Universidade Federal do Ceará

Mestrado em Ciência da Computação

Replicação de Mídias em Ambientes Distribuídos Utilizando a Arquitetura CORBA

Rodrigo de Oliveira Lopes

Dissertação de Mestrado

Fortaleza-CE

Dezembro / 2002

Replicação de Arquivos de Mídias em Ambientes de Educação a Distância e de Realidade Virtual

Rodrigo de Oliveira Lopes

Dissertação de Mestrado

Replicação de Arquivos de Mídias em Ambientes de Educação a Distância e de Realidade Virtual

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Rodrigo de Oliveira Lopes e aprovada pela Banca Examinadora.

Fortaleza, 23 de Dezembro de 2002.

José Neuman de Souza, Dr. (Orientador)

Dissertação apresentada ao Mestrado em Ciência da Computação, UFC, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Mestrado em Ciência da Computação

Universidade Federal do Ceará

Replicação de Arquivos de Mídias em Ambientes de Educação a Distância e de Realidade Virtual

Rodrigo de Oliveira Lopes

Dezembro de 2002

Banca Examinadora:

- José Neuman de Souza, Dr. (Orientador)
  Universidade Federal do Ceará - UFC
- Rossana Maria de Castro Andrade, Dra.
  Universidade Federal do Ceará - UFC
- Francisco Vilar Brasileiro, Dr.
  Universidade Federal da Paraíba - UFPB

Agradecimentos

Agradeço inicialmente, e principalmente, a Deus por me iluminar e por ter me dado forças para a conclusão deste trabalho.

Agradeço também a meus pais - Lucas e Auristela - e a meus irmãos - Rachel, Renato e Rafael - pelo incentivo e confiança no desenvolvimento deste trabalho.

Ao professor Dr. José Neuman de Souza, pela disposição na orientação deste trabalho. Ao Melo Júnior e Camilo pelas dicas, esclarecimentos e discussões iniciais importantes para esta dissertação.

À Poliana pelo estímulo, paciência e compreensão.

Aos amigos e colegas mestrandos, deixo meus agradecimentos pela convivência e troca de experiências. Camila, Felipe, Allan, Yuri, Debora, aos que sempre estiveram ao meu lado, me apoiando e incentivando nesta caminhada.

Aos *pelejantes*, pelos momentos de descontração. Àqueles que que deram força e apoio principalmente nos momentos de maior dificuldade do desenvolvimento deste trabalho.

À CAPES que financiou este trabalho e ao Departamento de Computação da UFC que forneceu a estrutura necessária para execução deste trabalho. Ao Orley por sua presteza. Aos professores Dr. Francisco Vilar Brasileiro e Dra. Rossana Maria de Castro Andrade por terem colaborado na avaliação desta trabalho.

A todos vocês, mais uma vez, muito obrigado.

Resumo

A revolução das telecomunicações e da informática e a popularização da Internet possibilitaram a criação de um ambiente propício para o desenvolvimento de aplicações distribuídas. Porém, a baixa largura de banda se apresenta como uma deficiência no uso da Internet como base de comunicação para uma aplicação distribuída, sobretudo aplicações de realidade virtual e educação a distância que demandam canais de comunicação rápidos e eficientes. Para possibilitar a utilização da Internet como meio de comunicação destas aplicações, algumas propostas têm feito uso de técnicas de replicação de arquivos e de *caches* locais para armazenamento das informações.

Replicação sob demanda é uma técnica oriunda dos sistemas de armazenamento de bancos de dados, que permite a cópia de informações específicas, através da rede, à medida que as mesmas se fazem necessárias. *Caches* locais são regiões de memória que armazenam dados necessários a algum tipo de processamento específico. Sempre que essas informações se fizerem necessárias, as mesmas são acessadas diretamente da *cache* local, sem a necessidade de novo *download*. O uso de *caches* locais implica em uma menor utilização da rede.

Neste trabalho é feita a análise de algumas arquiteturas que definem o uso de servidores específicos para disponibilização de mídias, cada uma com suas características próprias, com o uso ou não de replicação e de *caches* locais. O objetivo dessa dissertação

é definir, baseado nas propriedades apresentadas por cada arquitetura analisada, e implementar, usando CORBA, um servidor de fornecimento de arquivos de mídias que atenda às necessidades de aplicações distribuídas que precisem de tal serviço.

Palavras-Chaves: *Cache*, Replicação, Ambientes Distribuídos, CORBA.

Abstract

The revolution of telecommunications and computers and the Internet's popularity made possible the creation of an ideal environment for development of distributed applications. Unfortunately, low bandwidth shows up as a deficiency in the use of the Internet as communication basis of a distributed application, specially virtual reality and distance learning, which require fast and efficient communication channels. Some proposals have used local caches and replication of files as a way to make possible the use of Internet to these applications.

Replication on demand is a deriving technique of data bases storage systems that allows the copy of specific information through the network, as it is necessary. Local caches are memory regions that store data necessary to some type of specific processing. Whenever this information is necessary, it is obtained directly from the local cache, without the necessity of download. The use of local caches implies in a better network usage.

In this work we analyze some architectures that define the use of specific servers for providing media, each one with its proper features, as the use or not of replication and local caches. The aim is to propose and implement a media file replication server, based in the features presented by each architecture, using CORBA, that takes care of the necessities of some architectures that need such service.

Keywords: Cache, Replication, Distributed Environments, CORBA.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1. Motivação

A baixa largura de banda se apresenta como uma deficiência no uso da Internet como estrutura base para uma aplicação distribuída. O desempenho das aplicações cai drasticamente à medida que mais pessoas começam a acessar a rede. Daí surgiu a necessidade de se propor arquiteturas que possibilitem o desenvolvimento de aplicações distribuídas que utilizam satisfatoriamente uma rede com baixa largura de banda como meio de comunicação entre seus componentes.

Entre essas arquiteturas destacamos algumas que viabilizam a construção de ambientes virtuais ([1] e [2]) e de ambientes de ensino a distância ([3]). Estas arquiteturas definem componentes especializados que distribuem o processamento pela rede e aplicam técnicas que minimizam a utilização da rede de comunicação de forma a obter um melhor aproveitamento da estrutura de distribuição.

Porém existem melhorias que não foram apresentadas nas arquiteturas pesquisadas e que podem reduzir ainda mais a utilização da rede de comunicação da aplicação distribuída. Pode-se diminuir a quantidade de acessos que são feitos aos servidores a partir dos clientes. Além disso, estas arquiteturas ainda são dependentes de servidores específicos para execução de determinadas tarefas. Na falta de um destes servidores, o sistema pode entrar em colapso. É necessário tornar estes sistemas mais tolerantes a falhas.

1.2. Objetivos

O presente trabalho propõe a utilização de *caches* locais para armazenamento de mídias utilizadas em ambientes virtuais e sistemas de educação a distância.

Inicialmente são apresentadas algumas arquiteturas distribuídas para desenvolvimento de ambientes virtuais e de aplicações para educação a distância, dando ênfase, sobretudo, a como estas arquiteturas tratam o problemas da disponibilização de mídias aos clientes de suas aplicações.

Posteriormente, foi feito um estudo de algumas plataformas que viabilizam a construção de aplicações distribuídas. Foi executada uma análise geral sobre CORBA (Common Object Request Broker Architecture) e um breve estudo de outras possibilidades como DCOM (Distributed COM) e Java RMI (Remote Method Invocation) seguido de uma conclusão a respeito da plataforma escolhida para dar base à proposta deste trabalho.

A partir de todas as informações obtidas, foi proposto o servidor de replicação de mídias que pode ser acoplado a quaisquer das arquiteturas antes apresentadas e ainda apresenta alguns benefícios em relação aos que foram propostos anteriormente, como o uso de *caches* locais e de replicação como forma de reduzir o uso da rede e tornar o sistemas mais tolerante a falhas. A partir desta proposta foi implementado o servidor, utilizando-se CORBA como plataforma de distribuição, para validação da proposta.

1.3. Estrutura da Dissertação

Esta dissertação está dividida em 5 capítulos, incluindo este introdutório, e um anexo, conforme explicamos a seguir:

- **Capítulo 2 - CORBA: Common Object Request Broker Architecture**

  O capítulo 2 apresenta a arquitetura CORBA, definida pela OMG e que possibilita a implementação de objetos distribuídos que podem se comunicar entre si de forma transparente e independente de em qual linguagem foram implementados ou de em qual plataforma estão executando.

- **Capítulo 3 - Arquiteturas para Distribuição de Aplicações de Realidade Virtual e de Educação a Distância**

  - **Seção 3.3 - Arquitetura ARVORE**

    Nesta seção é apresentada a arquitetura ARVORE (Arquitetura para Distribuição de Realidade Virtual através de Replicação) [2], uma arquitetura que viabiliza a utilização de redes de pequena largura de banda para disponibilização de elementos de multimídia e de realidade virtual (RV) em ambientes multiusuários. A ARVORE também apresenta um módulo responsável pela transferência dos arquivos de mídia e de RV entre os outros módulos da arquitetura. Este módulo é exposto com mais detalhes neste capítulo.

  - **Seção 3.4 - Arquitetura Distribuída para Desenvolvimento de Aplicações de Educação a Distância**

    A seção 3.4 apresenta uma arquitetura distribuída para o desenvolvimento de aplicações de educação a distância, definida em [3], dando ênfase especial ao esquema de funcionamento das transferências de mídias.

  - **Seção 3.5 - Arquitetura Ataxia**

    A seção 3.5 apresenta a arquitetura ATAXIA [1], que define componentes especializados para possibilitar a distribuição do processamento de um ambiente virtual em rede e assim, viabilizar o uso de NVEs (*Networked Virtual Environments* - Ambientes Virtuais em Rede) para educação a distância em uma rede de baixa largura de banda.

- **Capítulo 4 - Servidor de Replicação**

  O capítulo 4 apresenta a principal contribuição desta dissertação, que é a descrição de um servidor de replicação de arquivos de mídia que apresenta propriedades que o tornam capaz de ser utilizado em quaisquer das arquiteturas apresentadas nos capítulos anteriores e também em outras.

- **Capítulo 5 - Conclusões e Trabalhos Futuros**

  Neste capítulo são apresentadas considerações finais a respeito do servidor proposto e nele é finalizada a dissertação com a enumeração de alguns trabalhos a serem realizados futuramente a fim de dar continuidade ao descrito nesta dissertação.

Capítulo 2

CORBA: Common Object Request Broker Architecture

No início, os sistemas de computação eram marcados pela independência de comunicação. *Hardware* e *software* eram proprietários. As aplicações atendiam a propostas bastante específicas. Os recursos computacionais eram bastante caros e sua capacidade de compartilhamento era mínima [4].

Após esta fase, surgiram as redes de computadores. Estas, ainda imaturas, utilizavam protocolos de comunicação proprietários, que impediam o intercâmbio de informações entre diferentes sistemas. Os sistemas, caracterizados por sua heterogeneidade, sendo compostos por aplicações implementadas em diferentes linguagens de programação, em computadores de várias arquiteturas e conectados por diversos elementos físicos de redes, tinham deficiências marcantes, tais como a falta de integração.

As plataformas de suporte à distribuição surgiram então com o objetivo de possibilitar que as aplicações se comunicassem umas com as outras e de uma forma transparente, sem que o desenvolvedor do sistema se preocupasse como foi desenvolvida uma aplicação da qual necessitasse obter dados externos. E ainda sem se preocupar onde e em que tipo de sistema operacional está sendo executada, uma vez que as plataformas distribuídas permitiam a característica de interoperabilidade entre objetos.

A plataforma CORBA, resultado da arquitetura OMA, desenvolvida por uma orga-

nização internacional constituída por empresas de computação e telecomunicações - a OMG (Object Management Group) [5], fornece mecanismos que garantem interoperabilidade entre objetos, fazendo isso transparentemente, através de pedidos e respostas em um ambiente distribuído. É basicamente um ambiente de suporte a aplicações distribuídas orientadas a objeto.

As principais características de CORBA são:

- Seu modelo de programação é simples e eficiente, onde são permitidas invocações a métodos de forma estática e dinâmica;
- Transparência local e remota, tanto no que diz respeito à linguagem de programação, como na capacidade de interoperabilidade entre objetos construídos em ambientes operacionais distintos;
- Trata-se de uma arquitetura que permite coexistência com outros sistemas existentes. A utilização de CORBA trata código existente como um objeto, mesmo se este foi implementado utilizando, por exemplo, linguagens de programação como COBOL e C++.

2.1. Arquitetura

A figura 2.1 mostra os quatro principais elementos que formam a arquitetura CORBA.

O elemento fundamental desta arquitetura é o ORB (Object Request Broker). Quando um componente de um aplicação deseja utilizar um serviço de um outro componente, primeiro ele deve obter uma referência para o objeto que fornece o serviço. Após esta referência ter sido obtida, o componente pode fazer chamadas aos métodos desse objeto, utilizando assim o serviços fornecidos. A responsabilidade primária do ORB é resolver requisições para as referências de objetos remotos possibilitando que componentes das aplicações estabeleçam conexão entre si.

Após um componente de uma aplicação obter uma referência para um objeto cujos serviços o componente deseja utilizar, o componente pode fazer chamadas aos métodos

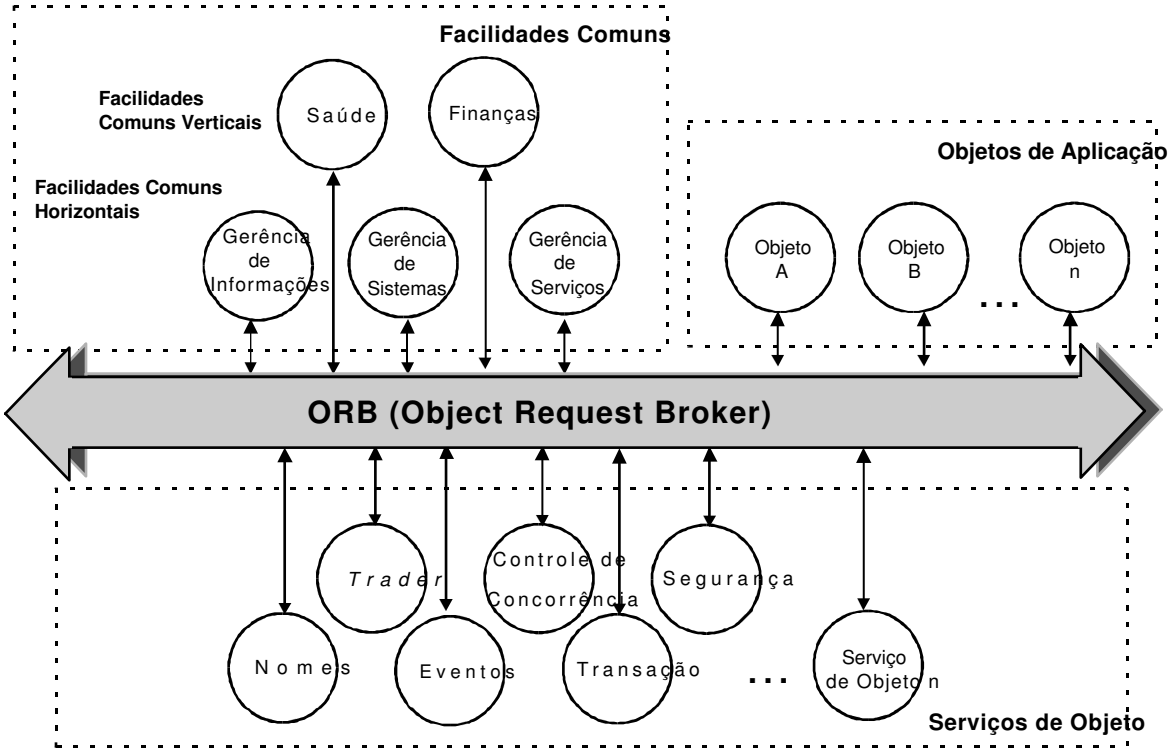

Figura 2.1. *A arquitetura da plataforma CORBA*

desse objeto. Geralmente esses métodos possuem parâmetros de entrada e retornam valores de saída. Outra responsabilidade do ORB é receber os parâmetros que o componente está enviando e "traduzir"(*marshalling*) estes parâmetros para um formato que possa ser transmitido para o objeto remoto. O ORB também faz a tradução inversa (*unmarshaling*) dos valores retornados, convertendo-os para o formato aceito pelo componente que chamou o método. Todo esse processo de tradução dos parâmetros de entrada, invocação do método do objeto remoto e tradução dos dados de retorno é executado pelo ORB automaticamente de maneira transparente.

Um consequência do processo de tradução dos dados é que a comunicação entre os componentes é independente de plataforma, uma vez que os parâmetros são convertidos para um formato independente de plataforma durante a transmissão e depois traduzido para o formato dependente da plataforma na recepção. Quaisquer diferenças de plataforma, seja de *software* ou *hardware*, são tratadas transparentemente pelo ORB.

Note que a OMG apenas define a especificação de CORBA. A partir das especificações, empresas implementam seus ORBs. Atualmente, já existem cerca de 70 ORBs, dando suporte a 15 linguagens de programação em diversas plataformas operacionais.

Além do ORB, a arquitetura define ainda mais 3 componentes, descritos a seguir:

Serviços de Objeto: Os serviços de objetos CORBA especificam os serviços que são básicos às aplicações. São definidos 15 serviços CORBA pela OMG. Dentre eles, os principais (destacados na figura 2.1) são descritos na tabela 2.1 [6].

A descrição detalhada a respeito de cada um dos serviços de objetos da plataforma CORBA encontra-se nos documentos de especificação da OMG [5].

Facilidades Comuns: As facilidades comuns são coleções de objetos que proporcionam serviços a serem utilizados diretamente pelas aplicações, englobando funcionalidades mais ricas do que as dos serviços de objetos CORBA.

São divididas em facilidades horizontais e verticais. As horizontais implementam objetos para gerenciamento de informações, gerenciamento de sistemas e gerencia-

Serviço	**Função**
Serviço de Nomes	Para fazer a localização de um serviço, é informado um nome lógico ao servidor de nomes que mantém uma base de dados com informações de objetos e seus nomes lógicos.
Serviço *Trader*	Em vez de fazer uma busca informando nomes de objetos, são informados os serviços que os objetos oferecem.
Serviço de Eventos	Permite que objetos registrem ou removam dinamicamente seu interesse em eventos específicos
Serviço de Controle de Concorrência	Descreve como um objeto pode controlar acessos simultâneos feitos por mais de um cliente. Seu funcionamento acontece através de *locks* que são *tokens* que permitem o acesso a um determinado recurso. Este serviço foi projetado para ser utilizado em conjunto com o serviço de transações, de forma a coordenar as atividades de transações concorrentes.
Serviço de Transações	Permite que os contextos de uma transação sejam propagados transparentemente em uma aplicação CORBA. Clientes utilizam um conjunto de interfaces simples para criarem ou abortarem transações, enquanto internamente, interfaces definem implementações que suportam o serviço.
Serviço de Segurança	Suporta mecanismos de encriptação, autenticação e autorização pelos quais aplicações podem restringir e, consequentemente, proteger o acesso às operações de um objeto.

Tabela 2.1. *Descrição das funções e serviços oferecidos por CORBA*

mento de serviços. Além disso, criam objetos de interface gráfica para a definição de um padrão para desenvolvimento de aplicações. As verticais provêem interfaces definidas para segmentos especializados, tais como as áreas de saúde, transportes, seguros, entre outras.

Objetos de Aplicação: Estes objetos correspondem a aplicações desenvolvidas por usuários e não são padronizados pela OMG.

Uma característica importante da arquitetura CORBA é que ela tem um alto poder de integração de serviços. Serviços definidos como objetos da aplicação poderão interagir, por exemplo, com serviços de objetos CORBA e com algumas facilidades comuns através do ORB.

2.2. Modelo de programação

CORBA facilita o desenvolvimento independente de componentes heterogêneos através de uma linguagem de definição de interface, mais conhecida como IDL (Interface Definition Language).

A IDL permite que desenvolvedores definam interfaces para seus programas e objetos de forma padronizada. Com a IDL, são mapeadas as definições e os tipos para linguagens de programação, tais como C, C++, Smalltalk e Java. Em conseqüência, CORBA oferece aos desenvolvedores o mecanismo de transparência de linguagem, permitindo, assim, que objetos clientes e objetos servidores, escritos em linguagens diferentes, possam interagir entre si através do ORB.

A sintaxe da OMG IDL é derivada da linguagem C++, sendo acrescentadas novas funcionalidades para a especificação de ambientes distribuídos.

Em um arquivo IDL, pode-se identificar quatro elementos básicos:

- Modules - provêem um conjunto de descrições das interfaces (classes de CORBA).
- Interfaces - definem um conjunto de métodos que um objeto cliente pode invocar. A

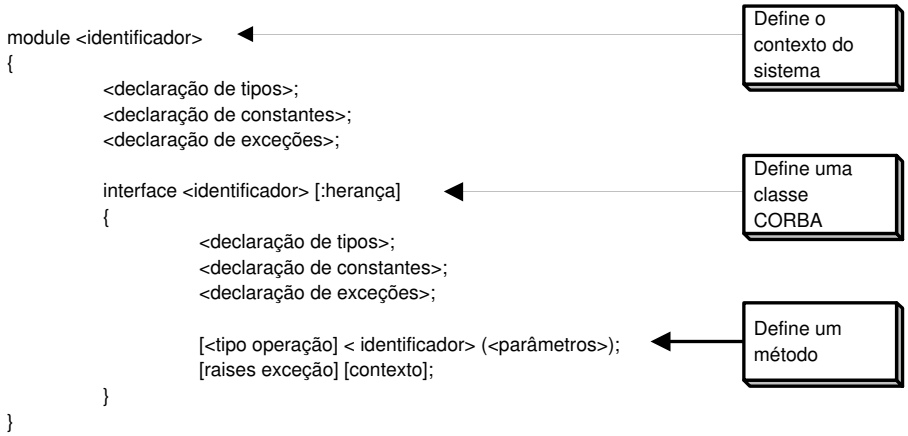

Figura 2.2. *A estrutura de um arquivo IDL*

interface pode declarar também exceções que indicam que uma operação não pôde ser executada com sucesso.

- Operações - denotam um serviço que o cliente pode invocar. Na declaração de uma operação são indicados os parâmetros e seu retorno.

- Tipos de dados - são usados para descrever os valores de parâmetros, atributos, exceções e tipos de retorno.

A estrutura de um arquivo IDL é apresentada na figura 2.2. IDL é uma linguagem estritamente declarativa. Ela é utilizada para descrição de objetos que compõem a aplicação distribuída.

O processo de criação de uma aplicação distribuída utilizando CORBA dá-se conforme a ilustração a seguir.(Figura 2.3)

Inicialmente, é necessário definir uma interface IDL para cada objeto da aplicação. Após a especificação da interface, é necessário fazer requisição a um pré-compilador, que traduzirá o código da interface para um conjunto de classes escritas em uma linguagem específica do compilador (por exemplo, Java ou C++).

Dentre estas classes criadas, encontram-se os códigos dos stubs e skeletons. Estes dois objetos são usados pelo ORB para permitir a comunicação entre os objetos cliente

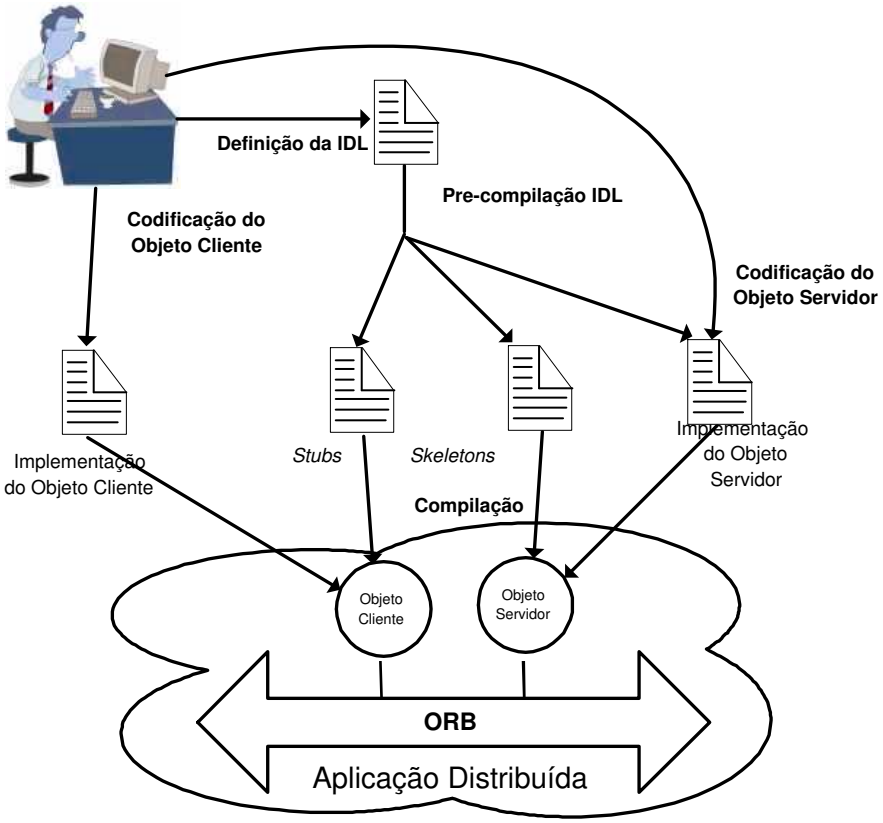

Figura 2.3. *Criação de uma aplicação distribuída usando CORBA*

e servidor, sendo o stub utilizado pelo objeto cliente e o skeleton utilizado pelo objeto servidor. Ou seja, os stubs e skeletons ligam as aplicações cliente e servidoras e o ORB.

Além da criação de stubs e skeletons, o compilador do código IDL cria uma classe especial onde deve ser realizada a implementação do objeto servidor. Após a fase de pré-compilação, o desenvolvedor do ambiente deve realizar a implementação dos objetos clientes e o objeto servidor. Após a compilação geral de stubs, skeletons e implementação dos objetos clientes e servidores, obtém-se a aplicação distribuída.

2.2.1. Exemplo de aplicação CORBA

Como exemplo de desenvolvimento de uma aplicação com o uso de CORBA, desenvolvemos a seguinte aplicação: Um objeto remoto disponibilizará um método que receberá uma *string* (sequência de caracteres) como parâmetro e retornará esta sequência com a ordem invertida. Por exemplo, caso o parâmetro seja a *string* 'CORBA', o retorno será a *string* 'ABROC'.

Primeiro, deve-se definir a IDL do objeto (Figura 2.4):

```
module exemplo {
    interface Inverte {
        string inverteString( in string param);
    };
};
```

Figura 2.4. *IDL do Objeto Inverte*

O objeto *Inverte* define o método inverteString que recebe uma *string* como parâmetro e que retorna uma *string* como saída.

Em seguida, deve-se usar o compilador IDL para serem gerados os *stubs* e *skeletons*:

```
idlj Inverte.idl;
```

Depois é necessário prover uma implementação para esta interface (Figura 2.5):

A classe que implementa a interface definida na IDL, deve herdar as funcionalidades do *skeleton* que foi gerado pelo compilador IDL. Além, é claro, de prover uma implementação

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*
import exemplo.*

class InverteImpl extends _sk_Inverte implements Inverte {
    InverteImpl() {
    }

    public String inverteString(String param)
        throws org.omg.CORBA.SystemException {
        StringBuffer strbuf = new StringBuffer(param);
        String retorno = (strbuf.reverse()).toString();
        return retorno;
    }
}
```

Figura 2.5. *Implementação da IDL do objeto Inverte*

para o método *inverteString* definido na IDL.

A seguir, é necessário criar uma aplicação que torne disponível o objeto Inverte para que ele seja acessível remotamente através do ORB (Figura 2.6):

Por último, implementamos uma aplicação cliente que irá utilizar o objeto Inverte que está disponibilizado remotamente (Figura 2.7):

Para executar o exemplo, é necessário compilar (*javac*) os stubs e skeletons, a aplicação servidora e a aplicação clientes:

```
javac exemplo/*.java

javac InvertImpl.java

javac InvertServer.java

javac InvertCliente.java
```

Depois deve-se executar o servidor de nomes (*tnameserv*) e a aplicação servidora, que registrará o objeto Inverte no servidor de nomes.

```
start tnameserv

start java InvertServer
```

Finalmente, executa-se a aplicação cliente que fará uso do objeto remoto.

```
java InvertClient
```

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*
import exemplo.*

class InverteServer {
    public static void main(String[] args) {
        try {
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
            InverteImpl inv = new InverteImpl("Inverte");
            orb.connect(inv);
            org.omg.CORBA.Object nameServiceObj =
                    orb.resolve_initial_references ("NameService");
            org.omg.CosNaming.NamingContext nameService =
                    org.omg.CosNaming.NamingContextHelper.narrow(nameServiceObj);
            NameComponent[] invName = {new NameComponent("Inverte", " ")};
            nameService.rebind(invName, inv);
            Thread.currentThread().join();
        }
        catch(Exception e) {
            System.err.println(e);
        }
    }
}
```

Figura 2.6. *Aplicação servidora*

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*
import exemplo.*

class InverteClient {
    public static void main(String[] args) {
        try {
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
            org.omg.CORBA.Object nameServiceObj =
                    orb.resolve_initial_references ("NameService");
            org.omg.CosNaming.NamingContext nameService =
                    org.omg.CosNaming.NamingContextHelper.narrow(nameServiceObj);
            NameComponent[] invName = {new NameComponent("Inverte", " ")};
            Inverte inverte = InverteHelper.narrow(nameService.resolve(invName));

            BufferedReader in = new BufferedReader(
                    new InputStreamReader(System.in));
            for(;;) {
                System.out.print("> ");
                String line = in.readLine();
                if ((line==null) || line.equals("exit"))
                    break;
                String ret = inve.inverter(line);
                System.out.println("Servidor retornou> " + ret);
            }
        }
        catch(Exception e) {
            System.err.println(e);
        }
    }
}
```

Figura 2.7. *Aplicação cliente*

2.3. Serviço de Nomes

O Serviço de Nomes CORBA é um serviço padrão para implementações da arquitetura CORBA definido na especificação dos serviços CORBA, pela OMG [7]. Ele permite que se associe nomes às referências de objetos CORBA e possibilita que clientes encontrem estes objetos procurando pelos respectivos nomes. É o principal mecanismo para que objetos no ORB localizem outros objetos. O Serviço de Nomes faz, então, um mapeamento entre o nome e a referência para objeto.

O mapeamento entre nomes e objetos é denominado *binding*. Um *Contexto de Nomes* é um ambiente onde o nome de um determinado objeto é único. O Serviço de Nomes permite que se crie uma hierarquia de nomes, através da qual os clientes podem navegar por diferentes Contextos de Nomes até encontrar o objeto que procuram.

Um objeto pode ser referenciado usando-se uma sequência de nomes que formam uma árvore hierárquica de nomes (Figura 2.8). Na figura, cada nó escuro é um Contexto de Nomes. O nome de um objeto consiste de uma sequência de nomes (ou componentes) que formam um nome composto. Cada componente, exceto o último, nomeia um contexto. O último é o nome simples do objeto [8].

2.4. Serviço Trader

Sistemas distribuídos geralmente executam num ambiente de plataformas de software, hardware e rede heterogêneos. Para utilizarem os serviços disponíveis nesses sistemas, os usuários devem conhecer quais são estes serviços e onde os mesmos estão localizados. Além disso, em grandes sistemas, a versão e a localização dos serviços pode mudar com uma certa frequência, o que torna essencial a existência de mecanismos de localização e acesso dinâmicos aos serviços.

O *Serviço Trader* fornece um serviço de localização de objetos. Os provedores de

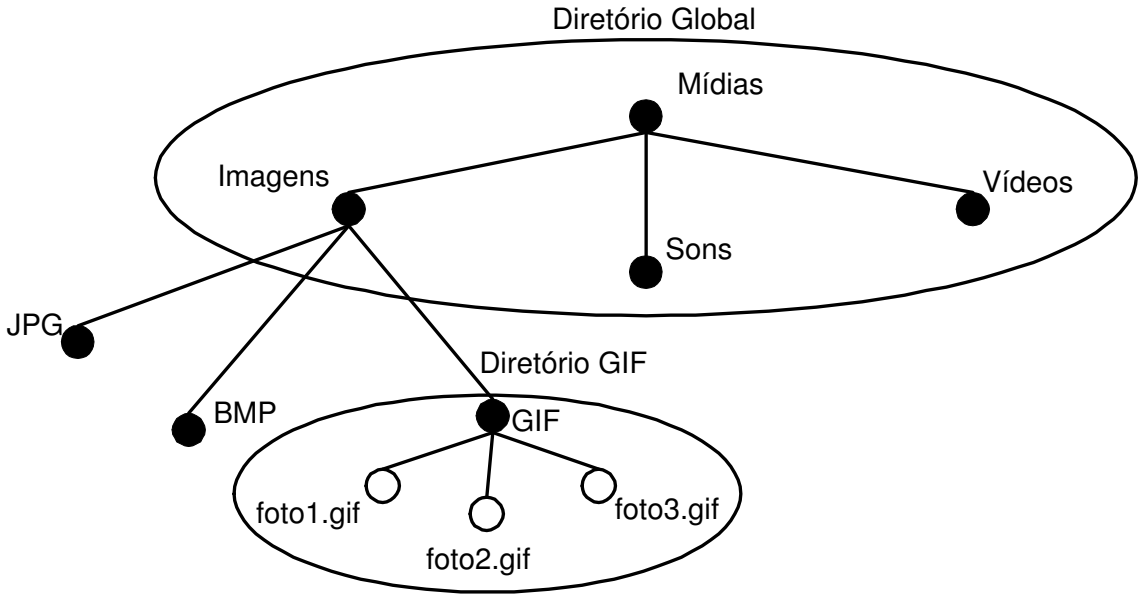

Figura 2.8. *Hierarquia de nomes no Serviço de Nomes CORBA*

serviços utilizam o *Serviço Trader* para anunciar seus serviços e os usuários consultam o *Serviço Trader* para obter informação sobre determinados serviços, baseados no tipo do serviço e em características específicas de cada serviço.

Por exemplo, um banco divulga sua contas de poupança através do anúncio de sua taxa de juros e dos serviços que ele oferece ao cliente (cheque, cartão etc). Enquanto isso, um cliente deseja encontrar um banco que ofereça a melhor taxa e os melhores serviços. o papel do *Serviço Trader* é apresentar o banco ao cliente.

2.4.1. Ofertas de Serviços

Em CORBA, um serviço é fornecido por um ou mais objetos, definidos em uma ou mais interfaces. Para divulgar e localizar serviços num sistema distribuído, é necessário primeiro descrever os serviços. Uma *oferta de serviço* é uma descrição deste serviço contendo:

- detalhes de cada interface do serviço, consistindo de:
  - Referência para a interface - Para utilizar um serviço, um cliente precisa possuir uma referência para as interfaces que fornecem este serviço.

  - Tipo de interface - CORBA exige que as interfaces sejam fortemente tipadas. Para utilizar um serviço, o cliente precisa saber quais atributos e operações existem nas suas interfaces.

- Outras características do serviço - Mesmo que duas interfaces tenham o mesmo tipo, as suas implementações podem ser diferentes, o que fornece uma base para comparação de serviços semelhantes. A OMG define duas formas para caracterizar interfaces individualmente:

  - atributos que são definidos de forma estática na definição da interface;
  - propriedades que são dinamicamente definidas pelo uso do *Serviço de Propriedades*.

O *Serviço de Propriedades* provê operações para armazenar recuperar propriedades, através da interface *PropertySetDef*. Uma propriedade consiste de:

- Um nome
- Um valor tipado
- Um modo, que pode ser:

  normal: Propriedade pode ser alterada ou excluída

  read_only: Propriedade não pode ser alterada mas pode ser excluída

  fixed_normal: Propriedade pode ser alterada mas não pode ser excluída

  fixed_readonly: Propriedade não pode ser alterada nem excluída

Como é possível armazenar tipos, referências e atributos de interfaces no *Serviço de Propriedades*, a interface *ServiceOffer* do *Serviço Trader* estende a interface *PropertySetDef* do Serviço de Propriedades. Sendo assim, a interface *ServiceOffer* possui métodos para definir propriedades, com seus valores e modos; obter valores e modos de propriedades; e apagar propriedades. Além disso, a interface *ServiceOffer* possui uma operação para avaliar expressões envolvendo os valores das suas propriedades.

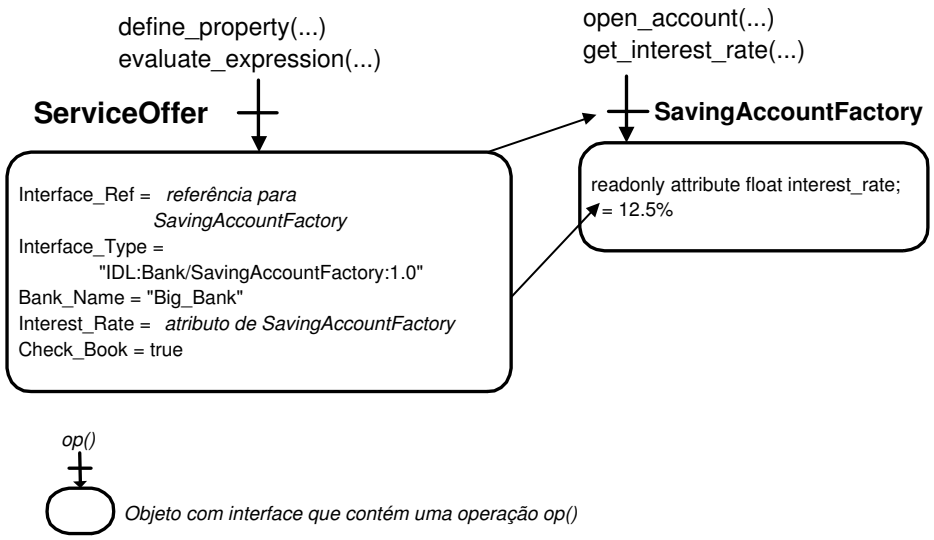

Figura 2.9. *Uma oferta de serviço para o objeto SavingAccountFactory*

Quando um cliente deseja uma conta bancária uma alta taxa de juros e com serviço de talão de cheques, ele pode verificar se a oferta do serviço se satisfaz à condição "*Taxa de juros* $\geq$ *10 e Talão de cheques*". No exemplo da figura 2.9, a oferta atende à requisição do cliente.

2.4.2. Traders

Um *Trader* possui um registro de todas as ofertas de serviços; ele armazena referências para interfaces *ServiceOffer*. O cliente utiliza o Trader para localizar ofertas de serviços que satisfaçam seus requerimentos.

As duas principais funcionalidades de um *Trader* são:

- Registrar/Divulgar ofertas de serviços

  Geralmente, um objeto servidor registra seu serviço no *Trader*, mas esse registro pode ser feito por um outro objeto. O objeto que registra o serviço é denominado como o *Exportador*. Existe a operação correspondente para remover a oferta de serviço do registro do *Trader*. As operações de registro (*register(...)*) e remoção (*withdraw(...)*) estão disponíveis na interface *Exporter*.

Se as propriedades do serviço são modificadas, a oferta do serviço é atualizada através das operações da interface *PropertySetDef*; o *Trader* não é envolvido.

- Importar (localizar) ofertas de serviços
  Um objeto pode importar ofertas de serviços a partir do *Trader*; este objeto é definido como o *Importador*. O Importador pode pesquisar por uma coleção de ofertas de serviços que atendam aos requerimentos do cliente ou selecionar uma única oferta atenda a esses requerimentos e melhor satisfaça ao *critério de seleção*. As operações de pesquisa (*search(...)*) e de seleção (*select(...)*) estão definidas na interface *Importer*.

O *Trader* implementa as interfaces *Importer* e *Exporter*.

Por exemplo, o banco registra sua oferta de serviço com o *Trader*. O cliente deseja abrir uma conta que ofereça a maior taxa de juros e que ofereça a facilidade de talão de cheques. O cliente executa uma operação de seleção da interface *Importer* para localizar um serviço que satisfaça à restrição "*Taxa de juros* $\geq$ *10 e Talão de cheques*" e que melhor atenda ao critério de seleção "*max(Taxa de Juros)*". Quando o banco decide não mais oferecer seu serviço de conta bancária, ele usa a operação *withdraw* da interface *Exporter* para remover sua oferta de serviço do registro do *Trader*.

2.5. Considerações

CORBA se apresenta como uma boa escolha para o desenvolvimento do trabalho pelo fato de ser um padrão aberto, com transparência e independência de linguagem de programação e de sistema operacional. A aplicação desenvolvida neste trabalho deverá interagir com aplicações anteriormente desenvolvidas e a independência de linguagem e de plataformas apresentadas pelo CORBA darão suporte para que isso seja possível.

Além disso, os serviços de *Nomes* e *Trader* serão de grande valia para a implementação da replicação dos arquivos, uma vez que para cada arquivo estará disponível um objeto no ORB, que será localizado através desses serviços.

Capítulo 3

Arquiteturas para Distribuição de Aplicações de Realidade Virtual e Educação a Distância

Distribuir elementos multimídia e de realidade virtual em redes de baixa velocidade é uma tarefa complexa devido às características desses tipos de mídia, como a necessidade de canais de comunicação rápidos e a capacidade de negociar QoS. Adaptar estruturas de distribuição, baseadas no paradigma cliente-servidor, e aplicar tecnologias atuais a fim de se viabilizar esse tipo de distribuição é importante.

Normalmente, a distribuição de recursos de multimídia e de realidade virtual (RV) não se aplica a redes de computadores de baixa velocidade.

Neste capítulo são apresentadas três arquiteturas voltadas para a distribuição de ambientes de realidade virtual e/ou aplicações de educação a distância. Inicialmente apresentamos a arquitetura ARVORE, uma arquitetura desenvolvida para viabilizar a utilização de redes de pequena de largura de banda para distribuição de elementos de realidade virtual em ambiente virtual em rede (NVE - Networked Virtual Environment). Em seguida é apresentada uma arquitetura distribuída para o desenvolvimento de aplicações de educação a distância e finalmente é descrita a arquitetura ATAXIA, que define componentes especializados para possibilitar a distribuição do processamento de um ambiente virtual e assim, viabilizar o uso de NVE's para educação a distância.

3.1. Ambientes Virtuais em Rede

Os NVE's (Networked Virtual Environments - Ambientes de Realidade Virtual em Rede) são sistemas que permitem a interação entre usuários geograficamente distantes no interior de um ambiente virtual compartilhado. Cada participante faz uso de uma cópia específica desse ambiente, geralmente através de um sistema computacional próprio. Os NVE's são oriundos da junção de três grandes áreas de aplicação da computação: a de aplicações compartilhadas através de redes de computadores, a de computação gráfica e a de realidade virtual [9].

Em um NVE, um usuário pode interagir com objetos do próprio ambiente ou com os outros usuários no interior do ambiente. Assim, sempre que um evento provocar alteração do estado de uma das cópias do ambiente virtual compartilhado, ele é transmitido automaticamente aos computadores utilizados pelos participantes, para que as cópias do ambiente virtual sejam atualizadas. Dessa forma, a consistência geral do NVE é mantida e os participantes têm a impressão de compartilhar um único ambiente virtual. Essa impressão de compartilhamento, que existe em um NVE, estimula fortemente a colaboração entre os participantes para a execução das mais variadas tarefas [10].

NVE's têm sido empregados com sucesso em: [11; 12]

- teleconferência com troca de objetos multimídia;
- trabalhos colaborativos envolvendo modelos tridimensionais;
- ambientes de jogos multiusuários;
- teleshopping, envolvendo modelos tridimensionais, imagens, sons;
- aplicações médicas (diagnósticos a distância, cirurgias virtuais para treinamento);
- agência de viagens virtual;
- estúdio virtual de vídeo com integração de mídias através de rede;

- aprendizagem e treinamento a distância.

Nos últimos anos, a significativa evolução do poder de processamento gráfico dos computadores e o grande aumento da capacidade de transmissão de dados em redes de computadores têm viabilizado as aplicações em NVE's, tornando-as cada vez mais atrativas e usuais. A gama de aplicações potenciais em NVE's tem atraído o interesse de pesquisadores que têm focalizado suas pesquisas em aspectos específicos dos NVE's, tais como: a escalabilidade e as topologias de rede; a estruturação eficiente do espaço; as simulações em tempo-real; e o sentimento de presença [1].

[13] compilou um conjunto condições que, a seu ver, deveriam ser atendidas para a manutenção de um nível aceitável de fidelidade entre o mundo virtual e o mundo real em um NVE:

Fidelidade de realismo visual: o ambiente deve ser o mais realista possível;

Fidelidade de modelagem: os objetos devem apresentar proporções corretas entre si e ser movimentados em velocidades realistas, sendo seus comportamentos replicados de forma precisa a todos os participantes;

Fidelidade de tempo: deve haver um atraso mínimo entre a ação de uma determinada entidade virtual (objetos, personagens, etc) e a retransmissão da mesma às cópias de ambiente de todos os participantes;

Fidelidade de informação: a quantidade e a consistência das informações distribuídas no NVE devem ser suficientes para o desenvolvimento de todas as situações previstas e para o suporte à tomada de decisões, tanto pelos diversos participantes do NVE, como pelos possíveis personagens controlados por computadores;

Fidelidade do comportamento de personagens: o comportamento de possíveis personagens controlados por computadores deve simular as ações desempenhadas por personagens controlados por humanos face às mesmas circunstâncias;

Fidelidade física: o NVE deve incluir efeitos físicos existentes no mundo real;

Fidelidade sensorial: informações visuais, auditivas e háptico-cinestésicas apresentadas aos participantes devem replicar o mundo real da melhor forma possível;

Fidelidade dos dispositivos de entrada: o hardware utilizado para interagir com o NVE deve responder de acordo com as ferramentas simuladas do mundo real; e,

Fidelidade do sistema: as atividades e respostas dos diversos participantes do NVE (humanos ou máquinas) devem, quando consideradas como um todo, apresentar um comportamento geral próximo àquele do mundo real.

De acordo com [10], para minimizar a complexidade do NVE e torná-lo mais eficiente, em geral, algumas das condições de fidelidade descritas anteriormente são relaxadas ou desconsideradas. Eles sugerem os seguintes aprimoramentos das condições de fidelidade de um NVE para que elas sejam mais bem exploradas:

- a padronização pelo próprio NVE do comportamento de certos objetos deve ser estabelecida, a fim de minimizar a complexidade da definição do mundo virtual (Ex.: gravidade);
- o tratamento de colisões de componentes do ambiente deve ser considerado, uma vez que ele melhora significativamente a qualidade das simulações nos ambientes virtuais;
- a atribuição de direitos de acesso individualizado deve ser considerada, para minimizar os problemas de acesso a objetos e a partes do mundo virtual que ocorrem quando o NVE é compartilhado por um grande número de participantes. Esses direitos de manipulação de objetos e de entrada em determinados recintos virtuais devem ser assegurados a partir das necessidades de cada participante;
- técnicas refinadas de seleção e manipulação de objetos ou de suas partes [14] devem ser utilizadas a fim de melhorar a interação entre o participante do NVE e os

elementos do mundo virtual; e,

- estímulos háptico-cinestésicos devem ser explorados, através de force feedback, pois provêem os usuários com um maior grupo de sensações e, conseqüentemente, um maior realismo final.

3.2. Ambientes de Educação a Distância na Web

A forma mais fácil de disponibilizar informação na Web é através de livros on-line, os quais requerem apenas uma preparação eletrônica dos livros sem necessidade de conhecimentos de programação HTML (HyperText Markup Language) ou Java, por exemplo, pois existem programas que permitem a exportação desses livros a partir de um processador de textos. Porém, a disponibilização de conhecimento nestes termos é considerada a forma mais ineficiente para o sucesso da aprendizagem[3]. Estes tipos de sistema, em termos pedagógicos, não são aconselháveis. Utilizar apenas material didático exposto como um livro sem qualquer interação do aluno com outro participante de um curso é prejudicial para o aluno já que este não pode trocar suas experiências e compartilhar suas dúvidas com outras pessoas. Estes tipos de ambientes baseiam-se unicamente na forma de comunicação assíncrona.

A forma de comunicação assíncrona é caracterizada pelo fato de não ser necessário a presença simultânea do professor e estudantes no sistema. O e-mail, os fóruns de discussão e os grupos de notícias são bastante utilizados neste tipo de comunicação.

Outra classe de sistemas é conhecida como ambientes do tipo sites educacionais. Os sites educacionais reúnem um conjunto de funcionalidades, tais como biblioteca de software educacional, espaços para comunicação, software para *download*, *links* para outras referências de interesse, dentre outras coisas. Nestes ambientes, além de se empregar a comunicação assíncrona, é utilizado o conceito de comunicação síncrona. Neste tipo de comunicação, existe uma interação em tempo real entre os alunos e professores. Os estudantes e professores utilizam câmeras de vídeo e facilidades de áudio para receber as

aulas de forma *on-line* e ao vivo, em lugares remotos.

[15] enumera muitos exemplos de sites educacionais e destaca duas modalidades principais para estes ambientes de ensino. São elas:

Sistemas de autoria para cursos a distância: Estes sistemas oferecem ao professor um conjunto integrado de ferramentas para criação e aplicação de cursos. Aos usuários de ambientes de autoria são disponibilizados recursos como conversa *on-line*, *whiteboard* e videoconferência.

Salas de aula virtuais: Estes ambientes provêem um conjunto de aulas, cujo conteúdo é apresentado como um conjunto de transparências, textos, vídeos e imagens. A interatividade é garantida pelos serviços que a Internet provê, tais como o correio eletrônico e listas de discussão, e também através de sessões de bate-papo (*chat*) e videoconferência.

Alguns exemplos de ambientes de EAD existentes são o AulaNet, o LearningSpace, TopClass, FirstClass, Virual-U, WebCT e Persyst. Todos são analisados em [15] e em [16].

Estas ferramentas vêm sendo bastante utilizadas em universidades e empresas uma vez que elas possuem recursos bastante completos e podem ser muito bem empregados no processo de aprendizagem. No entanto, estes ambientes apresentam algumas deficiências. Os principais problemas dizem respeito à centralização do conteúdo da informação, à dependência de sistema operacional e à falta de suporte à interação entre múltiplos usuários [3].

3.3. Arquitetura para Distribuição de Realidade Virtual através de Replicação

A distribuição de recursos de multimídia e de realidade virtual (RV) usualmente não é aplicada em redes de computadores de baixa velocidade. A transmissão de mídias contínuas

(*streams* de áudio e vídeo) em tempo real e o envio de imagens digitais processadas requerem maiores velocidades de transmissão e uma qualidade de serviço mais eficiente. [2]

ARVORE (Arquitetura para Distribuição de Realidade Virtual através de Replicação) apresenta uma arquitetura que viabiliza a utilização de redes de pequena largura de banda para disponibilização de elementos de Multimídia e de RV em ambientes multiusuários através de aplicações-clientes que permitem ao usuário interagir no interior de um ambiente de RV com personagens controlados por outros indivíduos ou mesmo por máquinas. Essa arquitetura foi idealizada a partir do conjunto de requisitos necessários à implementação das ferramentas propostas no Projeto AVAL (Ambientes Virtuais para o Aprendizado de Línguas)[17], que tem como principal objetivo a utilização de aplicações distribuídas de RV para o treinamento a distância de guias de turismo do Estado do Ceará.

Uma das principais características dessa arquitetura é a distribuição, a baixo custo, dos elementos de RV e de multimídia através de estruturas especializadas de replicação, que os enviam, sob demanda, para o armazenamento local e persistente nos diversos componentes previstos. [2]

Os sistemas gerados a partir da arquitetura ARVORE são compostos por clientes e servidores com funções especializadas, integrados através de uma entidade configurável, responsável pelo controle geral do fluxo de informações. Extensões a estes sistemas podem ser facilmente realizadas através da reconfiguração dessa entidade controladora.

Esses sistemas são gerenciáveis a partir de módulos específicos. Isso é possível, pois muitos dos componentes da ÁRVORE mantêm informações sobre seu próprio funcionamento e/ou do sistema como um todo, disponibilizando-as, posteriormente, às aplicações de gerenciamento do sistema.

Os clientes e servidores definidos na arquitetura ARVORE, destacados na figura 3.1, são descritos nas tabelas 3.1 e 3.2.

O Servidor de Roteamento (SRo) armazena as informações sobre os serviços disponíveis

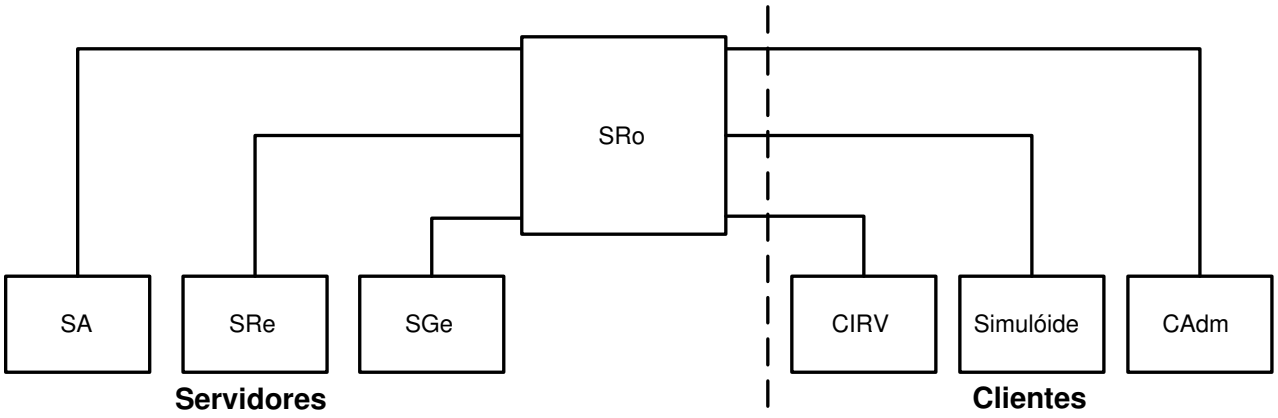

Figura 3.1. *Modelo lógico da arquitetura ÁRVORE*

Servidor	**Descrição**
Servidor de Roteamento	Elemento básico de integração da arquitetura, controlando o fluxo de informações entre os diversos componentes do sistema distribuído.
Servidores de Ambiente	Responsáveis pelo gerenciamento do comportamento do ambiente tridimensional, reagindo automaticamente às interações dos usuários.
Servidor de Replicação	Responsável pela distribuição, sob demanda, dos elementos de multimídia e de RV para o conjunto de clientes da arquitetura.
Servidores de Gerenciamento	Fornecem serviços para o controle e a documentação do funcionamento geral do sistema distribuído.

Tabela 3.1. *Descrição dos servidores definidos na ÁRVORE*

Cliente	**Descrição**
Cliente de Interação em RV	Disponibiliza o conjunto de serviços do ambiente de RV ao usuário comum do sistema.
Simulódides	Clientes controlados por máquinas, funcionando como interfaces que interligam aplicações externas ao ambiente RV.
Cliente de Administração	Ferramenta de gerenciamento, remoto ou não, e configuração do funcionamento de todo o ambiente.

Tabela 3.2. *Descrição dos clientes definidos na ÁRVORE*

em uma lista persistente que contém as relações entre serviços e suas respectivas localizações no ambiente distribuído. Esta estratégia de roteamento é fundamental para a extensibilidade da arquitetura, uma vez que basta incluir os novos elementos e reconfigurar a lista de serviços do SRo. Esta reconfiguração se dá através do Cliente de Administração.

Os Servidores de Ambientes (SA's) mantêm o estado de todas as estruturas virtuais que esbocem algum tipo de reação pré-programadas. As interações entre os personagens no interior do ambiente disparam eventos que são tratados pelos SA's. Os SA's também são responsáveis pela pelos eventos periódicos, eventos que são disparados independentemente da ação de personagens virtuais e cujo emprego está associado a fatores temporais.

O Servidor de Replicação (SRe) é utilizado sempre que um usuário tenta fazer uso de um elemento de multimídia ou de RV e não o encontra armazenado localmente. (Figura 3.2). Neste caso, um elemento genérico (coringa) é exibido e concomitantemente o cliente requisita ao SRe o elemento que não foi encontrado localmente. O SRe retorna o elemento diretamente ao cliente que o armazenará em sua base local. A inclusão de novos elementos a serem disponibilizados pelo SRe é feita pelo cliente de Administração.

O Servidores de Gerenciamento são responsáveis por prover serviços de validação de acesso aos diversos elementos que compõem a arquitetura e ao ambiente virtual como um todo. Além disso armazenam informações necessária à administração do sistema.

O cliente de interação em RV (CIRV) é personificado por uma entidade de um mundo

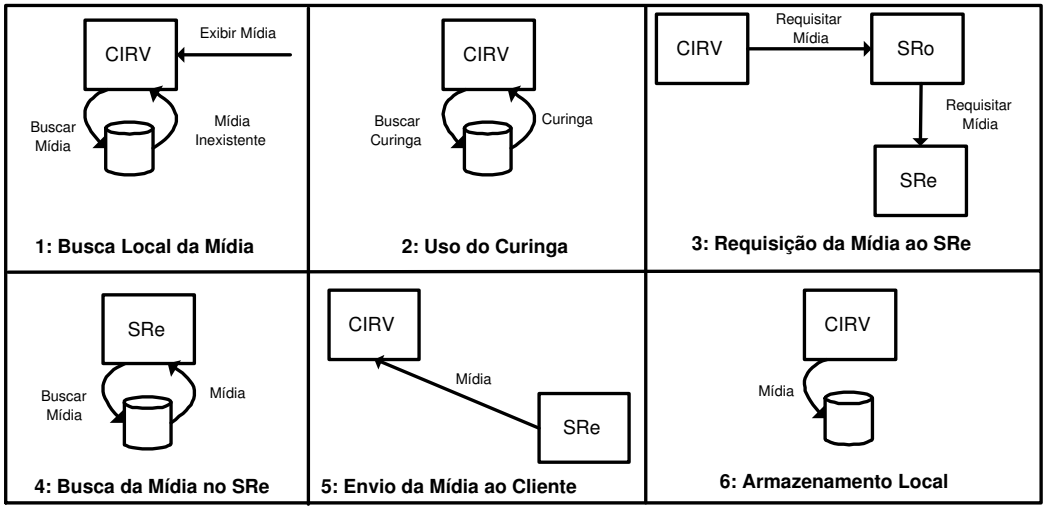

Figura 3.2. *Sequência de Requisição e Replicação de Objetos*

virtual que representa um usuário humano (avatar). Este cliente utiliza uma interface gráfica baseada em RV que, considerando as características do avatar e sua posição no ambiente, e processa as estruturas tridimensionais do mesmo. A comunicação entre os personagens do ambiente de RV é feita basicamente através de distribuição de texto, som, imagem ou animação. Os simulóides são clientes controlados por máquinas que funcionam como interface para acesso a aplicações externas ao ambiente. Os outros personagens interagem com os simulóides a fim de usufruirem de serviços fornecidos por aplicações externas aos mesmos.

A fim de mostrar a viabilidade da arquitetura ARVORE foi implementado um protótipo de um ambiente virtual para atender às necessidades do projeto AVAL e baseado na arquitetura proposta. Este primeiro protótipo contempla a construção de um modelo tridimensional de parte de um aeroporto e um simulóide que representa o comportamento de uma atendente de um balcão de informações.

O sistema de distribuição desse protótipo utiliza a Internet como meio de comunicação entre seus elementos. Mensagens de texto, contendo informações sobre os serviços utilizados, formam as unidades básicas de interação entre os diversos elementos do protótipo. Estas mensagens são transmitidas através do protocolo TCP. A informação distribuída

é formada basicamente por diálogos (texto) entre os usuários, arquivos de som e URL's (Universal Resource Locations) de páginas na Internet. O servidor de replicação foi implementado como um servidor FTP sem haver o uso de caches locais por parte dos clientes.

3.4. Arquitetura Distribuída para Desenvolvimento de Aplicações de Educação a Distância

EAD é uma das técnicas mais modernas em termos de prática educacional, sendo bastante difundida em escala mundial. A maioria dos sites educacionais é caracterizada por um modelo centralizado, onde praticamente todos os serviços de um curso encontram-se em um único local. Este modelo apresenta deficiências, tais como sobrecarga nos servidores e falta de robustez e tolerância a falhas. Visando solucionar o problema de centralização do conteúdo, em [3] é definido um cenário básico de uma aplicação EAD e seus serviços básicos e, posteriormente, proposta uma arquitetura na qual estes são disponibilizados por um conjunto de objetos distribuídos e que podem estar localizados em diferentes servidores.

A descentralização de processamento vem proporcionar importantes aspectos para as aplicações, tais como uma maior robustez e tolerância a falhas, dada a possibilidade de replicação de serviços, e a capacidade de atender a um maior número de usuários devido à diminuição da sobrecarga de tarefas nos servidores. No modelo cliente-servidor, em geral, isso não é possível, uma vez que as requisições de um usuário são processadas por um único servidor que, quando falha, compromete todo o sistema.

Uma aplicação de EAD neste novo cenário é constituída de um conjunto de objetos que podem estar distribuídos em diferentes localidades. Os objetos são considerados entidades que encapsulam estruturas de dados e servem como unidades básicas de informação. Para o usuário da aplicação, não há uma preocupação com a localização desses objetos devido à característica de transparência de localização dos objetos inerentes à aplicação de EAD.

Baseado nos conceitos de objetos distribuídos e plataformas abertas de suporte à dis-

tribuição, foi construído um modelo de estruturação de ambientes de ensino a distância. Este modelo identifica serviços necessários à implantação de um ambiente de EAD. Dentre estes serviços, pode-se citar: a) os serviços de suporte de mídias, responsáveis pela transferência, processamento e armazenamento de objetos de mídia, como textos, imagens, áudios e vídeos; b) serviços de suporte geral, responsáveis por autenticação de usuário, gerenciamento de informações do sistema, localização de serviços, gerenciamento dos eventos de usuários, gerenciamento da qualidade de serviço e escalonamento de aplicações; e c) serviços de aplicações inerentes à EAD, como dicionários, biblioteca de aulas, um sistema de avaliação e um sistema de autoria de aulas. (Tabela 3.3)

Grupo de Objetos	**Função**
Objetos de Aplicação para EAD	Estes são objetos específicos de aplicações de EAD. Como exemplo, pode-se citar objetos como dicionário, avaliação, biblioteca, e co-autoria de aulas.
Objetos de Suporte de Mídias	É uma coleção de objetos que provêem serviços especializados a dar suporte ao armazenamento, processamento e transferência de mídias, tais como áudio, vídeo, texto e imagem.
Objetos de Suporte Geral	Dentre os objetos de suporte geral, pode-se citar os responsáveis pela localização de objetos (objeto servidor de nomes e o Internet Trader), pelo tratamento de eventos (objeto servidor de eventos), pelo escalonamento de requisições a objetos (Meta-Scheduler), pelo gerenciamento de qualidade de serviço da aplicação (QoS Manager), pela segurança (autenticador) e pelo gerenciamento das informações do sistema de ensino (gerenciador de informações cadastrais).

Tabela 3.3. *Grupos de objetos de um ambiente de EAD*

Para cada um dos serviços identificados foi definido um conjunto de objetos de suporte a tais serviços. Na figura 3.3 são exibidos os objetos que dão suporte ao serviços especificados.

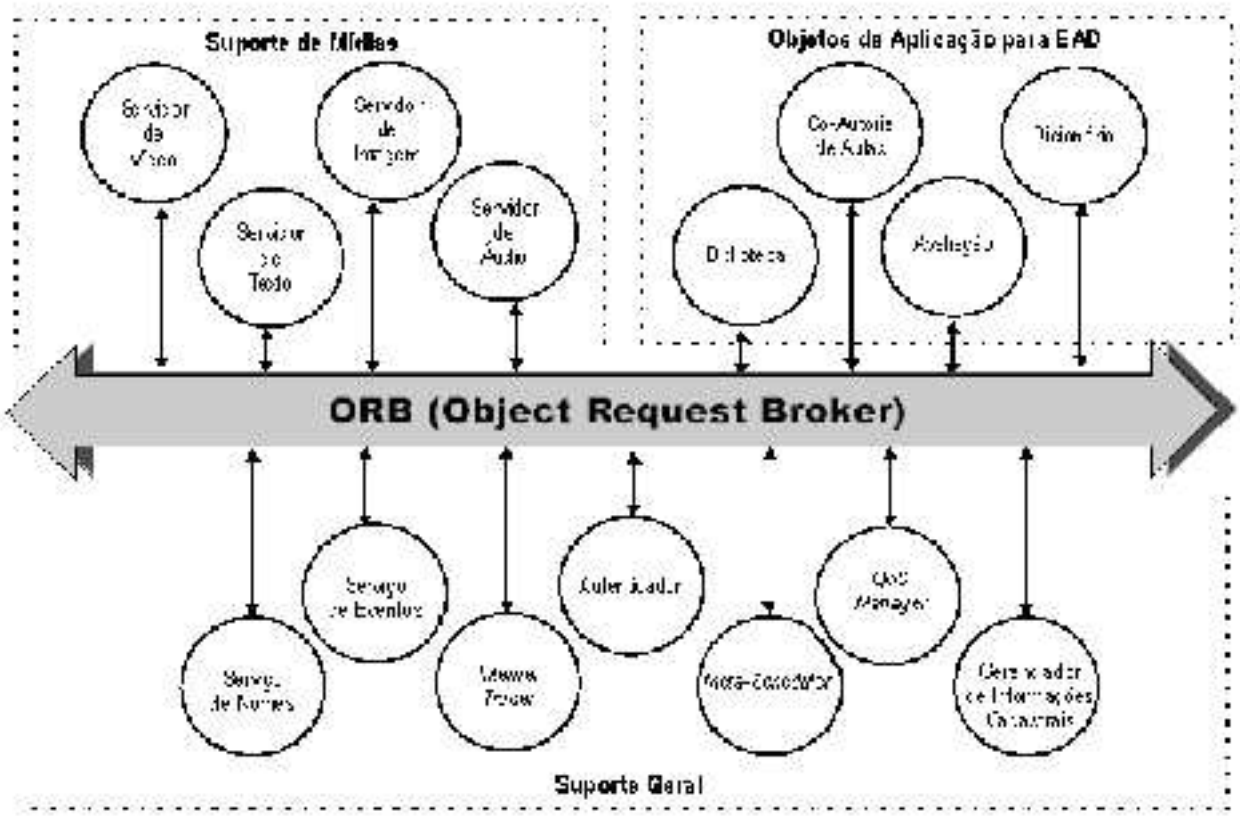

Figura 3.3. *Arquitetura de aplicação EAD*

Uma característica importante que pode ser observada é a transparência de localização. Um objeto cliente requisita ao servidor de textos ou ao servidor de imagens algum objeto sem se preocupar com a localização física destes objetos. Cabe aos servidores de mídia localizar os objetos requisitados e transferi-los.

Define-se o uso de servidores de bancos de dados multimídia ou servidores de arquivos para o armazenamento das mídias. A transferência de mídias configura-se como um problema produtor-consumidor, onde a fonte de dados é considerada o elemento produtor de stream. Cada objeto de mídia chega ao consumidor na forma de um conjunto de pacotes conhecidos como frames. Estes são escritos em um buffer. A aplicação lê os elementos do buffer, decodifica-os e apresenta-os ao usuário.

3.4.1. O Protótipo Inglês On-Line

A fim de validar a arquitetura proposta em [3], foi implementado um protótipo de um sistema de EAD, voltado para o ensino da língua inglesa. O protótipo foi denominado Inglês On-Line.

Na tabela 3.4 são mostrados os objetos que foram implementados no protótipo Inglês On-Line fazendo a devida correspondência aos objetos que são definidos na arquitetura previamente estabelecida.

Tabela 3.4. *Objetos utilizados no protótipo Inglês On-Line*

Grupos de Objetos	**Objetos Implementados**
Objetos de Aplicação para EAD	• Dicionário. • Biblioteca • Avaliação
Objetos de Suporte de Mídias	• Servidor de Texto • Servidor de Imagem
Objetos de Suporte Geral	• Autenticador • Gerenciador de informações cadastrais • Servidor de nomes CORBA

Alguns objetos definidos na arquitetura apresentada na seção 3.4, tais como o QoS

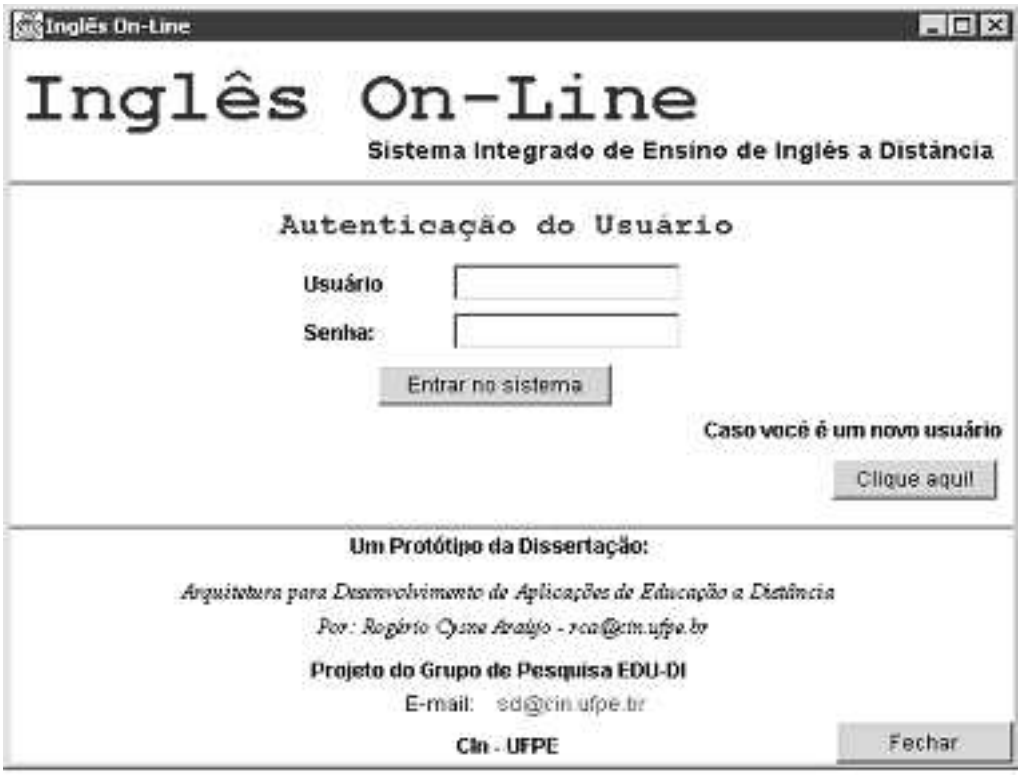

Figura 3.4. *Interface do* Inglês On-Line

Manager, o Meta Scheduler e o Internet Trader, não foram integrados ao Inglês On-Line. Isto ocorreu pelo fato destes elementos ainda estarem em fase de desenvolvimento. Porém como ambiente integra elementos dos três grupos de objetos da arquitetura proposta, pode-se afirmar que este protótipo é suficiente para avaliar a aplicabilidade da arquitetura.

Os objetos foram desenvolvidos utilizando CORBA como plataforma de suporte à distribuição. Vale ressaltar que o modelo proposto poderia ter sido implementado em qualquer outra plataforma de distribuição, tal como Java/RMI ou Microsoft COM/DCOM. A escolha por CORBA é justificada pelo fato de ser um padrão aberto, com transparência e independência de linguagem de programação e de sistema operacional.

3.5. Arquitetura Ataxia

O aprimoramento dos recursos empregados na educação a distância tem permitido a instrução de indivíduos em qualquer parte do mundo. A introdução de novas alternativas de ensino, particularmente a partir do surgimento da Internet, está possibilitando a capacitação de pessoas em praticamente qualquer ponto do planeta. No entanto, a grande maioria dos cursos realizados a distância tem esbarrado em um sério problema de motivação de alunos e professores. Nesse contexto específico, a Realidade Virtual (RV) surge como um complemento capaz de estimular o aprendizado como um todo.

São inúmeras as novas possibilidades de exposição de conteúdo através de ambientes virtuais, onde podem ser destacadas: a recriação de locais remotos, em épocas antigas; a simulação do comportamento de equipamentos; a representação de fenômenos específicos; e a interação com personagens virtuais controlados por computadores. Se for considerada, ainda, a introdução de recursos de comunicação através de redes aos ambientes virtuais, as possibilidades de exposição de conteúdo tornam-se ilimitadas. Grupos de participantes, formados por alunos, professores e mesmo máquinas, podem interagir tanto com o ambiente virtual compartilhado quanto entre si, explorando situações, trocando experiências e criando idéias.

A justificativa final para o uso de RV na educação a distância advém da possibilidade

de apresentar a complexidade de situações do mundo real de tal maneira que pessoas possam observar diretamente os resultados de suas próprias decisões. [1]

O uso da realidade virtual, no entanto, esbarra em sérios problemas, dentre os quais podem ser destacados: o alto custo na aquisição de hardware especial, a complexidade na definição e implementação de estruturas computacionais de suporte à geração de estímulos a indivíduos, o controle de respostas a eventos oriundos de interações e a caracterização realista de ambientes. Some-se a isso as necessidades específicas provenientes da utilização da Internet como meio de suporte ao compartilhamento de espaços virtuais a múltiplos usuários e os parcos recursos oriundos dos geralmente baixos orçamentos das instituições de ensino atuais, principalmente as públicas. Assim se caracteriza um sério problema que é o de como viabilizar a utilização de recursos de realidade virtual através da Internet para a educação a distância.

Inicialmente, foi proposta a ESCREV - **ESC**ola **RE**almente **V**irtual, que consiste de um modelo básico que descreve todas as necessidades de uma escola virtual voltada para educação a distância através da Internet. Da análise das necessidades da ESCREV surge a arquitetura ATAXIA.

A arquitetura ATAXIA define componentes especializados capazes de distribuir, através da Internet, todo o processamento do ambiente virtual; obtendo-se, assim, um grande poder computacional a um baixo custo. Além disso, esta arquitetura emprega técnicas de otimização do acesso a informações remotas, diminuindo o tráfego da rede e, conseqüentemente, viabilizando o uso de NVE's voltados para a educação a distância.

3.5.1. Características da Arquitetura

Para que seja possível o processamento a baixo custo de todas as funcionalidades NVE (Networked Virtual Environment - Ambiente Virtual em Rede), faz-se necessário o uso de componentes especializados responsáveis pela realização de determinadas tarefas. A distribuição de processamento em sistemas computacionais de baixo custo, apresenta uma relação custo/benefício bem superior àquelas que empregam caros sistemas centralizados

de grande porte. [18]

A interligação dos diversos componentes da arquitetura dá-se tanto através de conexões do tipo cliente-servidor como também do tipo ponto-a-ponto; dessa forma a arquitetura ATAXIA pode ser considerada uma arquitetura híbrida. Para um melhor aproveitamento da largura de banda disponível, o modelo de protocolo empregado nesta arquitetura utiliza PDU's (Protocol Data unit - Unidade de Dados do Protocolo) de tamanho bastante pequeno.

A ATAXIA é uma arquitetura aberta, que permite a incorporação de novos serviços e aplicações previamente disponíveis. Isto pode ser feito através de diferentes artifícios: simulóides podem realizar a interligação de aplicativos externos ao NVE; alguns componentes da arquitetura podem ser re-configurados para permitir o fluxo de informações entre eles e os novos aplicativos, nos dois sentidos; e o protocolo empregado na arquitetura prevê a possibilidade do uso de aplicativos não pertencentes ao NVE.

Como a arquitetura ATAXIA é um padrão aberto e faz uso de aplicações especializadas que podem ser interligadas através da Internet, esta arquitetura permite a implementação de seus componentes em diversas plataformas. Assim, diferentes soluções de software e de hardware podem ser integradas, possibilitando um melhor aproveitamento dos recursos disponíveis.

Uma outra característica da arquitetura é o baixo uso de *streams* de áudio ou de vídeo. Streams são canais de distribuição de mídias como áudio e vídeo através de redes de computadores que usam estruturas de controle especializadas, e exigem que tanto o emissor quanto o receptor da mídia realizem uma certa quantidade de processamento para suportá-los. As estruturas de controle geralmente consomem bastante processamento e largura de banda. [1] Em substituição ao *streaming*, são empregadas técnicas de armazenamento de dados em caches locais e replicação sob demanda. [19]

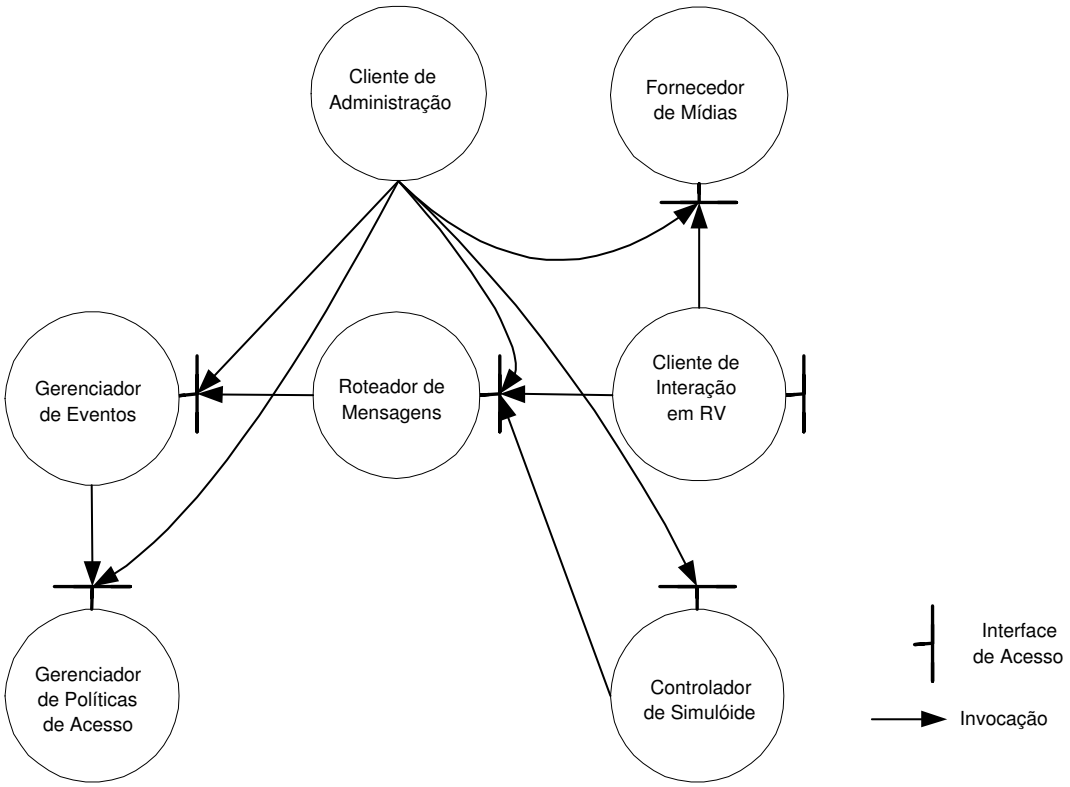

Figura 3.5. *Componentes da arquitetura ATAXIA*

3.5.2. Componentes da Arquitetura

A arquitetura ATAXIA define um conjunto de componentes que desempenham funções especializadas e trabalham de forma integrada para tornar disponível o ambiente virtual, através da troca de PDU's específicas. Estes componentes foram divididos em clientes e servidores. Estes componentes são apresentados na figura 3.5

Entre os servidores estão o Roteador de Mensagens, o Gerenciador de Eventos, o gerenciador de Políticas de Acesso e o Fornecedor de Mídias. O Cliente de Interação em Realidade Virtual (CIRV), o Controlador de Simulóides e o Cliente de Administração são os clientes da arquitetura.

O Roteador de Mensagens é o responsável por disponibilizar conexões de um determinado NVE aos seus diversos participantes. O critério de roteamento é baseado em uma lista persistente de serviços disponíveis, presente no roteador de mensagens e na análise

das PDU's que trafegam na rede, que podem indicar sua origem e seu destino. O roteador de mensagens informa aos participantes qual fornecedor de mídias (descrito posteriormente) deve ser utilizado para a obtenção de arquivos específicos. Além disso, o roteador de mensagens é um dos componentes responsáveis por sincronizar as diversas cópias do ambiente virtual.

O Gerenciador de Eventos é o componente responsável por iniciar o funcionamento do ambiente virtual e por tratar todos os eventos oriundos das interações de seus participantes. Para isso, o gerenciador de eventos armazena os estados e posições de cada um dos objetos disponíveis no ambiente virtual.

O Gerenciador de Políticas de Acesso é o componente especializado nas restrições de ações dos participantes do ambiente virtual.

O Fornecedor de Mídias torna disponível aos participantes do NVE o conjunto de arquivos utilizados em interações no interior do ambiente. O Fornecedor de Mídia funciona basicamente como um sistema de armazenamento de arquivos de mídias específicas (imagens, sons, vídeos etc) que podem ser enviados diretamente às aplicações empregadas pelos participantes do NVE. Todos os arquivos de mídias são organizados em categorias específicas e indexados de forma a agilizar o processo de busca dos mesmos.

Todos os recursos do ambiente virtual são tornados disponíveis aos usuários humanos através do Cliente de Interação em Realidade Virtual (CIRV), que se conecta diretamente ao roteador de mensagens. Essa aplicação cliente faz uso de uma GUI (Graphical User interface - Interface Gráfica do Usuário) que realiza todo o processo da geração e apresentação do mundo virtual.

O CIRV mantém uma cache de mídias local para o armazenamento persistente de todas as mídias utilizadas, inclusive a própria arquitetura tridimensional do ambiente virtual. Essas mídias são replicadas sob demanda através da rede através de uma conexão direta com o Fornecedor de Mídias.

Personagens virtuais controlados pelo computador também podem estar disponíveis no ambiente virtual. Para que isto seja possível, é necessário, para cada um deles, o emprego

de componente Controlador de Simulóide, que é diretamente conectado ao Roteador de Mensagens.

O Cliente de Administração é o componente da arquitetura que agrega recursos básicos para realização das tarefas de manutenção e configuração do ambiente, tais como: configuração da lista de serviços disponíveis no Gerenciador de Roteamento, cadastramento de usuários do NVE junto ao Gerenciador de Eventos, inclusão de elementos de mídia disponíveis no Fornecedor de Mídias, manutenção da política de direitos de acesso, análise de ocorrências específicas e auditoria do sistema.

3.6. Conclusões

Verificou-se que as arquiteturas apresentadas procuram, através de suas definições em componentes modulares, solucionar o problema da centralização do processamento. E também buscam possibilitar a utilização de aplicações que normalmente exigem grande largura de banda, como aplicações de realidade virtual e de educação a distância, em redes que não apresentam esta propriedade.

Além disso, estas arquiteturas apresentam em comum a definição de um componente especializado no fornecimento de arquivos de mídias para os outros módulos da arquitetura, principalmente os módulos clientes.

Capítulo 4

Servidor de Replicação

Este capítulo engloba a principal contribuição deste trabalho. Aqui é apresentado um novo serviço de replicação de arquivos de mídias definido e implementado a partir da análise das estruturas descritas no capítulo 3.

De acordo com o que foi apresentado no capítulo 3, pode-se ver que todas as arquiteturas apresentadas fazem uso de um componente responsável pela disponibilização de arquivos de mídias aos outros componentes de suas respectivas arquiteturas. Em cada uma é dado um nome diferente para este componente: Servidor de Replicação, Fornecedor de Mídias, Objetos de Suporte a Mídias.

Na ARVORE (Seção 3.3), o Servidor de Replicação (SRe) é um dos servidores especificados naquela estrutura. É através do SRe que os clientes têm acesso aos elementos de multimídia e de realidade virtual que forem necessários para utilização do sistema.

O SRe é único no sistema e não é acessado diretamente pelos clientes. Estes devem fazer ao Servidor de Roteamento (SRo) qualquer solicitação que deva ser dirigida ao SRe.

Todos os elementos de multimídia e de RV enviados aos clientes são armazenados localmente, ficando disponíveis para usos futuros e evitando seus reenvios quando forem novamente necessários. O SRe é utilizado sempre que um usuário tenta fazer uso de um elemento de multimídia e não o encontra armazenado na sua base local de armazenamento. Neste caso, o cliente utiliza um elemento genérico, denominado coringa, que substituirá, naquele momento, o elemento solicitado. Ao mesmo tempo o cliente requisita

ao SRe o elemento não encontrado localmente. A partir do momento em que o elemento requisitado encontra-se armazenado localmente, posteriores acessos àquele elemento não mais acarretarão na utilização do coringa.

Embora proposto na definição da ARVORE, o uso de caches locais e de replicação de arquivos não foi implementado no protótipo utilizado para validação daquela arquitetura.

Em [3], a arquitetura especificada faz uso do CORBA como plataforma para viabilização da distribuição de seus vários componentes. Os componentes são, na verdade, conjuntos de objetos que juntos fornecem os serviços providos pelos servidores da arquitetura. Entre eles estão os objetos de suporte a mídias.

Quando o cliente necessita de algum arquivo de mídia, este faz a solicitação do arquivo diretamente aos objetos de suporte a mídias, através do ORB CORBA. O arquivo, então, não é armazenado em uma base de armazenamento local ao cliente. Ou seja, um novo acesso ao mesmo arquivo de mídia acarretará numa nova solicitação aos objetos de suporte a mídias.

O uso do CORBA como arquitetura de suporte à distribuição de objetos e de seu Serviço de Nomes tornou implícita a indexação das mídias, como sugerido em [1], facilitando a localização dos arquivos.

Em [1], a especificação da arquitetura ATAXIA sugere o uso de caches locais para armazenamento das mídias, mas deixa em aberto a possibilidade do uso de uma política de replicação sob demanda, apontando possibilidades para a transmissão de mídias contínuas, como som e vídeo. O Fornecedor de Mídias funciona basicamente como um sistema de armazenamento de arquivos de mídias específicas (imagens, sons, vídeos etc) que podem ser enviados diretamente às aplicações empregadas pelos participantes do NVE. Todos os arquivos de mídias são organizados em categorias específicas e indexados de forma a agilizar o processo de busca dos mesmos.

4.1. Caches Locais

O desempenho das aplicações da Web cai drasticamente à medida que mais pessoas começam a acessar a internet, uma vez que a infraestrutura da web não projetada para suportar um grande número de usuários. [20] O uso de caches é uma solução óbvia para este tipo de problema, pois não só reduz a quantidade de dados que trafegam pela rede e o tempo de espera pela carga de determinado documento mas também resulta em um sistema mais tolerante a falhas e com um melhor balanceamento da carga do servidor.

A dificuldade maior aparece quando as requisições de todos os usuários têm que ser atendidas por um único servidor. E pior ainda: se este único servidor falha, o acesso à informação é perdido.

A idéia por trás do uso de caches é a de trazer os dados o mais próximo possível do usuário de forma a minimizar a quantidade de acessos. Da primeira vez que determinados dados são acessados, eles são armazenados na cache e a partir do segundo acesso aos mesmos dados há um ganho de desempenho, uma vez que o acesso à cache é mais rápido e fácil. Devido às limitações de espaço, um dos problemas de um sistema de gerenciamento cache é decidir quais dados manter na cache. Além desse, há o problema de coerência da cache que ocorre quando os dados são modificados na origem sem que o sistema gerenciador da cache tome conhecimento disso.

Para solucionar o primeiro problema, são propostos diversos algoritmos para substituição dos dados da cache quando esta não possui espaço disponível para armazenar novos arquivos. Vários destes algoritmos levam em conta o há quanto tempo o arquivo está presente na cache ou há quanto tempo o arquivo não acessado ou modificado para determinar qual arquivo deve dar lugar ao novos arquivos que precisam ser armazenados na cache.

Quanto ao problema da coerência, para solucioná-lo, deve-se verificar se a cópia do arquivo presente na cache é a mesma que existe no servidor. Com a finalidade de se reduzir o número de acessos ao servidor para verificar a versão dos arquivos, pode ser definido com que frequência esse verificação é feita, por exemplo, sempre que o arquivo é

acessado; uma vez a cada utilização da aplicação pelo usuário ou uma vez por dia.

O uso de caches, porém, só é realmente útil a partir da segunda vez que os dados são acessados. Na primeira vez que determinada informação é necessária ela deve ser obtida do servidor através da rede. Em uma abordagem eficiente do uso de caches, parte dos dados está disponível na cache local no momento que o primeiro usuário faça a primeira requisição de dados ao servidor.[20] A determinação de quais dados estariam disponíveis inicialmente na cache, pode ser feita com base em estimativas ou em utilizações anteriores do sistema.

Existem alguns problemas que não são resolvidos apenas com o uso de caches. Se os dados disponíveis não são acessados com muita frequência, ou seja, se na maioria das vezes é feito um único acesso aos dados disponíveis, este único acesso será feito diretamente ao servidor uma vez que nenhuma requisição anterior foi executada e, logo, a informação desejada não estará na cache.

O aumento de disponibilidade das informações também não é conseguido apenas com o uso de caches. Não é possível verificar se os dados da cache estão atualizados se o servidor não estiver acessível. E, caso a informação não esteja presente na cache, não será possível recuperá-las do servidor se o acesso a este não estiver disponível. Deste problema surge a necessidade do uso de replicação em conjunto com a utilização das caches.

4.2. Servidor de Replicação - SRE

Nesta seção é apresentado um novo serviço para disponibilização de arquivos de mídias (Servidor de Mídias - SRE), que pode ser utilizado em qualquer uma das arquiteturas propostas no capítulo 3 em substituição aos seus respectivos componentes que apresentam esta funcionalidade. O serviço aqui proposto apresenta a união de algumas das características apresentadas nas arquiteturas anteriormente descritas, buscando, dessa forma, se adequar a quaisquer aplicações que necessitem deste tipo de serviço, como, por exemplo, aplicações de educação a distância e de realidade virtual.

O serviço foi implementado na forma de objetos CORBA distribuídos. Através da de-

finição das interfaces dos objetos, estes podem ser utilizados pelos clientes sem que estes tomem conhecimento de como o serviço está implementado. Além disso, após publicar sua interface, um objeto CORBA pode ser facilmente localizado pelo cliente que esteja querendo utilizá-lo. Isto torna possível que clientes de diferentes aplicações possam fazer uso dos métodos disponibilizados pelos componentes distribuídos no ORB. Outra vantagem do uso de objetos CORBA distribuídos é a possibilidade de o serviço estar executando em qualquer plataforma. Além da linguagem de definição das interfaces ser independente de plataforma, a implementação das interfaces foi feita em linguagem Java, que também é independente de plataforma.

A implementação do SRE deve suportar o uso de caches locais para armazenamento dos arquivos de mídias pelos clientes. Sempre que um cliente necessitar de determinado arquivo, primeiro tentará localizar este arquivo em sua base local de arquivos. Caso o arquivo exista localmente, o cliente utilizará este arquivo e, ao mesmo tempo, solicitará ao SRE informações sobre a versão do mesmo arquivo que está armazenado no servidor. Se a versão da mídia existente no servidor for diferente da versão local do cliente, o cliente solicitará ao servidor a nova versão do arquivo, que será utilizada em futuros acessos àquele elemento, não sendo mais necessário um novo acesso ao servidor para obter o arquivo.

A existência de caches locais torna o sistema mais tolerante a falhas. Caso o SRE esteja indisponível para o cliente, este poderá obter os arquivo de mídia de que necessite da sua cache local. A fim de implementar uma maior disponibilidade dos arquivos de mídias, mais de uma cópia do SRE pode ser colocada em execução, ao mesmo tempo, em máquinas diferentes da rede. Caso uma dessas cópias esteja inacessível pelo cliente, outras versões do SRE estarão disponíveis. O elemento ORB da arquitetura CORBA se responsabiliza pelo balanceamento da carga entre os vários servidor existentes.

A cada arquivo solicitado ao servidor, um novo objeto será instanciado no ORB e uma referência para o mesmo será devolvida para o cliente. A partir de então, o objeto que corresponde ao arquivo de mídia é quem responderá às requisições do cliente. O cliente poderá solicitar o arquivo propriamente dito ou apenas informações sobre o arquivo para

que possa comparar com a versão que existe na cache local.

A árvore de diretórios onde estão localizados os arquivos no servidor será mapeada em contextos do servidor de nomes, ou seja, quando o cliente solicitar ao servidor o arquivo que está em /midias/imagens/jpg/rosto.jpg, será instanciado um objeto, que será identificado pelo nome rosto.jpg e que, no serviço de nomes, estará no contexto de nomes midias/imagens/jpg.

De forma a evitar que o cliente precise saber a exata localização do arquivo (o diretório onde este está), objeto deve apresentar meios de ser localizado pelo SRE caso o cliente forneça apenas o nome do arquivo, por exemplo. Para tal será utilizado um serviço de localização (Trader) disponível na arquitetura CORBA.

Para cada arquivo disponibilizado pelo servidor, será divulgada uma oferta de serviço (Seção 2.4.1) com informações sobre o arquivo para que o mesmo possa ser localizado pelo cliente sem que este precise saber em qual diretório o arquivo se encontra. Embora diminua a preocupação do cliente em saber a localização exata do arquivo, este procedimento pode provocar algum erro caso existam mais de um arquivo com mesmo nome, mas localizados em locais diferentes.

Quando um cliente obtém um arquivo do servidor e o armazena em sua cache local, ele também instancia um novo objeto do tipo Media associado ao arquivo armazenado em sua cache e publica uma referência desse objeto no ORB. Dessa forma, esta cópia do arquivo estará disponível para outros clientes, além da cópia disponível no servidor. Assim consegue-se a replicação dos arquivos na rede, aumentando sua disponibilidade e sua tolerância a falhas, uma vez que, se o servidor não estiver disponível, os clientes podem obter os arquivos de que necessitem a partir das caches locais dos outros clientes.

A fim de atender ao que foi apresentado, definiu-se a interface IDL dos objetos a serem implementados como é mostrado na figura 4.1

Foram definidas duas interfaces: *Server* e *Media*. A interface *Server* representa o SRE. Ela apresenta os métodos que serão utilizados pelo cliente para solicitar um arquivo de mídia de que esteja precisando. Ao receber esta solicitação, o objeto *Server* instanciará um

```
module sre
{
    typedef sequence <octet> SetOfBytes;

    interface Media
    {
        SetOfBytes getFile();
        string getFileInfo();
    };

    interface Server
    {
        void loadMediaByFullName( in string mediaName );

        void loadMediaBySimpleName( in string mediaName );
    };

};
```

Figura 4.1. *IDL do Servidor de Replicação*

objeto *Media* associado ao arquivo solicitado pelo cliente, exportará este objeto mídia para o ORB e devolverá para o cliente uma referência para o mesmo. O cliente pode solicitar um arquivo ao servidor fornecendo o caminho completo de onde o arquivo se localiza na base do servidor, ou fornecendo apenas o nome do arquivo. Para isto são definidos os métodos *loadMediaByFullName* e *loadMediaBySimpleName*, respectivamente.

A interface *Media* possui os métodos necessários para que o cliente obtenha informações sobre os arquivos de mídia, além do arquivo propriamente dito. É através de métodos do objeto Media que o cliente pode comparar a versão do arquivo existente em sua base local com a versão do arquivo disponibilizada pelo servidor.

4.3. Modelo de Implementação

Para fins de validação da proposta, foram executadas algumas modificações no aplicativo **Papo3D** para que este utilizasse o SRE como seu componente fornecedor de mídias. Papo3D é uma aplicação distribuída de bate-papo (*chat*) na qual os usuários interagem uns com os outros por meio de um ambiente virtual. Ele foi desenvolvido em conjunto pela

empresa SoftBuilder Informática(http://www.softbuilder.com.br) e pelo Departamento de Computação da UFC no âmbito do projeto AVAL [17]. Esta aplicação foi desenvolvida tendo como base a arquitetura ATAXIA, apresentada na seção 3.5. Foi implementada utilizando-se a linguagem Delphi e o fornecedor de mídias foi implementado como um simples servidor de arquivos. O cliente do aplicativo não utilizava caches locais para armazenamento dos arquivos obtidos do fornecedor de mídias, por isso foram necessárias algumas modificações no código do cliente implementado anteriormente.

A linguagem de programação utilizada foi Java (com compilador JDK - Java Development Kit - 1.4.0). Esta escolha deve-se ao fato de que é uma linguagem projetada para uso em ambiente Internet, podendo ser utilizada para criar aplicações que podem ser executadas em um único computador ou ainda em um ambiente distribuído. A máquina virtual Java (JVM - Java Virtual Machine) possui implementações para diferentes plataformas, podendo estar embutidas em outros programas (browsers) na forma de applets. Além da característica multiplataforma, Java é também uma linguagem de programação orientada a objetos bastante completa, o que a torna uma arma poderosa para o desenvolvimento de aplicações voltadas para a Internet. Diversas bibliotecas, mais conhecidas como APIs (Application Program Interface), são continuamente desenvolvidas e disponibilizadas para atender às necessidades de desenvolvimento de aplicações. [3]

A figura 4.2 apresenta a interface gráfica do servidor de replicação que foi implementado. Na interface são exibidos os objetos que já foram carregados no ORB mediante solicitações dos clientes.

A plataforma de distribuição utilizada no desenvolvimento deste protótipo é CORBA. Foi utilizado o ORB implementado pela Sun, presente no JDK, versão 1.4.0, que implementa a versão 3.0 da especificação de CORBA. Também utilizamos o serviço de nomes presente no JDK, versão 1.4.0.

Além disso, para desenvolvimento do protótipo, foram utilizadas as seguintes ferramentas de software:

JCreator - Para a implementação das classes Java;

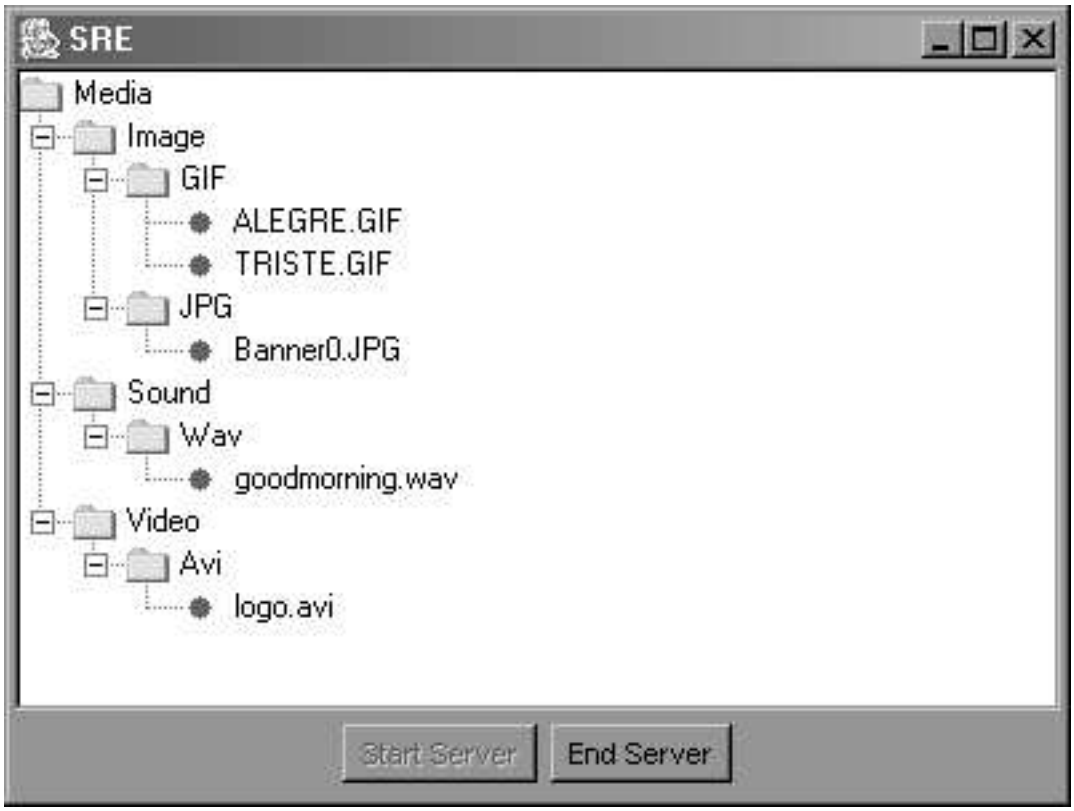

Figura 4.2. *Interface Gráfica do Servidor de Replicação*

Rational Rose ®2001 - Para a modelagem dos diagramas de seqüência e de atividade baseados no padrão UML (Unified Modelling Language).

4.4. Análise de Desempenho

Para análise do desempenho do servidor implementado considerou-se a seguinte situação: O cliente irá selecionar, aleatoriamente, 100 arquivos dentre os arquivos de mídias disponibilizados pelo servidor, podendo haver repetição de um mesmo arquivo, e solicitá-los um a um ao servidor. Será contado o tempo total decorrido entre a instante em que é feita a solicitação do primeiro arquivo ao servidor por parte do cliente até o momento em que o cliente obtém do servidor o arquivo solicitado na centésima requisição.

No primeiro cenário foram utilizados um cliente sem caches locais e o servidor implementado conforme definido na especificação da arquitetura ARVORE (Seção 3.3). Ou seja, o cliente se comunica com um Servidor de Roteamento, que direciona a solicitação do cliente para o Servidor de Replicação, que devolve o arquivo solicitado ao cliente. A comunicação entre o cliente e os servidores de Roteamento e de Replicação é feita através de Sockets, utilizando-se o protocolo TCP.

No segundo cenário utilizou-se um cliente com uma cache local e o servidor SRE implementado. A comunicação entre cliente e servidor é feita através do ORB CORBA sem a utilização de um servidor de roteamento.

As implementações do primeiro e do segundo cenário foram executadas várias vezes e o tempo gasto na execução do segundo cenário foi, em média, 60% menor que o tempo decorrido na execução do primeiro cenário, conforme o gráfico apresentado na figura 4.3.

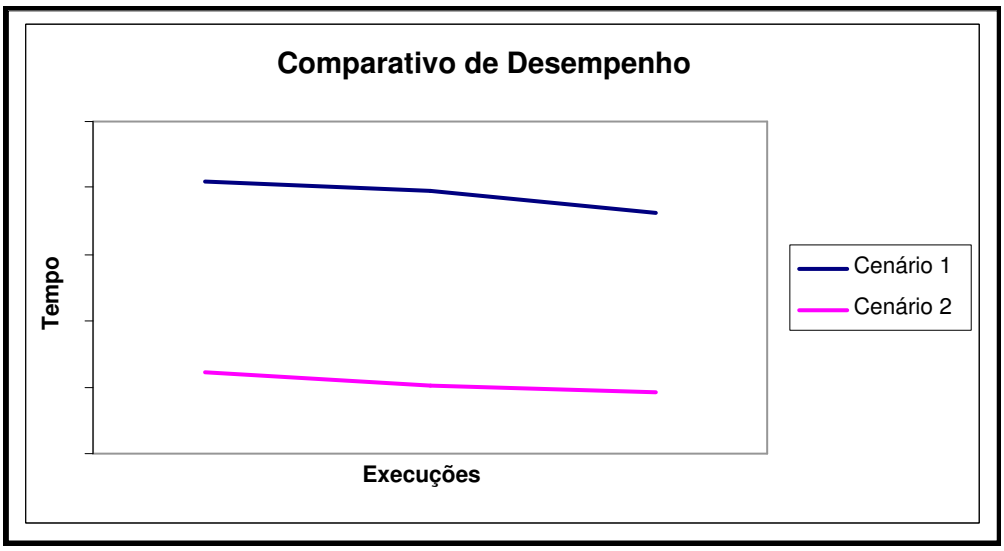

Figura 4.3. *Análise do desempenho do servidor de Replicação*

Capítulo 5

Conclusões e Trabalhos Futuros

Este capítulo finaliza esta dissertação, apresentando considerações a respeito do servidor de mídia proposto no capítulo 4. São analisados também os objetivos alcançados e enumeradas sugestões de trabalhos futuros.

5.1. Conclusões

A utilização de objetos distribuídos na construção de sistemas tem demonstrando importantes benefícios, antes não encontrados no modelo de disponibilização centralizado de informações, geralmente encontrado na maioria dos ambientes de EAD e de RV. Dentre estes benefícios, destaca-se a capacidade de processamento distribuído, a transparência de localização de recursos e a escalabilidade.

A utilização do serviço *Trader* da arquitetura CORBA possibilitou uma indexação mais simples dos arquivos de mídia. Um objeto e, consequentemente, o arquivo de mídia correspondente, pode ser obtido não mais apenas através do seu nome (identificador) mas também através de propriedades do objeto, o que corresponde a característica do arquivo. Nas implementações apresentadas em [1], [3] e [2], o cliente precisava saber o nome completo do arquivo, inclusive o caminho onde o mesmo se localizava no servidor de arquivos para poder solicitá-lo ao servidor. Com o uso do serviço *Trader*, os arquivos podem ser requisitados simplesmente através de seus nomes.

O uso de caches, ou seja, o armazenamento local dos arquivos utilizados pelos clientes das aplicações reduziu o número de acessos feitos ao servidor, uma vez que um arquivo passa a ser recuperado da cache depois que uma primeira requisição deste arquivo foi feita ao servidor.

Uma outra vantagem inserida na aplicação com a implantação de caches de armazenamento local é a possibilidade de transmissão de mídias contínuas (*streams*). Sem que a mídia esteja armazenada localmente, é necessário que existam no cliente e no servidor estruturas de controle especializadas na apresentação deste tipo de mídia. Estas estruturas consumiriam processamento e largura de banda. Com a mídia armazenada totalmente no lado cliente, onde será apresentada, o uso destas estruturas não se faz mais necessário.

5.2. Trabalhos Futuros

Como sugestão de continuação do presente trabalho vale ressaltar a implementação dos outros componentes das arquiteturas que foram apresentadas com o uso de objetos distribuídos. Por exemplo, se todos os componentes da arquitetura ATAXIA estivesse disponíveis através de um ORB, o Roteador de Mensagens talvez possa simplesmente ser substituído pelo ORB. O mesmo se aplicaria para o Servidor de Roteamento da arquitetura ARVORE, caso os elementos desta arquitetura também fossem implementados como objetos CORBA.

As caches dos clientes foram implementadas sem nenhuma política de gerenciamento. O algoritmo implementado foi o mais simples: caso a cache esteja cheia e um novo arquivo precisa ser armazenado na cache, o arquivo de maior tamanho que está na cache é removido. Isso se repete até que haja espaço disponível para armazenar o novo arquivo na cache. Seria interessante a implementação de um algoritmo melhor.

Como foi descrito na seção 2.1, um dos componentes da arquitetura CORBA são as *facilidades comuns verticais*, que são coleções de objetos que proporcionam serviços a serem utilizados diretamente por outras aplicações de segmentos específicos como as áreas de finanças, saúde, transportes etc. Cabe aqui investigar a possibilidade de uma

nova definição do servidor de mídias para que o mesmo possa ser submetido à avaliação da OMG de forma a ser considerado uma facilidade comum da arquitetura CORBA a ser utilizados por aplicações de realidade virtual, educação a distância e outras quaisquer que necessitem de tal serviço.

Estudar a tecnologia de *Web Services* e a futura disponibilização do serviço de replicação de mídias na forma de um *Web Service*.

Um *Web Service* é uma interface que descreve uma coleção de operações que são acessíveis através da rede com o uso mensagens que utilizam XML (e**X**tensible **M**arkup **L**anguage) como linguagem padrão. *Web services* atendem a uma determinada tarefa ou um conjunto de tarefas.

Assim como os componentes distribuídos, um *Web Service* apresenta uma funcionalidade encapsulada que pode ser utilizada sem necessidade do conhecimento de como esta funcionalidade está implementada. Os clientes que acessam um determinado Web Service podem estar implementados em qualquer plataforma e em qualquer linguagem de programação desde que sejam capazes de produzir e de consumir as mensagens definidas pela interface do *Web Service*.

CORBA, COM e RMI são grandes tecnologias para integrar objetos distribuídos numa rede corporativa. No entanto, o problema de integração de aplicações via internet é melhor solucionado através do uso de Web Services. Isto porque a tecnologia de Web Services atua sobre protocolos abertos da Internet, como HTTP e SMTP.[21]

Referências Bibliográficas

[1] Leite Jr., A. J. M. *ATAXIA: Uma Arquitetura para a Viabilização de NVE's Voltados para a Educação à Distância Através da Internet*. Dissertação (Mestrado) — Universidade Federal do Ceará, Fortaleza/CE, 2000.

[2] VIDAL, C. A. et al. *ÁRVORE - Uma Arquitetura de Baixo Custo para Distribuição de Elementos de Realidade Virtual e Multimídia em Ambientes Multiusuários*. [S.l.], 2000.

[3] CYSNE, R. A. *Arquitetura Distribuída para Desenvolvimento de Aplicações de Educação a Distância*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife/PE, 2000.

[4] SEETHARAMAN, K. The CORBA Connection. *Communications of the ACM*, v. 41, n. 10, p. 34–36, 1998.

[5] OMG. *CORBA - The Common Object Request Broker*. [S.l.]. Disponível em: <http://www.omg.org>. Acesso em: 2000.

[6] SIEGEL, G. OMG Overview: CORBA and the OMA in Enterprise Computing. *Communications of the ACM*, v. 41, n. 10, p. 37–43, 1998.

[7] OMG. *NamingService Specification*. [S.l.]. Disponível em: <http://www.omg.org>. Acesso em: Fevereiro/2001.

[8] ORFALI, R.; HARKEY, D. *Client/Server Programming with Java and CORBA*. Second. [S.l.]: John Willey & Sons, 1999.

[9] DURLACH, N. I.; MAVOR, A. S. *Virtual Reality: Scientific and Technological Challenges*. [S.l.]: Comitee on Virtual Reality Research and Development, National Research Council, National Academy of Sciences Press, 1995.

[10] ÇAPIN, T. K. et al. *Avatars in Networked Virtual Environments*. [S.l.]: John Wiley and Sons, 1999.

[11] DOENGES, P. K. et al. MPEG-4: Audio/Video and Synthetic Graphics/Audio for Mixed Media. *Image Communication Journal*, v. 5, n. 4, p. 433–463, 1997.

[12] ZYDA, M.; SHEEHAN, J. *Modeling and Simulation: Linking Entertainment and Defense*. [S.l.]: National Academy Press, 1997.

[13] STYTZ, M. R. Distributed Virtual Environments. *IEEE Computer Graphics and Applications*, v. 16, n. 33, p. 19–31, 1996.

[14] BOULIC, R.; REZZONICO, S.; THALMANN, D. Multi-Finger Manipulation of Virtual Objects. In: *Procedures of ACM VRST'96*. [S.l.: s.n.], 1996.

[15] SANTOS, N. Estado da Arte em Espaços Vituais de Ensino e Aprendizagem. *Revista Brasileira de Informática na Educação*, n. 04, p. 75–94, Abril 1999.

[16] Grupo EDU-Cin. *Análise comparativa de ferramentas de educação a distância*. Disponível em: <http://www.di.ufpe.br/ sd/ead/tabela.html>. Acesso em: 2002.

[17] VIDAL, C. A. *Projeto AVAL: Ambientes Virtuais para o Aprendizado de Línguas*. 1999. Projeto aprovado junto ao PROTEM-CC/CNPq. Disponível em: <http://www.lcg.dc.ufc.br/aval/>.

[18] TANEMBAUM, A. S. *Distributed Operating Systems*. [S.l.]: Prentice Hall, 1995.

[19] SCHWEBER, E. von. Escape from VRML Island. In: *SIGGRAPH 98*. [s.n.], 1998. Disponível em: <http://www.infomaniacs.com/SQL3D/-SQL3D-Escape-From-VRML-Island.htm>.

[20] BAENTSCH, M. et al. Enhancing the Web's Infraestructure – From Caching to Replication. *IEEE Internet Computing*, v. 1, n. 2, p. 18–27, 1997. Disponível em: <http://citeseer.nj.nec.com/baentsch97enhancing.html>.

[21] GRAHAM, S. et al. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*. [S.l.]: Sams, 2002.

[22] BOUCHER, K. D. CORBA: From Vision to Reality. *Standard View*, v. 6, n. 1, p. 14–16, March 1998.

[23] HELD, J. J.; SUSCH, C. A. T.; GOLSHAN, A. What Does the Future Hold for Distributed Object Computing. *Standard View*, v. 6, n. 1, p. 17–21, March 1998.

[24] VINOSKI, S. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, v. 14, n. 2, p. 46–55, 1997.

[25] GAMMA, E. et al. *Design Patterns*. [S.l.]: Addison Wesley, 1994.

[26] ORFALLI, R.; HARKEY, D.; EDWARDS, J. *The Essential Distributed Objects Survival Guide*. [S.l.]: John Willey & Sons, 1996.

[27] SINGHAL, S.; ZYDA, M. *Networked Virtual Environments: Design and Implementation*. [S.l.]: Addison Wesley, 1999.

[28] TANEMBAUM, A. S. *Sistemas Operacionais Modernos*. [S.l.]: Prentice-Hall do Brasil, 1996.

[29] Microsoft Corporation. *The Component Object Model Specification*. Disponível em: <http://www.microsoft.com/oledev/>. Acesso em: Outubro/2002.

[30] Open Software Foundation. *The OSF Distributed Computing Environment*. Disponível em: <http://www.osf.org/dce/>. Acesso em: Outubro/2002.

[31] Microsoft Corporation. *OLE-DCOM Architecture*. Disponível em: <http://www.microsoft.com/technet/prodtechnol/winntas/plan/dcomarch.asp>. Acesso em: Outubro/2002.

[32] Microsoft Corporation. *DCOM Technical Overview*. Disponível em: <http://www.microsoft.com/technet/prodtechnol/winntas/plan/dcomtowp.asp>. Acesso em: Outubro/2002.

[33] Sun Microsystems. *Java IDL*. Disponível em: <http://java.sun.com/j2se/1.4/docs/guide/idl>. Acesso em: Abril/2002.

[34] SILVA, J. L. de Castro e; SILVA, A. N. da; SOUZA, J. N. de. *Designing and Developing Distributed Application within the CORBA and Java Environments: Practical Issues and Critical Analysis*. 1998.

[35] KIRNER, C. *Realidade Virtual: Dispositivos e Aplicações*. 1996. Disponível em: <http://www.dc.ufscar.br/ grv>. Acesso em: 2000.

[36] KIRNER, C. *Sistemas de Realidade Virtual*. 1997. Disponível em: <http://www.dc.ufscar.br/ grv>. Acesso em: 2000.

[37] RAJ, G. *A Detailed Comparison of CORBA, DCOM and Java/RMI*. Disponível em: <http://www.execpc.com/ gopalan/misc/compare.html>. Acesso em: Novembro/1999.

[38] Sun Microsystems. *Java Remote Method Invocation RMI*. 1997. Disponível em: <http://www.javasoft.com/products/jdk/1.1/docs/guide/rmi>. Acesso em: 2000.

[39] JGURU. *Fundamentals of RMI - Short Course*. Disponível em: <http://developer.java.sun.com/developer/onlineTraining/rmi/>. Acesso em: Dezembro/2002.

[40] Sun Microsystems. *The Java Web Services Tutorial*. Disponível em: <http://java.sun.com/webservices/tutorial.html>. Acesso em: Outubro/2002.

[41] WADDINGTON, D. G.; HUTCHISON, D. *Supporting Multimedia in Distributed Object Environments*. 1999. Computing Department, Lancaster University, Reino Unido. Disponível em: <ftp://ftp.comp.lancs.ac.uk/pub/mpg/MPG-99-17.pdf>.

[42] CORBA Trading Service OMG Team. *Trading Service*. [S.l.], 1995.

Apêndice **A**

Modelagem do Servidor de Replicação (SRE)

Os diagramas a seguir apresentados (Figuras A.1 e A.2) foram especificados em UML (Unified Modelling Language - Linguagem de Modelagem Unificada) e apresentam a maneira com a qual o cliente da aplicação pode obter um determinado arquivo arquivo de mídia através da utilização do servidor de mídias. Como apenas um caso de uso foi identificado e apenas duas classes (*Server* e *Media*)foram implementadas, não foi necessário a modelagem de diagramas de casos de uso nem de classes.

A.1. Diagrama de Sequência

O seguinte diagrama exibe como se dá a interação entre os objetos utilizados na implementação do servidor de mídias.

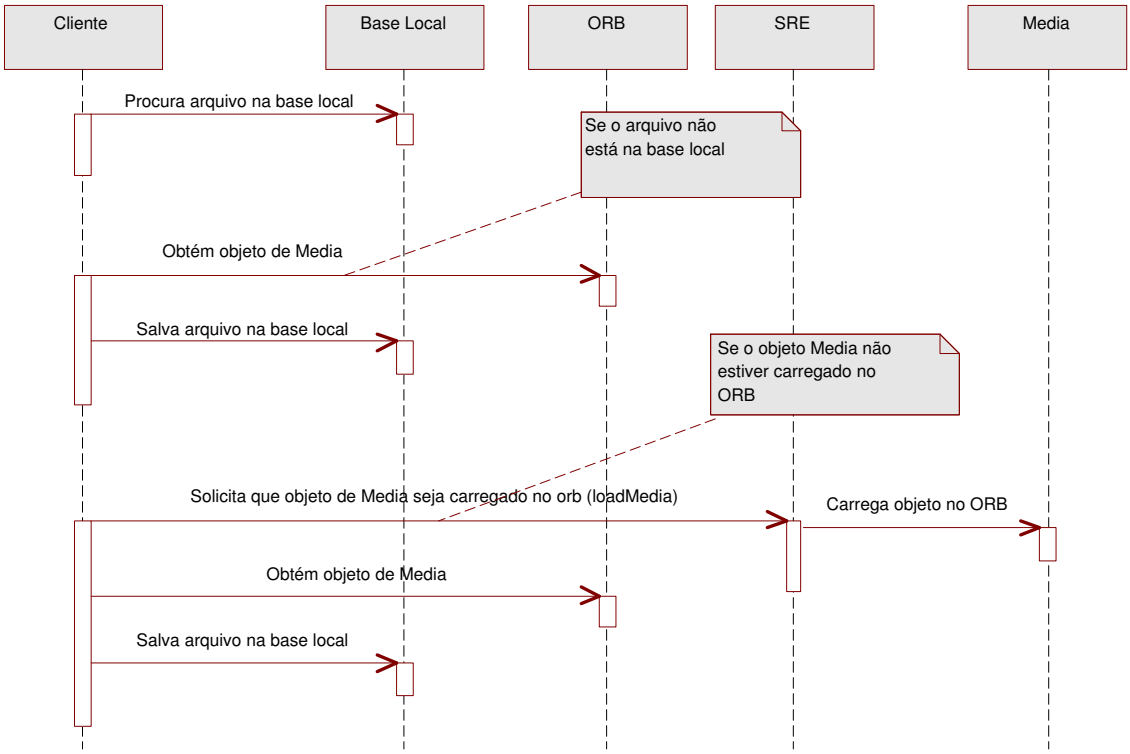

Figura A.1. *Diagrama de Sequência: Obter arquivo de mídia*

A.2. Diagrama de Atividade

O diagrama de atividade a seguir mostra a sequência de ações que são executadas para obtenção de um arquivo de mídia pelo cliente.

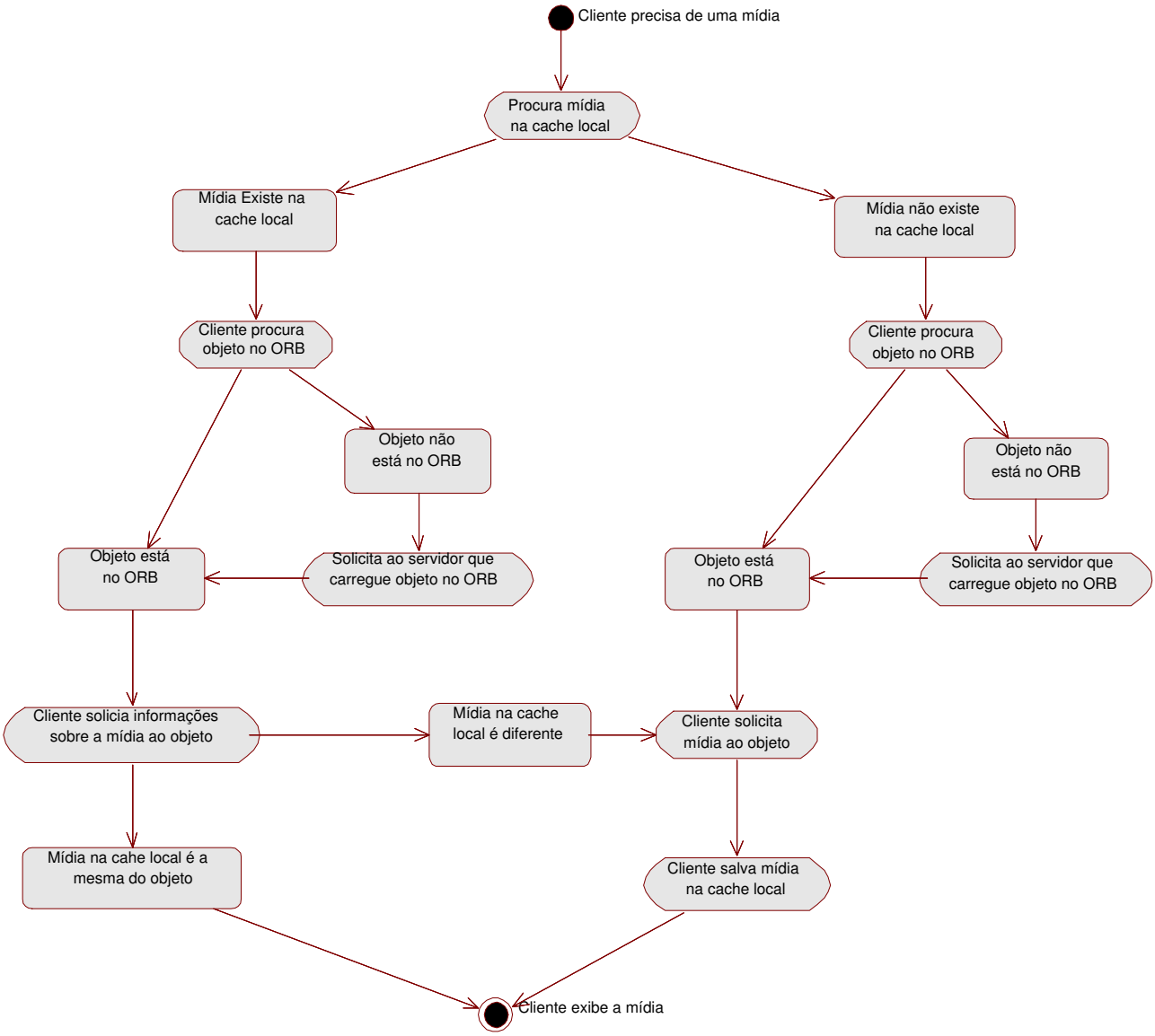

Figura A.2. *Diagrama de Atividade: Obter arquivo de mídia*