UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA - CIn

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

2006.1

UNINDO SISTEMAS E BANCOS DE DADOS DISTRIBUÍDOS

Por

BRUNO RODRIGO CUNHA DE ABERU

T r a b a l h o d e G r a d u a ç ã o em Computação Distribuída

Recife

Outubro, 2006.

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA - CIn

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

2006.1

BRUNO RODRIGO CUNHA DE ABREU

Unindo Sistemas e Bancos de Dados Distribuídos

Este trabalho foi apresentado ao programa de graduação em Ciências da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Orientador: Carlos André Guimarães Ferraz

Co-Orientador: Fernando da Fonseca de Souza

Recife

Outubro, 2006.

Agradecimentos

"Felicidade não é um lugar ao qual se quer chegar, mas sim uma forma de viajar. Viva feliz e nunca deixe de sonhar, pois tudo aquilo que hoje é realidade é porque foi sonho um dia".

Antes de tudo a Deus, por ter me dado a oportunidade de fazer o curso dos meus sonhos, e ter permitido que eu conseguisse chegar aqui, a sua conclusão.

A minha família, meus pais e meus irmãos, que tanto amo, que sempre me motivaram, reclamaram, acompanharam, que sempre estiveram e sempre estarão ao meu lado em qualquer situação.

Aos todos os meus amigos da igreja, em especial, Kelly, minha namorada, Simonete, Valéria, Luciano, Katheryne, Gyselli e Polyanna, por toda força que me deram.

A minha turminha da Visão: Luis, Maria, Clécia, Juliano, João Marcelo, Rafaela, Juliana, que sempre torceram por mim.

Aos meus amigos do CIn, mister Marcos, Dinallllllllllllldo, doutora Hirakava, Lucas, Guilherme, Victor, João Rodrigo, João Marcelo, Daniel, Patrícia, Bado e ao meu amigão Cristiano, que me acompanharam durante todo este período de curso.

A todos meus amigos da vida, da Chesf, da Vila Tamandaré e Inocop, espalhados por Recife, pelo Brasil e pelo mundo, que de uma forma ou de outra, estiveram sempre rezando e torcendo para que eu pudesse realizar este sonho.

Aos meus amigos “chefes” conseguidos no Nutes, Prefeitura e Chesf, em especial, Paulo Beltrão e Eya Miranda, por todo carinho e preocupação comigo e com meu curso.

E claro, aos meus professores do Centro de Informática, em especial a quatro professores amigos, dentre estes meus dois orientadores, Carlos Ferraz e Fernando Fonseca e a Alexsandro Gomes e Patrícia Tedesco, pois além de formação e apoio como professores, se preocupavam como amigos.

Resumo

O ramo da computação distribuída faz-se presente no nosso dia-a-dia, com isso diversas características de distribuição, como transparência, segurança, velocidade entre outras, tornam-se uma realidade. Porém, nesse grande ramo da computação distribuída dois ramos menores podem ser observados: os sistemas distribuídos e os bancos de dados distribuídos. Percebe-se o crescimento dessas duas áreas em separado, e se elas trazem tantos benefícios isoladamente, que benefícios elas não podem trazer trabalhando em conjunto, unindo sistemas e bancos de dados distribuídos?

Assim esse trabalho se propõe a observar as características dos sistemas e bancos de formas separadas e verificar como elas podem trabalhar juntas. Além disso, o trabalho também propõe-se a analisar um pouco sobre a televisão digital, no aspecto das mídias distribuídas e ver como ela se encaixa nesse contexto

Palavras-chaves: sistemas distribuídos, bancos de dados distribuídos, middleware, padrão de projeto, sistemas multimídia distribuídos, televisão digital.

Abstract

The branch of the distributed computation becomes present in ours days. Thus, several characteristics of distribution how transparency, security, speed, among at others, becomes a reality. However in this great branch of the distributed computation two lesser branches can be observed: distributed systems and distributed databases. The growth of these two areas is perceived separately and if they bring as many benefits separately, what benefits they could bring working together, joining systems and distributed databases?

Therefore, this work is arrived at observing the characteristics of distributed systems and databases separately and verifying how they can work together. Moreover the work also considers to analyze briefly the digital television, as far as distributed medias are concerned and verify how it fit in this context as well.

Keywords: distribuited systems, distribuited databases, middleware, standard of project, distribuited multimedia systems, digital television

Sumário

Lista de Figuras

1 Introdução

A quantidade de serviços oferecidos pela informática cresce a cada dia. Dentre esses, serviços de internet, e-mails, controle de processos e equipamentos em industrias, sistemas financeiros e bancários, sistemas de gestão, entre outros diversos, que se tornaram presentes no dia a dia, utilizando-se para isso de redes de computadores cada vez complexas.

As redes possibilitaram a distribuição do processamento entre computadores diferentes. Além da simples divisão de tarefas, permitiu a repartição e a especialização das tarefas entre os computadores conforme a natureza de cada um. E, cada vez mais, o uso de redes locais e da Internet, que ocorre desde as grandes empresas até o uso doméstico, permite a criação de sistemas distribuídos. Segundo Tanenbaum [TAN95], um sistema distribuído é "uma coleção de computadores independentes que parecem ao usuário como um único computador".

Um ambiente distribuído possui um conjunto de processadores conectados por uma rede, sem memória compartilhada. Essa ausência de memória requer que a interação entre as tarefas seja feita através da troca de mensagens. E, para aproveitar esses sistemas da melhor forma possível, é necessário fornecer o suporte adequado.

Comparados aos ambientes centralizados, os ambientes distribuídos são mais complexos, já que existe pouca experiência em projetos de aplicações distribuídas; é difícil tratar questões de redes e, um dos principais problemas, a questão de segurança. Porém, a série de vantagens em contraposição às desvantagens é muito grande: aproveitamento de máquinas de baixo potencial; algumas aplicações têm natureza distribuída; maior facilidade de crescimento; possibilidade de replicação, resultando em disponibilidade e tolerância a falhas, entre outras. Uma vez que a programação distribuída é mais complexa que a programação seqüencial, diversos trabalhos são desenvolvidos tanto com o objetivo de explorar o paralelismo de forma implícita ou automática, quanto com o intuito de tornar a expressão do paralelismo mais simples.

Um exemplo desses sistemas distribuídos, muito presente nos dias de hoje é a arquitetura cliente/servidor, onde diversos clientes comunicam-se com os servidores, os quais são especializados na execução de algumas tarefas.

1.1 O Trabalho

Quando se aborda computação distribuída, sistemas distribuídos, ambientes distribuídos ou dados distribuídos, muitas dessas palavras se confundem. Então para o melhor entendimento, considere-se **computação distribuída** como a grande área de distribuição na qual todos os projetos distribuídos se incluam; **sistemas distribuídos** como as aplicações, a parte que cuide do processamento e transmissão de informações e, **bancos da dados distribuídos** como os sistemas que cuidem dos dados, de sua replicação, de sua consistência, de seu armazenamento.

Por que se fazer essa divisão? Durante o curso de ciências da computação, ao estudar a grande área da computação distribuída, foi possível observar a real existência dessas duas subáreas, de sistemas e de bancos de dados, e ao tentar aproximá-las, apareceram algumas dificuldades.

Então, surgiu o desafio nesse trabalho: aproximar e, por que não dizer? Unir os sistemas e os bancos de dados distribuídos, verificando como eles podem trabalhar em conjunto, quais as vantagens dessa união e quais as desvantagens. Além de, também observar como a Televisão Digital (TVD) se encaixa nesse contexto, já que, ao se falar de TVD, trata-se de mais uma área da distribuição, **mídias distribuídas**.

Antes de tratar o assunto principal, um estudo sobre a área da computação distribuída se faz necessário, de modo que se possa entender todo o contexto do trabalho; analisar os sistemas e os bancos de dados distribuídos, para aí sim chegar na sua união.

1.1.1 Estrutura da Monografia

A monografia é composta desse capítulo introdutório e de mais seis capítulos.

O capítulo 2 tem por objetivo apresentar a área da computação digital, suas características, vantagens e desvantagens e as estruturas existentes.

O capítulo 3 apresenta os sistemas distribuídos, mostrando as características de um middleware e suas funcionalidades, e descreve alguns padrões de projetos.

O capítulo 4 apresenta os bancos de dados distribuídos, suas principais características e formas de utilização.

O capítulo 5 mostra as formas como sistemas e bancos de dados distribuídos podem trabalhar juntos, e exibe algumas experiências universitárias e profissionais do autor sobre essa união.

O capítulo 6 apresenta as mídias distribuídas, principalmente à televisão digital e mostra como esta faz parte da união de sistemas e bancos de dados distribuídos.

O capítulo 7 expõe uma série de conclusões, e sugestões que podem ser aplicadas a trabalhos futuros.

2 Computação Distribuída

O crescimento da computação distribuída tem sido guiado pela evolução da tecnologia e pela expansão da utilização e interligação das redes de computadores. E quanto maior for a evolução na utilização de redes, mais tende-se a ter novas e melhores soluções distribuídas. Pode-se observar isso ao se falar da internet que, a cada dia, cresce o número de usuários, permitindo que esses utilizem diversas aplicações distribuídas, além do crescimento da largura de banda, permitindo uma comunicação mais fácil e mais rápida entre os componentes distribuídos.

Para perceber esse crescimento, basta reparar um pouco na história, nos anos 70 um computador era tudo. E rede? Quem usava aquilo? Cheia de problemas, erros e muito lenta. Hoje, "The network is the computer"(John Gage - Sun Microsystems) [WIKI].

A computação distribuída permite o compartilhamento de recursos, como impressora ou *scanners*; possibilita distribuir tarefas entre os computadores participantes; decompõe um sistema complexo num conjunto de sistemas mais simples. Essas características motivam, cada vez mais, investimentos nessa área. Pois, com essas características, a computação distribuída propicia diversas vantagens comparadas à computação centralizada.

Entre essas vantagens, tem-se, um custo inferior, que ao invés de se utilizar um supercomputador, utiliza-se várias máquinas de menor potência, para atender a mesma demanda ou até a uma demanda maior. E com uma quantidade maior de computadores, pode-se conseguir um melhor desempenho, já que com vários computadores pode-se ter um paralelismo na execução do sistema, e com mais computadores, também se têm uma maior disponibilidade, visto que, se uma máquina falhar, o sistema como um todo não é abalado, além de se que pode ter formas de dar suporte a falhas, ou seja, mesmo que estas venham a ocorrer, o sistema continua funcionando normalmente. Isto será observado um pouco mais à frente neste trabalho.

Outra vantagem é que o sistema pode ser estendido, tanto no sentido computadores, ou seja, novos computadores serem inseridos ao sistema, para fornecer um maior suporte, uma maior disponibilidade; ou no sentido de novas funcionalidades, ou seja, o sistema é incrementado com novas funções, referentes a segurança, transparência, ou novas funções do próprio sistema, incrementando as funções por este já fornecidas.

Porém, como se está tratando de computação distribuída, os componentes acabam tendo uma visão parcial do sistema como um todo, ou seja, ao se ocorrer uma alteração em alguma parte do sistema, demora-se um certo tempo, diferente de zero, para que essa informação chegue a todo o sistema. Além desse pequeno atraso, um problema maior vem ao se referir às falhas, quando ocorre alguma falha de comunicação, ou alguma falha de outra natureza, partes diferentes do sistema podem ter informações incoerentes.

Esses problemas, referentes à distribuição fazem com que os ambientes distribuídos venham a necessitar de alguns requisitos. A heterogeneidade é um deles, dado que não se fala apenas de um computador e sim de vários, o sistema deve dar suporte a sistemas operacionais diferentes, hardware variados, topologias de redes, protocolos de comunicação, e, em alguns casos, a diferentes linguagens de programação.

Esses ambientes devem ser transparentes ao usuário, de forma que este o utilize como se estivesse em um ambiente centralizado. Para isso, deve haver transparência de acesso, escondendo as diferenças no modo de acesso a um recurso; transparência de localização, escondendo a localização do recurso; transparência de replicação, escondendo que o recurso é replicado; transparência de concorrência, escondendo que o recurso pode estar sendo utilizado por outros usuários, entre outros tipos de transparências, que fazem com que o usuário perceba o sistema como um todo e como uma coleção de componentes independentes.

Outro requisito é a sincronização e concorrência, pois, como o ambiente é distribuído, usuários diferentes tentam um acesso simultâneo a um recurso, ou a uma tarefa. Com isso, há a necessidade de sincronizar esse acesso, de forma a manter a coerência e consistência do sistema como um todo. Para isso, utiliza-se de diversas técnicas de sincronização e

controle de concorrência, como ordenação de eventos, filas, relógios, prioridades, diversos algoritmos existentes para tratar desse requisito, mas que não serão abordados nessa monografia.

Outra necessidade é a segurança. Como foi dito anteriormente, a segurança é um dos principais problemas, por tratar-se de um ambiente distribuído. Com isso, há uma necessidade de dedicar-se a esse aspecto com bastante cuidado. Como os recursos e as informações estão espalhados por todo o ambiente, técnicas de segurança como, autenticação, proteção de ataques, criptografia, *firewalls*, vêm a se fazer bastante necessários, para que, a confidencialidade, a integridade e a disponibilidade sejam características do ambiente final.

E por fim, esses ambientes devem apresentar alguma forma de tratamento de falhas. Como visto, o sistema pode falhar, comportando-se de uma forma indevida devido a algum erro, porém, como se está tratando de ambientes distribuídos, essas falhas normalmente são isoladas, facilitando o seu tratamento. Para isso são utilizados algoritmos de detecção de falhas, como *checksum*, e ao detectar-se alguma falha, o ambiente deve tratá-la de alguma forma, definida pelo seu desenvolvedor.

Uma estrutura geral de um ambiente de computação distribuída pode ser observada na Figura 2.1. Além disso, pode-se observar os pontos de vista de cada um dos participantes do ambiente.

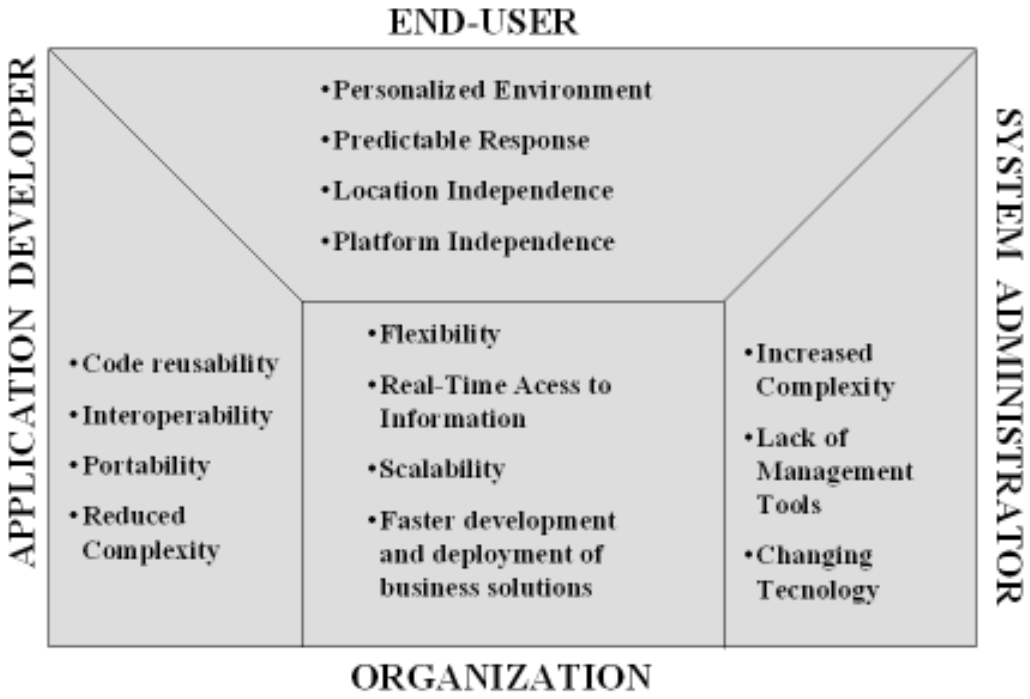

Figura 2.1. Estrutura geral de um ambiente distribuído [KHA94]

Na computação distribuída tem-se três componentes principais: dados, referente às informações armazenadas; processamento, componente que processa a informação, que a transmite, que a protege; e a interface, ou camada de apresentação, camada pela qual o usuário interage com o ambiente distribuído.

Em um ambiente centralizado, esses componentes se encontram conforme a Figura 2.2, onde os três componentes encontram-se em um único computador.

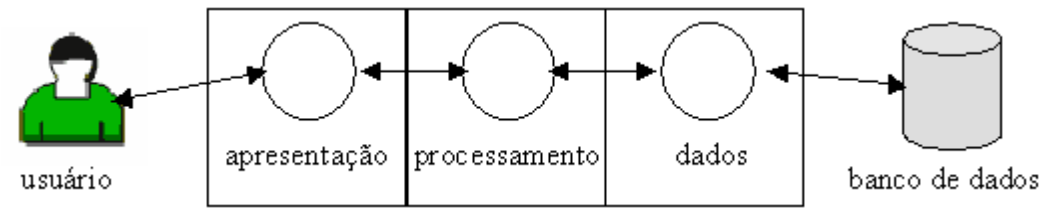

Figura 2.2. Ambiente Centralizado

Já, para os ambientes distribuídos, esses componentes podem se apresentar com diversas estruturas, como será exibido nas Figuras 2.3 à 2.8, e que posteriormente serão explicadas.

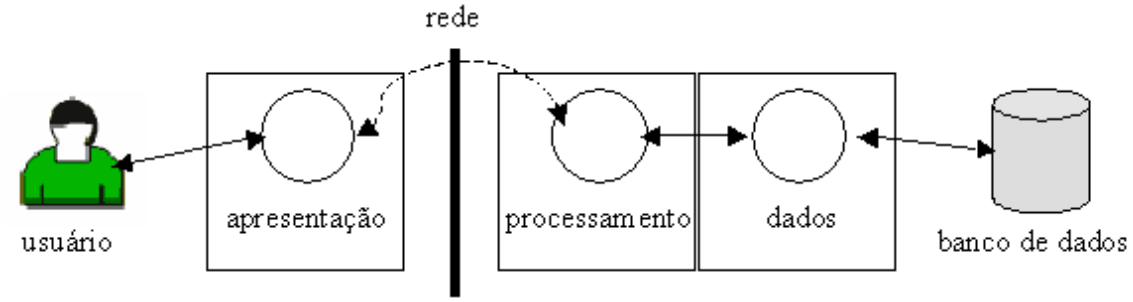

Figura 2.3. Componente de apresentação apenas no cliente

A Figura 2.3 mostra a estrutura do modelo cliente/servidor, a interface fica localizada apenas do lado do cliente, que faz algumas solicitações ao servidor. Este processa a solicitação, acessa os dados caso seja necessário e devolve a resposta ao cliente.

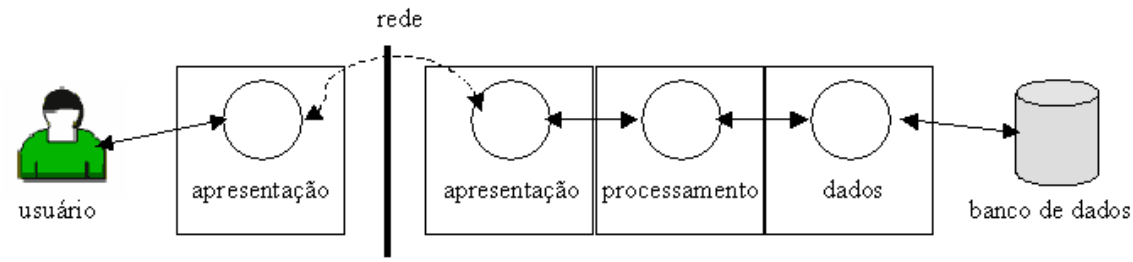

Figura 2.4. Componente de apresentação dividido entre cliente e servidor

A estrutura da Figura 2.4 é muitas vezes esquecida, na qual a camada de apresentação também se encontra do lado do servidor, e isso permite que o servidor visualize alguma informação, ou ate mesmo que interaja com a solicitação feita pelo cliente. Para que após isso a solicitação seja processada, acessando ou não aos dados, de modo que haja o retorno ao cliente.

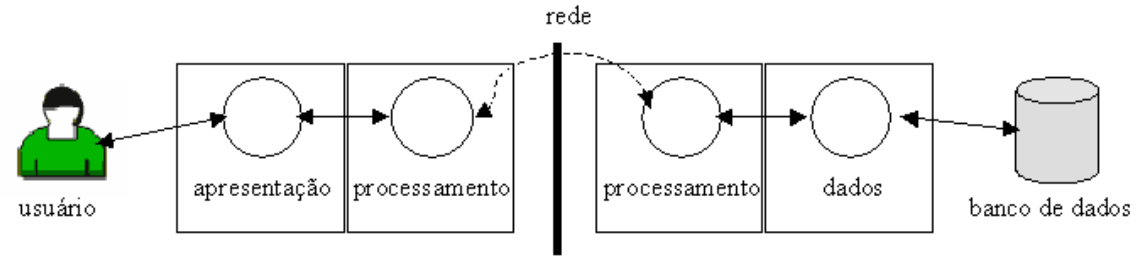

Figura 2.5. Processamento distribuído

Atualmente a estrutura que mais cresce no ramo da computação distribuída é a da Figura 2.5. Muitas vezes, o lado do cliente pode executar algum processamento, para acrescentar facilidades ou segurança à informação a ser transmitida. Com isso, faz-se necessário o processamento da informação antes dessa ser enviada.

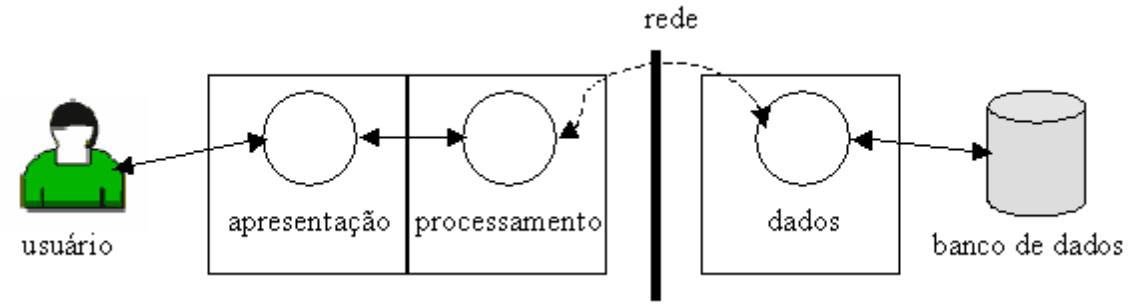

Figura 2.6. Acesso a dados remotamente

Muitas vezes o processamento não é o ponto chave do ambiente distribuído, e sim os dados. Então, a estrutura apresentada na Figura 2.6 vem a mostrar a forma de organização quando o processamento é feito todo do lado do cliente, e através de um acesso remoto aos dados, ocorre uma seleção, uma inserção, remoção ou modificação nos dados existentes. Muitas vezes do lado do cliente existe alguma forma de armazenamento temporário desses dados, ou em memória ou em um banco local, para que, o acesso remoto só venha a ser executado quando houver a real necessidade. Até lá, o processamento é feito sobre os dados armazenados temporariamente.

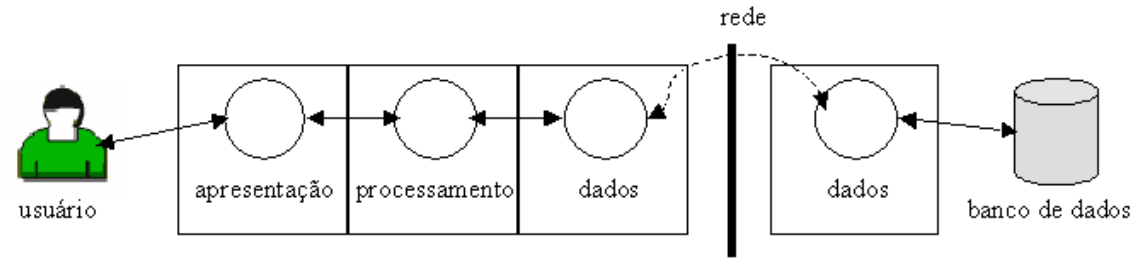

Figura 2.7. Gerenciamento de dados distribuídos

A Figura 2.7 mostra a estrutura típica dos bancos de dados distribuídos. Onde os dados estão distribuídos por todo o ambiente, e são gerenciados pelos sistemas de gerenciamento de bancos de dados distribuídos.

Das estruturas apresentadas acima, da Figura 2.3 à 2.6, todas se encaixam no perfil de sistemas distribuídos, ou aplicações distribuídas. E serão tratadas como um todo no próximo capítulo, para que se possa perceber como funcionam os sistemas distribuídos. Já a Figura 2.7, que é a estrutura típica dos bancos de dados distribuídos, será abordada no capitulo seguinte.

E após isso, pode-se estudar como os sistemas e os bancos de dados distribuídos podem trabalhar juntos, gerando assim, a nova estrutura de ambientes distribuídos, apresentada na figura 2.8, abaixo.

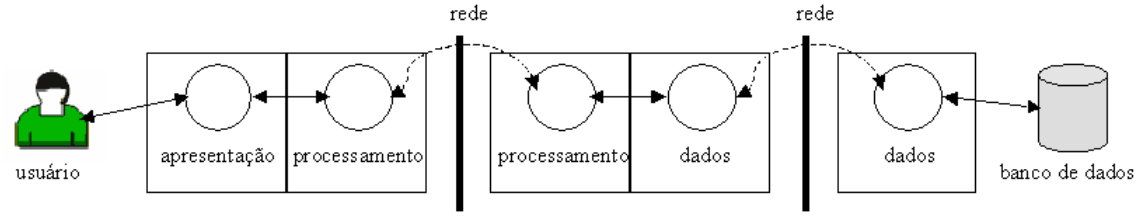

Figura 2.8. Sistemas e Banco de dados distribuídos

3 Sistemas distribuídos

Como citado anteriormente, esse abordará os sistemas distribuídos ou aplicações distribuídas. E não há como falar de sistemas distribuídos sem falar sobre **middleware,** que tem como função facilitar a construção de sistemas distribuídos. Um middleware pode ser definido como, “Uma camada de software que permite a comunicação entre aplicações (distribuídas)” ou “Um conjunto de serviços que fornece comunicação e distribuição de forma transparente” (Carlos Ferraz) [FERR]. Ou seja, um middleware é uma ferramenta utilizada em sistemas distribuídos para esconder a heterogeneidade, para mascarar a complexidade dos mecanismos de rede

O middleware implementa seus próprios protocolos e está localizado entre as camadas de aplicação e transporte, ou seja, ele implementa as camadas de sessão e apresentação. O middleware possui uma coleção de serviços e estes permitem que haja a interação nos sistemas distribuídos, como pode ser visto na Figura 3.1.

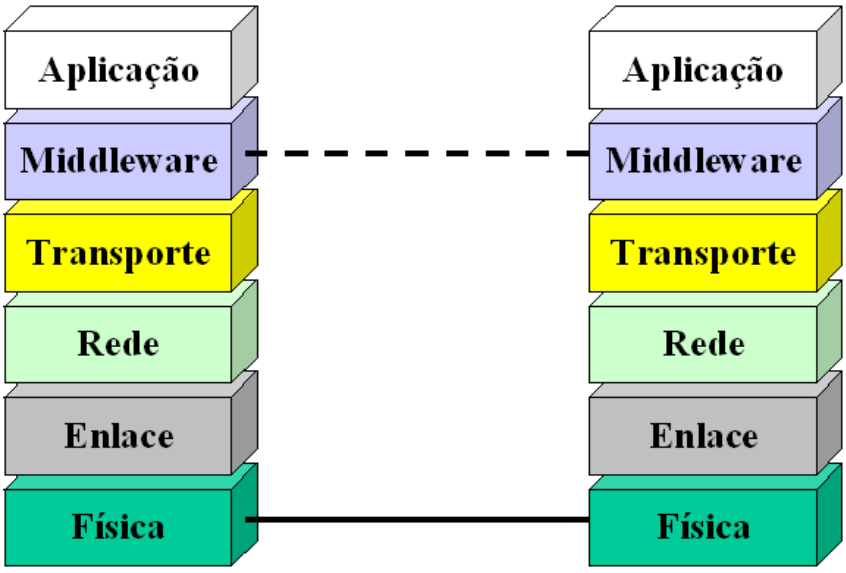

Figura 3.1. Localização e comunicação do middleware

Para que o middleware possa realizar as funções apresentadas anteriormente, ele conta com uma estrutura de quatro camadas (Figura 3.2): infra-estrutura, cuida da parte de concorrência e comunicação, além permitir que o software seja portável; distribuição, camada que permite as operações remotas; serviços comuns, ou seja, serviços de alto nível, como, segurança, transação, controle de concorrência, sincronização; e a ultima camada, de serviços específicos, que como o próprio nome diz, realiza tarefas específicas, próprias ao sistema distribuído que o utiliza.

Figura 3.2. Estrutura geral de um middleware

Middleware pode ser classificado de diversas maneiras, dependendo de sua forma de comunicação, da sua forma de disponibilização, da sua linguagem, do seu ambiente de execução. Dentre essas classificações, os modelos mais comuns são: middleware orientado a transações, middleware orientado a mensagens (MOM), middleware procedural, middleware baseado em eventos, middleware baseado em objetos e middleware multimídia.

Middleware transacional utiliza comunicação síncrona ou assíncrona, através da chamada de procedimentos remotos. Normalmente é utilizado em sistemas que exigem que a execução desses procedimentos remotos seja muito rápida, consistente e coerente, pois acessam bancos de dados muito importantes. Um exemplo disso são os sistemas bancários.

Um outro middleware bastante utilizado é o middleware orientado a mensagem (MOM), como o próprio nome diz, utiliza-se de mensagens para estabelecer a comunicação, e com isso permite a comunicação assíncrona, dando liberdade ao sistema

sem fazer com que ele fique parado esperando algum retorno. Para isso, utiliza-se de filas, que armazena a informação até que se tenha a garantia de que esta chegou no seu local de destino, e, muitas vezes, utiliza-se te técnicas de verificação de transmissão e de re-envio.

E outro middleware a se falar, de bastante importância para esse trabalho, é o middleware orientado a objetos, onde métodos de objetos remotos são invocados. Isto é possível devido a uma interface dos serviços oferecidos por esses objetos. Exemplos desse middleware são Corba, Rmi e Ejb [ROSA].

Após uma breve visão em *middleware,* o trabalho focalizará sistemas distribuídos, ao estudar alguns padrões de projetos. A partir desses padrões, notando suas características e suas aplicabilidades, pode-se perceber como inserir bancos de dados distribuídos a eles.

3.1 Padrões de projetos

Dentre os diversos padrões de projetos, alguns foram selecionados devido a suas características, que podem vir a permitir a união entre os sistemas e bancos distribuídos, dentre esses estão:

- **Wrapper Facade ►** É um padrão de projeto (Figura 3.3) que encapsula funções e dados fornecidos por API, criando uma nova API de mais alto nível e melhor definida. Com isso, ganha-se portabilidade, coesão, consistência, tratamento de erros, entre outros benefícios, se comparados à antiga API de baixo nível.

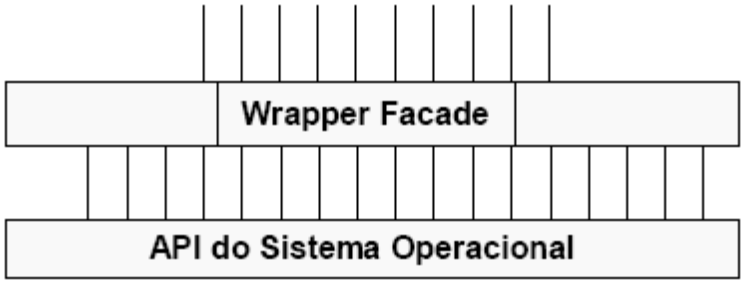

Figura 3.3. Wapper Facade

- **Component Configurator** ► padrão usado em aplicações que necessitam que seus componentes sejam iniciados, pausados, reiniciados e finalizados sem abalar a execução do sistema como um todo (Figura 3.4). Esses componentes inicializados ficam armazenados em um repositório. Esse repositório é gerenciado por um configurador que também gerencia todos os componentes presentes no repositório.

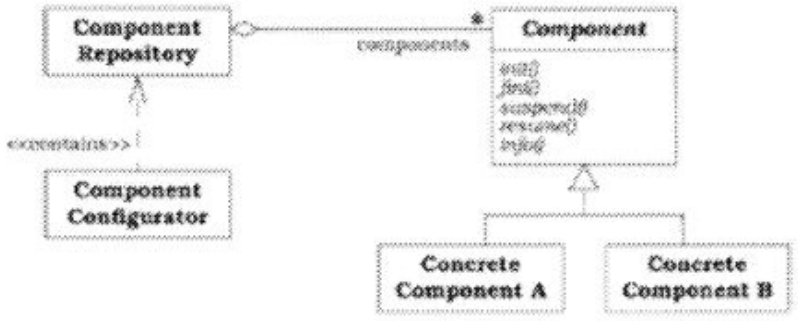

Figura 3.4. Component Configurator [ROSA]

- **Reactor** ► Padrão para tratar sistemas que recebem muitas solicitações simultâneas dos usuários (Figura 3.5). Essas solicitações são tratadas de forma síncrona e serial. Contudo, o servidor não pode ficar bloqueado ao executar uma solicitação, e nem pode deixar de ser transparente ao usuário.

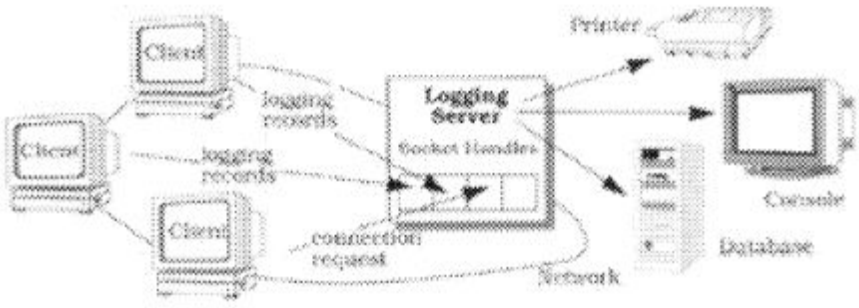

Figura 3.5. Ambiente para uso do padrão Reactor [ROSA]

- **Proactor** ► Similar ao Reactor, onde a grande diferença é que essas solicitações devem ser tratadas de forma assíncrona (Figura 3.6). Porém, isso faz com que a programação torne-se bastante complexa de se construir, testar e utilizar.

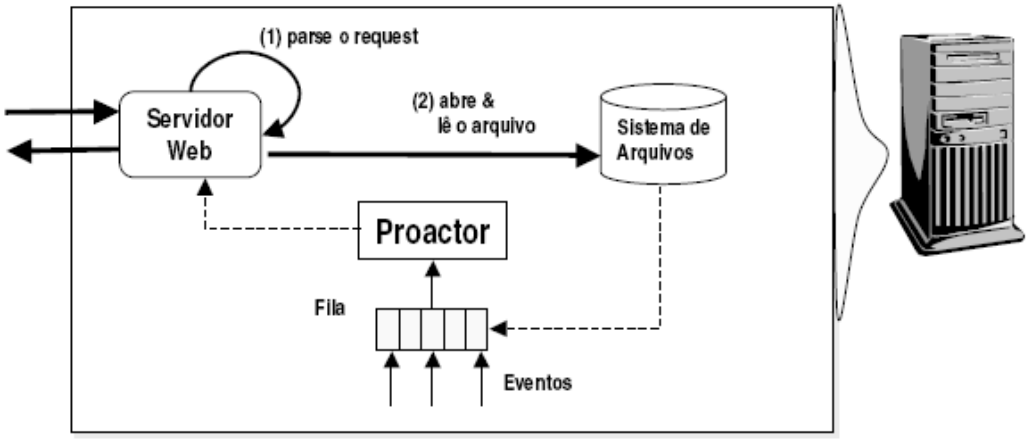

Figura 3.6: Proactor [ROSA]

- **Scope Locking** ► Padrão que cria um lock quando a execução de um sistema entra em uma seção critica, e libera esse lock quando a execução deixa a seção. Muito utilizado em aplicações concorrentes e com compartilhamento de recursos, para impedir erros de coerência e consistência. Vale ressaltar, que esse padrão deve se observado com bastante cautela, pois esses locks podem gerar deadlocks.

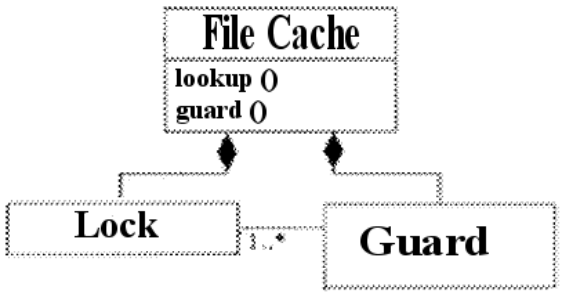

Figura 3.7 Scope Locking

- **Monitor Object**: Padrão semelhante ao scope locking, porém ele não trata de seções e sim de objetos, sincronizando o acesso aos mesmos. Normalmente utilizado em sistemas com muitos *threads*, trabalhando para que esses acessem concorrentemente um objeto compartilhado.

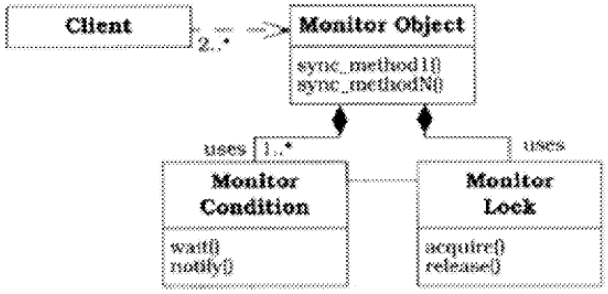

Figura 3.8 Monitor Object [ROSA]

Esses padrões de projetos serão retomados novamente quando for abordada a união entre sistemas e banco de dados distribuídos. A seguir, serão abordados os bancos de dados distribuídos, de modo a completar o embasamento conceitual que permitirá aproximar essas duas grandes sub-áreas.

4 Bancos de dados distribuídos

Já foram, de uma forma geral, abordadas as vantagens que a distribuição tem a oferecer. Serão acrescentadas ao estudo mais algumas características específicas dos bancos de dados distribuídos. Sabe-se que a tendência atual é que a quantidade de dados cresça cada vez mais. Esse crescimento faz com que o processamento e acesso a esses dados tenda a ser mais lento, visto que, com um grande número de dados, acessar um simples dado pode exigir mais tempo do que em um banco pequeno. Além do que, está se aumentando em muito o número de usuários desse banco, se a quantidade de solicitações ao banco for muito grande, alguns desses usuários necessitaram esperar bastante tempo para ter a sua informação.

Observando tudo isso, é que os bancos de dados distribuídos ganham força para o seu desenvolvimento. Com redes muito mais potentes, e com os custos de comunicação cada vez menores, distribuir e, até mesmo, replicar os dados para ganhar velocidade de acesso, técnicas de prevenção de falhas, permissão a acesso concorrente e paralelo, já que agora os dados não se encontram em apenas um local. Porém, como todo ambiente distribuído, tem-se que prevenir de alguns problemas. Dentre esses, controle de replicações, sincronização, segurança, tolerância a falhas e manutenção da integridade dos dados, problemas esses a serem observados e tratados durante a construção do banco de dados distribuídos.

Segundo Özsu e Valduriez [OZS01] “Banco de dados distribuído (BDD) é uma coleção de múltiplos bancos de dados logicamente inter-relacionados e distribuídos sobre uma rede de computadores”. E esses bancos de dados distribuídos são gerenciados por um Sistema de Gerenciamento de Banco de Dados Distribuído(SGBDD).Ainda segundo Özsu e Valduriez [OZS01], “SGBDD é um sistema que gerencia um BDD e provê um grau de transparência para os usuários”. Um BDD é ilustrado na figura 4.1.

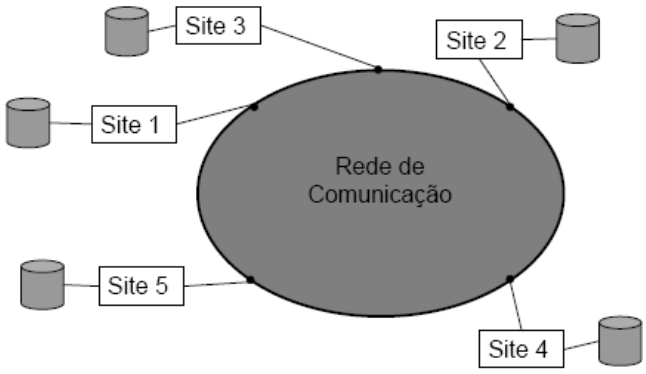

Figura 4.1 Ambiente de um SGBDD

Esses BDD devem ser transparentes aos usuários. Essa transparência, ao se falar de dados, pode ser de três formas: transparência de rede ou localização, transparência de replicação e transparência de fragmentação.

A transparência de rede ou localização, faz com que o usuário não necessite saber onde se encontra o dado que ele acabou de solicitar, e que lhe pareça que o dado é local. A transparência de replicação é mantida quando mesmo ao se possuir replicações espalhadas pelo sistema, essas réplicas sejam todas consistentes. E a transparência de fragmentação mantém a consistência e integridade em cada fragmento de dados. Aqui, duas palavras vêm a fazer-se bastante presentes no contexto dos bancos de dados distribuídos, fragmentação e replicação.

A decomposição de dados em fragmentos permite que cada um seja tratado como uma unidade isolada, favorecendo a execução de transações concorrentes sobre o banco de dados como um todo. Além disso a fragmentação pode permitir a execução paralela de uma única consulta, visto que seus dados estão divididos na rede. Então, ao executar uma consulta, essa pode ser dividida entre os sistemas distribuídos e com isso executar de forma paralela.

Fragmentar pode trazer benefícios, mas também pode trazer desvantagens. Então, faz-se necessário estudar como se irá fragmentar os dados e se vale a pena. Muitas vezes, uma consulta pode requerer a união de dois fragmentos, o que pode ser bastante dispendioso, além da necessidade de manter a integridade dos dados fragmentados.

Algumas características sobre fragmentação devem ser observadas, como: o tamanho dos fragmentos; as formas de acesso e a freqüência de acessos aos fragmentos; a localidade do acesso, se elas forem muito distantes pode-se fazer necessário à mudança do local de armazenamento do fragmento; capacidade e custos para armazenar e processar; custos para manter a integridade; requisitos de rede como, largura de banda, latência, sobrecarga e gerencia das transações.

Observando essas informações, no desenvolvimento do projeto uma séria de estudos deve ser feita para verificar a relação custo/beneficio de se fazer a fragmentação. Mas, além de, fragmentar outra técnica usada em banco de dados distribuídos é a replicação, onde dados ou fragmentos são replicados e distribuídos pelo sistema. Muitas vezes, há a necessidade da replicação, pois uma quantidade de dados, que sofre modificação com pouca freqüência, pode vir a ser acessada com bastante freqüência em diversas localidades, havendo com isso a necessidade da replicação.

A replicação, assim como a fragmentação, é uma questão de projeto, havendo uma necessidade de se estudar a aplicabilidade de uma replicação sobre um determinado dado, pois essa replicação deve manter a consistência e coerência no sistema global.

Uma vez abordadas de forma geral os sistemas e bancos de dados distribuídos, pode-se estudar uma forma deles trabalharem em conjunto.

5 Unindo sistemas e bancos de dados distribuídos

A partir das considerações expostas nos capítulos anteriores, passa-se a estudar formas dos sistemas e banco de dados distribuídos trabalharem em conjunto. Inicialmente, pode-se fazer referência aos já citados padrões de projetos, mostrando como eles podem facilitar a interação.

5.1 Wrapper Facade

Esse padrão de projeto é muito interessante ao se tratar com bancos de dados, pois muitas vezes há uma série de funções para se acessar o banco de dados, e com o *wrapper facade* tudo isso pode ser reduzido a uma única classe, a qual deve-se chamar, e esta executa todas as API de baixo nível para realizar aquele procedimento.

Como se está tratando de bancos de dados distribuídos há uma série de processos a serem executados para realizar uma seleção, como: decomposição da consulta, localização dos dados, otimização da consulta e até mesmo alguma diretiva de segurança. Esses processos podem ser considerados como parte da API de baixo nível a ser implementada pelo *wrapper facade* como na Figura 5.1.

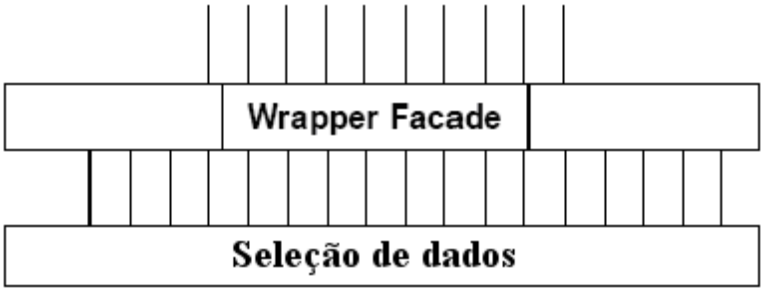

Figura 5.1 Wapper Facade e Banco de dados

Na elaboração desse middleware, pode-se levar em conta todas as características para a seleção da informação, ou seja, um serviço específico, ou pode-se apenas desenvolver os

mecanismos gerais para a execução de alguma seleção no sistema como um todo, ou seja, um serviço comum, permitindo que esse possa ser utilizado em outras partes do projeto, ou até mesmo em outros projetos.

Esse padrão pode ser adotado para a seleção, inserção, modificação ou remoção de algum dado, mas também pode ser adotado para outras finalidades como mecanismos de controle de replicação ou técnicas de tolerância à falha. Para tanto, basta ao desenvolvedor fornecer a API de baixo nível a ser utilizada pelo middleware.

5.2 Component Configurator

Como citado anteriormente, este padrão trabalha com componentes que são inicializados, pausados, reiniciados e finalizados. Isto pode ser adotado ao se criar componentes que cuidem do banco de dados, como técnicas de tolerância à falhas, que podem ser iniciadas pelo sistema em um determinado contexto, quando ocorrer um determinado problema. Ao se solucionar o problema, esse componente pode ser pausado e posteriormente reiniciado, caso o sistema ache adequado.

Um exemplo, para ficar mais claro: se o sistema apresenta uma falha, um dos mecanismos de tolerância é a anulação da requisição. Por exemplo, em uma transação bancária, se ocorre algum erro durante a transação um mecanismo de prevenção é o cancelamento total da transação. Um componente pode ser inicializado assim que a transação é solicitada, e pausado para que ele guarde as variáveis iniciais. Caso haja algum problema durante a transação, esse componente é re-iniciado para recuperar o sistema, ou caso não haja nenhum problema esse componente é finalizado.

5.3 Reactor e Proactor:

Padrões utilizados em sistemas que recebem muitas solicitações simultâneas. A diferença entre eles, como citado anteriormente, é a forma de tratamento dessas

solicitações. No caso do reactor ela é síncrona, enquanto que no caso do proactor ela é assíncrona. E como bancos de dados distribuídos são criados para darem suporte a diversas solicitações simultâneas, então, os padrões reactor e proactor podem ser utilizados para tratamento dessas solicitações.

5.4 Scope Locking:

Muitas vezes um banco de dados pode entrar em uma seção crítica, bloqueando o acesso à determinada área do sistema durante um período de tempo para que a integridade total do sistema possa ser mantida.

5.5 Monitor Object:

Padrão mais próximos aos bancos de dados que o Scope Loking. Como citado, ele é usado para cuidar da integridade de objetos. Muitas vezes, algum objeto é acessado simultaneamente por dois ou mais usuários, então se faz necessário algum mecanismo de sincronização a ser realizado pelo middleware para manter a integridade. Vale ressaltar a questão da replicação, pois com isso, o dado não se encontra em um único local e sim em réplicas espalhadas pelo sistema. Então, a sincronização deve ser feita levando-se em conta todas as replicas, para que elas se mantenham consistentes e coerentes.

5.6 Experiências do Autor

Durante a realização de duas disciplinas do curso de Ciências da Computação da Universidade Federal de Pernambuco: bancos de dados móveis e distribuídos [FONS] e tópicos avançados em sistemas distribuídos [ROSA], o autor desta monografia tentou aproximar as duas áreas. Contando com a colaboração dos professores das disciplinas, foi

possível observar fatos interessantes nessa aproximação, alguns um pouco decepcionantes, mas também outros bastantes motivantes quando da aproximação de ambas.

Tratava-se de um servidor de aplicação [ABRE1], com as seguintes características:

- Cliente ► este tinha a interface para que ele pudesse interagir com o ambiente, juntamente com um middleware de comunicação, que tratava das questões de comunicação, como organização da solicitação, criptografia, compressão e envio dos dados (Figura 5.2).

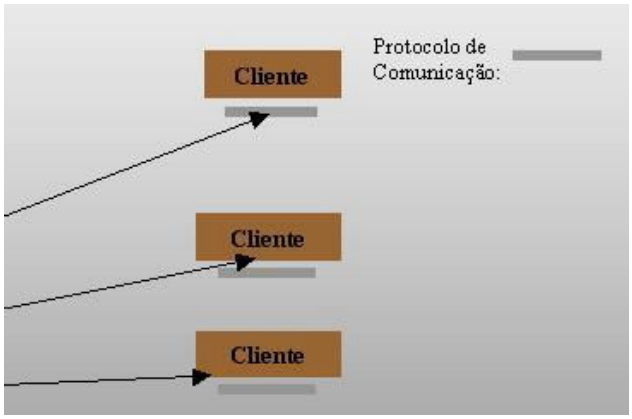

Figura 5.2. Clientes no Servidor de Aplicação

- Servidor de Aplicação ► servidor que possuía um container de middleware, contendo middleware de comunicação, de serviços específicos, além de possuir diversas aplicações registradas no mesmo. Dentre essas, um serviço de acesso ao SGBDD. A idéia inicial seria ter um grande banco de dados distribuídos para todo o servidor de aplicação. Entretanto, estar-se-ia saindo do padrão do servidor de aplicações (Figura 5.3). Então, o sistema de gerenciamento de banco de dados distribuídos funcionou de forma isolada à sua aplicação (Livro Net) [ABRE1].

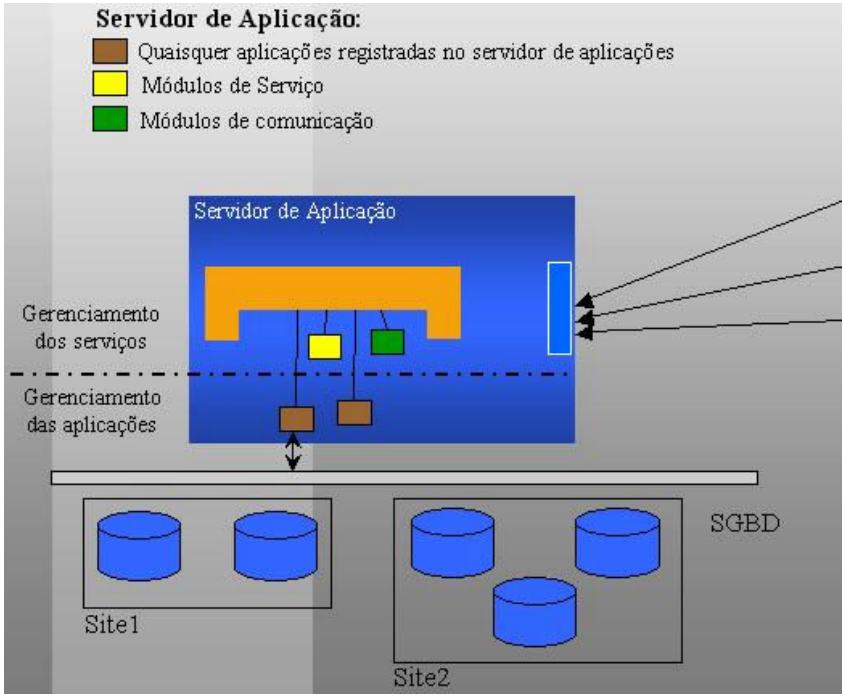

Figura 5.3 Servidor de Aplicação e SGBDD

Ao observar a Figura 5.3, percebe-se a forma de interação entre a aplicação distribuída e o banco de dados distribuídos, que se limita à comunicação feita entre este e uma única aplicação de todo o servidor de aplicações.

Ao iniciar esse desenvolvimento acreditava que a interação seria bem maior entre os sistemas distribuídos e os bancos de dados distribuídos, não foi o que aconteceu. Contudo, foi possível a criação de um sistema que apresentassem ambas as áreas.

Algumas observações puderam ser feitas, dentre essas, a principal foi a observação de que se estava tratando de um servidor de aplicação como o sistema distribuído maior, que foi o que bloqueou a maior interação. Então o que poderia ser feito para melhorar a interação nesse caso? Ao invés de tratar o servidor de aplicação como o sistema distribuído, poder-se-ia ter a aplicação LivroNet como o sistema distribuído. Então, a aplicação utilizaria o banco de dados distribuídos de forma habitual.

Como um exemplo, será considerada a empresa CHESF (Companhia hidrelétrica do São Franscisco), empresa na qual o autor desse trabalho atua há dois anos. Atualmente, o

referido autor faz parte de uma das equipes de desenvolvimento responsável pela criação de alguns sistemas Intranet.

Sobre a responsabilidade do referido autor, está o sistema de controle da tarifação, que controla todas as ligações dos telefones da CHESF. Também, há um sistema de agendamento de videoconferências, que agenda e gerencia todas as videoconferências a serem realizadas pela companhia. Além desses dois sistemas, deve-se destacar mais dois: primeiro o Safe ou prod, que controla a carga horária de todos os funcionários, e segundo o serviço de e-mails Notes da empresa.

O que essas quatro aplicações têm a ver com o assunto desse trabalho? Todas elas funcionam de forma distribuída e todas usam, de certa forma, de uma mesma base de dados.

O sistema de tarifação, além de armazenar todas as ligações, disponibiliza-as de forma que possam ser verificadas e estudadas. Este sistema é baseado nos telefones da CHESF, porém cada telefone está associado a um usuário, esta informação é registrada numa base de dados do próprio sistema.

No sistema de videoconferências, que se encontra distribuído, qualquer funcionário pode requisitar o agendamento de uma videoconferência. Esse sistema também necessita da lista de todos os funcionários da CHESF. Para não replicar a informação como o sistema anterior, esse sistema faz acesso remoto ao sistema de controle da carga horária dos funcionários.

O Safe, além da informação de todos os funcionários da empresa, tem o registro do horário que cada funcionário entra e sai da referida empresa.

E, por último, o serviço de e-mails, que também necessita da lista de todos os funcionários da empresa. Entretanto esse sistema tem sua base de dados de funcionário própria.

Com isso, pode-se perceber a existência de quatro grandes sistemas distribuídos, que de uma certa forma, poderiam estar trabalhando em conjunto, pelo menos no que se refere aos bancos de dados. E com o estudo desse trabalho, pode-se perceber que grandes melhorias

podem ser feitas nesses sistemas. Como por exemplo, no Safe, muitas vezes acessá-lo é bastante lento, visto que, da mesma forma que o sistema de videoconferência acessa sua base de dados, outros sistemas também acessam. Então, se poderia inserir algumas melhorias técnicas de bancos de dados distribuídos, como replicação e fragmentação.

Já no sistema de tarifação, não seria necessária a replicação de toda a base de funcionários, uma vez que essa está presente em outro local, e nunca é alterada por esse sistema. Então, poder-se-ia manter uma associação do ramal do telefone com o usuário, não mais localmente, mas distribuída.

E quanto ao sistema de e-mails, que por si só já apresenta uma série de problemas, visto que esse possui uma base de dados muito grande, poder-se-ia ter a informação distribuída dos funcionários. Além de procurar melhorar a estrutura atual do sistema, distribuindo sua própria base de dados (e-mails), e acrescentar novas funcionalidades, para facilitar a comunicação, melhorar a segurança e diminuir a quantidade de *spams*.

De uma forma geral uma estrutura final de um grande sistema distribuído com dados distribuídos poderia ser semelhante à apresentada na Figura 5.4.

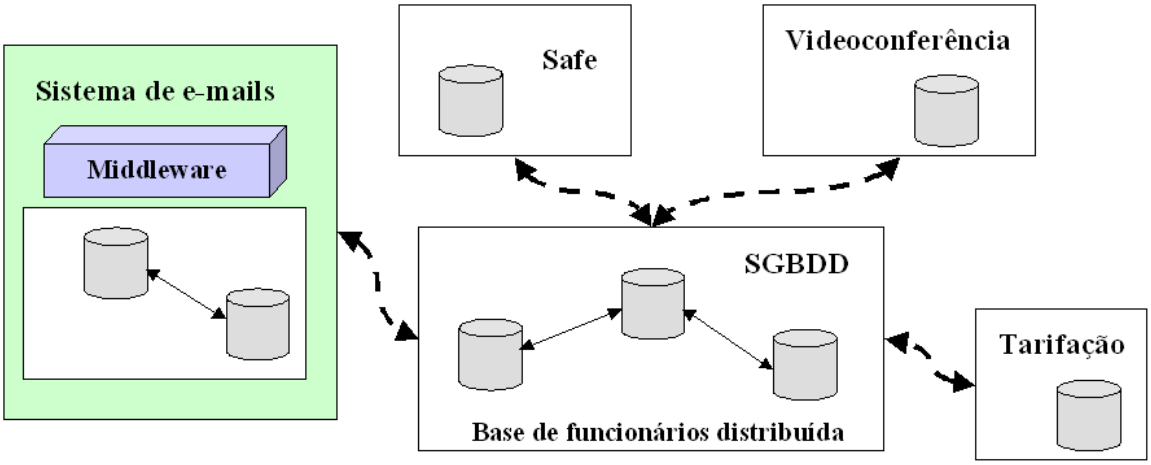

Figura 5.4. Melhorias da união para a Chesf

Além desses estudos, não se poderia deixar de mencionar os consagrados sistemas bancários. Sabe-se que eles apresentam falhas e ainda têm muito a melhorar, mas é visível a forma de utilização dos sistemas com a união proposta nesse trabalho, onde seus sistemas cada vez mais distribuídos acessam e gerenciam uma enorme base de dados distribuída, procurando trazer os melhores serviços a seus usuários e à empresa bancária principalmente.

Para finalizar o trabalho, será tratada no próximo capítulo a relação das mídias distribuídas com a televisão digital, observando como encaixa-la no contexto desse trabalho.

6 Mídias distribuídas e Televisão Digital

Os recursos multimídias exigem grande capacidade de processamento, transmissão, armazenamento e compressão, então se deve tomar os cuidados necessários ao se distribuir esse tipo de informação. Pois, ao se tratar de mídias, essas devem ser transmitidas da forma mais rápida possível, buscando a latência mínima, muitas vezes em velocidade de transmissão de tempo real. A transmissão das mídias é ilustrada na figura 6.1.

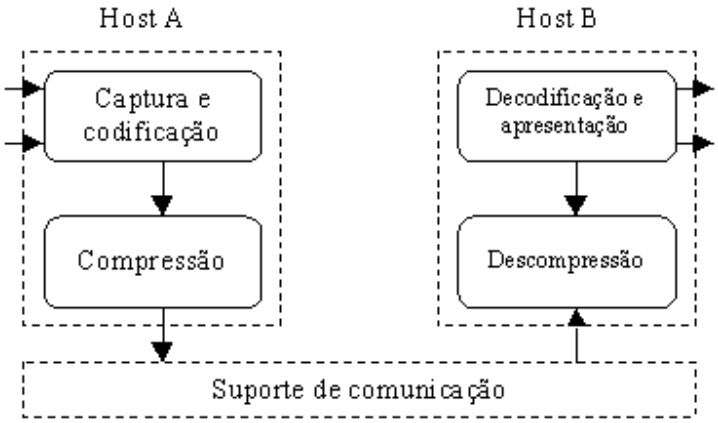

Figura 6.1 Transmissão de mídias

As redes ATM (Asynchronous Transfer Mode) [WIKI] apresentam-se como uma solução que busca resolver grande parte das limitações impostas às aplicações emergentes da Internet, procurado oferecer maior capacidade de transferência de dados através da rede, além de vários serviços exigidos por uma classe de aplicações que surge de áreas como redes de banda larga, comunicação de grupo e sistemas distribuídos: os **Sistemas Multimídia Distribuídos**.

Um sistema multimídia distribuído (SMD) integra uma variedade de informações multimídia, que estão espalhadas e conectadas via rede, dentro de uma aplicação, possibilitando que um usuário acesse remotamente diferentes tipos de informações e serviços, interagindo com o ambiente. O usuário pode decidir quando receber uma informação ou controlar o fluxo da apresentação, o que difere radicalmente das aplicações

convencionais de difusão, como TV ou rádio, onde não existe interação ou qualquer tipo de interferência na apresentação. Entre as aplicações de SMD, encontra-se: vídeo sob-demanda, jogos interativos multi-usuário, ensino a distância, videoconferência, e a **Televisão Digital**.

Nos sistemas multimídia distribuídos tem-se as três camadas de um sistema distribuído: apresentação, processamento e dados. Mais atenção será dada a duas camadas: a camada de apresentação, pela qual o usuário irá interagir com o sistema, e a camada de dados, também conhecida como servidores multimídia.

As tarefas básicas de um servidor multimídia são: receber solicitações, por exemplo, de áudio e vídeo, dos vários terminais conectados e retornar as informações solicitadas, com garantia de desempenho; e servir múltiplos usuários simultaneamente e prover garantias de serviço. Algumas das características dos servidores multimídias são: tempo de resposta mínimo; capacidade de processamento rápida e alta taxa de acesso; confiabilidade e disponibilidade; alta capacidade de armazenamento; capacidade para minimizar o tráfego e a carga na rede; e suporte à interação com o usuário.

A televisão digital, baseado no projeto desenvolvido na disciplina de Sistemas Multimídias Distribuídos [FERR], pode ser vista na ilustração mostrada pela Figura 6.2.

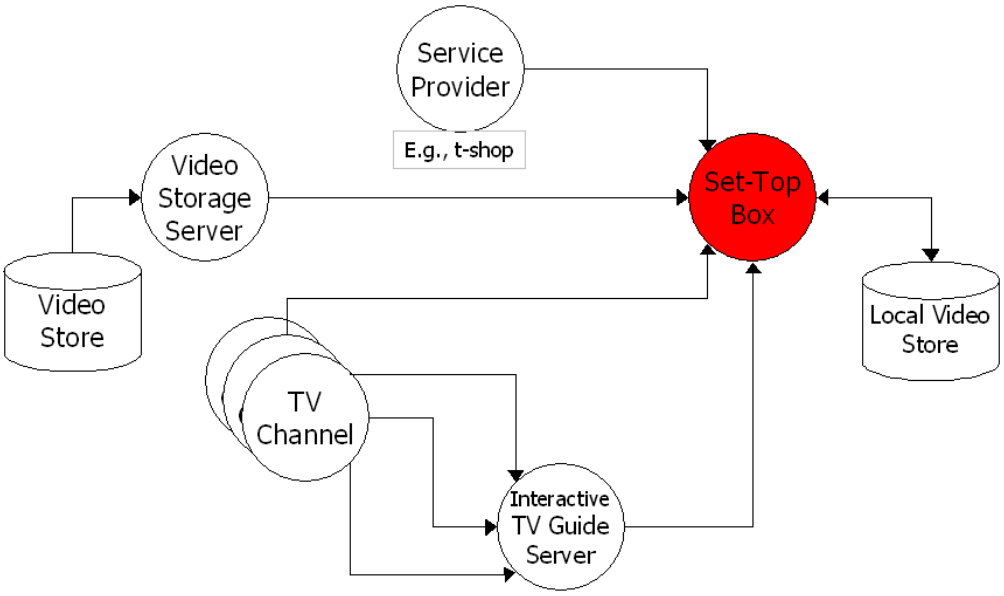

Figura 6.2 Estrutura do projeto de Televisão digital

Como pode ser observado na Figura 6.2, tem-se um grande sistema distribuído. Nesse, há uma camada de apresentação, baseada no controle remoto, na televisão e no guia interativo, pelos quais o usuário pode assistir ou interagir com a televisão digital. Além de haver um pequeno local de armazenamento, para que vídeos ou programas possam ser gravados num instante de tempo determinado pelo usuário.

Além disso, há na Figura 6.2, dois servidores multimídias. Um deles é o que armazena o vídeo para que esse possa ser enviado ao cliente. Esses vídeos podem ser a própria transmissão de um canal, ou filmes que possam estar armazenados para que um cliente compre e assista no horário desejado.

E o outro servidor multimídia é o provedor de serviços, que envia serviços para que o cliente possa interagir ou não com o sistema. Esses serviços podem ser serviços de bancos, serviços de informações sobre o clima, entre diversos outros que podem ser fornecidos pela televisão digital. Portanto, a televisão digital pode ser vista como um grande sistema distribuído, que também une sistemas e bancos de dados distribuídos na sua estrutura, como um todo.

7 Conclusões

Este trabalho discutiu uma maneira de como os sistemas e os bancos de dados distribuídos podem trabalhar de forma conjunta. Percebe-se que essa já é uma realidade em muitos locais, dentre esses, nos sistemas multimídias distribuídos, com a televisão digital. Este ramo da computação distribuída, como citado, é um ramo que cresce a cada dia, trazendo cada vez mais vantagens aos seus utilizadores. Para tanto, basta que os conhecedores desse grande ramo, o apresentem à sociedade e desenvolvam sistemas cada vez melhores, aproveitando todas as características possíveis desse ramo.

Na computação distribuída, dois novos ramos vêm tornando-se mais forte, dentre esses, um que já pode ser visto no dia-a-dia, é o ramo da computação móvel, que trabalha juntamente com a computação distribuída. Isto está gerando uma nova área de computação móvel e distribuída, utilizando middleware de computação móvel para realizar as suas funcionalidades. Além disso, os bancos de dados móveis também vêm crescendo, fazendo com que a união proposta nesse trabalho venha a ser estimulada.

Além desses ramos, uma área bastante crescente e que se utiliza dos dois ramos comentados, é a computação ubíqua. Esta tem por objetivo tornar o uso do computador mais agradável, fazendo que muitos computadores estejam disponíveis em todo ambiente físico, mas de forma invisível para o usuário. Ou seja, para que a computação ubíqua possa vir a ser uma realidade, a computação móvel e distribuída deve crescer em muito a funcionalidade da transparência, mas isso é um grande tema para um próximo trabalho.

Com esse trabalho, o autor conseguiu mostrar que muitas vantagens podem ser conseguidas quando se observa a inter-disciplinalidade, e que devemos buscar essas vantagens. Neste trabalho, tratando-se da área da computação distribuída, mas isso pode ser estendido para outras áreas, unindo diversos ramos da computação. Além de que num trabalho futuro, poder-se-ia aprofundar sobre esses dois ramos, analisando, testando e colocando em prática novas técnicas sobre essa união.

8 Anexos

Classe GuiCliente que acessa os serviços da aplicação LivroNet no servidor de Aplicação [ABRE1].

```
package LivroNet;
import java.awt.Frame;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.LinkedList;
import javax.swing.JOptionPane;
import LivroNet.dados.Livro;
import proxy.*;

public class GuiCliente {
   @SuppressWarnings("unchecked")
   public static void main (String args[]){
         try {
               Socket socket = new Socket("localhost",1099);
               ObjectOutputStream          out          =          new
ObjectOutputStream(socket.getOutputStream());
               ObjectInputStream           in           =           new
ObjectInputStream(socket.getInputStream());
               LinkedList list =  new LinkedList();
               list.add(0,"GetProxy");
               list.add(1,"Hello");
               out.writeObject(list);

               DadosProxy p = (DadosProxy) in.readObject();

               Proxy               m               =               new
Proxy(p.getObject(),p.getHost(),p.getPort());


               LivroNet        proxy                =        (LivroNet)
java.lang.reflect.Proxy.newProxyInstance(

     Thread.currentThread().getContextClassLoader(),                    new
Class[]{LivroNet.class},      (Proxy)m);

               Frame frame = new Frame();

     JOptionPane.showMessageDialog(frame,((Livro)proxy.procurarAutor("Br
uno Rodrigo").firstElement()).getTitulo());
         }
         catch(Exception e){
               e.printStackTrace();    }     }}
```

9 Referências

[ABRE1] ABREU, Bruno Rodrigo C.; FREITAS, Osmany Barros. *Projeto – Middleware baseado em Servidores de Aplicação.*
Disponível em:
http://www.cin.ufpe.br/~brca/arquivos/ta sd/

[ABRE2] ABREU, Bruno Rodrigo C.; MELLO, Lucas Freire; CAVALCANTE, Cristiano Campos. *MPEG e Interface de uma televisão digital*
Disponível em:
http://www.cin.ufpe.br/~brca/arquivos/smd/

[ABRE3] ABREU, Bruno Rodrigo C.;GODIM, Flavio Melo. *Projeto - Livro Net (Vendas de Livros on-line)*
Disponível em:
http://www.cin.ufpe.br/~brca/arquivos/pcd/

[ABRE4] ABREU, Bruno Rodrigo C. FREITAS, Osmany Barros. *Projeto – Venda de Livros on-line usando bancos de dados distribuídos.*
Disponível em:
http://www.cin.ufpe.br/~brca/arquivos/bd/

[COU94] COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. *Distribuited Systems: Concepts and Design.* Addison-Wesley, 1994.

[FERR] FERRAZ, Carlos A. G. *Sistemas Multimídias Distribuídos*
Disponível em:
http://www.cin.ufpe.br/~cagf/smd/tvdigital/

[FONS] FONSECA, Décio; SOUZA, Fernando Fonseca. *Banco de Dados Distribuídos e Móveis.*
Disponível em:
http://www.cin.ufpe.br/~if694/

[KHA94] KHANNA, Raman. *Distributed Computing: Implementation and management strategies.* Prentice Hall, 1994.

[OZS01] ÖZSU, M. Tamer; VALDURIEZ, Patrick. *Princípios de Sistemas de Banco de Dados Distribuídos*; tradução Vandenberg D. de Souza. Editora Campus Ltda., 2001.

[SIM96] SIMON, Errol. *Distribuited Information Syste: from clent/server to distribuited multimedia.* McGraw-Hill, 1996.

[ROSA] ROSA, Nélson. *Tópicos avançados em sistemas distribuídos*.
Disponível em:
http://www.cin.ufpe.br/~if749/

[TAN95] TANENBAUM, A. S. *Distributed Operating Systems*. Prentice Hall, 1995.

[WIKI] WIKIPEDIA. *Jonh Gage* e *Redes ATM*
Disponível em:
http://en.wikipedia.org/wiki/John_Gage
http://en.wikipedia.org/wiki/Asynchronous_Transfer_Mode

Assinaturas

_____

Carlos Ferraz

Orientador

_____

Fernando Fonseca

Co-Orientador

_____

Bruno Rodrigo Cunha de Abreu

Aluno