Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Suporte a Ambientes Virtuais Colaborativos de Larga Escala em Redes Peer-to-peer, com Gerenciamento de Distribuição de Dados em Conformidade com o Padrão HLA

ALUNO: NÉSTOR DANIEL HEREDIA VIEIRA

ORIENTADORA: REGINA BORGES DE ARAUJO

CO-ORIENTADOR: AZZEDINE BOUKERCHE

SÃO CARLOS - SP

ABRIL - 2006

Suporte a Ambientes Virtuais Colaborativos de Larga Escala em Redes Peer-to-peer, com Gerenciamento de Distribuição de Dados em Conformidade com o Padrão HLA

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Suporte a Ambientes Virtuais Colaborativos de Larga Escala em Redes Peer-to-peer, com Gerenciamento de Distribuição de Dados em Conformidade com o Padrão HLA

Néstor Daniel Heredia Vieira

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Departamento de Computação, da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.
Área de concentração: Processamento de Imagens e Sinais - PIS.

SÃO CARLOS - SP

ABRIL – 2006

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

H542sa

Heredia Vieira, Néstor Daniel.
Suporte a ambientes virtuais colaborativos de larga escala em redes peer-to-peer, com gerenciamento de distribuição de dados em conformidade com o padrão HLA / Néstor Daniel Heredia Vieira. -- São Carlos : UFSCar, 2007.
68 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2006.

1. Ambientes virtuais. 2. DDM (Gerenciamento de Distribuição de Dados). 3. HLA (Arquitetura de Alto Nível). 4. RTI (Infra-estrutura de Tempo de Execução) I. Título.

CDD: 006 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Suporte a Ambientes Virtuais Colaborativos de Larga Escala em Redes Peer-to-Peer, com Gerenciamento de Distribuição de Dados em Conformidade com o Padrão HLA"

NÉSTOR DANIEL HEREDIA VIEIRA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Profa. Dra. Regina Borges de Araujo
(Orientadora – DC/UFSCar)

Prof. Dr José Hiroki Saito
(DC/UFSCar)

Profa. Dra. Regina Melo Silveira
(LARC/USP)

São Carlos
Maio/2006

"A vida é como uma brisa, que ao fechar os olhos a sentimos e, quando percebemos, ela já passou, mas nunca hesitemos em fechar os olhos senão, nem mesmo a brisa nós a teremos aproveitado"

(Néstor Daniel Heredia Vieira)

Dedico este trabalho:

À minha família, amigos e namorada.

Agradecimentos

Primeiro, gostaria de agradecer à minha família, pai, mãe, avó e irmãos, pelo incentivo, apoio em todos os momentos e na qual sempre me espelhei.

À minha orientadora, profª Drª Regina Borges de Araújo, pela motivação, encorajamento e direção durante os meus dois anos de estudos na pós-graduação. Seu entusiasmo ilimitado pela pesquisa me fez trabalhar com muito esforço e dedicação. Gostaria de agradecer também ao prof. Dr. Azzedine Boukerche, da *University of Ottawa* - Canadá, pela oportunidade oferecida para trabalho conjunto, durante três meses, em seu laboratório.

À minha namorada, Giovana Lourenção, pela paciência durante as inúmeras noites e finais de semana os quais passei à frente do computador.

Aos meus colegas do LRV*net* – DC, UFSCar, Altieres, Diego, Fernando e Ricardo e aos do PARADISE – SITE, *Uottawa*, Anis Zarrad e Raed Jodeci, pelas discussões que contribuíram para a realização deste trabalho. Agradecimento ao aluno Richard Werner pelo companheirismo nos três meses no Canadá, ao amigo Carlos Higa pela ajuda prestada e aos meus amigos que foram companheiros nesta jornada e sempre apoiaram para a conclusão deste mestrado, Rodrigo, Fernando, Guilherme e Anderson.

Aos professores do Departamento de Computação da UFSCar, principalmente aos que tive a oportunidade de conhecer durante meus 2 anos de estudos.

Aos colegas do laboratório de Redes e ao professor Hélio que não mediram esforços para auxiliar no funcionamento do *cluster* do DC.

Às funcionárias do DC, UFSCar, Cristina e Mirian, pelos favores concedidos.

Resumo

Em Ambientes Virtuais Colaborativos de Larga Escala (AVCs-LE), ambientes 3D sintéticos extensos são compartilhados entre um número muito grande de usuários que colaboram entre si para atingir um objetivo comum. Nesses ambientes, os usuários precisam ter uma resposta imediata às suas ações, e estas devem ser refletidas nos ambientes de todos os usuários participantes. Assim, o sucesso da aplicação depende não apenas de processamento gráfico poderoso, mas também, da capacidade da rede na entrega das informações a tempo. Algoritmos de gerenciamento de distribuição de dados, em conformidade com o padrão *High Level Architecture / Run Time Infrastructure* (HLA/RTI) para simulações paralelas e distribuídas, vêm sendo utilizados na redução da latência e como limitantes e controladores do volume de dados trocados durante simulações. A distribuição de dados de AVCs-LE é realizada normalmente por um destes modelos de comunicação: cliente/servidor ou *Peer-to-peer*. Diferentemente do modelo cliente/servidor no qual a figura do servidor pode caracterizar um gargalo na rede, nas soluções *Peer-to-peer* as atividades estão distribuídas e conseqüentemente, suas aplicações são escaláveis, ou seja, suportam uma quantidade crescente de usuários.

Motivados por essas tecnologias amplamente estudadas e comumente utilizadas, procurou-se criar uma solução tolerante a falhas e de baixa latência, dentro dos requisitos de simulações de ambientes virtuais colaborativos de larga escala. Isso foi feito em conformidade com o padrão HLA, em que usuários, utilizando seu próprio computador conectado a uma rede Gnutella, possam participar de sessões de simulações sem as limitações encontradas em kits de suporte à simulação distribuída, como o Kit RTI existente. Em razão disso, foi proposta uma arquitetura com gerenciamento de distribuição de dados em conformidade com o HLA, padrão comumente utilizado em simulações paralelas e distribuídas e que utiliza o modelo de comunicação *Peer-to-peer* da rede *Gnutella* para disponibilização e compartilhamento de tais ambientes, sobre redes móveis *Ad-Hoc* (*Mobile Ad-Hoc Networks* - MANETs). Para tanto, realizaram-se simulações comparativas entre o Kit RTI desenvolvido pela Georgia Tech, com objetos de apenas uma velocidade, contra o Kit RTI Adaptado, com objetos de velocidade variada, em um *cluster*. A avaliação do tempo total de execução da federação, o número total de mensagens *multicast* geradas e o número total de mensagens trocadas pela grade deu origem a gráficos que mostraram aumentos consideráveis, principalmente, no tempo

e no número de mensagens trocadas pela grade. Da mesma forma, foi avaliada a técnica proposta, tolerante a falhas.

Abstract

In Large Scale Collaborative Virtual Environments – LSCVEs, extensive synthetic 3D environments are shared among a large number of users that collaborate towards the same objective. As all users in these environments need immediate answer for their actions and these actions must be sent to all participating users, the application success depends not only on a strong graphic processing but also in the capacity of the network to deliver information in time. Data distribution management algorithms in conformity with the *High Level Architecture / Run Time Infrastructure* (HLA/RTI) pattern for parallel and distributed simulations have been used to reduce latency and to limit and control the data amount exchanged during simulations. The data distribution of LSCVEs is generally made by one of these communication models: client/server or Peer-to-peer. Differently of the client/server where the server can be a bottleneck of the network, in Peer-to-peer solutions the tasks are distributed and consequently applications are scalable, i.e., support a crescent client number.

Motivated by these largely studied and commonly used technologies, a fault tolerant and low latency solution was searched, addressing the strict requirements of large scale collaborative virtual environments simulations. This was made in conformity with the HLA pattern, which users, using their own computer connected at Gnutella network, can participate in simulation sessions without the limitations found in kits that support distributed simulations like the RTI-Kit existent. For this reason an architecture was proposed with data distribution management in conformity with the HLA/RTI and that use the Gnutella Peer-to-peer communication model to make available and to share these environments over *Mobile Ad-Hoc Networks* (MANETs). Towards this propose simulations were made comparing the RTI-Kit developed by Georgia Tech with one speed objects and the RTI-Kit Adapted with varied speed objects in a *cluster*. The evaluation of the total time of the federation execution, the total number of the *multicast* messages generated and the total number of messages exchanged by the grid originated graphics that show up considerable increasing in the time and the number of messages exchanged by the grid mainly. In the same way the fault tolerant technique was evaluated.

Sumário

Lista de Figuras, Gráficos e Tabelas

Abreviações

2D	Bidimensional
3D	Tridimensional
AVC	Ambiente Virtual Colaborativo
AVCs-LE	Ambientes Virtuais Colaborativos de Larga Escala
DDM	*Data Distribution Management* - Gerenciamento de Distribuição de Dados
HLA	*High Level Architecture* – Arquitetura de Alto Nível
MANET	*Mobile Ad-Hoc Network* – Rede Móvel Ad-Hoc
P2P	*Peer-to-peer*
RTI	*Run Time Infrastructure* – Infra-estrutura de Tempo de Execução
RV	Realidade Virtual

1 Introdução

Ambientes Virtuais Colaborativos de Larga Escala (AVCs-LE) podem ser compartilhados por um grande número de usuários participantes em aplicações do tipo treinamento militar e policial, cidades virtuais, shoppings virtuais, dentre outras. O ambiente é habitado por entidades dinâmicas que podem ser dirigidas por usuários através de Avatar – representação gráfica do usuário no Ambiente Virtual, ou por simulação – *scripts* ou Inteligência Artificial. O ambiente contém também entidades estáticas como prédios, árvores e outros, que podem sofrer modificações ao longo da aplicação AVC. Pode-se necessitar baixar terrenos, ou outros tipos de dados, enquanto a aplicação AVC estiver em progresso, ou seja, um usuário, por meio de seu avatar, pode ir de uma área para outra que não esteja presente em sua máquina local, requerendo ser baixado de outro servidor ou de outra máquina, a área requerida. As ações do usuário no ambiente devem ser refletidas, em tempo real, tanto localmente (*feedback* visual para o terminal do usuário), como remotamente para todos os terminais de usuários remotos. Para o sucesso dos Ambientes Virtuais Colaborativos, a sensação de imersão é fundamental. Gráficos 3D e integração multimídia, especialmente som, melhoram a sensação de "imersão", presença do usuário no AVC, mas não são suficientes. Os usuários necessitam ter um retorno visual para suas interações, em um tempo aceitável. Com grande latência, o usuário começa a perder a sensação de imersão, diminuindo sua taxa de interação e eventualmente perde seu interesse pela aplicação. Assim, o sucesso da aplicação dependerá não apenas do potencial de processamento gráfico, mas também da capacidade da rede na entrega das informações a tempo.

Em AVCs-LE, onde o número de usuários pode ser muito grande e as áreas geográficas simuladas muito extensas, a manutenção da consistência do ambiente torna-se um grande desafio para os desenvolvedores de AVCs-LE pois a rede pode rapidamente ser sobrecarregada com o tráfego gerado por todas as entidades da aplicação AVC. Assim, soluções que reduzem o tráfego da rede são desejadas. Algoritmos de gerenciamento de distribuição de dados, em conformidade com o padrão HLA/RTI para simulações paralelas e distribuídas, vêm sendo utilizados na redução da troca de dados e manutenção da consistência do sistema.

A distribuição de dados de AVCs-LE é realizada normalmente por um destes modelos de comunicação: cliente/servidor ou *Peer-to-peer*. O modelo cliente/servidor é

o mais utilizado por ser mais simples de implementar e de gerenciar. Entretanto, este modelo é centrado em um servidor e está assim, sujeito a sofrer gargalos, aumentando o atraso no acesso ao serviço, às vezes chegando ainda, à negação do serviço. Esta desvantagem se agrava quando o número de usuários aumenta muito. Já o modelo de comunicação *Peer-to-peer* possibilita que dois ou mais nodos numa rede colaborem entre si, de igual-para-igual, sem a necessidade de uma coordenação central. Desta forma, problemas em um nodo não comprometem a aplicação. Mais ainda, as soluções *Peer-to-peer* reduzem o problema de congestionamento inerente ao modelo cliente/servidor, uma vez que não há ponto central no sistema, reduzindo assim a latência e, conseqüentemente, diminuindo o tempo de resposta do usuário participante. A rede *Gnutella* é uma implementação de código-fonte aberto do modelo de comunicação *Peer-to-peer* e foi adotada como parte da solução neste trabalho.

As motivações no estudo e possibilidade de integração destas tecnologias, que vêm sendo amplamente estudadas tanto no ambiente acadêmico como no comercial, levaram ao grande objetivo deste trabalho e à sua concretização e estão descritos a seguir.

1.1 Motivações e Objetivos

Em simulações 3D em rede, em particular nos Ambientes Virtuais Colaborativos de Larga Escala – AVCs-LE, é importante reduzir a troca de dados irrelevantes entre as entidades simuladas sem comprometer a exatidão da simulação, assim como é importante também garantir tolerância a falhas. Simulações distribuídas, em conformidade com o padrão HLA são tipicamente implementadas (na versão de software livre) em clusters, como é o caso do kit RTI (conjunto de bibliotecas de suporte ao desenvolvimento de simulações paralelas e distribuídas) e possuem várias limitações. Mais ainda, a interface para esses sistemas é tipicamente 2D e, até onde o autor pôde pesquisar, não há uma preocupação de visualização da simulação em diferentes dispositivos, inclusive móveis. Assim, podemos sintetizar as motivações para este trabalho como sendo:

- A possibilidade de levar simulações de larga escala, grande número de usuários e objetos simulados, em conformidade com o modelo HLA, comumente adotado para simulações distribuídas em ambientes militares e industriais, para o ambiente das redes IP e móveis *Ad-Hoc,* incluindo a Internet, através da rede *Peer-to-peer*;

- A superação das limitações atuais da Infra-estrutura de Tempo de Execução - RTI para que usuários possam entrar e sair de forma assíncrona da simulação;
- O foco do ambiente visual que, embora tenham sido tratadas apenas simulações 2D, tem como objetivo futuro o suporte a aplicações de ambientes virtuais 3D colaborativos do tipo shoppings virtuais, jogos multi-usuário, treinamento de forças policiais, militares, entre outras;
- A necessidade de filtragem dos dados trocados entre as entidades participantes da simulação para viabilizar a execução das simulações na Internet;
- A necessidade de tolerância a falhas de maneira a manter a consistência e execução da simulação mesmo com a entrada e saída de usuários.

Mediante essas motivações, o objetivo principal deste trabalho foi propor uma arquitetura de suporte a AVC-LE como simulações distribuídas, com as seguintes características:

- tolerância a falhas;
- implementação em redes *Peer-to-peer* (a rede *Gnutella* foi a rede *Peer-to-peer* adotada neste trabalho);
- gerenciamento de distribuição de dados *(Data Distribution Management - DDM)* em conformidade com o padrão HLA (comumente utilizado em simulações paralelas e distribuídas).

1.2 Organização do Documento

Este trabalho está organizado da seguinte maneira: no capítulo 2 são discutidos os Ambientes Virtuais Colaborativos de Larga Escala (AVCs-LE). No capítulo 3, são descritas as Simulações Paralelas e Distribuídas, o padrão para simulações distribuídas HLA *(High Level Architecture)*, a interface de implementação do HLA, RTI *(Run Time Infrastructure)*, os algoritmos de gerenciamento de distribuição de dados (DDM - *Data Distribution Management)* existentes no Kit RTI e o Kit RTI em si. No capítulo 4, é apresentado o modelo de rede *Peer-to-peer*, a descrição das redes *Peer-to-peer* descentralizadas e híbridas, os tipos de aplicações *Peer-to-peer* e a rede *Gnutella*. No capítulo 5, é descrita uma Arquitetura de Suporte a Ambientes Virtuais Colaborativos em redes *Peer-to-peer*, com gerenciamento de distribuição de dados em conformidade com o padrão HLA, onde são apresentados a arquitetura concebida e os experimentos realizados. No capítulo 6 são descritas as contribuições geradas, dificuldades encontradas e considerações finais, seguido de Referências Bibliográficas.

2 Ambientes Virtuais Colaborativos de Larga Escala

Ambientes Virtuais Colaborativos são sistemas baseados em computador que, suportam ativamente a colaboração e a comunicação humana, com interpretação de contexto realizada por computador. Uma classe particular destes ambientes com suporte a um grande número de usuários simultâneos distribuídos geograficamente são os Ambientes Virtuais Colaborativos de Larga Escala – AVCs-LE [JOS03].

2.1 Características de AVCs-LE

Dentre as características que são comuns aos Ambientes Virtuais Colaborativos de Larga Escala têm-se [GRE04, SIN99]:

- O suporte a múltiplos usuários geograficamente dispersos;
- A capacidade dos usuários de se comunicarem e colaborarem de diferentes maneiras. Isto porque há uma noção de espaço ou mundo – o ambiente virtual – no qual essa atividade está situada;
- A representação ou "embarcamento" de cada usuário dentro do mundo virtual onde se torna visível (e audível, etc.) para outros usuários por meio deste "embarcamento". Esta representação ou "embarcamento" é conhecida como Avatar;
- A autonomia (independência) de cada usuário para mover-se dentro do ambiente virtual. Esta movimentação é realizada através do Avatar;
- A capacidade de um participante de ver o comportamento dos outros no momento em que isto ocorre, ou seja, o ambiente virtual deve suportar interação em tempo real.

Além das características comuns dos AVCs-LE, os usuários também apresentam características comuns em suas ações dentro dos AVCs-LE (Segundo o projeto proposto pela Living World [LIN97]):

- Modificações na cena que podem ser iniciadas por qualquer cliente;
- Modificações que devem, potencialmente, ser sincronizadas em todos os clientes;

- Para todas as modificações, cada objeto, é responsável por refletir atualização localmente e enviar mensagens de sincronização para as instâncias deste mesmo objeto em todas as máquinas de usuários participantes, ou é um receptor destas mensagens de sincronização de outros usuários participantes e deve realizar a ação apropriada;
- Nem todas as modificações devem obrigatoriamente ser realizadas no instante em que são recebidas; a otimização do desempenho depende de saber-se qual objeto é mais provável ser alterado posteriormente;
- Nem todas as notícias de modificação precisam ser comunicadas; a otimização depende também de ter-se o conhecimento de quem necessita saber o quê e como.

2.2 Requisitos e Desafios de AVCs-LE

Os AVCs-LE têm sua importância em duas perspectivas [GRE04]. Primeiro, a partir da perspectiva de colaboração e segundo, a partir da perspectiva de redes de computadores.

Em termos de suporte à colaboração, os AVCs-LE têm as seguintes características:

- *Suporte à comunicação espacial "natural"*. O espaço tem um significado social que é importante para interações do mundo real. Isso porque, significantes elementos de comunicação, como a atenção do olhar e gestos, dependem de uma referência espacial. E, por isso, o espaço deve ser visto como um recurso para gerenciamento das atividades e interações [BEN95]. Todo dia, exemplos do uso do espaço como recurso, pode ser visto na flexível formação de grupos de conversação em encontros sociais informais. Fatores espaciais como posições e orientações relativas dos participantes, postura e velocidade de movimento, consciente e inconscientemente expressam informações tais como a disponibilidade para conversar, interesse e ações de intenção. Espaço compartilhado é um conceito fundamental em AVCs, que busca ser possível o seu uso para suportar alguma dessas funções e tornar acessível o uso social desse espaço.
- *Suporte à ciência periférica.* O usuário tem noção do que está a sua volta dentro do AVC mesmo que não esteja interagindo diretamente.

- *Unificação da comunicação e informação.* Um bom trabalho deve ser feito na área de visualização – criando, geralmente graficamente, representações da informação para realçar o acesso e a compreensão. Tais visualizações podem estar situadas dentro de um AVC tal que, os usuários tenham acesso à informação e, ao mesmo tempo, tenham facilidades para comunicação e cooperação. Tais ambientes, onde usuários virtualmente co-localizados com a informação, com as quais estão trabalhando, são conhecidos como Terrenos de Informação Povoados (TIP - Populated Information Terrains (PITs)) [BEN95].
- *Conservação da autonomia.* Dentro do AVC cada usuário mantém sua respectiva perspectiva e independência de movimento e atividade. Isso é um contraste quando comparado ao método puro, onde se tem como premissa "O que eu vejo é o que você vê" (WYSIWIS -What You See Is What I See), cujo suporte colaborativo é baseado em janelas 2D para multi-usuários [STE87] nos quais cada colaborador possui exatamente a mesma visão e, por isso, corresponde, de uma maneira mais precisa, a uma simples atividade.
- *Escalável, ie, suporte a um grande número de participantes.* A vantagem final dos AVCs, à partir da perspectiva de colaboração, é que há um claro potencial para suportar extremamente, grande número simultâneo de participantes. Nos efeitos do mundo físico, por exemplo, a perspectiva, a oclusão e a dispersão permitem tomar consciência e interagir com um subconjunto relativamente grande, contínuo e consistente, do ambiente total.

Em termos computacionais e, em particular, nas redes de computadores, os AVCs-LE têm muitos aspectos interessantes e desafiadores que motivam os seus exploradores e desenvolvedores, dentre eles:

- *O tráfego misturado de tipos.* Os AVCs-LE disponibilizam um contexto natural de maneira a combinar serviços de dados distribuídos com fluxo de áudio e vídeo, todos no contexto de interações de tempo real com usuários geograficamente dispersos. Para cada fluxo de informação, diferentes níveis de fidelidade podem ser requeridos por diferentes observadores ao mesmo tempo e pelo mesmo observador em tempos diferentes (dependendo da proximidade dentro do mundo virtual).
- *A participação em grupo.* Os AVCs-LE ao ter como um dos objetivos o suporte da colaboração entre grupos arbitrários de pessoas, faz que aumente o interesse

em apropriadas representações que dêem suporte para esse tipo de comunicação em grupos nos domínios computacional e de redes.

- *A conexão dinâmica e as associações.* Os AVCs-LE devem permitir que os usuários, à partir de motivações sociológicas, se movimentem continuamente nos e entre os mundos virtuais, assim como, tenham a liberdade de entrar e sair deles. Por isso, os requisitos da comunicação são altamente dinâmicos.
- *A segurança e a consistência*. Há várias saídas para a escolha apropriada de tecnologias para alcançar um equilíbrio adequado entre teorias corretas e oportunidades nos AVCs-LE (ex: ocupação gradual na entrega segura e regular de mensagens defronte à velocidade da interação dentro e com o mundo virtual).

2.3 Aplicações de AVCs-LE

Ambientes Virtuais Colaborativos de Larga Escala podem beneficiar uma infinidade de aplicações, como treinamentos colaborativos, shoppings virtuais, comunidades virtuais, aplicações de preparação para emergências, treinamento de forças policiais (para atuação em campos de futebol, shows etc), corpo de bombeiros, militares e outras. Esta seção apresenta alguns exemplos de aplicações de AVCs-LE.

2.3.1 Treinamentos Colaborativos

Ambientes colaborativos oferecem meios de modificar objetos no interior de um ambiente virtual, bem como criar novos objetos. Todos usuários têm os mesmos privilégios e capacidades dentro do ambiente. Bons exemplos desta classe de aplicações são os treinamentos médicos e de manipulação de maquinário cuja compra antes do treinamento seria muito dispendiosa. Na Figura 1 pode-se ver o treinamento colaborativo de nova técnica cirúrgica.

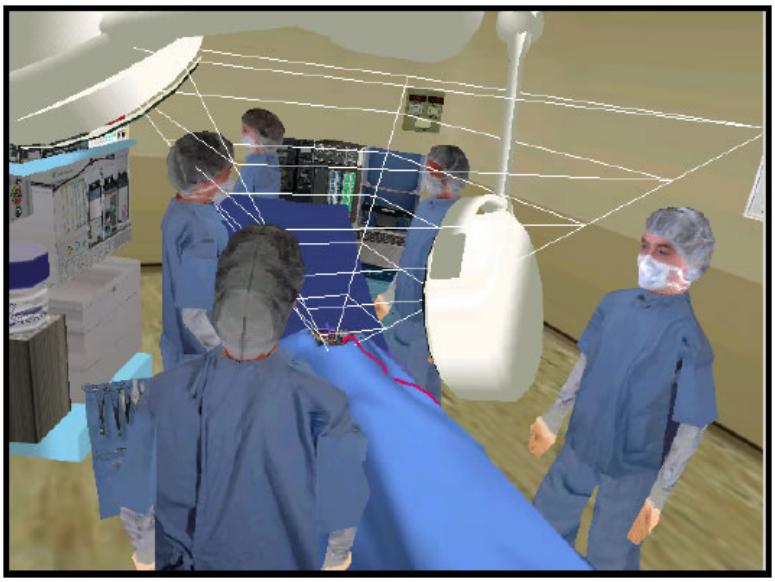

Figura 1. Treinamento Colaborativo.

2.3.2 Shopping Virtual

Um usuário pode entrar em uma loja virtual e interagir com produtos que estão à venda. Caso ele queira mais detalhes sobre o produto, poderá assistir a algum vídeo, ler um texto ou mesmo escutar as declarações dos clientes que já compraram aquele produto. Ele pode, ainda, chamar um vendedor, representado por um outro *avatar* ou um objeto simulado. A comunicação entre o cliente e o vendedor poderá ser feita através de texto, áudio ou vídeo. Na Figura 2 pode-se ver um exemplo de um shopping virtual utilizando AVC.

Figura 2. Shopping virtual.

2.3.3 Comunidades Virtuais

As comunidades virtuais diferem das outras aplicações de ambientes virtuais colaborativos pelo fato de os ambientes existirem ao longo de um período determinado e de as aplicações existirem de forma permanente. Devido às comunidades virtuais poderem ser habitadas por uma grande quantidade de usuários, verdadeiras cidades podem ser formadas, com a existência de prefeito, polícia e centro de informações, tal qual as cidades reais. Cada usuário pode ter um espaço próprio, onde possa realizar alterações, conforme as leis prescritas na "cidade virtual". A Figura 3 mostra um exemplo de uma comunidade virtual com três usuários *on-line*.

Figura 3. Comunidade virtual.

2.3.4 Treinamentos para Emergências

Os ambientes de treinamentos para emergências possuem diversas aplicabilidades, desde treinamentos de corpos de bombeiros, policiais, militares até ações em casos de desastres naturais e outras. Estas aplicações podem tanto existir para treinamentos rotineiros como serem implementadas para um treinamento específico. Neste tipo de aplicação, os objetos não avatares, ou seja, que não são 'usuários', têm grande participação na simulação, sendo assim, menos passivos que os mesmos tipos de objetos em aplicações de treinamentos colaborativos por exemplo. Na Figura 4 pode-se ver um ambiente de treinamento militar para controle de guerras civis.

Figura 4. Treinamento para Emergência.

2.4 Ambientes Virtuais Colaborativos de Larga Escala Existentes

Várias pesquisas vêm sendo realizadas em Ambientes Virtuais Colaborativos de Larga Escala. A seguir, são apresentadas algumas aplicações desenvolvidas.

O *Massive* é um implementação do modelo espacial de interações. Consiste de um sistema distribuído multi-usuários de Realidade Virtual – RV, objetivando aplicações de teleconferência, mas não de aplicações de propósito geral de RV [BEN95]. O principal conceito envolvido no controle interativo entre objetos é a "ciência". A ciência de um objeto em relação a outro quantifica a importância subjetiva ou relevância do outro objeto. Essa pode corresponder ao volume de um canal de áudio, ou um nível de detalhes de uma renderização gráfica. Em geral, mais atenção (e mais largura de banda e computação) será voltada a objetos com valores maiores de ciência.

Duas versões do *Massive* já foram desenvolvidas. Primeiramente o MASSIVE-1, um sistema mais limitado, dedicado a teleconferência em realidade virtual, suportando combinações de interações via texto, gráficos e áudios em tempo real sobre protocolos *unicast* de rede. O MASSIVE-2 é mais recente e de propósito mais geral. Adiciona à teleconferência um *framework* de desenvolvimento de aplicações gerais e emprega uma combinação de protocolos de rede do tipo *unicast* e *multicast*. Em suas duas versões o Massive foi implementado baseado no modelo de comunicação cliente/servidor, sendo necessário um servidor bastante robusto.

O projeto COVEN (*Collaborative Virtual Environments* – Ambientes Virtuais Colaborativos) é focado no desenvolvimento de um serviço computacional de 'teleworking' e presença virtual. O grande objetivo deste projeto é prover as facilidades necessárias de maneira a ajudar em futuros sistemas cooperativos de *teleworking*, ou

seja, a plataforma pode ser usada como base para futuras aplicações AVCs. Este projeto é baseado no modelo de comunicação cliente/servidor com o servidor implementado em *cluster Beowulf*. Uma aplicação já desenvolvida, a *Viagens em Londres* [STE99], consiste de um demonstrador que viaja dentro de um modelo virtual de Londres onde, participantes podem planejar rotas, investigar informações turísticas e se encontrar com acompanhantes turísticos ou guias.

Oliveira et al. (2000) [OLI00] exploram os AVCs para o Tele-Treinamentos Industrial e para o Comércio Eletrônico. Nos dois casos os usuários, representados por avatares, podem entrar nos mundos virtuais e então manipular e interagir com os objetos como no mundo real. Para os de propósito de treinamento, espera-se reduzir os gastos financeiros e de tempo enquanto treinam remotamente usuários/engenheiros para operar determinados equipamentos, tais usuários podem realizar também, algumas tarefas sob a supervisão de um treinador. Ao invés de trabalharem com objetos físicos, estes são representados por objetos virtuais que podem ser colocados em um ambiente virtual acessível a vários usuários. Os usuários, representados por seus avatares, podem então manipular e interagir com os objetos como no mundo real e ganhar importante experiência e treinamento antes de utilizar o equipamento real. Já no AVC de Comércio Eletrônico , o cliente deve ser capaz de ver os modelos que ele/ela deseja comprar. Se questões precisam ser respondidas, um Agente Inteligente pode aparecer e ajudar o usuário com informações detalhadas e caminhar entre as características dos objetos virtuais. Pode-se estender o mundo virtual tornando-o similar a um verdadeiro *shopping center*, o que deve facilitar a adaptação de usuários não acostumados às interfaces baseadas em navegador. Desta maneira busca-se disponibilizar ao usuário uma experiência mais proveitosa enquanto realiza compras *online*. Os dois ambientes possuem uma arquitetura comum baseada no modelo de comunicação cliente/servidor com o servidor implementado em *cluster*.

O StringCVE [MOL02, MOL03] utiliza o motor de um jogo multi-jogadores para desenvolver uma aplicação AVC para coordenar o *design* arquitetural e as críticas de *design*. A ênfase inicial foi prover baixo custo mas, de forma rica, tornou-se alternativa à Realidade Virtual comercial facilitando assim os estúdios virtuais de *design* na educação arquitetural. A proposta foi dar suporte aos estágios iniciais de design onde equipes possam colaborar e avaliar iterações a um relativo baixo nível de detalhes. Dessa maneira para tornar-se uma parte útil do ciclo de *design*, é crucial a existência de uma fácil troca de dados (geométrica e informações embutidas) do motor

de jogo para um software CAD 3D. Baseado no modelo de comunicação cliente/servidor, este projeto necessita de um servidor bastante robusto.

O CIMBLE – CADETT Ambiente Multi-Usuário Interativo de Aprendizado de Negócios [SEI99] disponibiliza um ambiente virtual distribuído e colaborativo para conduzir treinamento de equipes. Inicialmente desenvolvido como uma arquitetura cliente/servidor mas foi então redesenhado em conformidade com a HLA para melhorar o desempenho e a usabilidade. É baseado no princípio da constituição de equipes, formadas através da coordenação e cooperação na solução de problemas, com atividades que podem ser realizadas remotamente, se o sistema que integra os participantes for atraente, robusto e confiável. O CIMBLE é direcionado principalmente no aprendizado e na prática de habilidades básicas como comunicação e tomada de decisões.

O mWorld [DIA97] é um ambiente virtual colaborativo 3D multi-usuário. A aplicação é guiada através de edição e animação 3D em tempo real. Usuários podem pegar, adicionar, modificar e excluir objetos, mudar iluminação e simular objetos deformados, assim como, navegar pelo ambiente. O controle de sessão e o serviço de comunicação de dados é feito por uma interface de serviço comum. Apesar de possuir uma implementação por camadas e uma interface de aplicação clara, a implementação corrente não inclui mecanismos para reduzir a carga da rede. Além do mais, é baseada no modelo de comunicação cliente/servidor.

2.5 Considerações Finais

Ambientes Virtuais Colaborativos de Larga Escala (AVC-LE) podem ser definidos como simulações que suportam ambientes sintéticos, gerados por computador, que são compartilhados por um número muito grande de entidades (usuários, representados por avatares, e objetos simulados) que interagem entre si e com o ambiente (que pode representar áreas geográficas extensas). Várias características e requisitos de AVC-LE foram descritos neste capítulo, mostrando que os requisitos, como latência, escalabilidade, sincronização, dentre outros, são estritos e desafiadores. A redução de tráfego trocado entre as entidades participantes de AVC-LE atende a alguns dos requisitos mencionados. Técnicas de gerenciamento de distribuição de dados são utilizadas para a redução de tráfego e conseqüente diminuição de latência em ambientes de simulação distribuída. No próximo capítulo são introduzidos os conceitos de Simulações Paralelas e Distribuídas, bem como o Modelo HLA de Referência, o RTI e o DDM que serão utilizados para atender os requisitos de AVCs.

3 Simulações Paralelas e Distribuídas e o Modelo HLA de Referência

O termo *simulação* refere-se à realização alternativa de um determinado sistema maior e mais complexo, de maneira a analisar e entender o comportamento deste sistema frente a várias ações ou decisões alternativas [SEI95]; tal realização é também chamada de modelo. Por exemplo, pilotos de aviões são treinados em simuladores de vôo – dispositivos físicos que recriam a resposta da aeronave a várias ações que o piloto pode tomar, permitindo assim, que os pilotos aprendam a controlá-la.

As simulações possuem vantagens e desvantagens que devem ser levadas em consideração para avaliar a criação ou não de um modelo e estão descritas a seguir [CEN01]:

- Uma vez que um modelo esteja construído, ele pode ser utilizado repetidas vezes para várias análises;
- Os dados de uma simulação são usualmente mais baratos que os dados do sistema real;
- Os métodos de simulação são usualmente mais fáceis de aplicar que os métodos analíticos;
- Os modelos de simulação não requerem tantas hipóteses simplificadas de métodos analíticos;
- Os sistemas de simulação podem ser custosos de serem construídos;
- Pela sua característica estatística são necessárias várias execuções do mesmo modelo para obter dados confiáveis;
- Uma vez que a metodologia esteja bem entendida, existe a tendência de usar a simulação mesmo que técnicas analíticas fossem suficientes.

Uma classe específica das simulações de interesse neste trabalho são as simulações paralelas e distribuídas e estão relacionadas com a distribuição da execução de um programa de simulação através de múltiplos computadores [FER95].

Simulação paralela é a execução de um simples programa de simulação em uma coleção de processadores fortemente ligados. Por exemplo uma plataforma computacional multi-processadora com memória compartilhada (Figura 5 (a)).

Simulação distribuída é a execução de um simples programa de simulação em uma coleção de processadores unidos de maneira livre. Isto inclui execuções em

computadores distribuídos geograficamente e interconectados via uma rede de larga escala como a Internet (Figura 5 (b)).

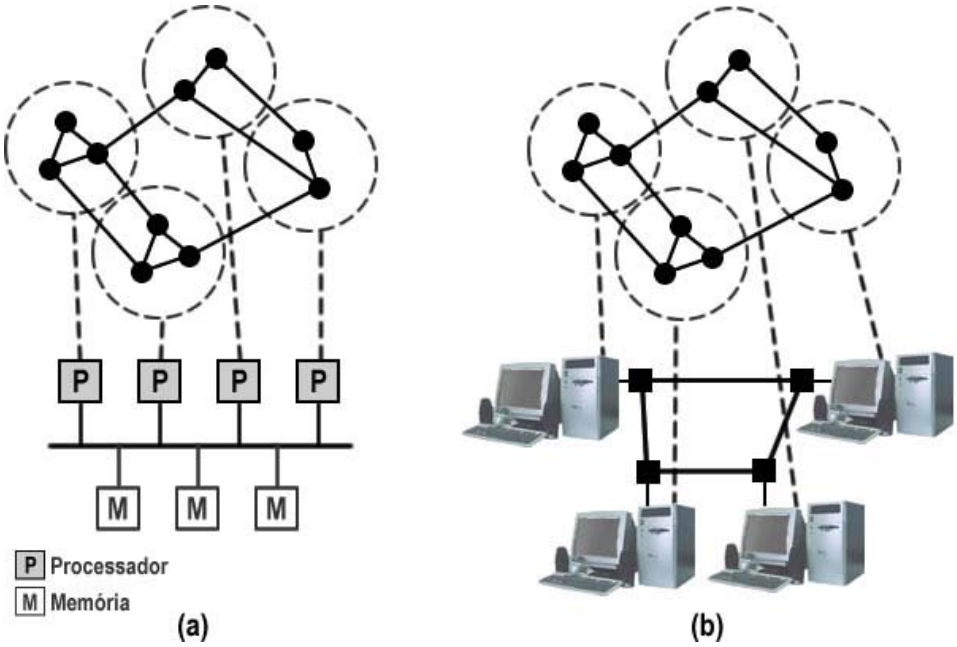

Figura 5.(a) Simulação Paralela. **(b)** Simulação Distribuída.

Existem duas principais categorias de simulações de interesse. A primeira são simulações primariamente usadas para análises, ou seja, para avaliar *designs* alternativos ou policiamentos controlados de um sistema complexo, por exemplo, uma rede de tráfego aéreo. Nestas simulações, o principal objetivo é computar resultados da simulação o mais rápido possível de maneira a melhorar o desempenho da ferramenta de simulação. A segunda categoria é a utilizada pra criar ambientes virtuais nos quais humanos e/ou dispositivos de hardware são acoplados. Tais ambientes são amplamente utilizados para treinamento, entretenimento, e teste de desempenho de dispositivos.

Sistemas de simulação paralela e distribuída podem prover benefícios substanciais para essas aplicações de diversas maneiras [FUJ01]:

- Os tempos de execução de sistemas analíticos pode ser reduzido subdividindo uma larga simulação computacional em várias sub-computações que podem ser executadas concorrentemente;
- Sistemas críticos, nos quais o tempo de resposta precisa ser muito rápido de maneira que importantes decisões possam ser tomadas, são beneficiados com a redução do tempo de resposta com a execução computacional subdividida;

- Ambientes virtuais cujas simulações devem ser executadas em tempo real, ou seja, um simulador deve ser capaz de simular um segundo de atividade em um segundo de tempo de um relógio comum de maneira que o ambiente virtual pareça realístico nas suas mudanças como o sistema atual, também são beneficiados com a redução do tempo de resposta com a execução computacional subdivida;
- Técnicas de simulação distribuída podem ser usados para criar ambientes virtuais distribuídos geograficamente, permitindo interação de humanos e/ou dispositivos como se estivessem co-localizados. Tais ambientes têm óbvios benefícios em termos de conveniência e redução de custos de viagens;
- Outro benefício de utilizar multiprocessamento é o aumento da tolerância a falhas. Se um processador falhar, pode ser possível continuar a simulação desde que elementos críticos não estejam presentes no processador falho.

Este conjunto de benefícios providos pelas simulações paralelas e distribuídas, auxiliam uma vasta gama de simulações de ambientes virtuais incluindo os de grandes áreas geográficas e com a participação de muitos usuários, os ambientes virtuais colaborativos de larga escala. Além do mais, algorimos de gerenciamento de distribuição de dados em conformidade com o modelo HLA de referência auxiliam, principalmente, na filtragem dos dados trocados desnecessariamente entre usuários participantes de ambientes virtuais colaborativos. Este modelo HLA de referência, assim como, a implementação de sua especificação de interface, o gerenciamento de distribuição de dados e o Kit RTI utilizado em nossos experimentos estão descritos na próxima seção.

3.1 Modelo HLA de Referência

A Arquitetura de Alto Nível (*High Level Architecture* – HLA) é uma arquitetura de propósito geral para reuso e interoperabilidade de simulações. Foi desenvolvida sob a liderança do Gabinete de Simulação e Modelagem de Defesa (*Defense Modeling and Simulation Office* – DMSO) para suportar o reuso e a interoperabilidade através do grande número de diferentes tipos de simulações desenvolvidas e mantidas pelo Departamento de Defesa Norte Americano (*United States Department of Defense* – DoD). As definições bases da HLA foram completadas em 21 de agosto de 1996. Foi aprovada pela *Under Secretary of Defense for Acquisition and Technology*

(USD(A&T)) como o padrão técnico de todas as simulações do DoD em 10 de setembro de 1996. A HLA foi adotada como a Facilidade para Sistemas de Simulação Distribuída 1.0 pelo Grupo de Gerenciamento de Objetos (*Object Management Group* – OMG) em Novembro de 1998 e atualizado em 2001 para refletir as mudanças resultantes da padronização comercial da especificação pelo IEEE (*Institute of Electrical and Electronic Engineers*). A HLA foi aprovada como um padrão aberto pelo IEEE – IEEE Standard 1516, em Setembro de 2000.

Com a intenção de ser aplicável a uma grande gama de simulações, nas mais diversas áreas, a HLA é baseada na premissa de que não é possível uma simulação satisfazer a todos os usos e usuários. O objetivo da HLA é prover uma estrutura com a qual o reuso das capacidades disponíveis em diferentes simulações possam ser aproveitadas, reduzindo o custo e o tempo requerido para criar novo ambiente, para nova finalidade.

Uma simulação individual ou um conjunto de simulações desenvolvidas para um propósito podem ser utilizadas em outra aplicação através do conceito HLA de Federação: um conjunto composto de simulações interagindo. E cada uma destas simulações representa um Federado [DAH97b]. Um exemplo de federação está representado na Figura 6 da próxima subseção.

A HLA não prescreve nenhuma implementação específica, nem estipula o uso de softwares ou linguagem de programação em particular, e sim, disponibiliza um *framework* para o desenvolvimento de arquiteturas de sistemas específicos de maneira que, com o avanço da tecnologia, novas e diferentes implementações possam ser possíveis com o uso deste *framework*. A HLA é formalmente definida por três componentes [DAH97b] : (1) o molde de modelos de objetos, (2) as regras HLA e (3) a especificação da interface. Este último componente é tratado nas seguintes subseções.

3.1.1 RTI (*Run Time Infrastructure*)

A especificação da interface HLA é implementada pela Infra-estrutura de Tempo de Execução (*Run Time Infrastructure* – RTI) e descreve os serviços de tempo de execução disponibilizados para os federados pelo RTI e pelos federados para o RTI [DAH97a]. Existem seis classes de serviços, a saber:

1. O serviço de *Gerenciamento de federação* oferece as funções básicas requeridas para criar e operar uma federação.

2. O serviço de *Gerenciamento de declaração* oferece eficiente gerenciamento de troca de dados por meio da informação fornecida pelos federados definindo os dados que eles irão disponibilizar e que irão requerer durante a execução da federação.
3. O serviço de *Gerenciamento de objetos* oferece a criação, exclusão, identificação e outros serviços no nível dos objetos.
4. O serviço de *Gerenciamento de posses* oferece a transferência dinâmica de posse de objetos/atributos durante a execução.
5. O serviço de *Gerenciamento de tempo* oferece sincronização da troca de dados da simulação em tempo real.
6. O serviço de *Gerenciamento de Distribuição de Dados* (*Data Distribution Management* – DDM), que oferece um roteamento de dado através dos federados durante a execução da federação.

A especificação da interface HLA define a maneira como esses serviços serão acessados, tanto funcionalmente como na interface de programação. Uma esquematização da interoperabilidade com a participação de dois simuladores, um monitor passivo e participantes de tempo real é apresentada na Figura 6. O serviço de Gerenciamento de Distribuição de Dados será o foco das subseções seguintes.

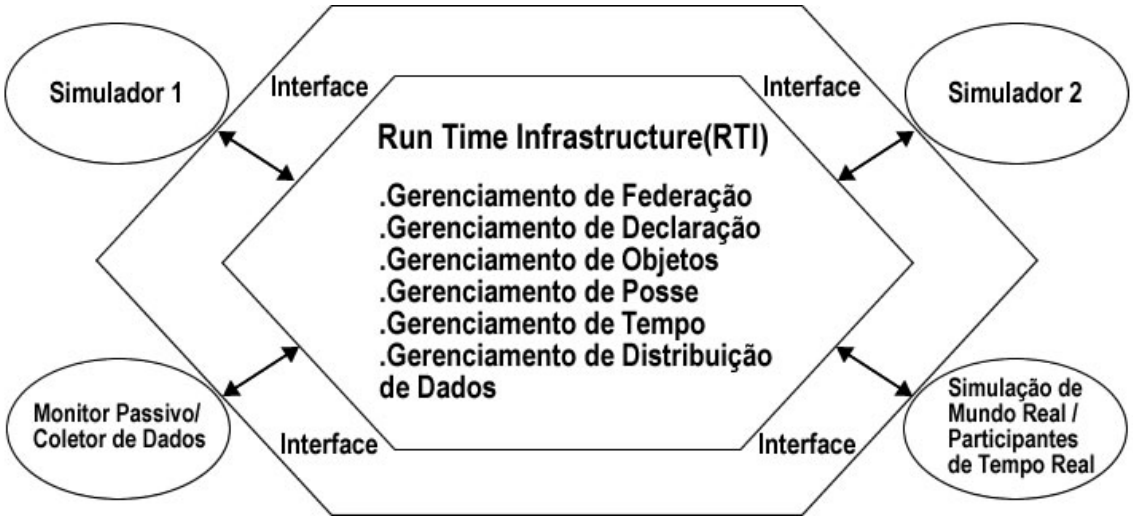

Figura 6. Federação: Interoperabilidade na HLA.

3.1.2 DDM (*Data Distribution Management*)

Os grandes objetivos do Gerenciamento de Distribuição de Dados (*Data Distribution Management* – DDM) são de limitar e controlar o volume de dados trocados durante uma simulação, reduzindo assim, os requisitos de processamento dos *hosts* da simulação, por meio do envio de informações de eventos e de estado apenas para aquelas aplicações que requerem tais informações. A observação fundamental sobre essa estratégia está nos objetos do mundo real que devem se interessar por apenas uma fração dos objetos que os cercam. Na Figura 7 pode-se observar um avião espião onde, em seu campo de visão está o Esquadrão A. Apesar da existência do Esquadrão B, ele não está interessado nas informações geradas por ele, pois está fora de seu campo de visão. Então, o avião espião receberá as informações geradas apenas pelo Esquadrão A.

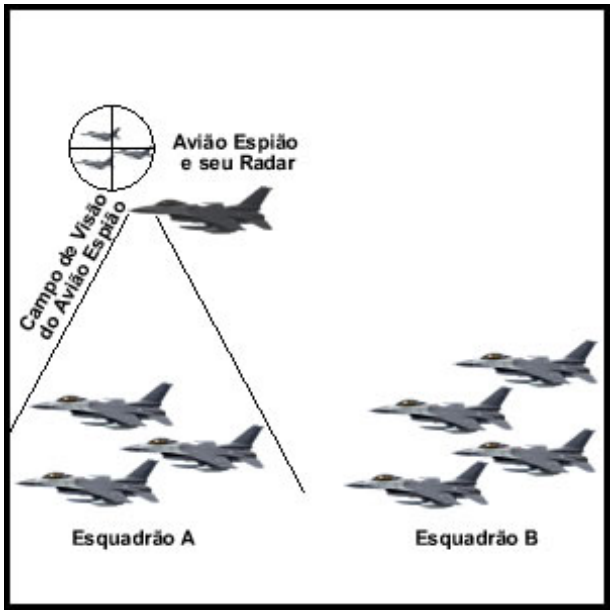

Figura 7. Região de Interesse do Gerenciamento de Distribuição de Dados – DDM.

Dentre as características que necessitam ser tratadas para a simulação dos ambientes virtuais colaborativos têm-se o espaço de roteamento, as regiões de subscrição e de publicação, a região de intersecção e a comunicação dos grupos por meio do *multicast*.

3.1.2.1 Espaço de Roteamento

Espaço de Roteamento é um sistema de coordenadas multidimensionais [DEP98]. Representa o mundo virtual no qual a simulação ocorre. É interessante ressaltar que ele não representa apenas coordenadas de localizações geográficas. Por exemplo, numa simulação de aviões, pode-se ter um sistema com cinco dimensões, sendo três delas representando localização geográfica num espaço 3D e outras duas um tipo de bomba com que o avião está equipado e a velocidade máxima que ele alcança. A escolha do número de dimensões que terá um espaço de roteamento e o que essas dimensões representarão são decisões importantes que deverão ser tomadas quando se está desenvolvendo uma simulação que usará algoritmos DDM.

3.1.2.2 Regiões de Subscrição, de Publicação e de Interesse

As entidades, objetos ou federados, possuem diferentes interesses nas regiões com as quais estão interagindo. Diante dessas diferenças, as seguintes terminologias são utilizadas para referenciar as regiões:

- *Região de Subscrição.* É uma abstração que referencia um conjunto de dados do mundo no qual uma entidade, objeto ou federado, está interessado;
- *Região de Publicação.* É uma abstração que referencia um conjunto de dados que uma entidade simulada está disponibilizando para o mundo;
- *Região de Interesse.* É uma abstração que referencia os dois interesses, Subscrição e Publicação, não importando exatamente qual.

3.1.2.3 Região de Intersecção e Comunicação dos Grupos por meio do *Multicast*

Uma *Região de Intersecção* é definida pela sobreposição de uma região de publicação com uma região de subscrição. A partir da identificação de tal região, um segundo tipo de comunicação necessita ser estabelecido para a troca de dados entre publicadores e subscritores. Este segundo tipo de comunicação *inter-host* é geralmente realizado utilizando a tecnologia de *multicast*, onde tem-se transmissão de uma origem para vários destinos. Mas, a identificação dessa área, e a subseqüente troca de dados de

estado, não é um processo trivial, pois as entidades envolvidas podem ser simuladas em diferentes *hosts*. Desta dificuldade, extraem-se dois problemas fundamentais que o DDM deve resolver: (1) Como uma intersecção de regiões é identificada precisamente? E (2) Como serão realizadas as trocas de dados entre publicadores e subscritores?. É na solução destes problemas que estão as grandes diferenças entre uma técnica e outra presentes na subseção seguinte.

3.1.2.4 Técnicas Existentes de Gerenciamento de Distribuição de Dados

As técnicas de gerenciamento de distribuição de dados – DDM podem ser classificadas em dois grandes grupos: Baseada em Região (*Region-Based*) [FUJ00, HOK98, TAN00a, TAN00b] e Baseada em Grade (*Grid-Based*) [ABR98, BER98, BOU02, BOU05, CAL94, MOR97]. A seguir serão apresentadas algumas técnicas baseadas em região, grade e uma combinação destas.

Baseada em Região (*Region-Based*)

Esta técnica compara diretamente todas regiões de publicação com todas de subscrição de modo a encontrar as intersecções, este processo é conhecido como *matching* [HOK09]. Dessa maneira essa técnica está associada a dois *overheads*: computacional e de custo de comunicação.

O primeiro problema é gerado pela ocorrência freqüente do teste de *matching*, que deve ocorrer sempre que uma região de publicação é modificada, porque necessita ser comparada com todas as regiões de subscrição. Por outro lado, quando uma região de subscrição é criada, deve ser comparada com todas as regiões de publicação de diferentes federados, gerando o segundo problema, pela comunicação *inter-host*, que necessita comparar regiões de diferentes federados.

Para possibilitar o funcionamento desta técnica, o sistema DDM deve ter, como extensão da federação, um Coordenador_DDM, para o qual todos federados enviam suas informações de interesse. Ele coleta as informações de interesse numa base de dados e realiza os *matchings* através das regiões de publicação e de subscrição, e comunica os resultados de volta aos federados, através de mensagens, convidando-os a se unir ou deixar os grupos *multicast*.

A presença de um Coordenador_DDM e uma base de dados podem influenciar o desempenho e a escalabilidade do sistema DDM, tornando-se o gargalo da funcionalidade, sendo por isso a maior desvantagem dessa técnica. Algumas táticas têm sido propostas, na literatura, de maneira a eliminar esse gargalo, como exemplo, o uso de agentes móveis inteligentes [TAN00b].

Na Figura 8 pode-se observar o funcionamento desta técnica. O avião espião subscreve seu interesse enviando ao Coordenador_DDM. Cada avião do Esquadrão A envia seu interesse de publicação para o Coordenador_DDM. O mesmo ocorre com o Esquadrão B. O Coordenador_DDM efetua o *matching* para encontrar as intersecções. Como no exemplo existe uma intersecção, um grupo *multicast* é criado, e tanto o avião espião quanto os aviões do Esquadrão A são convidados a unirem-se ao grupo *multicast*. Os dados são então trocados entre as partes. Numa nova movimentação, o processo é repetido.

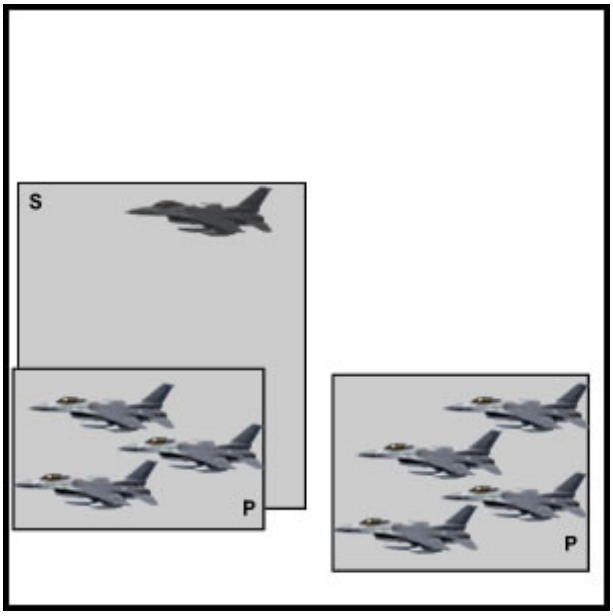

Figura 8. Baseada em Região (*Region-Based*), *matching* entre Esquadrão A e avião espião.
S – Subscritor, P – Publicador.

Baseada em Grade (*Grid-Based*)

Um sistema RTI usando a técnica baseada em grade mapeia cada região de interesse em uma grade multidimensional sobreposta ao terreno simulado, o qual representa o espaço de roteamento. Na Figura 9, pode-se observar este mapeamento onde, a subscrição do avião espião é mapeado nas células 1 e 3, a publicação de cada

avião do Esquadrão A é mapeada na célula 3 e similarmente as do Esquadrão B são mapeadas na célula 4. Células onde temos tanto publicação quanto subscrição, representam uma intersecção, verificada na célula 3.

Nos mais variados tipos de simulações é mais comum encontrarem-se regiões de subscrição que cobrem múltiplas células. Por outro lado, as regiões de publicação na maioria dos casos, como por exemplo, simulações militares, são pequenas. Projetistas de simulações podem forçar o requisito de que as publicações sejam confinadas a uma única célula que se chama de ponto de publicação, pois estão limitadas a um único ponto ou a uma pequena região.

O processo de sobrepor uma grade no terreno pode ser desempenhado por um componente RTI em cada *host*. Cada um realiza o mapeamento de região-células independente dos outros *hosts*, já que o número e o tamanho da grade de células já foi determinado antes do início da simulação e permanece constante ao longo da execução da simulação. Desta maneira, um Coordenador_DDM centralizado não é necessário, fazendo assim com que a técnica Baseada em Grade seja relativamente escalável, que é sua vantagem primária.

A maior desvantagem da técnica Baseada em Grade é que a precisão dos *matchings* tende a ser menor quando comparada com a Baseada em Região. Isso porque o mapeamento das regiões de interesse para as células da grade, podem não ser exatas. Assim, intersecções supérfluas podem ser geradas em algumas células e os respectivos publicadores podem enviar dados desnecessários aos subscritores.

Existem algumas maneiras de tentar lidar com essa imprecisão. Pode-se citar a filtragem dos dados desnecessários feitas pelo receptor, a utilização de apenas pontos de publicação e a utilizarão de margem de erro maior que o comprimento da célula. Desse modo, o tamanho das células da grade torna-se muito importante. Células pequenas produzem grande número de grupos *multicast* associados a cada região, enquanto células grandes produzem um menor número de grupos que requerem atualizações das regiões de interesse com mais freqüência.

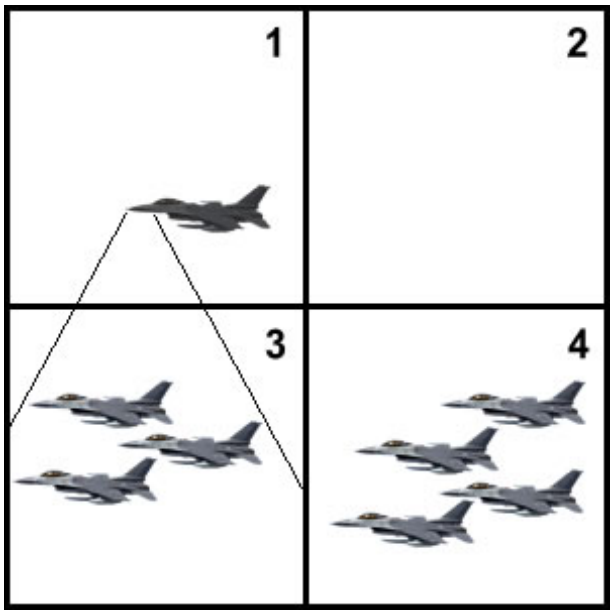

Figura 9. Baseada em Grade (*Grid-Based*), grade sobreposta para mapeamento de região-células.

Todas as técnicas Baseadas em Grade diferem entre si na maneira como utilizam as estratégias de comunicação *multicast* para transferir, de maneira eficiente, dados entre os *hosts*, que, por sua vez, produzem dados para os que consomem.

Fixa Baseada em Grade (*Fixed Grid-Based*)

Nesta técnica, um grupo *multicast* é associado a cada célula da grade desde a inicialização do sistema. Enquanto a simulação é executada, um componente RTI fica rodando em cada *host*, mapeia as regiões de interesse daquele federado (está sendo assumido um federado por *host*) para as células da grade e o federado se une aos grupos *multicast* que foram pré-definidos para aquelas células.

Uma característica única da técnica Fixa Baseada em Grade é que a intersecção das regiões é detectada apenas indiretamente, ou seja, sem o teste *matching*. Desta maneira, a comunicação entre federados é minimizada drasticamente. Os dados serão propriamente transferidos sempre que uma intersecção estiver presente porque, tanto publicadores como subscritores terão se unido ao mesmo grupo *multicast*. Na Tabela 1 mostra-se o funcionamento desta técnica, a partir do exemplo da Figura 9. Uma região de publicação/subscrição, presente em uma célula, é associada ao grupo *multicast* desta célula. Por exemplo, o avião espião que se subscreve nas células 1 e 3 é associado aos grupos *multicast* MG1 e MG3.

Entretanto, dados serão transferidos para todas as células com publicadores, mesmo quando não ocorrerem intersecções. O mesmo acontece quando, os limites da região não casarem com os limites da grade. Uma inexatidão pode ser causada na intersecção de algumas células de tal forma que publicadores enviam dados desnecessários aos subscritores. O custo dessa transmissão de dados irrelevantes pode ser alta, contra os benefícios da redução da dependência entre os *hosts*.

Tabela 1. Associação dos grupos *multicast* na técnica Fixa Baseada em Grade (*Fixed Grid-Based*).

Federados	Tipo da Entidade	Região de Interesse	Células	Associados
Fed1	Avião Espião	Subscrição	1,3	MG1, MG3
Fed2	Esquadrão A	Publicação	3	MG3
Fed3	Esquadrão B	Publicação	4	MG4

Técnica Híbrida (*Hybrid Approach*)

A técnica híbrida une as técnicas Baseadas em Região e Grade procurando reduzir os custos de suas desvantagens. Através do *matching* entre publicadores e subscritores que fazem parte de uma intersecção, procura reduzir o custo do *matching* excessivo associado à Baseada em Região e através do *matching* exato, procura reduzir o custo de atualização/mensagens desnecessárias da Baseada em Grade [TAN00b].

A técnica híbrida utiliza o seguinte algoritmo. Todos os federados enviam uma informação de interesse para um coordenador central, o Coodernador_DDM que, por sua vez, determina as intersecções mapeando as regiões de interesse em uma grade, como na técnica Baseada em Grade. Finalmente, o Coordenador_DDM realiza o *matching* com aqueles publicadores e subscritores que são mapeados em células com intersecção. Assim, para alguns casos, esta técnica mostra uma melhoria quando comparada com as duas técnicas separadas [TAN00b]. Mas, ela ainda requer o uso de um coordenador central, como na técnica Baseada em Região e, o uso deste coordenador, tende a limitar a escalabilidade do sistema DDM, transformando-se em uma desvantagem quando comparada à técnica Baseada em Grade, que não possui tal coordenador.

Dinâmica Baseada em Grade (*Dynamic Grid-Based*)

Como toda técnica Baseada em Grade, existe uma grade sobrepondo o terreno e definindo as células. Mas, diferente da técnica Fixa Baseada em Grade, os grupos de *multicast* são alocados dinamicamente com base nas correntes regiões de publicação e subscrição da simulação. Publicadores apenas unem-se e transmitem em um grupo se existir pelo menos um subscritor interessado naquele dado. Da mesma forma, subscritores apenas unem-se e recebem dados em um grupo se existir pelo menos um publicador transmitindo nele.

A estratégia utilizada na técnica Dinâmica Baseada em Grade é única porque apenas as células nas quais existem pelo menos um publicador e um subscritor, são associadas a um grupo *multicast*. As duas principais vantagens resultantes dessa técnica são: (1) publicadores são prevenidos de enviar dados desnecessários e (2) o número de grupos *multicast* a que um federado necessita se unir é reduzido de maneira bastante significativa.

Outros benefícios dessa técnica são a detecção de intersecções e o mecanismo de mensagens de acionamento (*triggering*) que são realizados pelos componentes RTI em cada *host*. Isto pode ser visto como uma coleção de Coordenadores_DDM distribuídos, onde cada um deles é responsável em manter um conjunto específico de células. Assim, não há uma base de dados nem coordenadores centrais, tornando essa técnica mais escalável que a Baseada em Região, que utiliza o Coordenador_DDM central.

Na Tabela 2, mostra-se o funcionamento dessa técnica, seguindo o exemplo da Figura 9. À partir da detecção da intersecção entre o Avião Espião e o Esquadrão A, um grupo *multicast* é criado dinamicamente para a célula 3, MG3, e os dois federados recebem mensagens de acionamento para unirem-se ao grupo. Uma diferença significativa pode ser observada com relação ao número de grupos *multicast*; isto porque, nesta técnica, apenas um grupo é criado contra os quatro criados na técnica Fixa Baseada em Grade.

Tabela 2. Associação dos grupos *multicast* na técnica Dinâmica Baseada em Grade (*Dynamic Grid-Based*).

Federados	Tipo da Entidade	Região de Interesse	Células	Associados
Fed1	Avião Espião	Subscrição	1,3	MG3
Fed2	Esquadrão A	Publicação	3	MG3
Fed3	Esquadrão B	Publicação	4	-

Grade Filtrada Baseada em Região (*Grid-Filtered Region-Based*)

Na técnica de Grade Filtrada Baseada em Região, o melhor das técnicas baseadas em região e grade são combinadas para oferecer um mecanismo de granularidade mais fina procurando reduzir ainda mais o tráfego de dados, com um teste de intersecção mais preciso, que considera o tamanho da célula da grade e a porcentagem da região que se sobrepôs à célula.

Esta técnica possui em sua estrutura de dados um valor de corte para a realização do *matching*, se a região cobrir a célula com uma porcentagem maior que o corte, a região é considerada uma região com intersecção, caso contrário, cada entidade é comparada explicitamente na célula em busca de intersecções – como na técnica Baseada em Região.

3.1.2.5 Comparação das Técnicas DDM e Técnica Utilizada

A técnica Baseada em Região, apesar de prover um teste de intersecção exato entre regiões de publicação e de subscrição, necessita de um coordenador central para realizá-la, o que pode se tornar um gargalo com o aumento do número de publicadores e subscritores atualizando suas áreas de interesse com freqüência. A técnica Híbrida reduz o custo do teste de intersecção da técnica Baseada em Região, mas não resolve o problema do coordenador central. A Fixa Baseada em Grade não possui um coordenador central, que é sua principal vantagem, mas não realiza um teste de intersecção exato; logo, pode enviar dados desnecessários que devem ser filtrados depois no destino. Além disso, como os grupos *multicast* são pré-alocados e associados às células, um número muito grande de grupos *multicast* são utilizados, quando comparado às outras técnicas.

Com a técnica Dinâmica Baseada em Grade esses problemas foram amenizados [BOU02]. Para simulações em larga escala, com um número crescente de entidades, o número de grupos *multicast* é excessivamente inferior em relação à técnica Fixa Baseada em Grade. Outra grande vantagem é que não é necessária uma base de dados nem um coordenador central; cada coordenador é responsável por manter um grupo específico de células da grade, o que torna a técnica Dinâmica Baseada em Grade mais escalável que técnicas apoiadas na Baseada em Região. No entanto, o teste de intersecção, da mesma forma que a técnica Fixa Baseada em Grade, não é exato. Esse

teste foi adicionado à Grade Filtrada Baseada em Região (GFBR) na presença do valor de corte assim, se a porcentagem da região sobre a célula não ultrapassar o valor de corte, o teste é efetuado como na técnica Baseada em Região.

Na Tabela 3, faz-se a comparação das técnicas de Gerenciamento de Distribuição de Dados, onde as variantes são fatores desejáveis para que exista a detecção de intersecção de maneira exata e não exaustiva e sem um coordenador central.

Tabela 3. Comparação das técnicas de Gerenciamento de Distribuição de Dados-DDM.

Variantes	Região	Fixa	Híbrida	Dinâmica	GFBR
Teste de Intersecção Exato	X		X		X
Sem Teste de Intersecção Exaustivo		X	X	X	X
Sem Coordenador Central		X		X	X
Detecção de Intersecção	X		X	X	X

O grande objetivo da utilização de algoritmos de Gerenciamento de Distribuição de dados é a filtragem de dados trocados entre as entidades participantes de simulações. Para tal, a técnica utlizada em nossos experimentos foi a Dinâmica Baseada em Grade pois reduz significantemente o *overhead* de mensagens e o número de grupos multicast utilizados [BOU00].

3.1.3 Kit RTI

O Kit RTI é a implementação da especificação da interface HLA utilizado em nossos experimentos e foi desenvolvido originalmente na Georgia Tech [GEO06]. Consiste de um conjunto de bibliotecas designadas para suportar o desenvolvimento de Infra-estruturas de Tempo de Execução para sistemas de simulação paralela e distribuída. Novos algoritmos de gerenciamento de distribuição de dados foram adicionados posteriormente por [BOU00, BOU02, BOU04b]. Vale ressaltar que tal Kit não possui distribuição livre e só foi possível sua utilização a partir da colaboração com o laboratório PARADISE, da *University of Ottawa,* Canadá. Outras implementações já foram realizadas mas são *softwares* proprietários.

A seguir será explicado o funcionamento do Kit seguido de um exemplo para melhor compreender. A partir do momento que todos os *hosts* possuam o Kit RTI, gera-se e executa-se um *script* com os seguintes dados (Observações: O arquivo *Spawner* atua em dois momentos: 1 – Geral e 2 – Local. Os passos de 3 a 5 são repetidos *Número de hosts* vezes, ou seja, para cada *host* participante da simulação):

1. *Spawner*. Arquivo executável responsável por distribuir a simulação entre todos os *hosts* participantes;
2. *Número de hosts*. O número de *hosts* que irão participar na simulação – após a *tag* -npe;
3. *Host*. O nome/endereço do *Host* participante na simulação – após a *tag* -nodes;
4. *Spawner*. Arquivo executável responsável por inicializar as rotinas do *Host* referentes aos *Argumentos* e inicializar a execução – após a *tag* -spawner;
5. *Arquivo_DDM*. Arquivo contendo os algoritmos DDM do *Host* – após a *tag* -exec;
6. *Argumentos.* Entre as *tags* –startags e -endargs.
    1 *Número de Unidades por Espaço Dimensional (Units per Spatial Dimension – UPSD). A partir deste valor calcula-se o número de células* que o *Host* terá sob sua responsabilidade da seguinte maneira: UPSD*UPSD*2/*Número de hosts*, onde 2 é a terceira dimensão representada pelo *Time* (descrito a seguir no item 6);
    2 *Número de objetos* totais na simulação. Este número será dividido entre todos os *hosts*, ou seja, cada *Host* possuirá *Número de objetos / Número de hosts* objetos;
    3 *Algoritmo DDM* utilizado na simulação. Definir um dentre todos os algoritmos implementados;
    4 *Mobilidade dos Objetos.* Móveis ou estáticos.
    5 *Tipo de Interesse*. Podendo ser publicação (pub), subscrição (sub) ou ambas (pubsub).
    6 *Time*. Duas possibilidades, vermelho (*red*) ou azul (*blue*).

Um *script* exemplo pode ser acompanhado no quadro a seguir e está explicado posteriormente:

```
/spawner-LINUX -npe 2 -nodes
computador1 -spawner /home/computador1/spawner-LINUX
–exec /home/computador1/alg_ddm
-startargs 100 10 dynamic move pub blue –endargs


computador2 -spawner /home/computador2 spawner-LINUX
–exec /home/computador2/alg_ddm
-startargs 100 10 dynamic move sub red –endargs
```

Na simulação executada pelo *script*, têm-se 2 *hosts* participando, computador1 e computador2. Cada *host* possuirá 10000 células (total de 20000 células na simulação); 5 objetos (total de 10 objetos na simulação); o algoritmo de gerenciamento de distribuição de dados escolhido foi o dinâmico baseado em grade; todos seus objetos serão móveis; os objetos do computador1 estarão publicando e fazem parte do time azul; os objetos do computador2 estarão subscrevendo e fazem parte do time vermelho. Com estas informações sabemos que os objetos do computador2 estão interessados nos objetos do computador1, isto porque, eles não apenas possuem interesses diferentes, mas também, fazem parte de times diferentes.

Apesar do Kit RTI ser extremamente útil em simulações distribuídas de grande porte (grande número de entidades participantes e áreas extensas), ele apresenta várias limitações, descritas a seguir:

- O número de objetos simulados deve ser igual em todos os *hosts*. Todos *hosts*, independente de capacidade e interesse, devem controlar o mesmo número de objetos, divididos entre todos os participantes na inicialização da simulação;
- A presença de apenas um tipo de objeto. AVCs com uma enorme gama de tipos de objetos estão limitados a um único tipo, o objeto *Tank*;
- A velocidade dos objetos é única. Nas aplicações AVCs é necessário o tratamento de diferentes objetos com diferentes velocidades;
- O posicionamento dos objetos na inicialização da simulação é randômico. Os *hosts* não possuem a liberdade de posicionar seus avatares dentro do ambiente;
- A impossibilidade de interrupção da simulação e continuação posterior. Simulações interrompidas precisam ser iniciadas novamente;
- A impossibilidade de alteração, após iniciada, da simulação, por exemplo, no número de objetos simulados ou no número de *hosts* participantes, uma vez que, se algum *host* sair, a simulação pára;
- A área do ambiente simulado é dividida e distribuída igualmente entre todas as entidades participantes. Assim, tanto dispositivos com baixos recursos como computadores potentes serão responsáveis pela mesma carga de trabalho;
- A simulação não é orientada a eventos mas a um número fixo de iterações, onde em cada iteração todos os objetos realizam sua movimentação. Numa simulação AVC, *hosts* precisam ter liberdade de movimentação de seus avatares.

3.2 Considerações Finais

Foram apresentados neste capítulo, os conceitos de simulação paralela e distribuída, nos quais a execução de simulações é distribuída através de múltiplos computadores, provendo benefícios substanciais para aplicações como os sistemas analíticos e os ambientes virtuais. Também foi apresentado o modelo HLA de referência, uma arquitetura comum para reuso e interoperação entre simulações. A especificação da interface HLA é implementada pelo RTI, sendo o Kit RTI, desenvolvido na Georgia Tech, a implementação utilizada neste trabalho. O RTI oferece seis serviços, dentre os quais o Gerenciamento de Distribuição de Dados – DDM. O grande objetivo do Gerenciamento de Distribuição de Dados é reduzir e controlar o volume de dados trocado durante uma simulação, o que atende aos requisitos de AVCs-LE, conforme mostrado no capítulo anterior.

Apesar do Kit RTI ser muito útil no suporte a simulações distribuídas, possui várias limitações. De modo a superar estas limitações (mostradas na seção 3.1.3 deste capítulo), uma arquitetura foi concebida que suporta simulações distribuídas, como AVC-LE, em redes *Peer-to-peer*. Este modelo de comunicação e, particularmente, a implementação *Gnutella* do modelo de redes *Peer-to-peer* é descrito e discutido no próximo capítulo.

4 Modelo de Comunicação *Peer-to-peer*

O *Peer-to-peer* (P2P) é um modelo de comunicação que possibilita dois ou mais pares colaborarem espontaneamente numa rede de pares (máquinas consideradas em nível de igualdade), usando informações apropriadas e um sistema de comunicação sem a necessidade de uma coordenação central [MIL02].

O *Peer-to-peer* tem o potencial de acelerar os processos de comunicação, explorar recursos inativos (ociosos) e facilitar a troca de informações recentemente criadas e distribuídas [SCH03]. Cada computador participante num modelo de comunicação P2P é chamado de par (*peer*), indicando que os participantes interagem de igual-para-igual. Esses pares podem apresentar algumas regras como, por exemplo, quando acessam informações, eles são clientes; quando servem informações a outros clientes, eles são servidores; e quando encaminham informações para outros, eles são roteadores [KUB03].

Várias aplicações surgiram como implementações do modelo de comunicação P2P, algumas tais como o *Napster* [OPE04] e o *Gnutella* [GNU04]. Embora o Napster se caracterize como um modelo *Peer-to-peer*, ainda existe a figura do servidor que oferece a habilidade de procurar arquivos em particular e iniciar a transferência direta entre os clientes. Já a rede *Gnutella* é descentralizada e logo, mais robusta porque elimina a dependência em servidores que são pontos onde facilmente podem existir falhas Esta foi a principal característica que levou-se em consideração na escolha da rede *Gnutella* para este trabalho. As seções seguintes descrevem os modelos existentes de comunicação P2P e a rede P2P *Gnutella*.

4.1 *Peer-to-peer* Descentralizado

O modelo *Peer-to-peer* descentralizado, também chamado de P2P puro de acordo com [SCH01], trabalha sem a ajuda de nenhum tipo de servidor ou repositório central de informações. Todos os pares são auto-suficientes e podem assumir simultaneamente as funções de cliente e servidor, como pode ser visto na Figura 10 [ZAI04].

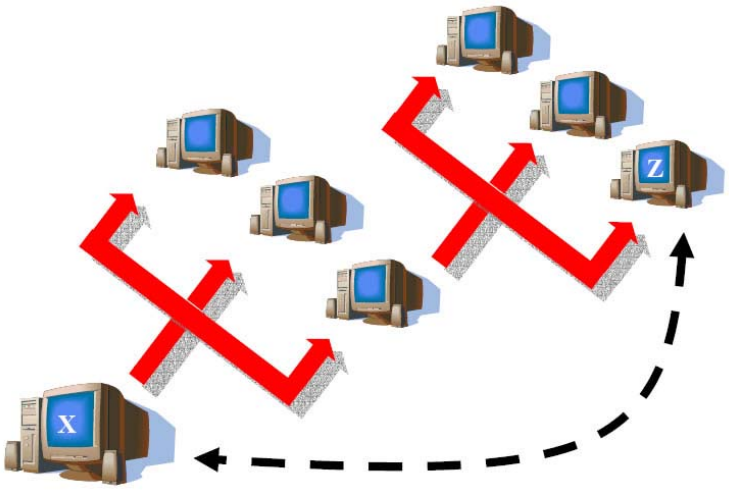

Figura 10. P2P – Modelo Descentralizado.

Neste modelo, os pares efetuam solicitações aos pares vizinhos que estão ao seu alcance, no caso, "X" sai procurando nas máquinas com as quais tem uma conexão. Por sua vez, estes se conectam a outros pares, até que o recurso seja localizado, no caso, em "Z". Quando isto acontece, é passado um apontador para o par inicial, que estabelece uma conexão direta com o par onde o recurso foi encontrado, podendo assim fazer a transação sem intermediários.

O *Peer-to-peer* descentralizado, devido a sua complexidade, é raramente usado conforme a definição acima. É muito mais freqüente a implementação de servidores para tarefas específicas e com pouco *overhead*, o que se trata do *Peer-to-peer* híbrido [ZAI04].

4.2 *Peer-to-peer* Híbrido

No modelo híbrido, são usados servidores para tarefas como autenticação de usuários, serviços de diretório e mapeamento de recursos disponíveis. A idéia básica é que os pares possam contatar algum servidor para iniciar a transação, ou para alguns dos serviços disponíveis na rede. O modelo híbrido é mostrado na Figura 11.

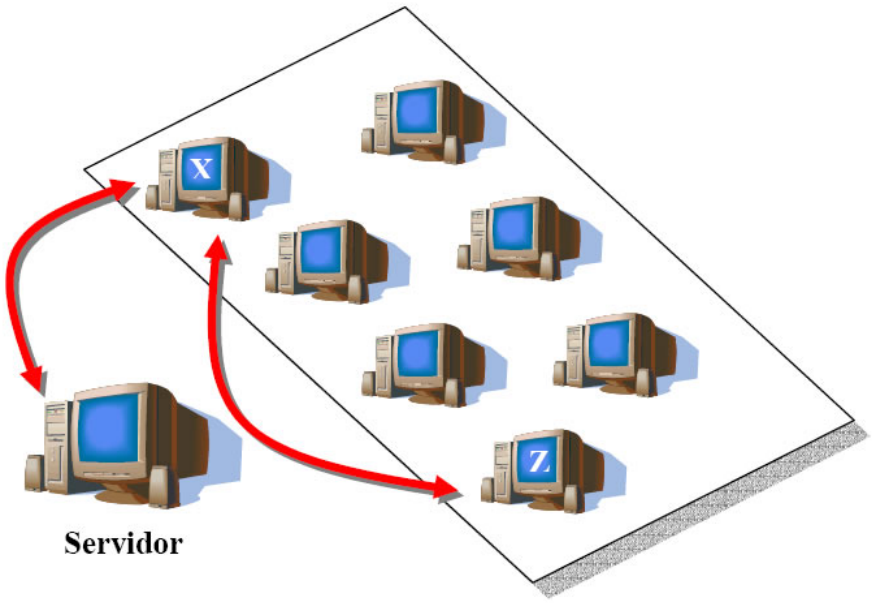

Figura 11. P2P – Modelo Híbrido.

Como mostra a Figura 11, "X" contata o servidor para saber quem tem um determinado arquivo ou informação. Em seguida, o servidor devolve ao par inicial alguma informação pertinente que permita a ele conectar-se diretamente com outro par, "Z" e assim efetuar a transação [ZAI04].

O modelo híbrido é usado na grande maioria dos sistemas P2P. No caso do Napster, o primeiro sistema de compartilhamento de arquivos que causou furor entre usuários do mundo todo e grandes dores de cabeça à indústria fonográfica, por exemplo, um servidor central armazena todos as contas de usuários e as listas de arquivos compartilhados. Ao efetuar-se a busca por um arquivo, o servidor central consulta a sua base e aponta um par que possui este arquivo. Daí em diante, a comunicação e transferência é processada diretamente, par a par [ZAI04].

Existem hoje várias aplicações P2P, as quais podem ser divididas em grupos de acordo com suas funções [CIG02]. Serão vistos a seguir os diferentes tipos existentes, suas características e tecnologias.

4.3 Tipos de Aplicações *Peer-to-peer* Existentes

As subseções seguintes descrevem os quatro principais tipos de aplicações *Peer-to-peer* [MIL02].

4.3.1 Computação Distribuída

Computação em grade é uma coleção de recursos heterogêneos e distribuídos possibilitando que sejam utilizados em grupo para executar aplicações de larga escala. Um *software* servidor divide as tarefas em pequenas partes as quais são divididas entre os computadores que estão conectados ao sistema. Os computadores executam as tarefas associadas apenas quando estão ociosos e depois de prontas, enviam-nas de volta ao servidor. Podem ser chamados de sistemas orientados a pares (*peers*).

Uma questão comumente destacada é se os sistemas de computação distribuída, no caso grade, são realmente *Peer-to-peer*. O argumento contra, é que um servidor central é necessário para controlar os recursos, os pares não operam como servidores e não ocorre comunicação entre os pares. O argumento a favor é que uma parte significante do sistema é executada nos pares, com alto anonimato. [CIG02, MIL02] consideram as aplicações distribuídas (grade) como sendo aplicações *Peer-to-peer*. Uma das maiores limitações da computação distribuída é que ela requer trabalhos que possam ser divididos em pequenas partes independentes que não necessitem de comunicação entre pares. As aplicações atuais consistem de aplicações Processo Simples Múltiplos Dados (*Single Process Multiple Data* – SPMD), ou seja, problemas em que um dado trabalho possa ser executado em muitos conjuntos de entrada de dados diferentes. Isso inclui, na maioria das vezes, aplicações como simulações. Dentre as várias aplicações existentes, podem ser citadas o Projeto Genoma Humano [GEN01], as pesquisas de novas drogas contra o Câncer [UNI04], o Projeto SETI@Home [SET04] do Laboratório de Ciências Espaciais da Universidade da Califórnia, Berkeley, a primeira tentativa de utilizar computação distribuída de larga escala para realizar uma busca sensível por sinais de rádio de civilizações extraterrestres, entre outras.

4.3.2 Compartilhamento de Arquivos

O compartilhamento de arquivos é uma aplicação típica P2P. A idéia é direta: substituir o disco rígido local de um computador por pontos de expansão de armazenamento ao longo da Internet. Assim, um computador interage com vários pares da rede para obter ou doar informações. Napster e *Gnutella* são as aplicações que têm sido muito usadas pelos usuários da Internet para evitar limitações de largura de banda que transformam a transferência de grandes arquivos em algo inaceitável.

As principais questões relacionadas a aplicações de compartilhamento de arquivos envolvem consumo de largura de banda, segurança e capacidades de busca. De acordo com [MIL02] existem hoje três principais modelos que são: modelo centralizado, como é o caso do Napster, o modelo de busca por *flooding*, como é o caso do *Gnutella*, e o modelo de roteamento de documento, como é o caso do FreeNet. Todas as aplicações de compartilhamento de arquivos podem ser classificadas dentro de um desses três modelos citados, embora existam variações, como por exemplo o KazaA, o qual permite uma melhor escalabilidade de sua aplicação e menos estresse na rede através de um componente, o *Supernode* – computador que possui uma lista dos arquivos sendo compartilhados por seus nós vizinhos e quando um destes deseja realizar uma busca, esta é encaminhada ao *Supernode*.

4.3.3 Colaboração *Peer-to-peer*

Aplicações colaborativas *Peer-to-peer* têm como principal objetivo permitir a colaboração ao nível de aplicação entre os usuários. Essas aplicações abrangem de *chats* e mensagens instantâneas, jogos e aplicações compartilhadas que podem ser usadas comercialmente e até mesmo por um grupo de amigos em suas casas.

As aplicações colaborativas são baseadas em eventos. Os pares formam grupos e iniciam uma determinada tarefa. O grupo pode incluir apenas dois pares colaborando diretamente, ou pode ser um grupo maior. Quando uma mudança ocorre num par um evento é gerado e enviado ao resto do grupo. Na camada de aplicação, cada interface de um par é atualizada de acordo com cada aplicação.

4.3.4 Plataforma *Peer-to-peer*

Os sistemas operacionais estão se tornando cada vez menos relevantes como ambientes para aplicações. Soluções, tais como Java Virtual Machines, ou Web browsers e servidores são os ambientes dominantes que são dos interesses dos usuários bem como dos desenvolvedores de aplicações. Em consideração, é provável que futuros sistemas em crescimento dependerão de algum outro tipo de plataforma que deverá ser um denominador comum para os usuários e serviços conectados a Web ou numa rede *ad-hoc* [MIL02].

As plataformas *Peer-to-peer* atualmente disponíveis oferecem suporte para os seguintes componentes primários P2P: descoberta, comunicação, segurança, e agregação de recursos. Elas têm uma dependência de sistema operacional, mesmo que mínima. A maioria das aplicações P2P ou estão rodando em Linux ou então são baseadas no Windows. Dentre as plataformas *Peer-to-peer*, uma das que mais se destacam é a plataforma JXTA, que é um conjunto de protocolos abertos e generalizados P2P que permitem a conexão de qualquer dispositivo na rede - de telefones celulares, PDAs e PCs - para se comunicarem e colaborarem como pares. Os protocolos JXTA são independentes de qualquer linguagem de programação e múltiplas implementações existem para diferentes ambientes. O JXTA é baseado no software JavaTM 2 Plataform, J2SE [JXT04].

4.4 Desafios do Modelo *Peer-to-peer*

Além do desafio inerente de se ter aplicações comunicando-se de igual para igual, outros desafios podem ser identificados no modelo P2P, entre eles, citam-se a segurança, interoperabilidade, controle de acesso, integridade dos dados, sincronização de uma aplicação entre duas ou mais máquinas (pares), além do grande desafio de se ter controle e dados distribuídos numa rede P2P [AND04, BHA05]. A seguir, temos as descrições desses desafios:

- *Segurança.* Devido à ampla quantidade de ameaças conhecidas nos sistemas de rede, mecanismos de segurança são extremamente necessários. Como o P2P está tornando-se interessante para o uso comercial, técnicas e métodos para autenticação, autorização, disponibilidade, integridade dos dados e confiança devem ser integradas ao P2P;
- *Interoperabilidade*. Atualmente, as aplicações utilizam protocolos e interfaces específicos. Como resultado, a interoperabilidade que se estende além de uma simples aplicação ou rede é rara nas atuais redes P2P. O desafio é encurtar o tempo de desenvolvimento e habilitar as aplicações a serem implementadas facilmente nos sistemas existentes;
- *Controle de acesso*. Outro desafio é o controle do acesso ao sistema, permitindo ou não acesso de usuários às informações;

- *Integridade dos dados.* caso os dados estejam distribuídos entre os vários pares existentes na rede, deve haver uma preocupação com a integridade desses dados, que podem ou não depender uns dos outros para formar alguma informação;
- *Sincronização.* Um dos grandes desafios dos sistemas P2P é a sincronização dos dados, uma vez que vários pares podem possuir um mesmo dado;
- *Distribuição dos Dados.* A distribuição dos dados pode ser feita de várias formas, entre elas a replicação (uma alteração de estado é passada para as réplicas) e particionamento (a aplicação é dividida em várias partes e cada máquina fica encarregada de uma);
- *Distribuição do Controle.* O controle em redes P2P é necessário e pode ser aplicado num servidor, como por exemplo em Napster, ICQ entre outras aplicações de rede P2P híbridas, ou então pode ser aplicado nos próprios clientes, eliminando a presença de um servidor, como em redes P2P puras. O desafio é principalmente em se implementar esse controle em redes P2P puras, ou seja, como administrar sem ter um servidor, como eleger uma máquina para que essa passe a ter o controle, e se essa máquina se desconectar da rede, para quem é passado o controle.

4.5 Vantagens e Desvantagens do *Peer-to-peer*

O modelo de comunicação *Peer-to-peer* apresenta algumas vantagens com relação ao modelo cliente/servidor que são descritas a seguir [AND04, GON02]:

- A falha num cliente afeta apenas a participação dos usuários ligados neste cliente, podendo ou não afetar a operação do sistema como um todo (depende da topologia);
- Elimina o gargalo, uma vez que as atividades estão distribuídas e não centralizadas;
- Dependendo da topologia, a distância entre os clientes pode ser reduzida;
- É escalável, podendo suportar um número crescente de clientes, mas pode apresentar problemas de engarrafamentos na rede;
- Não apresenta barreiras para se criar um sistema P2P, uma vez que ele não requer nenhum arranjo administrativo ou financeiro, ao contrário do cliente/servidor;

- Oferecem uma forma de agregar e fazer uso de enormes recursos computacionais e de armazenamento através dos computadores pela Internet;
- A natureza descentralizada e distribuída do sistema P2P lhe dá o potencial de ser tolerante a falhas e/ou ataques intencionais.

Porém esse modelo, assim como o cliente/servidor, apresenta desvantagens, a saber:

- Maior complexidade do *software*, custos mais altos de desenvolvimento e menor desempenho em cada cliente, devido a distribuição da coordenação das atividades entre os clientes;
- Cada cliente tem que saber os endereços dos outros clientes, para que possam se comunicar diretamente;
- Maior complexidade no gerenciamento de entrada e saída de novos clientes;
- O controle das versões de *software* é mais difícil de ser mantida, por estar distribuída entre os clientes.

Com a apelação de sair do tradicional cliente/servidor e utilizar mais o potencial das máquinas que ficam nas pontas da rede (pares), existem várias aplicações hoje utilizando o modelo de comunicação P2P, algumas tais como o *Napster* e o *Gnutella*. Nestas aplicações, os arquivos ficam na máquina do cliente, nunca passando por um servidor. No *Napster,* o servidor oferece apenas a habilidade de procurar arquivos em particular e iniciar a transferência direta entre os clientes [OPE04]. Enquanto um servidor central garante uma busca rápida e eficiente, o sistema também possui um único ponto de entrada. Nesse caso, pode haver gargalo na rede ou então a perda de conexão se o servidor, ou servidores, se tornarem incapacitados. Além disso, o modelo pode fornecer informações desatualizadas por um período de tempo, uma vez que a atualização é periódica e pode deixar de fornecer o estado mais atual quando, por exemplo, da saída de um usuário da rede. Já a rede *Gnutella* é descentralizada e logo, mais robusta que o modelo centralizado porque elimina a dependência em servidores que são pontos onde facilmente podem existir falhas [GNU04]. Este e outros fatores levaram à sua escolha como a rede P2P deste trabalho. A rede *Gnutella* está descrita a seguir.

4.6 *Gnutella*

Ao contrário do *Napster*, a *Gnutella* não usava, em sua concepção inicial, um servidor central para rastrear todos os arquivos dos usuários [GNU04]. Para utilizar a rede *Gnutella* um usuário precisa de uma aplicação que promove a conexão com outros pares, formando uma rede espontânea, sem controle centralizado, e que age tanto como cliente quanto servidor de conteúdo. Um exemplo de seu funcionamento está detalhando a seguir e representado na Figura 12.

Para compartilhar arquivos o terminal de um usuário (par) utiliza um computador na rede, chamado aqui de "A". Todos os pares podem ser chamados de "servent", pois o programa atua como uma combinação de servidor e cliente. O computador "A" então se conectará a outro computador na rede, "B" e avisará que está procurando um determinado arquivo. O computador "B" por sua vez, comunicará a todos os computadores com quem ele está conectado sobre o "A". Enquanto o alcance dessa rede pode ser infinito, ele é , na verdade, limitado por um mecanismo denominado tempo-de-vida (*time-to-live* - TTL), que nada mais é do que o número de computadores que a busca atingirá. A maioria dos computadores (pares) rejeitarão qualquer mensagem na rede que tiver um TTL muito alto [OLI04].

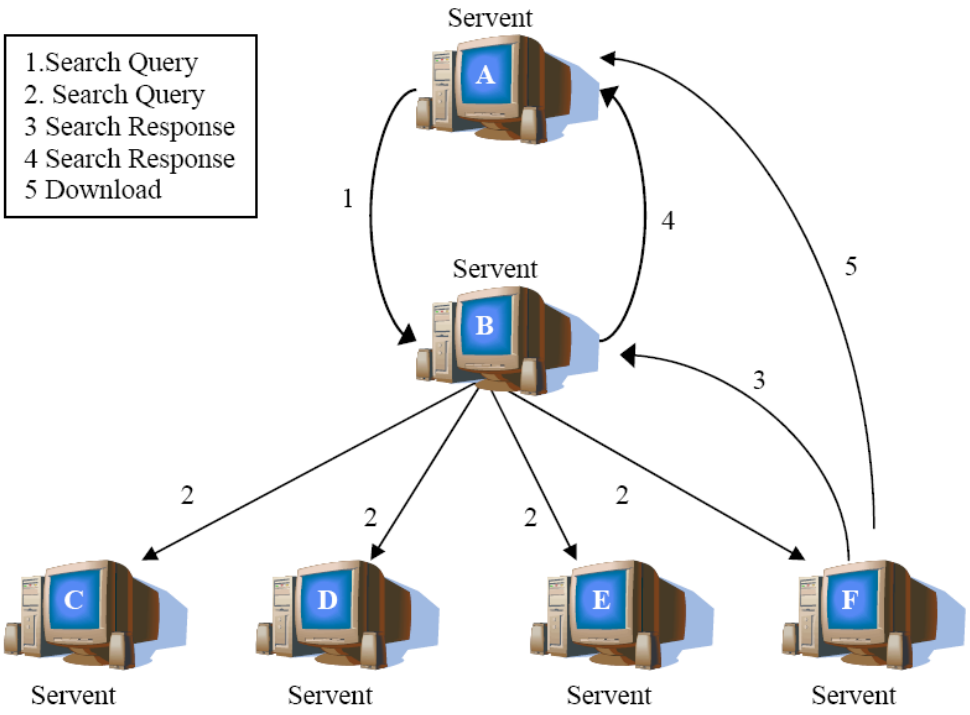

Figura 12. Rede *Gnutella*.

A rede *Gnutella* tem um número significativo de vantagens sobre os outros modelos *Peer-to-peer*. Ela é descentralizada e logo, mais robusta que o modelo centralizado porque elimina a dependência em servidores que são pontos onde facilmente podem existir falhas. Enquanto o *Napster* tinha apenas um *software* como Cliente, o *Gnutella* apresenta uma variedade, dentre os quais: *BearShare*, *Gnucleus*, *KazaA*, *Limewire*, *Morpheus*, *WinMX*, *XoloX*, *JTella*. Atualmente, a rede *Gnutella* e seus softwares clientes sofreram modificações para oferecer melhorias a seus usuários, a principal modificação é o conceito *Ultrapeer* e é apresentado na seguinte subseção.

Apesar da rede *Gnutella* ter sido um pouco obscurecida por redes eficientes como o *FastTrack* (protocolo proprietário da aplicação *KazaA*) ela foi escolhida como alvo deste trabalho pela simplicidade conceitual e por ser um software livre, o que proporciona que experimentações das mais diversas e interessantes possam ser realizadas e avaliadas, como é o caso da implementação de ambientes virtuais colaborativos.

A seguir são apresentados o conceito *Ultrapeer* na rede *Gnutella* e um de seus *softwares* clientes, o *JTella*.

4.6.1 O Conceito *Ultrapeer* na Rede *Gnutella*

O protocolo *Gnutella* original trata todos os pares (*peers*) indiferentemente de suas capacidades de largura de banda, poder computacional e outras propriedades (capacidades de conexão, *uptime*, etc). Os pares na rede comunicam-se usando o protocolo *Gnutella* versão 0.4 [GNU04], ou uma versão melhorada que utiliza novos mecanismos de estabelecimento de conexão [BIL01], pois ambos apresentam propriedades interessantes. Novos pares podem entrar na rede a qualquer momento e os que estiverem *online* poderão sair a qualquer momento. A habilidade de se ter uma rede segura, sem a dependência de um par em particular (ou um conjunto deles), é uma característica notável que o *Gnutella* traz com sua popularidade. Juntamente com isso, há outras características da rede *Gnutella* [RIT01] que a distingue de outros sistemas. Porém apesar das boas características que possui, a rede *Gnutella* apresenta o problema da escalabilidade. Por exemplo, um grande número de *queries* são enviadas para muitos pares, numa tentativa de assegurar a resposta de poucos deles, causando problema de largura de banda. E para tentar resolver isso, foi proposto pelos desenvolvedores *Gnutella* o esquema de *Ultrapeers*. Um *Ultrapeer* é um par como os outros, porém

apresentando características como alto poder de processamento e alta largura de banda, podendo assumir a "carga de trabalho" de outros pares mais fracos, PC´s ou até mesmo PDA´s e telefones celulares (Figura 13). Os *Ultrapeers* aliviam também os pares normais da recepção da maioria do tráfego de mensagens, reduzindo assim o tráfego total da rede e deixando-a mais rápida.

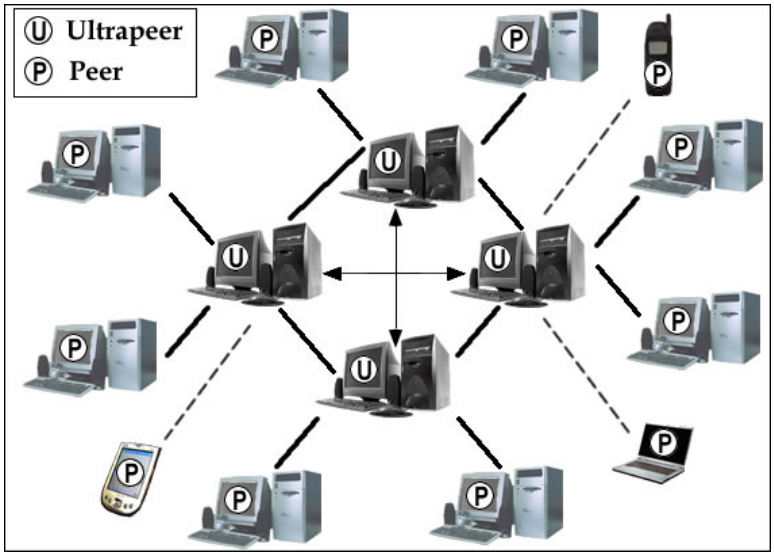

Figura 13. *Ultrapeers* e *peers*.

4.6.2 *Software* Cliente *JTella*

O *JTella* é um software "servent" (do inglês *server* e *client* – servidor e cliente) da rede *Gnutella*. Sua primeira versão consistia de um conjunto de bibliotecas Java criadas por Ken Mccrary [MCC00], lançadas como *JTella* 0.7. Seu objetivo era prover uma maneira fácil de acessar a rede *Gnutella* e realizar buscas de arquivos sem ditar como esses *dowloads* seriam tratados e sem *interface* gráfica. Também tinha o objetivo de ser de fácil entendimento consistindo de apenas 36 classes, a maioria delas pequenas e bem comentadas. Oferecia suporte ao protocolo *Gnutella* 0.4 [GNU04]. Posteriormente Daniel Meyers [MEY04] atualizou o *JTella*, chamando de *JTella* 0.8, adicionando o suporte ao novo protocolo *Gnutella* 0.6 [GNU06], a possibilidade de *download* de arquivo a partir de várias fontes, assim como, uma interface gráfica. Finalmente, em 2004, Stuart Andrew Beatty [BEA05] acrescentou novas funcionalidades ao *JTella*, o tratamento de *downloads* e a possibilidade de compartilhamento de arquivos e não apenas busca como anteriormente.

O fácil entendimento do software *JTella*, em menor espaço de tempo, foi crucial na sua escolha como base para nossos estudos, apesar de sua baixa utilização, quando comparada a outros softwares clientes Java, de código livre, como o *Limewire* [LIM04], que possui mais de 100 classes, a maioria delas muito complexas, e o *Phex* [PHE04]. A Figura 14 mostra a *interface* do cliente *JTella*. Na aba 'Connection' vizualizam-se as conexões estabelecidas com outros pares (*peers*). Na aba 'Searching' realizam-se as buscas e mostram-se os arquivos encontrados. Na aba 'Downloading' mostram-se os arquivos sendo baixados pelo usuário. Na aba 'Options' defini-se parâmetros como endereço IP, porta e se Ultrapeer ou não. E a aba 'Sessions' foi incluida neste trabalho para tratar as sessões de AVCs e será melhor explicada no próximo capítulo.

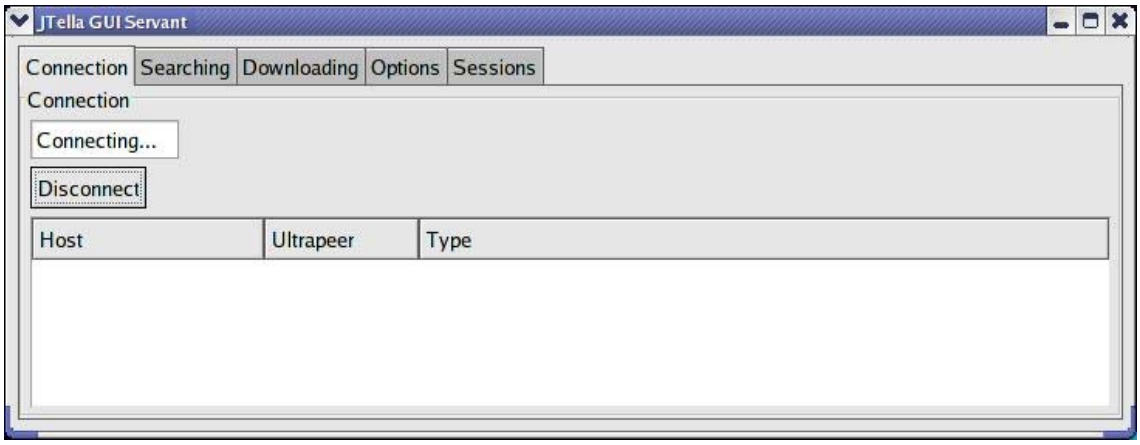

Figura 14. *Interface* do *JTella*.

4.7 Considerações Finais

Neste capítulo, foi descrito o modelo *Peer-to-peer* e suas características. A rede *Gnutella* foi escolhida como a rede *Peer-to-peer* utilizada neste trabalho, devido a sua característica descentralizada, o seu código fonte livre e pela sua simplicidade, apresentando características como os *ultrapeers* os quais podem ser utilizados como repositório para os mundos virtuais e entidades participantes da simulação. Por sua vez, o *software* cliente *JTella* foi utilizado também pela sua simplicidade e curto tempo necessário de aprendizado quando comparado a outros *softwares* clientes como o *Limewire*.

Uma arquitetura de suporte a AVC-LE foi projetada para atender os estritos requisitos dessas simulações, com tolerância a falhas, baixa latência, e redução e controle de volume de tráfego e que supera as limitações de estruturas atuais de

execução de simulações distribuídas, como é o caso do Kit RTI utilizado neste trabalho. O próximo capítulo descreve esta arquitetura.

5 Projeto de uma Arquitetura de Suporte a AVCs-LE

Este trabalho é parte do projeto "*Emergency Preparedness*", envolvendo o Laboratório de Realidade Virtual em Rede (LRVnet), do Departamento de Computação da UFSCar e o laboratório PARADISE, do SITE da *University of Ottawa*, Canadá. O objetivo deste projeto é pesquisar arquiteturas de suporte para soluções de preparação para emergência em aplicações que envolvem um número grande de usuários participantes, bem como áreas geográficas extensas em redes MANETs *(Mobile Ad-Hoc networks)*. Exemplos desse tipo de aplicação incluem: treinamento de forças policiais, militares etc. Três alunos estão envolvidos neste projeto: um aluno de doutorado do laboratório PARADISE (Anis Zarrad) e dois alunos de mestrado do LRVNet no Departamento de Computação da UFSCar (Ricardo Ferrari e este autor). O aluno de doutorado está trabalhando na arquitetura como um todo, enfatizando soluções de redes MANETs para o suporte aos AVCs distribuídos. O aluno do LRVNet, Ricardo Ferrari, está trabalhando no gerenciamento de simulações distribuídas, em conformidade com o HLA, através de *web services*. Este trabalho propôs uma arquitetura de suporte a AVCs, distribuídos em conformidade com o modelo de referência HLA, utilizando uma rede *overlay Peer-to-peer (Gnutella)*.

O objetivo deste trabalho foi criar uma solução tolerante a falhas para simulações distribuídas em conformidade com o padrão HLA, em que usuários, utilizando seu próprio computador conectado à rede *Gnutella*, possam participar de sessões de simulações sem as limitações encontradas em kits de suporte à simulação distribuída, como o Kit RTI existente. No Kit RTI, é necessário conhecer todo o cenário simulado antes de iniciar a simulação e esta configuração deve permanecer até o fim, sendo que, a saída de um federado leva a simulação a parar. Com a utilização da rede *Gnutella*, adiciona-se flexibilidade ao sistema, permitindo sua reconfiguração conforme andamento e dá-se liberdade aos usuários, que podem entrar e sair livremente (ou inadvertidamente) da sessão de simulação. Uma solução foi desenvolvida em que outros nós da rede *Gnutella*, nós especiais denominados Replicadores (sendo estes *ultrapeers*), assumem as informações referentes à respectiva área de simulação do federado que saiu da sessão de simulação ou foi desconectado; assim, os outros participantes não são impedidos de continuar participando da sessão de simulação.

Experimentos foram realizados para avaliação de latência no ambiente RTI, sendo executado em um cluster de computadores, *Beowulf*, com quatro máquinas Dual Xeon. Outros experimentos foram realizados para avaliação de latência observada quando a simulação é feita na rede *Gnutella* e nós contendo parte da simulação são desconectados.

Apesar de ter sido considerado neste trabalho os requisitos estritos de AVCs e o projeto da arquitetura ter sido concebido de modo a atender esses requisitos, o ambiente de visualização considerado foi o 2D - um ambiente 3D de visualização é parte de trabalho futuro do LRVnet.

As próximas seções descrevem a arquitetura proposta e seus componentes, além das adaptações realizadas no Kit RTI de apoio a simulações distribuídas para que este suportasse avatares de velocidades diferentes nos AVCs. Técnica de tolerância a falhas foi proposta para que a simulação seja continuada mesmo com a saída de federados.

5.1 Arquitetura de Suporte a Simulações HLA Distribuídas em Redes *Gnutella*

A partir das motivações apontadas e estudos realizados, foi proposta uma arquitetura de suporte a simulações HLA distribuídas (para AVCs) em redes *Peer-to-peer* (*Gnutella*). Diferentemente da implementação do Kit RTI de suporte a simulações distribuídas em conformidade com o HLA existente, na arquitetura aqui proposta, usuários distribuídos pela Internet são conectados através de uma rede *overlay Gnutella* e constituem-se em usuários participantes de ambientes virtuais compartilhados em que cada um retém parte da simulação distribuída (objetos fixos, avatares, etc.). O conjunto desses usuários forma uma federação.

Nessa solução, usuários subscrevem-se em sessões existentes ou criam sessões e convidam ou divulgam a nova sessão para outros usuários participantes. Uma vez participante de uma sessão, esse usuário recebe atualizações de outros usuários (federados) da rede *Gnutella*. De modo a minimizar o tráfego gerado na rede para a atualização do ambiente de simulação, técnicas de Gerenciamento da Distribuição de Dados (DDM) são utilizadas. Nessas técnicas, federados interessados em receber informações de uma entidade, subscrevem-se naquela entidade e passam a receber informações apenas sobre ela (e não de outras que o usuário eventualmente não tenha interesse). Por sua vez, entidades que desejam ter suas informações difundidas publicam-nas para que interessados possam se subscrever para recebê-las. Como na rede

Gnutella usuários entram e saem livremente da rede, técnicas que garantem a continuidade da simulação mediante a saída de usuários foram concebidas de tal forma que quando um nó da rede é desconectado, nós especiais, os Replicadores (*Ultrapeers*), assumem o papel de responder pela parte da simulação contida na máquina do usuário que acabou de sair de uma sessão de simulação. Para o suporte de tal solução, a arquitetura mostrada na Figura 15 foi concebida e será melhor descrita e avaliada ao longo deste capítulo (a área sombreada denota o foco deste trabalho).

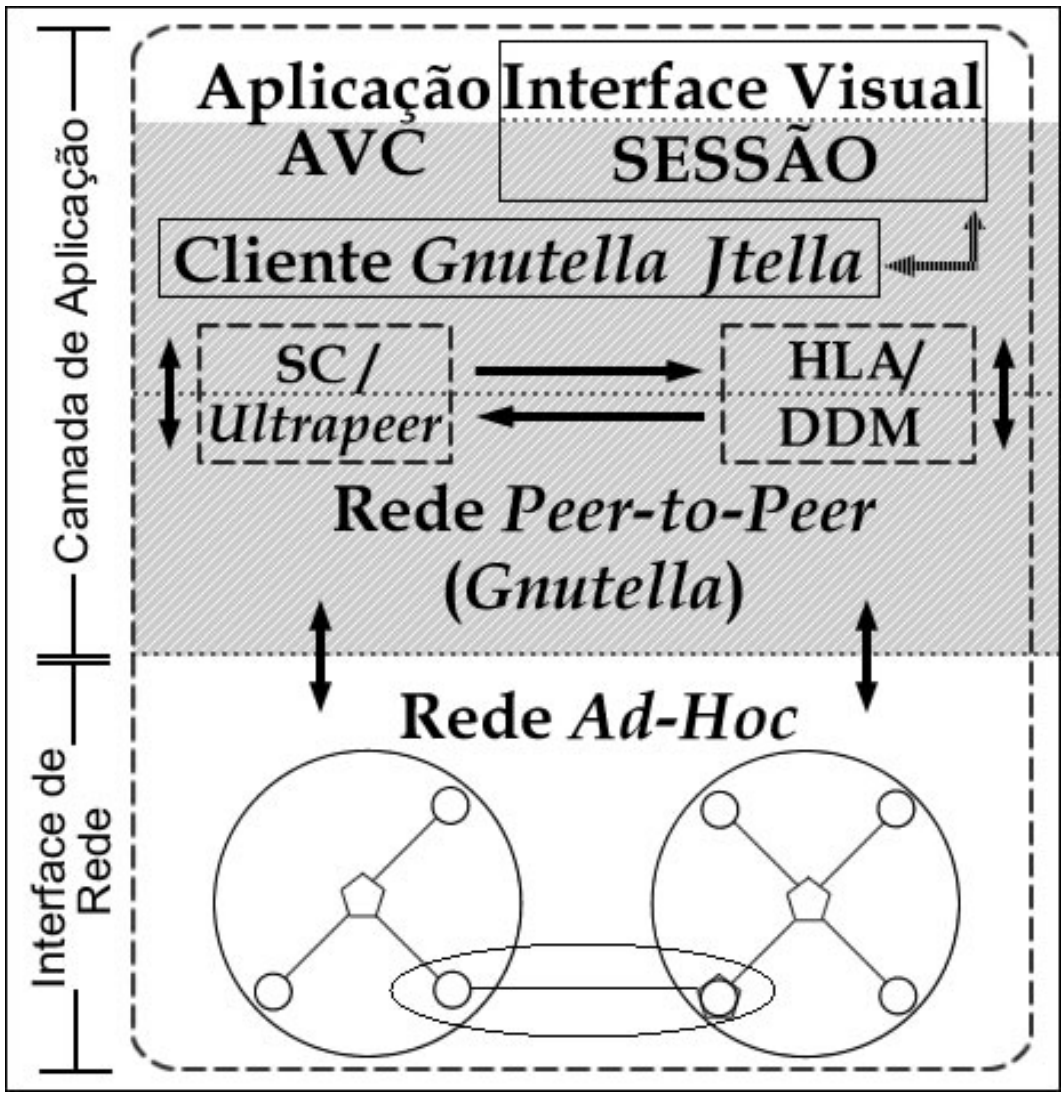

Figura 15. Arquitetura proposta de suporte a simulações distribuídas na rede *Gnutella*.

Para melhor compreensão da arquitetura, os seguintes conceitos são definidos:

- *Zona*: É o ambiente virtual colaborativo em si, ou seja, o cenário e os objetos estáticos que não sofrem mudanças durante a simulação;

- *Sessão*: É a execução de uma zona. Nela estão presentes uma coleção de usuários conectados por uma rede, ou seja, todas as entidades participantes, todos os avatares com suas atuais localizações dentro do ambiente.

Os módulos da arquitetura mostrada na Figura 15 são definidos a seguir:

- Cliente *JTella*: É o *software* cliente da rede *Peer-to-peer Gnutella* utilizado pelos usuários para disponibilizar e buscar sessões de AVCs;
- *Ultrapeer* Controlador de Sessão (*Session Controller* – SC): É o módulo que controla o acesso às zonas numa sessão, isto é, gerencia solicitações de entrada/saída de usuários participantes de uma sessão;
- HLA/DDM: São os algoritmos de gerenciamento de distribuição de dados responsáveis pela filtragem na sincronização e atualização dos dados trocados entre os usuários;
- Rede *Gnutella*: É a rede *Peer-to-peer* utilizada pelos usuários para disponibilizar e buscar sessões de AVCs;
- Rede Móvel *Ad-Hoc* (MANETs): São redes de dispositivos móveis que se comunicam entre si via conexões sem fio sem contar com uma infra-estrutura pré-determinada.

Os *hosts, peers* na rede *Gnutella,* hospedam os AVCs e seus objetos, além de manter cópias atualizadas destes dados em *Ultrapeers*. Esta manutenção sincronizada dos dados, permite que um *host* assuma a posição de outro que saia do sistema (Técnica tolerante a falhas que será detalhada na seção 5.2.4). Vários *hosts* perfazem uma federação. Estes *hosts* estão conectados através de uma rede móvel *Ad-Hoc*, entretanto, uma rede *overlay* (*Peer-to-peer Gnutella* sobre esta rede MANET) é utilizada para a distribuição dos AVCs. A consistência do ambiente, filtragem e distribuição dos dados é gerenciada através de algoritmos DDM.

Para o funcionamento da arquitetura, as seguintes regras foram definidas, bem como as ações que podem ser tomadas pelos *hosts*: quem pode disponibilizar zonas, quem são os controladores e outras.

- Todos participantes sejam eles *Ultrapeers* ou *peers* e independente do tipo de conexão, podem disponibilizar *zonas* através das *sessões*;
- Sessões são buscadas através do *JTella* da mesma maneira que são realizadas buscas de qualquer outro arquivo na rede *Peer-to-peer Gnutella*, via *flooding*.

No *flooding* a requisição de um *peer* é enviada diretamente aos *peers* diretamente conectados a ele e estes por sua vez repetem o processo até que o TTL (*Time to Live* – Tempo de vida) do pacote de requisição, decrementado por cada *peer* que repassa esta requisição, se torne zero (geralmente o TTL é iniciado com sete) [MIL02].

- Se nenhuma sessão estiver ativa e todos participantes que possuem uma zona estiverem desconectados, a zona não será encontrada pelo *JTella* e conseqüentemente não poderá ser baixada ou criada qualquer sessão;
- Se uma sessão for criada por um *Ultrapeer* ele será o responsável pelo controle da sessão (SC/*Ultrapeer*);
- Se uma sessão for criada por um *peer,* o controle será realizado pelo *Ultrapeer* mais proximamente a ele conectado;
- Quando a sessão é criada, o SC/*Ultrapeer* responsável fará um *download,* do disponibilizador da zona, como repositório caso este ainda não a tenha. Para uma maior confiabilidade, criará uma lista de SC/*Ultrapeers* potenciais e esta será atualizada constantemente de maneira que estes possam assumir o controle em caso de falhas ou desconexão. Maiores detalhes em [BOU04a];
- Todos os participantes, *Ultrapeers* e/ou *peers*, fazem o *download* da zona referente à sessão iniciada da qual farão parte, a partir do SC/*Ultrapeers* e outros usuários que já possuam essa zona.Caso o ambiente compreenda uma extensa área geográfica, os participantes farão *download* de apenas uma parte de seu interesse, referente à sua localização no ambiente, definida pelo SC/*Ultrapeer*.

Conforme mencionado, uma aplicação AVC em execução, ou seja, uma sessão, é atualizada constantemente através do gerenciamento de distribuição de dados (HLA/DDM), realizando uma filtragem nos dados trocados entre as entidades participantes da aplicação.

A seguir é apresentado o trabalho detalhado realizado com os componentes que foram integrados à arquitetura, o Kit RTI (HLA/RTI/DDM) e o *JTella*, citando limitações, dificuldades, adaptações realizadas e resultados obtidos.

5.2 Integração dos componentes

Após estudos do Kit RTI (HLA/RTI/DDM) e do *software* cliente da rede *Peer-to-peer Gnutella*, *JTella,* estes foram adaptados e integrados à arquitetura. Tais implementações são descritas a seguir.

5.2.1 O Kit RTI utilizado no Projeto

Conforme descrito no capítulo 4, a HLA não prescreve nenhuma implementação específica, nem estipula o uso de *softwares* ou linguagem de programação em particular para a implementação da especificação da interface HLA implementada pela Infra-estrutura de Tempo de Execução (*Run Time Infrastructure* – RTI). Assim, diferentes implementações do RTI foram realizadas, dentre as quais o Kit RTI desenvolvido originalmente na Geogia Tech e utilizado neste trabalho. Embora atenda parte dos requisitos de simulações distribuídas, incluindo algoritmos de gerenciamento de distribuição de dados, o Kit RTI possui várias limitações, já descritas na seção 3.1.3 do capítulo 3. Duas dessas limitações foram resolvidas e estão descritas na próxima subseção. Primeiramente, o tratamento de diferentes tipos de objetos e posteriormente a variação da velocidade entre esses objetos.

5.2.1.1 Adaptação do Kit RTI para Suporte a Diferentes Objetos de Velocidades Variáveis

A primeira limitação solucionada foi o tratamento de diferentes tipos de objetos. Até então, existia apenas o objeto *Tank*. Uma nova estrutura foi adicionada ao Kit RTI e acrescentada ao objeto; assim, cada objeto carrega agora o seu tipo.

A partir do tratamento de diferentes tipos de objetos, uma outra necessidade surgiu, a variação da velocidade entre os objetos. Como existia apenas um objeto, era justificável apenas uma velocidade, mas com a existência de novos objetos, é necessário também que eles apresentem velocidades diferentes. Adicionou-se então a velocidade à estrutura do objeto. Da mesma maneira, para efeito de testes, atrelamos as velocidades 3m/s, 10m/s e 100m/s para os objetos *Human*, *Tank* e *Aircraft*, respectivamente. O quadro a seguir mostra os trechos mais representativos de códigos alterados/adicionados com essas modificações. A Estrutura do Objeto (*ObjectStat*) recebeu dois novos atributos, uma *string Kind* (tipo do objeto) e um *double Speed* (velocidade do objeto). Randomicamente geram-se os três tipos de objetos e atribuem-se seus tipos e velocidade.

```
/* Modified Object Structure*/
typedef struct{
   int GridID;
   int CellID;
   int ObjID;
   double *Coords;
   string Kind; //Object Kind Added
   double Speed;//Object Speed Added
}ObjectStat;
/*Object Structure Modified*/

/*3 Kind of Objects*/

objectKind = rand () % 3;
if (objectKind == 0){          //Human
   TankStats[i].Kind=”human”;
   TankStats[i].Speed = 3;
}else if (objectKind == 1){    //Tank
   TankStats[i].Kind=”tank”;
   TankStats[i].Speed = 10;
}else{                         //Aircraft
   TankStats[i].Kind=”aircraft”;
   TankStats[i].Speed = 100;
}
/* 3 Kind of Objects*/
```

5.2.1.2 Algoritmos DDM de Gerenciamento de Distribuição de Dados

O Kit RTI utilizado possui alguns algoritmos de gerenciamento de distribuição de dados – DDM implementados. O algoritmo baseado em região foi originalmente implementado na Georgia Tech. Os demais, fixo baseado em grade, dinâmico baseado em grade e híbrido, foram implementados no laboratório PARADISE, *University of Ottawa*. Dentre estes, para a realização das simulações deste trabalho, utilizou-se o algoritmo dinâmico baseado em grade devido ao seu melhor desempenho para o cenário avaliado [BOU02].

5.2.1.3 Dificuldades Encontradas

Na realização deste trabalho, várias dificuldades necessitaram ser superadas, e que, de certa maneira, impactaram de forma negativa no bom andamento do desenvolvimento deste trabalho. Apesar da utilização do Kit RTI ser relativamente fácil, sua instalação é complexa, somado ao fato de não possuir distribuição livre, o que

restringe a presença de *helps* e fóruns, torna o aprendizado para realizar alterações uma tarefa árdua e toma bastante tempo. Além do mais, mesmo dominando a ferramenta, problemas de configuração de rede e/ou *cluster* chegam a impossibilitar a execução das simulações, necessitando uma ação conjunta entre usuários e administradores do *cluster* utilizado para solucionar tais problemas. Estes problemas foram enfrentados tanto no laboratório PARADISE e seu *cluster*, como no laboratório LRVnet e no *cluster* do Departamento de Computação. Pode-se citar os problemas com segurança devido às conexões SSH e RSH efetuadas pelo Kit RTI.

Embora tais problemas tenham surgido, a instalação do Kit RTI no *cluster* do departamento agregou bastante valor ao trabalho, firmando ainda mais a parceria com o laboratório PARADISE e possibilitando a partir disso, o estudo de novas frentes utilizando-se desta tecnologia.

5.2.2 Adaptação do *Software* Cliente *JTella*

Para atender aos requisitos do projeto, o *software* cliente da rede *Gnutella*, *JTella*, necessitou de algumas modificações no que diz respeito ao tratamento de sessão. Embora fosse possível realizar buscas, o *sofware* não reconheceria uma sessão e a trataria como um arquivo qualquer para *download*.

Os arquivos trocados pela rede *Gnutella* recebem um número *hash* único que os identifica. Uma busca é feita pelo nome do arquivo e como resposta têm-se todos arquivos com aquele nome e agrupados segundo seu número *hash*. Para identificar um arquivo de sessão, o número *hash* do arquivo foi alterado de maneira que o *JTella* possa diferenciá-lo. O quadro a seguir mostra o tratamento feito pelo *JTella* na posição de "servidor" quando uma busca por uma sessão é requisitada. A sessão em questão foi chamada de 'HLA'. Quando o "servidor" verifica a existência da sessão de nome 'HLA', ele cria uma mensagem de resposta ao "cliente" enviando o nome completo da sessão, 'HLA/RTI/DDM', o número *hash*, 'urn:sha1:SESSION' e informações sobre a sessão e sobre ele mesmo ("servidor"), como endereço e porta. É a presença da *string* SESSION no número *hash* que possibilita ao *JTella* tratar tais arquivos como sessões. Novas sessões devem ser criadas utilizando o mesmo número *hash* como base variando o nome da sessão.

```
searchReplyMessage searchReplyMessage =
new SearchReplyMessage(searchMessage,(short) 6347,
"200.171.20.141", DownloadConstants.DOWNLOADSPEED_T3,
"JTELLA");

//looking for the session HLA
if(searchMessage.getSearchCriteria().equals("HLA")){
//Create a Search Reply
SearchReplyMessage.FileRecord record =
new SearchReplyMessage.FileRecord(1, 1,"HLA/RTI/DDM",
"urn:sha1:SESSION", 40, 2);
```

Além da identificação dos arquivos de sessão, uma visualização da sessão e suas características por parte de todos participantes também se fazem necessárias. Para tal, foi adicionada uma nova aba no cliente *Jtella*, como pode ser observado na Figura 16. As colunas da esquerda para a direita representam respectivamente: O nome da sessão, o controlador da sessão, o endereço e porta do *Host*, o nome do *Host*, o número de objetos presentes na simulação, o time do qual o *Host* fará parte, a posição de seus objetos quanto a publicação e subscrição e o número de *Hosts* aguardados para iniciar a sessão.

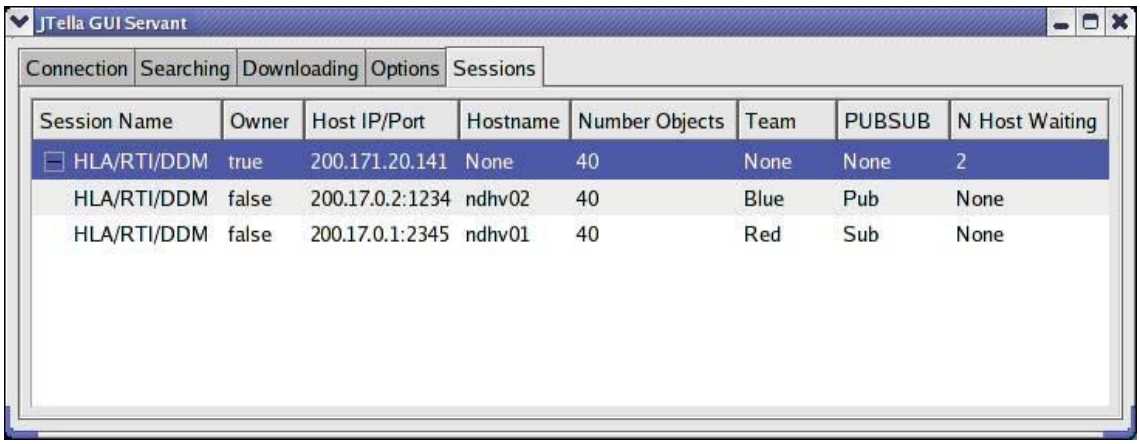

Figura 16. Aba Sessão do *JTella – visão do "servidor"*.

5.2.3 Integração dos Componentes e Funcionamento do Sistema

Após integração dos componentes adaptados, o funcionamento do sistema não se torna uma tarefa difícil, principalmente, para usuários já ambientizados com softwares

de compartilhamento de dados via Internet. Através do *software* cliente da rede P2P *Gnutella*, *JTella*, os usuários realizam buscas na rede procurando por sessões de AVCs-LE. Em caso de sucesso na busca de uma sessão, é retornado aos usuários o controlador desta sessão (tal controlador é explicado ainda nesta seção, na subseção referente ao *JTella*). Os usuários podem então entrar na sessão. A partir deste momento, o gerenciamento de distribuição de dados é responsável pela consistência do ambiente atualizando todos os usuários participantes com o estado atual da sessão e suas atualizações posteriores, de maneira transparente aos usuários, como mostra a Figura 17.

Apesar de a arquitetura evitar que *peers* e *hosts* móveis assumam o controle de sessões, qualquer tipo de *host* pode participar da aplicação AVC e a interrupção da conexão de algum destes pode trazer inconsistência ou até o bloqueio da execução da aplicação. Para tratar este problema, uma técnica tolerante a falhas foi proposta e está descrita a seguir.

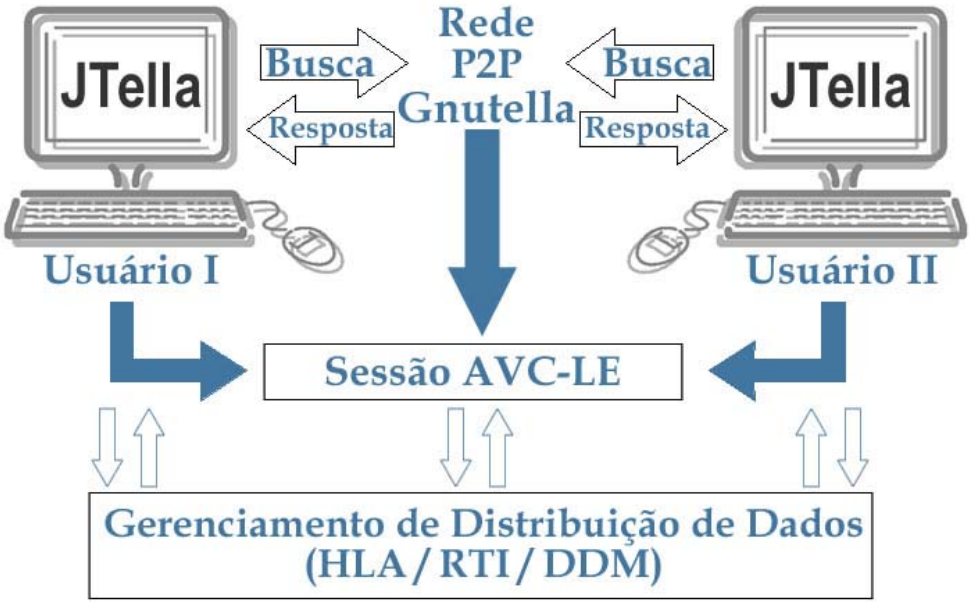

Figura 17. Integração da rede P2P *Gnutella* com o HLA/RTI/DDM.

5.2.4 Tolerância a Falhas

Como visto no capítulo 2, a consistência é um ponto fundamental a ser garantido para o bom funcionamento de AVCs-LE. De maneira a tratar a desconexão de *hosts*

participantes da simulação de AVCs, uma técnica tolerante a falhas foi proposta. Antes de introduzi-la, é interessante entender o conceito de rede *Overlay*. A Figura 18 mostra o encadeamento de redes *Overlay* – arquitetura de rede que executa sobre outra arquitetura de rede [HAU01], presente em simulações de AVCs através da rede *Peer-to-peer*.

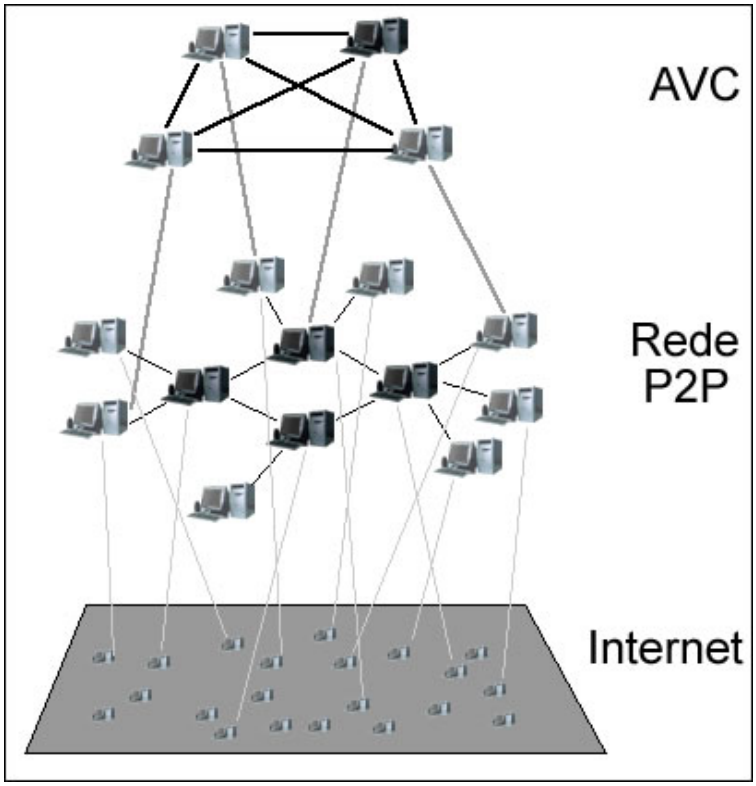

Figura 18. Redes *Overlay.*

A técnica proposta será descrita nos passos seguintes (acompanhe também a Figura 19):

- Todos *hosts* terão informações a respeito do ambiente, replicadas na rede P2P. Sejam essas informações a respeito do ambiente em si ou de objetos manipulados;
- *Hosts peers* terão seus dados replicados no *ultrapeer* a ele diretamente conectado (na Figura 19 são os *Hosts* da rede *Overlay* AVC com dados B, C e D);
- *Hosts ultrapeers* terão seus dados replicados em algum *ultrapeer* a ele conectado (na Figura 19 é o *Host* da rede *Overlay* AVC com dados A) ;

- Os replicadores guardarão também os endereços de todos participantes do AVC, pois caso este venha a assumir uma posição no ambiente, ele poderá estabelecer as conexões necessárias e manter o sistema consistente;
- Um replicador (*ultrapeer*) que por algum motivo se desconecte da rede *Gnutella* terá sua posição assumida por outro *ultrapeer*;
- Um *host* participante do ambiente, seja ele *peer* ou *ultrapeer*, que por algum motivo saia de uma sessão, terá sua posição assumida pelo seu replicador e este, por sua vez, passa a replicar suas informações do ambiente na rede P2P.
- Os *hosts* podem sair do sistema de duas maneiras: normal e anormal. Quando o *host* sai de maneira normal, ele avisa o *ultrapeer* diretamente a ele ligado para assumir sua posição, aguarda resposta e sai. O novo participante estabelece conexão com todos os outros clientes da sessão. Quando o *host* sai de maneira anormal, sua saída é detectada pelo *ultrapeer* a ele ligado através de mensagens de *ping* e após ausência de resposta, assume a posição, estabelecendo conexão com os outros clientes da sessão.

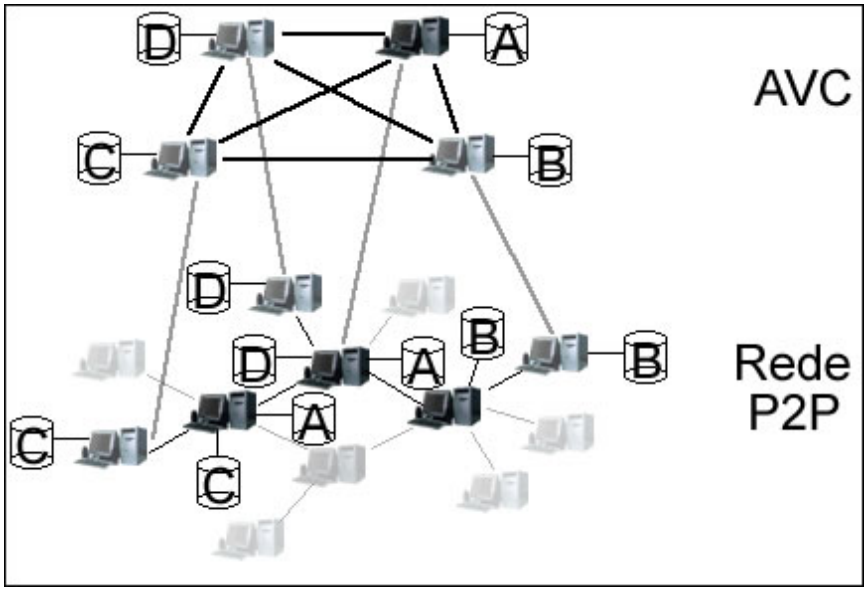

Figura 19. Replicação de Dados.

5.3 Experimentos Realizados

Simulações foram executadas em dois momento do trabalho: no primeiro, foram feitas avaliações de desempenho do Kit RTI original, utilizando objetos de velocidades

iguais, seguido de experimentos do Kit RTI Adaptado, com objetos de velocidade variada, em um *cluster*. O objetivo dessas simulações era de servir de comparativo frente às mesmas simulações na nossa arquitetura *Peer-to-peer* proposta. Num segundo momento, também foram realizadas simulações para avaliação de desempenho da técnica tolerante a falhas proposta quando da saída de um *host* da simulação distribuída. Os ambientes de simulação utilizados e os experimentos realizados são descritos a seguir.

5.3.1 Kit RTI Original *versus* Kit RTI Adaptado

As primeiras simulações comparativas realizadas foram entre o Kit RTI Original, com objetos de apenas uma velocidade, e o Kit RTI Adaptado, com objetos de velocidade variada. Tais simulações foram realizadas no *cluster* do Departamento de Computação da Universidade Federal de São Carlos (Beowulf, quatro máquinas Dual Xeon). Na Tabela 4, são listados os principais parâmetros utilizados na simulação.

Tabela 4. Parâmetros Utilizados nas Simulações.

Número de Federados	4 (1 por máquina Dual Xeon)
Número de Iterações	100
Área do Terreno	500 x 500 m^2
Algoritmo DDM	Dinâmico Baseado em Grade
Número de Times	2 (Azul e Vermelho)
Tempo	Segundos
Units Per Spatial Dimension (UPSD)	50, 100, 200, 500 e 1000
Número de Objetos	200 (50 por federado)

Os gráficos a seguir mostram os resultados obtidos. A federação com objetos de velocidade estável 4 m/s foi representada pela cor clara – rosa, e a federação com objetos de velocidade variável pela cor escura – azul. As velocidades utilizadas foram de 1m/s, 10m/s e 100m/s. Foram avaliados o tempo total de execução da federação, o número total de mensagens *multicast* geradas e o número total de mensagens trocadas pela grade (*grid*) representados pelos Gráficos 1, 2 e 3, respectivamente.

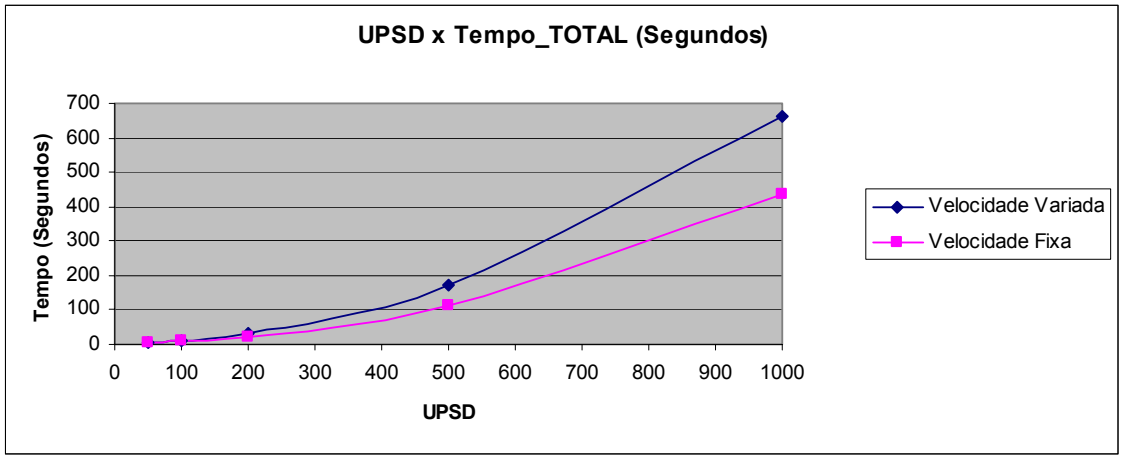

Gráfico 1. Variação do tempo com o aumento de UPSD.

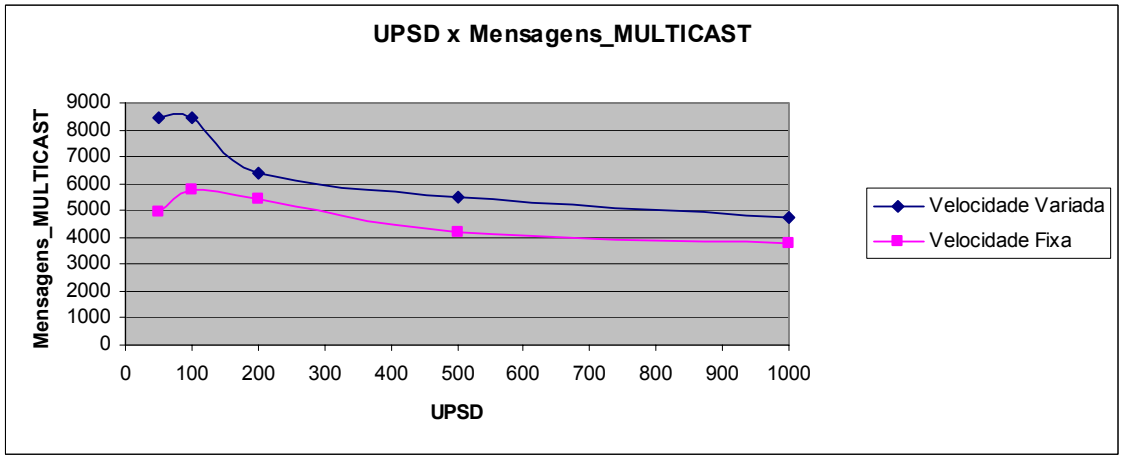

Gráfico 2. Variação do número total de mensagens *multicast* com o aumento de UPSD.

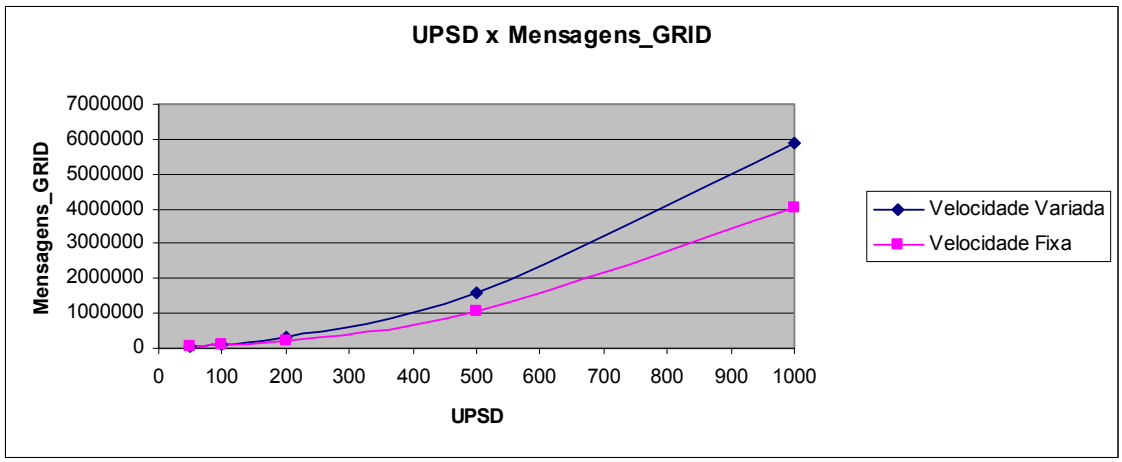

Gráfico 3. Variação do número total de mensagens trocadas pela grade com o aumento de UPSD.

Analisando os gráficos, pode-se ver um aumento considerável principalmente no tempo e no número de mensagens trocadas pela grade. Para compreender bem os resultados dos gráficos, é importante ter a noção que se o número de UPSD aumenta, o

número de células aumenta também e um número maior de células numa área de mesmo tamanho implica em células menores. O aumento no número de mensagens trocadas pela grade aumenta com o aumento do número de células e da velocidade, já que os objetos trocam mais rapidamente de células. O aumento no tempo é diretamente proporcional ao número de mensagens trocadas já que, um maior número de mensagens demanda um maior tempo de entrega e conseqüentemente de finalização da simulação. O número de mensagens *multicast* também aumenta com a velocidade mas não com o aumento do número de células pois, passa a ser mais difícil a presença de objetos publicadores e subscritores dentro de uma célula muito pequena.

Estes resultados justificam a necessidade de algoritmos de gerenciamento de distribuição de dados eficientes para filtragem das mensagens trocadas sem necessidade entre as partes participantes de uma simulação. Nos ambientes virtuais colaborativos de larga escala onde o número de usuários e o tamanho das áreas geográficas podem aumentar significantemente, este requisito de filtragem torna-se cada vez mais importante.

5.3.2 Avaliação de Desempenho do Sistema Quando da Saída de um *Host* da Simulação Distribuída

O sistema teve seu desempenho avaliado quando da saída de um *host* da simulação medindo-se o tempo gasto entre a saída do *host* e sua substituição. As métricas descritas abaixo representam o tempo gasto quando da saída normal e anormal respectivamente de um *host* da simulação.

(1) T_Total = T_notificação + T_conexoes + T_replicador

(2) T_Total = T_ativo + T_ping + T_conexoes + T_replicador

Onde:

- T_notificação – tempo gasto entre o envio da mensagem de saída e a recepção da resposta;
- T_conexoes – tempo gasto no estabelecimento da conexão com os clientes participantes da sessão;
- T_replicador – tempo gasto na definição de um replicador para este novo participante;
- T_ativo – tempo que um replicador aguarda mensagem de seu replicado para avisá-lo que ainda está ativo no sistema;
- T_ping – tempo gasto no envio de duas mensagens de *ping*;

Na realização desta avaliação, utilizaram-se os parâmetros apresentados na Tabela 5.

Tabela 5. Parâmetros Utilizados na Avaliação de Desempenho do Sistema Quando da Saída de um *Host* da Simulação Distribuída.

Número de Máquinas	4 (Intel Pentium IV)
Tempo	Milisegundos
Delay da Rede	50 ms (valor médio quando usuários estão interconectados através da rede academica do estado de São Paulo)
Número de *Hosts* por Sessão	10 a 60 (variando de 10 em 10). Estes foram simulados pelas quatro máquinas.

Nos cenários simulados, uma sessão é definida com um terço do número de *Hosts*. Para cada *Host* é associado um replicador, logo, mais um terço de *Hosts* estão ocupados. Os *Hosts* restantes são replicadores potenciais. Ao desconectar um *Host* da simulação, seu replicador assume sua posição e define à partir dos replicadores potenciais um para ser seu replicador nesta nova configuração do sistema. Essa operação é repetida Número de *Hosts* vezes.

Os resultados da avaliação podem ser vistos no Gráfico 4. Pode-se perceber aumento no tempo com o aumento do número de *hosts* por sessão. Isto ocorre principalmente pelo número maior de novas conexões que um novo participante (antigo *ultrapeer* replicador que está assumindo a posição de algum *host* que saiu) necessita fazer, sendo estas novas conexões as maiores tomadoras do tempo total da substituição. É desejável que os *hosts* saiam de maneira normal, embora isso nem sempre ocorra, dado que T_ativo + T_ping é maior que T_notificação. Embora a saída de um *host* e sua substituição gaste um determinado tempo, ele é quase imperceptível aos participantes de um AVC com até 60 *hosts*. A avaliação se mostrou satisfatória não apenas pelos baixos tempos apresentados, mas, por comprovar o funcionamento da técnica proposta, tolerante a falhas.

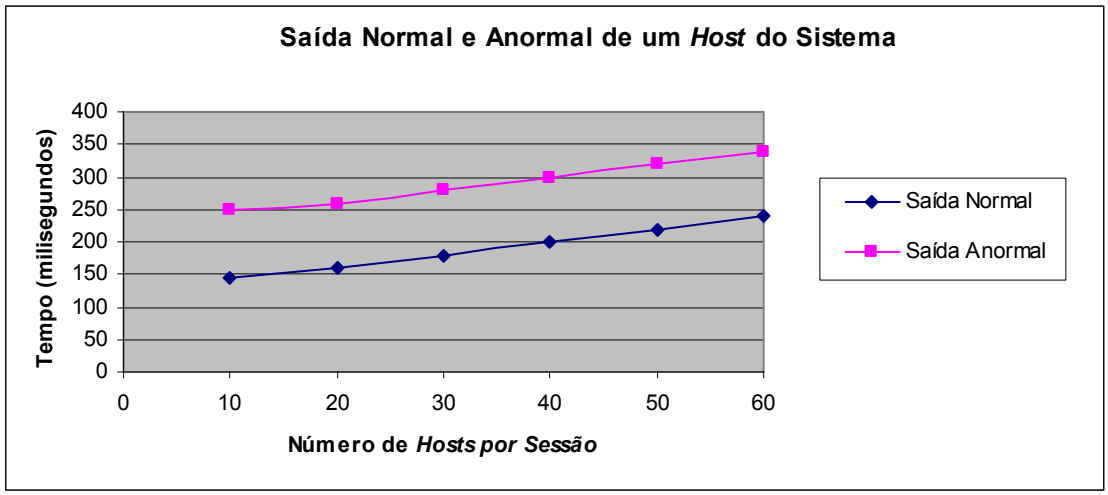

Gráfico 4. Tempo gasto entre a saída do *host* e sua substituição.

5.4 Considerações Finais

O objetivo deste trabalho foi propor uma arquitetura que oferecesse suporte aos AVC-LE, tolerante a falhas, com gerenciamento de distribuição de dados em conformidade com a HLA, padrão comumente utilizado em simulações paralelas e distribuídas e que se utiliza do modelo de comunicação *Peer-to-peer* da rede *Gnutella* para disponibilização e compartilhamento de tais ambientes, sem as limitações encontradas em kits de suporte à simulação distribuída, como o Kit RTI existente. Realizaram-se adaptações no *software* cliente da rede *Gnutella*, *Jtella*, para suportar controle de sessão. O *JTella* foi utilizado também como veículo de compartilhamento dos ambientes. Além disso, realizaram-se adaptações no Kit RTI desenvolvido pela Georgia Tech, com objetos de apenas uma velocidade, para suporte a objetos de velocidade variada, e posteriores simulações comparativas entre ambos em nocluster do Departamento de Computação da UFSCar. A avaliação do tempo total de execução da federação, o número total de mensagens *multicast* geradas e o número total de mensagens trocadas pela grade deu origem a gráficos que mostraram aumentos consideráveis, principalmente, no tempo e no número de mensagens trocadas pela grade. O sistema também teve seu desempenho avaliado quando da saída de um *host* da simulação medindo-se o tempo gasto entre a saída do *host* e sua substituição A avaliação se mostrou satisfatória não apenas pelos baixos tempos apresentados, mas, por comprovar o funcionamento da técnica proposta, tolerante a falhas.

6 Conclusões

Este trabalho descreveu uma solução para simulações distribuídas em redes *overlay* (*Gnutella*), que visa superar as limitações de estruturas de suporte a simulações distribuídas como o Kit RTI, um dos mais conceituados kits de suporte a simulações distribuídas baseadas no modelo HLA de referência.

6.1 Contribuições Geradas

- Melhor compreensão do Kit RTI com a sua instalação e adaptação no cluster de 16 computadores existentes no Departamento de Computação da UFSCar (agradecemos aos professores do GAPIS – Grupo de Arquitetura, Processamento de Imagens e Sinais, pela autorização de uso deste cluster);
- fortalecimento da colaboração entre o Laboratório de Realidade Virtual em Rede (LRVnet), do Departamento de Computação da UFSCar e o laboratório PARADISE, do SITE da *University of Ottawa*, Canadá, uma vez que este trabalho é parte de um trabalho maior envolvendo a interação entre alunos dos dois laboratórios.

Artigo submetido:

- Boukerche, A; Araujo, R.B.; Heredia-Vieira, N. D. Supporting HLA Distributed Simulations in Peer-to-peer Gnutella Network, submetido ao 10th DS-RT 2006, 2-6 October 2006, Torremolinos, Málaga, Spain.

6.2 Trabalhos Futuros

Algumas limitações do Kit RTI foram identificadas, algumas solucionadas, outras ficam como sugestão para trabalhos futuros, assim como o tratamento do ambiente visual, em destaque:

- A variabilidade no número de objetos simulados em cada *host*;
- O posicionamento, por parte dos usuários, dos objetos na simulação;
- A possibilidade de interrupção da simulação e continuação posterior;

- A possibilidade de alteração, após inicializada, da simulação, por exemplo, no número de objetos simulados;
- A implementação da técnica tolerante a falhas proposta permitindo a desconexão de *hosts* participantes da simulação;
- A divisão e controle do ambiente por entidades com maiores recursos;
- A orientação a eventos por parte da simulação;
- O tratamento do ambiente visual e de ambientes 3D.

6.3 Conclusões finais

Ambientes Virtuais Colaborativos de Larga Escala podem beneficiar uma infinidade de aplicações, como treinamentos colaborativos, shoppings virtuais, comunidades virtuais, aplicações de preparação para emergências, treinamento de forças policiais (para atuação em campos de futebol, shows etc), corpo de bombeiros, militares e outras. Nestes ambientes, extensas áreas 3D sintéticas são compartilhadas entre un número grande de usuários.

Para oferecer suporte aos estritos requisitos de simulações de ambientes virtuais colaborativos de larga escala, que exigem tolerância a falhas e baixa latência, neste trabalho propôs-se integrar um conjunto de tecnologias amplamente estudadas e comumente utilizadas. Em sua realização, várias dificuldades necessitaram ser transpostas, e que, de certa maneira, impediram um desempenho maior na implementação da arquitetura proposta. Contudo, implementações setoriais foram feitas, procurando contribuir com as pesquisas realizadas do projeto "*Emergency Preparedness*", envolvendo o Laboratório de Realidade Virtual em Rede (LRVnet), do Departamento de Computação da UFSCar e o laboratório PARADISE, do SITE da University of Ottawa, Canadá. O objetivo desse projeto é pesquisar arquiteturas de suporte para soluções de preparação para emergência em aplicações que vão de treinamento a monitoramento de ambientes cientes de contexto, em redes MANETs.

Realizaram-se adaptações no *software* cliente da rede *Gnutella*, *Jtella*, para suportar controle de sessão. O *JTella* foi utilizado também como veículo de compartilhamento dos ambientes. Além disso, realizaram-se adaptações no Kit RTI desenvolvido pela Georgia Tech, com objetos de apenas uma velocidade, para suporte a objetos de velocidade variada, e posteriores simulações comparativas entre ambos em um cluster. A avaliação do tempo total de execução da federação, o número total de

mensagens *multicast* geradas e o número total de mensagens trocadas pela grade deu origem a gráficos que mostraram aumentos consideráveis, principalmente, no tempo e no número de mensagens trocadas pela grade. O sistema também teve seu desempenho avaliado quando da saída de um *host* da simulação medindo-se o tempo gasto entre a saída do *host* e sua substituição. A avaliação se mostrou satisfatória não apenas pelos baixos tempos apresentados, mas, por comprovar o funcionamento da técnica proposta, tolerante a falhas.

Referências Bibliográficas

[ABR98] Abrams, H.; Watsen, K.; Zyda, M. Three-Tiered Interest Management for Large-Scale Virtual Environments. ACM Symposium on Virtual Reality Software and Technology, 1998.

[ALL97] Allen, R.; Garlan, D. Formal Modeling and Analysis of the HLA RTI. Proceedings of the 1997 Spring Simulation Interoperability Workshop, Orlando, FL, March 1997. p. 1153 – 1161.

[AND04] Androutsellis-Theotokis S.; Spinellis, D. A survey of Peer-to-peer Content Distribution Technologies. ACM Computing Surveys (CSUR) v. 36 Issue 4 , December 2004. p. 335 – 371.

[BHA05] Bhatia, K. Peer-To-Peer Requirements On The Open Grid Services Architecture Framework. Technical report, Global Grid Forum (GGF) GFD.49, July 2005. p.1 – 38.

[BEA05] Beatty, S. A. P2P Multi-Source Downloading for Gnutella. Thesis (BSc Computer Science), Lancaster University, Lancaster, UK, March 2005.

[BER98] Berrached, A.; Alternative Approaches to *Multicast* Group Allocation in HLA Data Distribution Management. 98S-SIW-198. Spring Simulation Interoperability Workshop (SIW), March 1998.

[BIL04] Bildson, G. Proposal: Handshaking Protocol, The Gnutella Developer Forum (GDF), August 2001.
Disponível em URL:
http://groups.yahoo.com/group/the_gdf/message/2010.
Consultado em 12/11/2004.

[BOU04a] Boukerche, A; Araujo, R.B.; Laffranchi, M. A Hybrid Solution to Support Multiuser 3D Virtual Simulation Environments in Peer-to-Peer Networks, DS-RT 2004, 2004. p. 61 – 68.

[BOU04b] Boukerche, A.; Dzermajko, C. Scalability and Performance Evaluation of An Aggregation/Disaggregation Scheme for Data Distribution Management in Large-Scale Distributed Interactive Systems. In: 37th Annual Simulation Symposium, Arlington, Virginia, April 2004. p. 238.

[BOU05] Boukerche, A.; McGraw N.J.; Dzermajko, C.; Lu, K. Grid-Filtered Region-Based Data Distribution management in Large- Scale Distributed Simulation Systems. In: 38th Annual Simulation Symposium (ANSS'05), San Diego, California, USA, April 2005. p. 259 – 266.

[BOU02] Boukerche, A.; Roy, A. Dynamic Grid-Based Approach to Data Distribution Management. Journal of Parallel and Distributed Computing, v. 62, 2002. p. 366 – 392.

[BOU00] Boukerche A.; Roy, A. In Search of Data Distribution Management in Large Scale Distributed Simulations. In: Summer Computer Simulation Conference (SCSC), Vancouver. The Society for Modeling and Simulation International (SCS), IEEE, 2000.

[CAL96a] Calvin, J.O.; Hook, D.J.V. AGENTS: An Architectural Construct to Support Distributed Simulation, 11 Workshop on Standards for the Interoperability of Distributed Simulations (DIS), September 1994.

[CAL96b] Calvin, J.O.; Weatherly, R. An Introduction to the High Level Architecture (HLA) Runtime Infrastructure (RTI). 14th Workshop on Standards for the Interoperability of Distributed Simulations, Orlando – USA, 1996. p. 705 – 715.

[CIG02] Ciglaric, M.; Vidmar, T. Position of Modern Peer-to-peer Systems Architecture. Conference Theme Communications and Information Technology for Development IEEE. MELECON, 7-9 May 2002.

[DAH97a] Dahmann, J.S. High Level Architecture for Simulation, 1st International Workshop on Distributed Interactive Simulation and Real-Time Applications (DIS-RT '97) Eilat, Israel, 1997. p. 9 – 14.

[DAH97b] Dahmann, J. S.; Fujimoto, R. M.; Weatherly, R. M. The Department of Defense High Level Architecture. Proceeding of the 1997 Winter Simulation Conference, December 1997. p. 142 – 149.

[DEP98] Department of Defense. High Level Architecture Interface Specification, Version 1.3, Defense Modeling and Simulation Office – DMSO, 1998.
Disponível em URL: http://hla.dmso.mil.
Consultado em 20/01/2005.

[DIA97] Dias, J.M.S.; Galli,R.; Almeida, A.C.; Bello, C.A.C.; Rebordão, J.M. mWorld: A multiuser 3D virtual environment. IEEE Computer Graphics and Applications, Mar – April 1997. p. 55-65.

[FER95] Ferscha, A. Parallel and Distributed Simulation of Discrete Event Systems. Contributed to the: HandBook of Parallel and Distributed Computing, McGraw-Hill, 1995. p. 1 – 65.

[FUJ98] Fujimoto, R.; Hoare, P. HLA RTI Performance in High Speed LAN Environments. Proceedings of the Fall Simulation Interoperability Workshop, Orlando – USA, September 1998. p. 501 – 510.

[FUJ00] Fujimoto, R.; Mclean, T.; Perumalla, K.; Tacic, I. Design of High Performance RTI Software. IEEE Distributed Interactive Simulation and Real-Time Applications, DS-RT'2000, August 2000.

[FUJ01] Fujimoto, R. Parallel and Distributed Simulation Systems. Proceedings of the 33nd Winter Simulation Conference, Arlington – USA, December 9 – 12, 2001. p. 147 – 157.

[GEN01] Genome@Home. 2001.
Disponível em URL: http://genomeathome.stanford.edu.
Consultado em 22/11/2004.

[GEO06] Georgia Tech, Parallel and Distributed Simulation, RTI Implementation.
Disponível em URL:
http://www-static.cc.gatech.edu/computing/pads/tech-highperf-rti.html
Consultado em 12/01/06.

[GNU04] Gnutella Protocol Specification, The Gnutella 0.4. Document Revision 1.2.
Disponível em URL:
http://www9.limewire.com/developer/Gnutella_protocol_0.4.pdf.
Consultado em 11/11/2004.

[GNU06] Gnutella Protocol Specification, The Gnutella 0.6.
Disponível em URL: http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html
Consultado em 11/11/2004.

[GON02] Gong, Li. Guest Editor's Introduction: Peer-to-Peer Networks in Action. IEEE Internet Computing, vol. 06, no. 1, January-February 2002. p. 37-39.

[GRE95] Greenhalgh, C.; Benford, S. MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading, in Proceedings of the 15th International Conference on Distributed Computing Systems (DCS'95), Vancouver, Canada, May 30-June 2, 1995. p. 27 – 34.

[GRE04] Greenhalgh, C. Large Scale Collaborative Virtual Environment. Nottingham, 1997. Thesis Doctor of Philosophy. University of Nottingham p. 209.

[HAU01] Hausheer, D.; Darlagiannis, V.; Stiller, B.; Strulo, B.; Efstathiou, E.; Pierce, M.; Simpson, S. Market Management of Peer-to-peer Services (MMAPPS), Design, Scalability, and Application of Peer-to-peer Overlay Networks. European Fifth Framework Project IST-2001-34201, 31 August 2001.

[HOK98] Hook, D.J.V.; Calvin, J.O. Data Distribution Management in RTI 1.3. 98S-SIW-206, Spring SIW, 1998.

[HOS04] Hosseini, M.; Georganas, N.D. End System *Multicast* Protocol for Collaborative Virtual Environments Presence: Teleoperators and Virtual Environments, v. 13, I.3, June 2004. p. 263 – 278.

[JOS03] Joslin, C.; Pandzic, I. S.; Magnenat-Thalmann, N. Trends in Networked Collaborative Virtual Environments, Computer Communications, v. 26, I.5, March 2003. p. 430 – 437.

[JXT04] JXTA, Project.
Disponível em URL: http://www.jxta.org.
Consultado em 28/11/2004.

[KUB03] Kubiatowics, J. Extracting Guarantees from Chaos. Communications of the ACM. Vol. 46. N 2, February 2003.

[LIM04] Limewire.Org.
Disponível em URL: http://www.limewire.org.
Consultado em 11/11/2004.

[LIV05] LivingWorlds.
Disponível em URL: http://medialab.it.fht-esslingen.de/da/goetz/da_goetz/ Anhang/Living-Worlds_Draft/draft_2/index.htm.
Consultado em 03/12/2005.

[MCC00] Mccrary, K. The Jtella Java API for the Gnutella Network, October, 2000.
Disponível em URL: http://jtella.sourceforge.net/
Consultado em 03/08/2005.

[MEY04] Meyers, D. A Multi-Source Download Capable Gnutella Servent. Thesis (BSc Computer Science), Lancaster University, Lancaster, UK, March 2004.

[MIL02] Milojicic, D.S.; Kalogeraki, V.; Lukose, R.; Nagaraja, K.; Pruyne, J.; Richard, B.; Rollins, S.; Xu, Z. Peer-to-peer Computing. HP Laboratories Palo Alto, HPL-2002-57, 8 March, 2002.

[MOL02] Moloney, J. StringCVE: Introducing a Virtual Design Studio Utilizing the Torque Game Engine. Proceeding of the 20th conference on Education in Computer Aided Architectural Design in Europe, ISBN 0 9541183 0 8, Warsaw, Poland, 18-21 September 2002. p. 522 – 525.

[MOL03] Moloney, J.; Amor, R. StringCVE: Advances in a game engine-based collaborative virtual environment for architectural design. Proceedings of CONVR 2003 Conference on Construction Applications of Virtual Reality, Blacksburg, USA, 24-26 September 2003.

[MOR97] Morse, K.L.; Steinman, J.S. Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering. Proceedings of SIW, Spring 1997.

[OLI04] Oliveira, D. Aplicações Peer-to-peer. Trabalho de Graduação para a disciplina Redes de Computadores II – Universidade Federal do Rio de Janeiro – 2002.
Disponível em URL:
http://www.gta.ufrj.br/~rezende/cursos/eel879/trabalhos/p2p/
Consultado em 28/11/2004.

[OLI00] Oliveira, J.C.; Shen X.; Georganas, N.D. Collaborative Virtual Environments for Industrial Training and e-Commerce. Proc. Workshop on Application of Virtual Reality Technologies for Future Telecommunication Systems, IEEE Globecom'2000 Conference, November – December 2000, San Francisco.

[OPE04] OpenNap: Open Source *Napster* Server.
Disponível em URL: http://opennap.sourceforge.net/.
Consultado em 26/11/2004.

[PHE04] Phex Development Team.
Disponível em URL: http://phex.kouk.de/mambo/
Consultado em 11/11/2004.

[RIT04] Ritter, J. Why *Gnutella* Can't Scale. No really.
Disponível em URL: http://www.darkridge.com/~jpr5/doc/*Gnutella*.html.
Fevereiro de 2001.
Consultado em 14/11/2004.

[SAN99] Santivañez, C; Stavrakakis, I. A Framework for a Multi-mode Routing Protocol for (MANET) Networks. IEEE Wireless Communications and Networking Conference WCNC'99, New Orleans, LO., September, 1999.

[SCH03] Schoder, D.; FischBach, K. Peer-to-peer Prospects. Communications of the ACM, February 2003.

[SCH01] Schollmeier, R. A. Definition of Peer-to-peer Networking for the Classification of Peer-to-peer Architecture and Applications. First International Conference on Peer-to-peer Computing, Suécia, 27 – 29 August, 2001.

[SEI99] Seidel, D.W.; Testani, S.; Wagner E. CIMBLE: Distributed Team Training via HLA. SIMULATION Vol. 73 N. 5, 1999. p. 304 – 310.

[SEI95] Seila, A. F. Introduction to Simulation. Proceeding of the 1995 Winter Simulation Conference, Arlington USA, December 3 – 6, 1995. p. 7 – 15.

[SET04] SETI@home: The Search for Extraterrestrial Intelligence.
Disponível em URL: http://setiathome.ssl.berkeley.edu/.
Consultado em 21/11/2004.

[SHE01] Shen X.; Nourian S.; Hertanto I.; Georganas N. vCOM: Virtual Commerce in a Collaborative 3D World. Proceedings of the ninth ACM international conference on Multimedia, 2001. p. 605 – 606.

[SIN99] Singhal, S.; Zyda, M. Networked Virtual Environments: Design and Implementation, ACM Press, New York, 1999.

[STE99] Steed, A.; Frecon E. Building and Supporting a Large-Scale Collaborative Virtual Environment. Proceedings of 6th UKVRSIG, University of Salford, 13th - 15th September 1999. p. 59 – 69.

[STE87] Stefik, M.; Bobrow, D. G.; Foster, G.; Lanning S.; Tatar, D. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. ACM Transactions on Office Information Systems, 5(2), ACM Press, 1987. p. 147 – 167.

[TAN00a] Tan, G.; Xu, L.; Moradi, F.; Zhang, Y. An Agent-based DDM Filtering Mechanism. In: Proceedings Eighth IEEE/ACM International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. p. 374 – 381.

[TAN00b] Tan, G.; Zhang, Y.; Ayani, R. A Hybrid Approach to Data Distribution Management. In: Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications, 2000. p. 55 – 61.

[UNI04] United Devices Cancer Research Project.
Disnonível em URL: http://www.grid.org/projects/cancer/.
Consultado em 21/11/2004.

[ZAI04] Zaiantchick, J. B. Análise sobre o impacto da tecnologia peer-to-peer.
Disponível em URL:
http://www.ime.usp.br/~is/ddt/mac339/projetos/2001/demais/bello.
Consultado em 07/11/2004.