Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

“Uma solução *Peer-to-Peer* para o gerenciamento da distribuição de dados baseada na Arquitetura de Alto Nível HLA”

RICARDO CESAR CÄMARA FERRARI

DEZEMBRO/2007

“Uma solução *Peer-to-Peer* para o gerenciamento da distribuição de dados baseada na Arquitetura de Alto Nível HLA”

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

"Uma solução *Peer-to-Peer* para o gerenciamento da distribuição de dados baseada na Arquitetura de Alto Nível HLA"

Ricardo Cesar Câmara Ferrari

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.

DEZEMBRO /2007

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

F375sp

Ferrari, Ricardo Cesar Câmara.
Uma solução *peer-to-peer* para o gerenciamento da distribuição de dados baseada na arquitetura de alto nível HLA / Ricardo Cesar Câmara Ferrari. -- São Carlos : UFSCar, 2009.
92 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2007.

1. Sistemas distribuídos. 2. Simulação de sistemas. 3. Serviços da web. 4. Arquitetura de redes de computador. 5. Peer-to-peer. I. Título.

CDD: 005.43 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

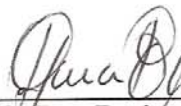
Programa de Pós-Graduação em Ciência da Computação

"Uma Solução Peer-to-Peer para o gerenciamento da distribuição de dados baseada na Arquitetura de Alto Nível HLA"

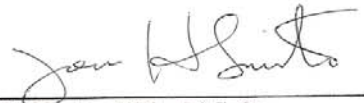
RICARDO CESAR CÂMARA FERRARI

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)

Prof. Dr. Aleardo Manacero Junior
(IBILCE/UNESP)

São Carlos
Dezembro/2007

Você aprenderá lições. Você está matriculado numa escola informal de período integral chamada vida. A cada dia nesta escola, terá a oportunidade de aprender lições. Você poderá gostar dessas lições ou considerá-las irrelevantes.

Dedico este trabalho:

À minha querida mãe, Terezinha Ferrari, pelo amor, incentivo e dedicação, sempre acreditando no meu sucesso.

Ao meu pai, Antonio Ferrari, certamente orgulhoso por mais um importante passo em minha vida.

Às minhas irmãs, Rose, Janete e Raquel, que sempre torceram para que tudo desse certo.

Às minhas avós, Sebastiana e Ida, por estarem sempre presentes quando precisei.

Aos meus queridos sobrinhos Mateus, Paula, Carol e Flávia, pelos quais tenho um amor imenso.

Agradecimentos

A Deus, acima de tudo e todos, por todas as oportunidades que tive e ainda terei em minha vida;

A meus pais, minhas irmãs, meus sobrinhos, minhas avós, meus cunhados, meus tios Ilso e Vilma e meus primos Rodolfo e William, pessoas sem as quais eu não seria o que sou hoje;

À profª. Drª. Regina Borges de Araújo, pela motivação, encorajamento, amizade e direção durante meus dois anos de estudos na pós-graduação;

A todos os meus familiares pela confiança irrestrita;

Aos meus amigos que passaram por momentos difíceis junto comigo durante essa etapa da minha vida sempre estando do meu lado, mesmo quando distantes;

A todos meus colegas do LRVNet, Leandro, Altieres, Fernando, Diego, Bruno, Fábio, Marcio, Igor, Rodolfo, Rafaela e José Eduardo, pelas discussões que contribuíram para a realização de nossos trabalhos;

Aos professores do Departamento de Computação da UFSCar, principalmente àqueles com quem tive mais/maior contato durante meus dois anos de estudos.

Resumo

Simulações paralelas e distribuídas podem ser utilizadas para criar ambientes virtuais colaborativos que envolvem seres humanos em aplicações de treinamento, entretenimento, etc. A Arquitetura de Alto Nível (*High Level Architecture* - HLA) desenvolvida pelo Escritório de Modelagem e Simulação de Defesa (DMSO) do Departamento de Defesa Norte-Americano – DoD e adotada pelo IEEE para modelagem e simulação de alto nível é um *framework* de padronização de simulações que visa, basicamente, o reuso e a interoperabilidade de/entre simulações. A HLA compreende os seguintes serviços: Gerenciamento de Federação (conjunto de federados), Gerenciamento de Declaração, Gerenciamento de Objetos, Gerenciamento de Posse, Gerenciamento de Tempo e Gerenciamento de Distribuição de Dados (DDM). A RTI (*Run-Time Infrastructure*) é a implementação da especificação de interface da HLA, cujo objetivo principal é separar a comunicação da simulação. Uma versão da RTI (kit RTI versão 1.3 do Instituto de Tecnologia da Georgia - Georgia Tech) foi instalada no *cluster* do DC da UFSCar. Esta versão consiste de um conjunto de bibliotecas de suporte a simulações de tempo-real distribuídas. O problema com esta versão da RTI é que ela possui várias limitações, dentre elas, a falta de suporte à interação. De modo a superar essas limitações uma arquitetura foi desenvolvida e avaliada, como parte deste trabalho, que utiliza a plataforma JXTA para iniciar federados (simulações) e federações (conjunto de federados) em tempo de execução por meio de serviços *web*. Esta arquitetura visa gerenciar a distribuição de dados entre os federados, um dos serviços mais importantes de simulações distribuídas.

Palavras chaves: Simulação distribuída, web services, HLA, JXTA.

Abstract

Distributed and parallel simulations can be used to create virtual collaborative environments that involve human beings in applications of training, entertainment, etc. The High Level Architecture - HLA, developed by the Defense Modeling and Simulation Office (DMSO) of the North American Department of Defense - DoD and adopted by IEEE for modeling and simulation of high level, is a framework of simulation standardization that basically aims at the reuse and interoperability of/among simulations. The HLA comprises the following services: Federation Management, Declaration Management, Object Management, Ownership Management, Time Management and Distribution of Data Management (DDM). RTI (Run-Time Infrastructure) is the implementation of the interface specification of HLA, whose main objective is to separate the communication from the simulation. A version of the RTI (RTI kit version 1.3 from Georgia Institute of Technology - Georgia Tech) was installed in the cluster of the Computer Department of the Federal University of São Carlos. This version consists of a set of support libraries for real-time distributed simulation. The problem with this version of the RTI is that it has many limitations, such as lack of support for human interaction. In order to overcome these limitations, an architecture was developed and evaluated, as part of this work, that uses the JXTA platform to initiate federates (simulations) and federations (set of federates) at runtime web services. This architecture aims to manage the distribution of data among federates, one of the most important services for distributed simulations.

Keywords: Distributed simulations, web services, HLA, JXTA.

Índice de figuras

Índice de tabelas

Lista de abreviaturas e siglas

AOL:	America Online
API:	Application Program Interface
COM:	Component Object Model
CORBA:	Commom Object Request Broker Architecture
DC:	Departamento de Computação
DCOM:	Distributed Component Object Model
DDM:	Data Distribution Management
WS-	
DDM:	Data Distribution Management Web Service
DM:	Declaration Management
DMSO:	Defense Modeling and Simulation Office
DOD:	US Department Of Defense
FOM:	Federation Object Model
HLA:	High Level Architecture
HTML:	Hypertext Markup Language
HTTP:	Hypertext Transfer Protocol
IBM:	International Business Machines Corporation
IDL:	Interface Definition Language
IEEE:	Institute of Electrical and Electronics Engineers
IP:	Internet Protocol
IRC:	Internet Relay Chat
JVM:	Java Virtual Machine
JXTA:	Justapose
LRVNet:	Laboratório de Realidade Virtual em Rede
NPSNet:	Naval Postgraduate School Network
OLE:	Object Linking and Embeddign
OMG:	Object Management Group
OMT:	Object Model Template

ORB:	Object Request Broker
ORPC:	Object Remote Procedure Call
P2P:	Peer-to-Peer
POC:	Point Of Contact
RMI:	Runtime Intrastructure
RO:	Receive Order
RPC:	Remote Procedure Calls
RTI:	Runtime Infrastructure
SGML:	Standard Generalized Markup Language
SIMNet:	Simulator Network
SOA:	Service Oriented Architecture
SOAP:	Simple Object Access Protocol
SOM:	Simulation Object Model
TCP:	Transmission Control Protocol
TSO:	Time Stamp Order
TTL:	time-to-live
UDDI:	Universal Description Discovery and Integration
UDP:	User Datagram Protocol
UFSCAR:	Universidade Federal de São Carlos
URI:	Unique Resource Identifier
UUID:	universal unique identifier
WSDL:	Web Service Description Language
WSIM:	Web Service Interest Management
XML:	Extensible Markup Language

Sumário

1. Introdução

O desenvolvimento de simulações em larga escala iniciou-se na área militar para apoiar o treinamento e simulações de guerra. Dentre os exemplos de sistemas desse tipo merece destaque o SIMnet, desenvolvido pelo Departamento de Defesa Americano (DoD), projetado e implementado entre 1983 e 1990 [Mil 95], [Pop 89], e o projeto NPSNET [Mac 95a, Mac 95b]. A Arquitetura de Alto Nível (*High Level Architecture)* (HLA), desenvolvida pelo Escritório de Modelagem e Simulação de Defesa (DMSO) do Departamento de Defesa Norte-Americano – DoD e adotada pelo IEEE para modelagem e simulação de alto nível, é um *framework* de padronização de simulações que visa, basicamente, o reuso e a interoperabilidade de/entre simulações.

A HLA é formada por três componentes básicos: regras da HLA, OMT (*Object Model Template*) e especificação de interface. A especificação de interface é dividida em sete serviços: gerenciamento de federação, gerenciamento de declaração, gerenciamento de objeto, gerenciamento de posse, gerenciamento de tempo, gerenciamento de distribuição de dados e serviços de suporte. O serviço de Gerenciamento de Distribuição de Dados *(Distribution of Data Management* - DDM) é um dos mais importantes, pois promove a comunicação entre federados (simulação) de uma federação (conjunto de federados). O DDM será discutido em detalhes por se tratar do serviço que foi implementado e avaliado neste trabalho, através de comunicação *peer-to-peer*.

1.1. Motivação

A princípio a HLA foi desenvolvido para modelagem de simulações de treinamento militar, mas, por ser um padrão que permite reuso e interoperabilidade entre simulações, acabou sendo adotado pelo IEEE, podendo ser usado em qualquer área de aplicação de simulações distribuídas [Vas 01], [Rei 2000].

Com a forte tendência de usar-se a *web* para executar simulações em tempo real [Ant 07], uma alternativa para fazer isso sem depender de software/hardware para a troca de mensagens pela Internet é usar tecnologias padronizadas, abertas e estáveis, nas quais os

serviços *web* são bons exemplos. Serviços *web* são considerados como solução em potencial para ser utilizada na integração de sistemas e na comunicação entre diferentes aplicações. Serviços web utilizam o protocolo SOAP *(Simple Object Access Protocol)* para trocar mensagens que são codificadas em XML, enviadas através de HTTP e que são descritas por meio de *Web Services Description Language* (WSDL). Outra tecnologia utilizada neste trabalho é o JXTA, que define uma plataforma para desenvolvimento de redes P2P *(Peer-to-Peer)* focalizada no uso eficiente dos recursos da Internet. O P2P procura resolver o gargalo gerado em redes centralizadas usando a cooperação de pares que realizam vários serviços.

Com o conceito de rede P2P, cada computador (nó) pode ser considerado como sendo um federado tendo a liberdade de passar novos parâmetros, podendo ter qualquer tipo de comportamento (imprevisível) controlado pelo usuário deste computador. Para acesso à simulação e interação o usuário precisa apenas conectar-se à Internet e possuir, em seu computador, o serviço *web* que permitirá sua participação efetiva na simulação e, então, começar a trocar informações entre os usuários participantes com os protocolos do JXTA. Entretanto, para que um participante da simulação possa interagir com a mesma, ele deve ter suporte para interferir na simulação. Os serviços disponíveis aos participantes que foram implementados são relacionados ao Gerenciamento de Distribuição de Dados – DDM, para suporte à interação e comunicação entre todos os participantes envolvidos, de maneira otimizada (a informação é trocada apenas entre participantes que têm interesses comuns, resultando na filtragem de uma grande quantidade de dados indesejáveis).

A motivação deste trabalho surge, principalmente, em decorrência das limitações existentes na versão da RTI do DC da UFSCar, a saber: suporta apenas o mesmo número de objetos simulados em todos os *hosts* participantes; suporta apenas um tipo de objeto de simulação; suporta apenas uma velocidade de objeto; não suporta persistência de dados; não suporta interação *(man-in-the-loop)*; a configuração inicial não pode ser mudada ao longo da simulação. De modo a superar algumas dessas limitações uma arquitetura é proposta, implementada e avaliada que suporta interação e um número ilimitado de objetos através do JXTA e serviços *web* em substituição ao cluster do DC da UFSCar.

1.2. Objetivos e Organização do Trabalho

O objetivo desse trabalho é a implementação e avaliação de um serviço de gerenciamento de distribuição de dados (DDM), baseado no padrão HLA, por meio de uma

rede *P2P*. Com isso, usuários participantes de simulações distribuídas poderão acessar e interagir com o ambiente de simulação, através da *web*. Este trabalho está organizado da seguinte maneira: o Capítulo 2 apresenta os conceitos de Simulação Paralela e Distribuída. O Capítulo 3 apresenta o padrão HLA, a especificação da Infra-Estrutura de Tempo de execução (RTI) e os seis serviços que esta oferece. O Capítulo 4 descreve em detalhes o serviço de Gerenciamento de Distribuição de Dados, que é parte da arquitetura implementada neste trabalho de mestrado. O Capítulo 5 descreve soluções de suporte à comunicação distribuída, como o JXTA e serviços *web*. No Capítulo 6 o JXTA é apresentado de forma detalhada, seguido do Capítulo 7 que descreve, também de forma mais detalhada, serviços *web*. A arquitetura de suporte a simulações distribuídas que foi desenvolvida neste trabalho é apresentado no Capítulo 8. A avaliação de desempenho da arquitetura é discutida no Capítulo 9, seguido de Conclusões e Referências Bibliográficas.

2. Simulação

Conceitualmente, a simulação modela um sistema do mundo real através de um programa de computador. A simulação permite que um sistema seja modelado em qualquer nível de detalhe: de uma tradução direta de um modelo de redes de filas à captura de todo o aspecto do comportamento do sistema. A simulação suporta qualquer coleção de métricas de desempenho que possa ser definida.

Originando de pesquisas básicas iniciadas nas décadas de 1970 e 1980, o campo de simulação paralela e distribuída tem amadurecido muito em poucas décadas. Hoje as simulações estão sendo usadas para aplicações como treinamento militar, análise de tráfego em redes de computadores e controle de tráfego aéreo [Ric 03].

A simulação é a geração de um histórico artificial de um sistema e a observação deste histórico para dedução de operações características do sistema real que está sendo representado. A simulação é uma metodologia indispensável para resolver problemas do mundo real. Ela é usada para descrever e analisar o comportamento de um sistema respondendo a questões do tipo “O que aconteceria com o sistema” e dando apoio ao projeto do sistema do mundo real. Sistemas conceituais e existentes, ambos podem ser modelados por meio de simulação [Jer 99].

Com os avanços na área de informática, modernos equipamentos e novas linguagens de programação e de simulação têm permitido empregar técnicas de simulação nas diversas áreas do conhecimento humano, fatos que têm propiciado: (a) projetar e analisar sistemas industriais, (b) avaliar performance de hardware e software em sistemas de computação, (c) analisar desempenho de armas e estratégias militares, (d) determinar freqüência de pedidos de compra para recomposição de estoques, (e) projetar e administrar sistemas de transportes como portos e aeroportos, e (f) configurar sistemas de atendimento em hospitais, supermercados e bancos, etc.

A implementação de modelos matemáticos de simulação a serem utilizados em computadores requer o uso das linguagens de programação como FORTRAN, C e PASCAL ou das linguagens de simulação como SLAM, GPSS, GASP, POWERSIM, ARENA e EXTEND.

2.1. Tipos de Modelos de Simulação

De modo geral, podem-se distinguir dois tipos de simulações, tendo em conta o modo de construir uma réplica do sistema, sendo eles [Sis 06]: 1) Os *modelos de escala,* nos quais, como o próprio nome sugere, o sistema é substituído por uma versão que é uma escala do próprio sistema, isto é, uma espécie de maquete é construída, sobre a qual experiências serão realizadas. Este método é muitas vezes usado quando se pretende simular o comportamento de grandes estruturas, como pontes, barragens, turbinas e aviões; 2) Os *modelos conceituais,* onde o sistema é substituído por uma versão conceitual do seu funcionamento, isto é, por um determinado conjunto de equações matemáticas ou relações funcionais de causa-efeito.

2.2. Modelos Conceituais

Os modelos matemáticos de simulação, ou simplesmente modelos de simulação, podem ser classificados em [Sim 06a]:

- Estáticos ou dinâmicos - denominam-se como modelos estáticos os que visam representar o estado de um sistema em um instante ou que em suas formulações não se leva em conta a variável tempo, enquanto os modelos dinâmicos são formulados para representar as alterações de estado do sistema ao longo da contagem do tempo de simulação;

- Deterministas ou estocásticos - são modelos determinísticos os que em suas formulações não fazem uso de variáveis aleatórias (a simulação ocorre em passos e não há noção de tempo), enquanto os estocásticos podem empregar uma ou mais variáveis aleatórias;

- Contínuos ou discretos - para os modelos contínuos o avanço da contagem de tempo na simulação dá-se de forma contínua, o que possibilita determinar os valores das variáveis de estado a qualquer instante. Nestes modelos os valores das variáveis podem mudar continuamente ao longo do tempo. Já os modelos discretos são aqueles em que o avanço da contagem de tempo na simulação se dá na forma de incrementos cujos valores podem ser definidos em função da ocorrência dos eventos ou pela determinação de um valor fixo. Nesses casos só é possível determinar os valores das variáveis de estado do sistema nos instantes de atualização da contagem de tempo. As variáveis continuam inalteradas ao longo de intervalos de tempo e mudam seus valores somente em momentos bem definidos.

2.3. Modelos Contínuos x Modelos Discretos

Como foi visto anteriormente nas definições dos modelos contínuos e discretos, nos modelos contínuos o estado do sistema muda continuamente em função do tempo. Utilizam-se geralmente equações diferencias para descrever iterações entre diferentes elementos do sistema. Exemplos: crescimento de populações, circuitos eletrônicos e reações químicas. Diferentemente dos modelos contínuos, nos modelos discretos o estado do sistema muda somente quando ocorre um evento. Para todos os demais instantes de tempo, nada muda no sistema. Esses sistemas podem ser modelados como filas de espera. Este tipo de simulação é conhecido como simulação de eventos discretos [Sim 06b], e é ilustrado no Figura 1.

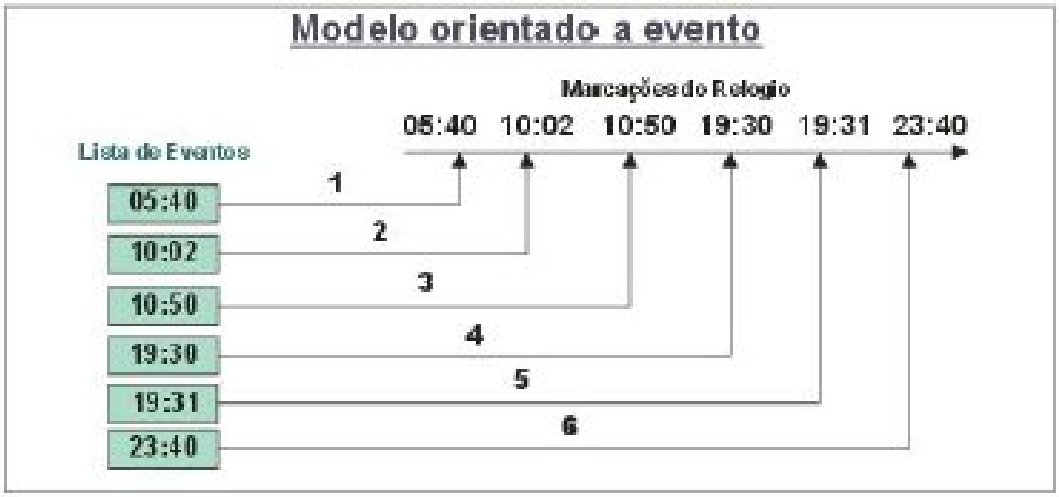


Figura 1 - Modelo orientado a evento

2.4. Simulação de Eventos Discretos

A simulação orientada a evento utiliza uma lista na quais os eventos são armazenados por ordem cronológica de acontecimento. Os eventos vão sendo disparados no meio simulado de forma seqüencial, fazendo com que a representação do tempo seja feita apenas em função do atributo tempo de ocorrência dos eventos. Caso estes eventos não possuam uma variável explícita de tempo de acontecimento, eles devem ser organizados simplesmente pela ordem de acontecimento. O relógio é incrementado com o atributo tempo do evento a cada evento disparado. O evento, após ser disparado, pode criar conseqüências ou alterações no estado do sistema, necessitando que outros eventos precisem ser disparados logo em seguida ou em uma ordem diferente da usual.

O avanço do tempo em simulações orientadas a evento acontece de forma discreta, o que permite que a evolução do tempo seja efetuada aos saltos, e não de uma forma contínua.

2.5. Simulação Paralela e Distribuída

Constantemente o conceito de Simulação Distribuída é confundido com Simulação Paralela da mesma forma que Processamento Distribuído é confundido com Processamento Paralelo. Esses conceitos estão intimamente ligados e, desta forma, é necessária a compreensão das diferenças entres computadores de processamento paralelo e computadores de processamento distribuído para a compreensão das diferenças entre as correspondentes simulações. Por exemplo, o atraso de comunicação entre as máquinas é a propriedade mais importante para diferenciar computadores paralelos e distribuídos. Este atraso é tipicamente de alguns micro-segundos a dezenas de mili-segundos em computadores paralelos e, muito maior em computadores distribuídos principalmente devido às longas distâncias físicas que o sinal deve atravessar e à complexidade do software de protocolos de comunicação necessário para se efetivar a conexão.

Simulações executadas em computadores com vários processadores são definidas como programas de simulação paralela, já simulações executadas em computadores distribuídos são definidas como simulação distribuída. A simulação distribuída é usada com objetivos analíticos ou mais popularmente para a construção de ambientes virtuais distribuídos.

2.6. Exemplos de Sistemas distribuídos

Em [Ryc 05] é apresentado o projeto e a prática de um sistema que suporte a execução da arquitetura de alto nível (HLA) em um ambiente de grade não confiável. O *framework* do sistema é dividido em três partes, sendo *Broker Support services* que define decisões iniciais na execução da simulação, *Migration Support Services* para tolerância a falhas e HLA *Management services* onde a biblioteca HLA é definida como um serviço de grade. [Kat 04] descreve um sistema em grade para monitorar aplicações baseadas em HLA para permitir o balanceamento de carga pela migração de federados.

Em [Pul 04] é apresentado uma extensão de *web services* para ambientes de simulação distribuído com HLA, integrando simulações heterogêneas em um ambiente em grade e [Wyt 03] apresenta uma solução inicial pra ligar os padrões, *web service* e HLA para um gerenciamento de simulação.

Já [Mor 04] propõe uma arquitetura chamada WSIM (*Web Services Internet Management*) compatível com HLA para gerenciamento de interesse, dividida em três funções, RBAC (*Role Based Access Control*), AOIM (*Area of Interest Management*) e AGIM (*Aggregation Interest Management*). [Cor 05] pretende explorar como uma infra-estrutura Peer-to-Peer pode ser usada para aumentar a disponibilidade de *web services.* Foi feita uma implementação desta infra-estrutura usando tecnologias das áreas w*eb services* e *Peer-to-Peer*, como a plataforma jxta.

Por fim, em [Ekl 04] é descrito o DRMS (*Decentralized Resource Management System)* baseado no jxta, que usa uma rede de computadores para executar e armazenar federações/federados em um ambiente p2p.

2.7. Considerações Finais

Introduzidos os principais tipos de simulações, de escala e conceituais, foram apresentados modelos de simulação conceituais dando ênfase aos modelos dinâmicos e discretos, sendo que o modelo discreto teve uma atenção maior, pois é o modelo que melhor reflete os ambientes virtuais colaborativos. A atualização dos atributos da simulação é feita apenas quando da ocorrência de um evento, seja este engatilhado por um objeto da simulação ou pela interação de um usuário. O próximo capítulo apresenta a Arquitetura de Alto Nível HLA, que permite o reuso e a interoperabilidade entre diferentes simulações. A solução de gerenciamento da distribuição de dados em simulações distribuídas proposta, implementada e avaliada como parte deste trabalho foi baseada na arquitetura HLA.

3. A Arquitetura de Alto Nível HLA – *High Level Architecture*

Muitas simulações complexas envolvem simulações individuais de diferentes tipos de sistemas, combinadas a outros aspectos do ambiente geral para serem simuladas. Normalmente, as simulações de alguns desses componentes já existem, tendo sido desenvolvidas para propósitos diferentes. Para ser possível reusar essas simulações, seria necessário fazer modificações extensivas para adaptar o componente no modelo da simulação de modo que este pudesse ser integrado a uma nova simulação, de forma combinada. Em alguns casos pode ser mais fácil programar uma simulação completamente nova de um componente do sistema do que modificar um já existente. Em outras palavras, nos modelos de simulações tradicionais geralmente faltam duas propriedades desejáveis: reusabilidade e interoperabilidade [Int 06].

A arquitetura de alto nível HLA é um *framework* que orienta um desenvolvedor a modelar uma simulação de forma padronizada, possibilitando o reuso dessa simulação, combinando-a com outra qualquer, desde que essa outra também esteja no padrão HLA, permitindo assim que estas sejam integradas em uma única simulação [Wyt 03].

3.1. Reusabilidade

Com a reusabilidade modelos de componentes da simulação podem ser reusados em diferentes cenários e aplicações de simulações. Junto à reusabilidade está a propriedade de interoperabilidade, na qual componentes reusáveis da simulação podem ser combinados com outros componentes sem a necessidade de recodificação [Tan 05].

3.2. Interoperabilidade

A interoperabilidade implica em combinar componentes de simulação em diferentes tipos de plataformas de forma distribuída, geralmente em tempo real. Componentes de simulação interagem entre si em um único programa tradicional, em um ambiente único

(não distribuído). Em outras palavras, a interoperabilidade consiste em um número de programas executando em computadores distribuídos de diferentes tipos e interagindo uns com os outros de forma distribuída [Tay 05].

3.3. Componentes Básicos da HLA

A HLA é constituído basicamente de três componentes:

- Regras da HLA - definem princípios e convenções que devem ser seguidas pelos federados (partes da simulação com um ou mais objetos) e pelas federações (formadas por vários federados interagindo) para que se tenha uma interação adequada.
- OMT *(Object Model Template)* - provê um *framework* comum para a documentação do modelo de objetos da HLA.
- Especificação de interface RTI *(Runtime Intrastructure)* - define como os seis serviços da HLA (Gerenciamento de Federação, Gerenciamento de Declaração, Gerenciamento de Objetos, Gerenciamento de Posse, Gerenciamento de Tempo e Gerenciamento de Distribuição de Dados) são acessados.

As regras da HLA, o modelo de objetos da HLA e a especificação da interface RTI são descritos em detalhe a seguir.

3.3.1. Regras da HLA

As regras para federações são as seguintes [Iee 00a]:

- As federações terão um HLA FOM (*Federation Object Model*), documentado de acordo com o HLA OMT. Todos os dados a serem trocados de acordo com a HLA serão documentados em um FOM. Um FOM documentará o acordo entre federados sobre os dados a serem trocados usando os serviços de HLA durante a execução da federação e o conjunto mínimo das condições de troca de dados (por exemplo, atualizações a serem emitidas quando as mudanças excedem algum

valor). Portanto, um FOM é um elemento essencial para definir uma federação.

- Em uma federação, toda a representação do objeto associado à simulação estará no federado, não na RTI. Na HLA, a responsabilidade para manter os valores de atributos do objeto da HLA ocorrerá no federado. Em uma federação HLA, todos os atributos dos federados associados serão propriedades dos federados, não da RTI. Entretanto a RTI pode possuir atributos associados como modelo de objeto de gerenciamento da federação. A RTI pode usar informações sobre atributos e interações para suportar serviços RTI, mas essas informações são usadas somente pela RTI.
- Durante a execução de uma federação, toda troca de dados entre federados ocorrerá via RTI. Serviços RTI serão usados para assegurar que a necessidade da coordenação de aplicações distribuídas esteja de maneira consistente através de todos os participantes em uma federação.
- Durante a execução de uma federação, federados interagirão com a RTI de acordo com a especificação de interface HLA. A HLA fornece uma especificação para uma interface padrão entre federados e RTI. Federados usarão esta interface padrão para acessar serviços RTI. A especificação definirá como federados interagem com a infra-estrutura. Entretanto, a interface e a RTI serão usadas por uma grande variedade de aplicações que requerem troca de informações de diversas características.
- Durante a execução de uma federação, um atributo será possuído por, no máximo, um federado em um instante de tempo. Isto é necessário para assegurar coerência nas informações através da federação. Um federado pode solicitar que a posse do atributo seja requerida ou negada, dinamicamente, durante a execução da federação. Esta posse pode ser transferida dinamicamente durante a execução, de um federado para outro.

As regras para federados são as seguintes [Iee 00a]:

- Os federados terão um HLA SOM (*Simulation Object Model*), documentado de acordo com o HLA OMT. O HLA SOM incluirá classes de objetos, classes de atributos e classe de interação de federados que pode ser publicada em uma federação. A HLA não prescreve qual informação é incluída no SOM, pois isto será responsabilidade do desenvolvedor de federados.
- Os federados serão capazes de atualizar e/ou censurar algum atributo e enviar e/ou receber interações, como especificadas em seus SOMs. A HLA permite aos federados fazer representações de objetos internos e interações disponíveis para uso externo como parte da execução da federação. Esta capacidade para interações externas será documentada no SOM para o federado. Se documentado no SOM, esta capacidade do federado incluirá a obrigação para exportar valores atualizados de atributos que são internamente calculados no federado.
- Os federados serão capazes de transferir e/ou aceitar posse de atributos dinamicamente durante a execução de uma federação, como especificado em seus SOMs. HLA permite posse de atributos de um objeto para serem transferidos dinamicamente durante a execução de uma federação. O atributo de um federado que pode ser também possuído ou censurado, cuja posse pode ser dinamicamente adquirida ou negada durante a execução, será documentado no SOM pelo federado.
- Os federados serão capazes de variar as condições sob as quais eles fornecem atualizações de atributos, como especificado em seus SOMs. A HLA permite aos federados possuir atributos de objetos representados nos federados e então deixar os valores disponíveis para outro federado através da RTI. Diferentes federações podem especificar diferentes condições sob as quais os atributos serão atualizados. As condições aplicáveis para atualização de atributos específicos possuídos por um federado serão documentadas no SOM pelo federado.
- Os federados serão capazes de gerenciar o tempo local de uma maneira que permitirá coordenar trocas de informações com outros membros da federação. Os desenvolvedores de federações identificarão o

gerenciamento de tempo delas como parte de seu projeto de implementação.

3.3.2. OMT (*Object Model Template*)

O OMT especifica que os modelos de objetos HLA devem ser documentados na forma de um número de tabelas que informa sobre as classes de objetos, seus atributos, suas interações e seus parâmetros.

As especificações OMT esperam que federações e simulações individuais (federados) usem todos os componentes OMT, embora algumas tabelas possam ser desconsideradas em alguns casos [Hla 06].

Um *framework* estrutural padronizado, ou *template*, é um componente essencial para especificar modelos de objetos HLA pelas seguintes razões [Iee 00b]:

- Fornece um mecanismo de fácil compreensão para especificar a troca de informações e coordenação geral entre os membros de uma federação.
- Oferece um mecanismo padronizado para descrever o potencial dos membros da federação.
- Facilita o projeto e a aplicação de ferramentas para desenvolvimento de modelos de objetos HLA.

O modelo de objetos pode ser usado para descrever um membro de uma federação individual (federado), criando um modelo de objeto de simulação (SOM), ou para descrever um conjunto de federados interagindo (federações), criando um modelo de objeto da federação (FOM). Em ambos os casos, o principal objetivo do *template* do modelo de objeto HLA (OMT) é facilitar a interoperabilidade entre simulações e reuso das simulações.

Durante o desenvolvimento de uma federação HLA, é importante que todos os membros da federação alcancem um entendimento comum nas comunicações requeridas entre os federados participantes. O objetivo principal de uma HLA FOM é fornecer uma especificação para troca de dados entre federados em um formato padronizado.

Um passo importante para a formação de uma federação é o processo de determinação da composição do federado. Um HLA SOM é uma especificação de tipos de

dados que um federado pode fornecer para federações HLA e informações que um federado pode receber de outros federados em federações HLA [Iee 00b].

O OMT *(template* do modelo do objeto) é formado por quatorze componentes [Hla 06]:

- Tabela de identificação de modelo de objeto.
- Tabela de estrutura de classe do objeto.
- Tabela de estrutura de classe de interação.
- Tabela de atributo.
- Tabela de parâmetro.
- Tabela de dimensão.
- Tabela de representação de tempo.
- Tabela de caracteres fornecidos pelo usuário.
- Tabela de sincronização.
- Tabela de tipo de transporte.
- Tabela de transferência.
- Tabela de tipo de dados.
- Tabela de aviso.
- Dicionário do FOM/SOM.

Esses componentes são descritos a seguir.

3.3.2.1. Tabela de Identificação de Modelo de Objeto

A Tabela de Identificação de Modelo de Objeto é uma tabela na qual é definido o modelo do objeto como FOM/SOM. Quando um desenvolvedor de federações precisa de informações detalhadas sobre como um federado ou uma federação foi construído, utiliza esta tabela como fonte.

3.3.2.2. Tabela de Estrutura de Classe do Objeto

A estrutura das classes de objetos é como um diagrama de classes na qual a classe de objeto é uma coleção de objetos com certas características ou atributos em comum. Se Aviões é superclasse de Caça e Caça é superclasse de F-22 Raptor, então Aviões é

superclasse de F-22 Raptor e F-22 Raptor é subclasse de Caça. Observe a tabela 2 na qual “s” e “p” representam subscrever e publicar.

Tabela 1 - Tabela de identificação de modelo de objeto

Categoria	**Informação**
Nome	ExemploEsquadrão
Tipo	SOM
Versão	1.0
Data de Modificação	07/08/2006
Propósito	Exemplo de um modelo de objeto para um esquadrão de aviões
Domínio da Aplicação	Operações do esquadrão
Responsável	Esquadrão
POC	Ricardo Ferrari
Organização do POC	LRVNet
Telefone do POC	3351-8625
E-mail do POC	ricardo@ufscar.br
Referencias	www.avioes.com.br
Outros	Veja guia dos aviões para mais informações

3.3.2.3. Tabela de Estrutura de Classe de Interação

Uma interação é definida como uma ação de um federado que pode ter algum efeito sobre outro federado dentro da federação. Estas interações serão especificadas na tabela de estrutura de classe de interação do modelo do objeto HLA nos limites de seus relacionamentos de classe para subclasse, na qual são definidas as interações de cada classe, sendo que todas as classes de interação terão uma superclasse Raiz de Interação HLA. Logo, quanto maior a derivação, maior será a especificação da classe determinando sua capacidade de subscrição e publicação.

3.3.2.4. Tabela de Atributo

A tabela de atributos é atualizada pela RTI e fornece seus dados para outros federados da federação. A tabela de atributos de um FOM descreve todos os atributos representados em uma federação, Já a tabela de um SOM descreve todos os atributos que são publicados e subscritos pelo federado. Observe a tabela a seguir sendo que N/A corresponde a Negar e Adquirir.

Tabela 2 - Tabela de estrutura de classe do objeto

<table>
<tr><td rowspan="11">Raiz do objeto HLA (N)</td><td rowspan="11">Aviões (P/S)</td><td rowspan="3">Caça (P/S)</td><td>AMX/A1 (P/S)</td></tr>
<tr><td>F-22 Raptor (P/S)</td></tr>
<tr><td>Mirage III (P/S)</td></tr>
<tr><td rowspan="3">Cargueiro (S)</td><td>AK (P/S)</td></tr>
<tr><td>AKR (P/S)</td></tr>
<tr><td>AKA (P/S)</td></tr>
<tr><td rowspan="3">Comercial (S)</td><td>Corvette (P)</td></tr>
<tr><td>Falcon 2000 (P)</td></tr>
<tr><td>Falcon 10 (P)</td></tr>
<tr><td>Espião (S)</td><td></td></tr>
</table>

3.3.2.5. Tabela de Parâmetro

Para cada classe de interação, o conjunto completo de parâmetros associados a esta interação está na tabela de parâmetros, na qual uma classe de interação herda parâmetros da superclasse Raiz de interação HLA.

Tabela 3 - Tabela de estrutura de classe de interação

Raiz de interação HLA (P/S)	Transação de caças (P/S)	Atacar (P)	Metralhadora (P)
		Receber ordens (S)	Base aérea (P)
			Comandante (P)
		Modo de ataque (S)	Aéreo (S)
			Terrestre (S)

3.3.2.6. Tabela de Dimensão

Na tabela de dimensões cada atributo ou interação que pode usar serviços DDM terá um conjunto de dimensões disponíveis, como indicado na tabela de atributos ou tabela de parâmetros. Cada conjunto de dimensões disponíveis é um subconjunto de todas as dimensões que estão disponíveis na tabela de dimensões e define um sistema de coordenadas multidimensionais no qual federados manifestam interesse em publicar ou subscrever informações. Estes sistemas de coordenadas indicam áreas de interesse por regiões, sendo que estas regiões podem ser de dois tipos: regiões de subscrição e regiões de publicação.

3.3.2.7. Tabela de Representação de Tempo

A HLA fornece serviços de gerenciamento de tempo que permite múltiplos federados com estratégias diferentes de progresso de tempo interno para uma interoperabilidade mais significativa. Com isso, é importante para todos os membros da federação a definição de como será representada a marca de tempo através da federação e a

documentação desse acordo na federação. A tabela de representação de tempo é usada para esse propósito.

Tabela 4 - Tabela de atributo

	Atributo	**Tipo de dado**	**Tipo de atualização**	**Condição de atualização**	**N/A**	**P/S**	**Dimensões disponíveis**	**Transporte**	**Ordem**
Raiz do objeto HLA	Privilégio HLA para apagar objetos	N/A	N/A	N/A	N	N	N/A	Confiável	Tempo marcado
Caça	Tamanho	Metros	Estático	Estático	N/A	P	N/A	Confiável	Tempo marcado
	Anos de uso	Data	Periódico	Anos + 1	N/A	P	10 anos e 5 anos	Confiável	Tempo marcado
	Peso	Quilos	Estático	Estático	N/A	P	N/A	Confiável	Tempo marcado

3.3.2.8. Tabela de Caracteres Fornecidos pelo Usuário

A RTI fornece um mecanismo para federados armazenarem dados em conjunto com certos serviços da HLA. Esses caracteres podem ser usados para permitir uma coordenação e controle adicional sobre esses serviços. A tabela de caracteres fornecidos pelo usuário provê um recurso de documentação dos acordos com a federação em relação ao tipo de dado a ser usado com esses caracteres.

3.3.2.9. Tabela de Sincronização

Na HLA RTI também existe um mecanismo de sincronização de atividades usando uma característica chamada ‘pontos de sincronização’. A tabela de sincronização disponibiliza um meio para que um federado descreva os pontos de sincronização que ele é capaz de cumprir e para a federação documentar acordos relativos a pontos de sincronização para serem usados.

Tabela 5 - Tabela de parâmetro

Interação	**Parâmetro**	**Tipo de dado**	**Dimensões disponíveis**	**Transporte**	**Ordem**
Aviões.	Velocidade	Km/h	Mirage III	Confiável	Marcador
	Exatidão	Coordenadas			

Míssil	Capacidade de destruição	Coordenadas			de tempo

Tabela 6 - Tabela de dimensão

Nome	**Tipo de dado**	**Lim. Sup. dimensão**	**Função de normalização**	**Valor *default***
Míssil	Beirute	4	Linear (míssil, 1, 4)	1
Metralhadora	Antiaérea	1000	Linear (metralhadora, 1, 1000)	100

Tabela 7 - Tabela de representação de tempo

Categoria	**Tipo de dado**	**Semântica**
Marca de tempo	Tipo de tempo	Valor de ponto flutuante em minutos
Antecipar	Constante	Valor de ponto flutuante em minutos

Tabela 8 - Tabela de caracteres fornecidos pelo usuário

Categoria	**Tipo de dado**	**Semântica**
Atualiza/descarta	N/A	N/A
Envia/recebe	N/A	N/A
Apaga/remove	ASCII	Razão para exclusão
Pedido de retirada	Nível de prioridade	Valor para transferência imediata
Completar retirada	N/A	N/A
Pedir aquisição	Nível de prioridade	Valor para transferência imediata
Antecipar	N/A	N/A

Tabela 9 - Tabela de sincronização

Rótulo	**Caractere de tipo de dados**	**Capacidade**	**Semântica**

Publicação inicial	N/A	Registra ponto de sincronização	Ativado quando todas as classes são publicadas e subscritas e todos os objetos atuais são registrados
Atualização inicial	N/A	Registra ponto de sincronização	Ativado quando os valores dos atributos para todos os objetos atuais são atualizados
Pausar execução	Tempo	Ativa e registra ponto de sincronização	Ativado quando o avanço do tempo após o tempo no caractere fornecido pelo usuário for alcançado; os pedidos de avanço de tempo então parariam.

3.3.2.10. Tabela de Tipo de Transporte

A tabela de tipo de transporte fornece meios para o programador de federados descrever os tipos de transportes que podem ser suportados, além de meios para que programadores de federados possam documentar acordos relativos a transporte de atributos e interações.

Tabela 10 - Tabela de tipo de transporte

Nome	**Descrição**
Confiável	Fornece entrega confiável de dados como TCP/IP entrega os dados com confiança
Melhor esforço	Faz um esforço para entregar dados como UDP fornece a entrega de melhor esforço

3.3.2.11. Tabela de Transferência

A tabela de transferência permite a especificação do ajuste inicial de cada transferência, indicando ou não se a RTI deve executar estas ações.

Tabela 11 - Tabela de transferência

Transferência	Direção
Fornecer automaticamente	Desabilitado
Grupo de designadores de região de transporte	Desabilitado
Espaço do atributo consultivo	Habilitado
Relevância do atributo consultivo	Habilitado
Relevância da classe do objeto consultivo	Habilitado
Relevância da interação consultiva	Habilitado
Relatar serviços	Desabilitado

3.3.2.12. Tabela de Tipo de Dados

Várias tabelas OMT fornecem colunas para especificar tipos de informações. As características destes tipos de informações são colocadas em tabelas adicionais.

Tabela 12 - Tabela de tipo de dados

Nome	Representação	Unidades	Resolução	Precisão	Semântica
HLAASCIIchar	HLAoctet	N/A	N/A	N/A	Padrão de caractere ASCII
HLAunicodeChar	HLAoctetPairBE	N/A	N/A	N/A	Padrão de caractere Unicode UTF – 16
HLAbyte	HLAoctet	N/A	N/A	N/A	Byte não interpretado de 8 - bit
CombustívelCaça	HLAfloat32BE	Minutos	0.015	N/A	Medida do consumo de combustível

3.3.2.13. Tabela de Aviso

Algumas entradas em algumas tabelas OMT podem ser anotadas com informações descritivas adicionais fora da estrutura imediata das tabelas. Estas características permitem que usuários associem as informações explicativas com tabelas de entrada que facilitam o uso efetivo de informações.

Tabela 13 - Tabela de aviso

Rótulo	**Semântica**
1	Ano de uso é um valor que somente será mudado no aniversário de cada avião.
2	O aumento de valor não é fornecido para qualquer intervalo de tempo.

3.3.2.14. Dicionário do FOM/SOM

O dicionário FOM/SOM é constituído de quatro tabelas de definições (tabela de definição de classe do objeto, tabela de definição de classe de interação, tabela de definição de atributo e tabela de definição de parâmetro) e é incluído no OMT para a semântica de informações necessárias para um entendimento das informações contidas nas tabelas OMT. Como estamos preocupados apenas em descrever o OMT de uma maneira geral, não há necessidade de comentarmos sobre as tabelas do dicionário do FOM/SOM.

3.3.3. Especificação da Interface

Essa especificação define os serviços padrão que serão usados pelos federados para suportar a troca de informações em uma federação distribuída. Esta especificação define a comunicação entre a RTI e os federados e como os serviços da RTI serão. A RTI fornece seis serviços básicos que são ativados na forma de procedimentos que recebem e retornam parâmetros. Fica a cargo do programador implementar os serviços na linguagem desejada.

De acordo com [Kuh 00], a HLA possui alto poder de abstração para que não seja uma tecnologia transitória, considerando-se que as linguagens de programação e tecnologias para comunicação em redes estão em constante evolução.

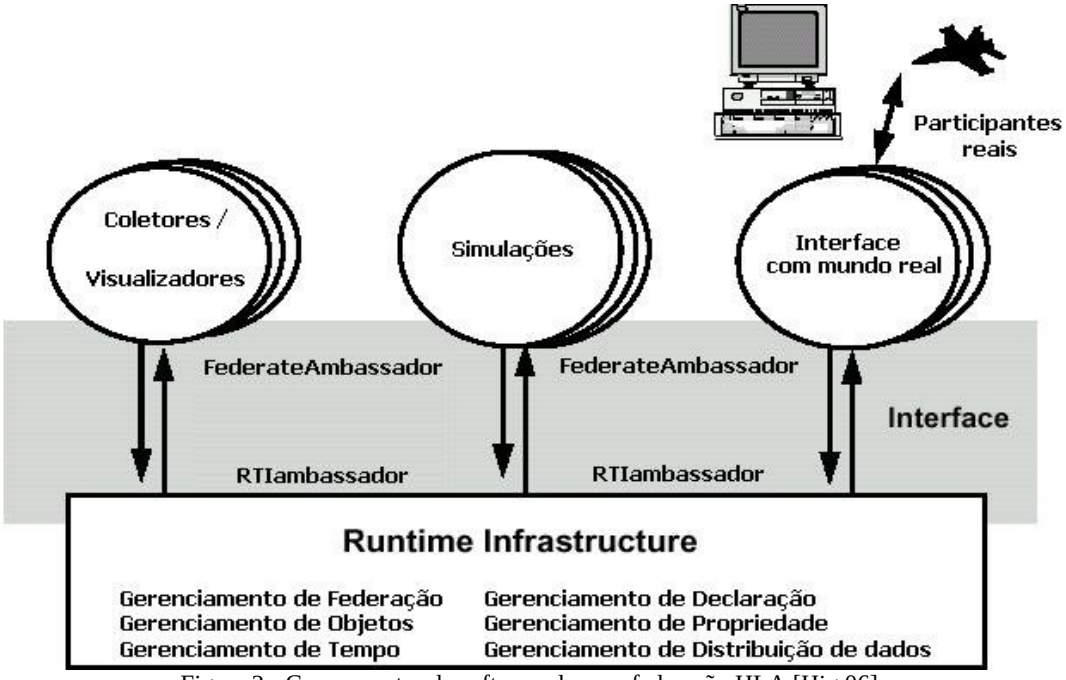


Figura 2 - Componentes de software de uma federação HLA [Hig 06].

Os objetos federados podem ser visualizados na Figura 2 como sendo os elementos que interagem com a RTI, ou seja, coletores/visualizadores, simulações, e objetos com interface para o mundo real, como mostra o exemplo.

A comunicação entre os federados e a RTI é feita através dos "embaixadores" que no exemplo anterior são o *Federate Ambassador* e o RTI *Ambassador*. Dentro do processo de especificação da interface é descrito o conjunto de classes de serviços que são fornecidos para os federados pela RTI [Dah 97].

A RTI fornece sete serviços básicos [Iee 00c], como segue:

- Gerenciamento de federação.
- Gerenciamento de declaração.
- Gerenciamento de objeto.
- Gerenciamento de posse.
- Gerenciamento de tempo.
- Gerenciamento de distribuição de dados.
- Serviços de suporte.

Esses serviços serão descritos a seguir.

3.3.3.1. Gerenciamento de Federação

O gerenciamento de federação refere-se à criação, controle, modificação e eliminação de uma federação.

A Figura 3 mostra os estados de uma federação, onde o primeiro estado é o de "federação não existente". É criada então a federação, que ainda não possui nenhum federado. Com a entrada do primeiro federado, a federação vai para o terceiro estado que já contém federados unidos, ficando neste estado até a saída do último federado, para que em seguida volte para o segundo estado e assim a federação seja destruída passando novamente para o primeiro estado.

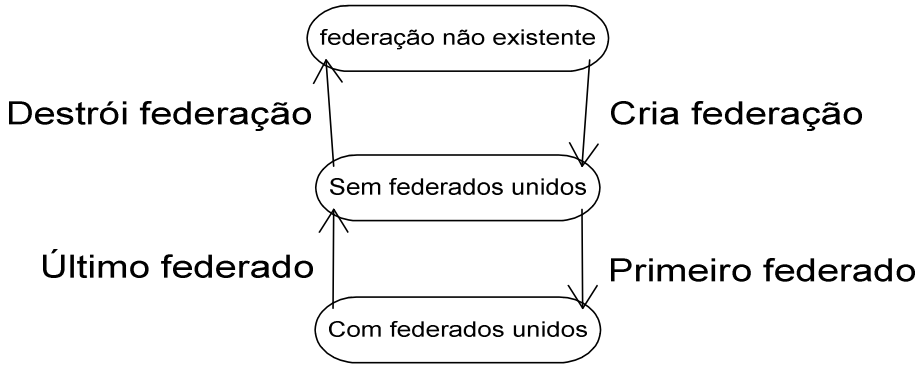


Figura 3 - Estados de uma federação.

3.3.3.2. Gerenciamento de Declaração

Os federados usam serviços de gerenciamento de declaração (DM) para declarar suas intenções e assim gerar informações. Um federado invoca serviços de DM apropriados para poder registrar um objeto, atualizar atributos e enviar interações.

Os federados também utilizam gerenciamento de distribuição de dados (DDM) ou DM para declarar suas intenções de receber informações, revelar objetos, ignorar atributos e receber interações. Serviços DM e DDM, juntos com serviços de gerenciamentos de objetos, gerenciamento de posse da classe de objeto e classe de interação determinam:

- Classes de objeto onde objetos podem ser registrados.
- Classes de objeto onde objetos podem ser revelados.
- Atributos que são disponibilizados para serem atualizados e ignorados.
- Interações que podem ser enviadas.
- Classes de interação onde interações são recebidas.

- Parâmetros que são disponibilizados para serem enviados e recebidos.

3.3.3.3. Gerenciamento de Objeto

O serviço de Gerenciamento de objeto da HLA trata do registro, modificação e exclusão dos objetos, bem como do envio e recebimento de interações.

A declaração do objeto é o principal conceito nesse grupo de serviço. Por exemplo, o objeto O terá uma classe de declaração do candidato no federado F unido se F for subscrito também na classe registrada de O ou para uma superclasse da classe registrada de O.

3.3.3.4. Gerenciamento de Posse

O gerenciamento de posse é usado pelos federados e pela RTI para transferir posse de atributos entre federados. A capacidade de transferir posse de atributos entre federados é requerida para suportar a modelagem cooperativa de um determinado objeto através de uma federação. Somente o federado que for dono de um atributo invocará o serviço de atualizar atributo para fornecer um novo valor para este atributo, receberá invocações do serviço que fornece atributo atualizado e receberá invocações dos serviços: a) ativar atualizações para o objeto e b) desativar atualizações para o objeto relativo ao atributo.

3.3.3.5. Gerenciamento de Tempo

O serviço de gerenciamento de tempo e mecanismos associados fornecem, para uma federação, meios para ordenar a liberação de mensagens por toda a federação. Estes mecanismos permitem que as mensagens sejam enviadas de diferentes federados para serem liberadas em uma ordem consistente e para que outro federado na federação possa receber essas mensagens de forma ordenada. Existem dois tipos de ordenação de mensagens: TSO (*TimeStamp*) e RO (*Receive*). O tipo de ordenação será determinado por diversos fatores, tais como, tipo preferido da ordem, presença de um marca tempo, estado do tempo do federado e tipo da ordem da mensagem enviada.

3.3.3.6. Serviços de Suporte

Existem quarenta tipos de suportes diferentes (não comentados neste trabalho) que descrevem vários serviços utilizados pelos federados para melhorar algumas ações como:

- Transformação de nome para apelido e de apelido para nome.
- Ambiente de troca consultiva.
- Manipular regiões.
- Abrir e fechar RTI.

3.3.3.7. Gerenciamento de Distribuição de Dados

Os serviços de gerenciamento de distribuição de dados (DDM) podem ser usados pelos federados para reduzir tanto a transmissão como também a recepção de informações irrelevantes. Em [Azz 01], o DDM é um serviço que gerencia a distribuição de atualizações, interações de informações em simulações distribuídas de larga escala, limites e controle de volume de dados durante a troca de dados na simulação. Os principais algoritmos de DDM são baseados em região, em grade, força bruta, híbridos e baseados em tipo [Rac 05].

O gerenciamento de distribuição de dados foi implementado e avaliado como parte deste trabalho de mestrado. Este serviço é descrito em detalhes no próximo capítulo.

3.4. Considerações finais

Este capítulo apresentou uma visão geral do padrão HLA, com os três componentes básicos, a saber: regras da HLA, OMT e especificações da interface, e todos os serviços da Infra-estrutura de Tempo execução (RTI). Dentre esses serviços, o gerenciamento da distribuição de dados (DDM) é um dos mais importantes, pois é o que permite a comunicação entre os federados. O próximo capítulo descreve esse serviço em detalhes, que foi um dos objetos principais deste trabalho de mestrado.

4. O Serviço de Gerenciamento da Distribuição de Dados - DDM (*Data Distribution Management*)

O gerenciamento de distribuição de dados é um serviço da HLA/RTI que controla a distribuição de informações em simulações distribuídas e a troca dos dados durante a execução da simulação [Azz 01].

Os objetivos principais do serviço de gerenciamento da distribuição de dados é limitar e controlar o volume de informações que são trocadas durante a simulação e reduzir o processamento entre os federados. Vários métodos DDM realizam este objetivo de diferentes formas, mas a chave para um DDM eficiente é limitar as mensagens que são emitidas para que somente as mensagens de interesse dos federados cheguem até eles. Em um ambiente de simulação distribuída, cada ação ou movimento que ocorrer na simulação, que pode afetar ou ser de interesse de outro participante na simulação, requer uma mensagem. Tais ações podem ser incluídas, mas não estão limitadas a movimento, criação e/ou destruição de entidades simuladas. Em uma simulação distribuída de larga escala, simular muitas entidades através de muitos federados diferentes, identificando as entidades que são de interesse de outras entidades pode resultar em um aumento de comunicação através de uma rede [Azz 03].

4.1. Espaço de Roteamento

O Espaço de Roteamento é um sistema de coordenadas multidimensionais [Dep 98] que representa o mundo virtual no qual a simulação ocorre. É interessante ressaltar que ele não representa apenas coordenadas de localizações geográficas. Por exemplo, em uma simulação de aviões, pode-se ter um sistema com cinco dimensões, sendo três delas usadas para representar localização geográfica em um espaço 3D e as outras duas usadas para representar um tipo de bomba com o qual 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.

4.2. 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 interessada.
- Região de Publicação - é uma abstração que referencia um conjunto de dados que uma entidade simulada disponibiliza para o mundo.
- Região de Interesse – é uma abstração que referencia os dois interesses, Subscrição e Publicação, não importando exatamente qual.

4.3. 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. Esse segundo tipo de comunicação *inter-host* é geralmente realizado utilizando a tecnologia de *multicast*, na qual existe 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*. Dessa 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.

4.4. Técnicas Existentes de Gerenciamento de Distribuição de Dados

As técnicas de gerenciamento de distribuição de dados – DDM podem ser classificadas em dois grandes grupos: Baseadas em Regiões (*Region-Based*) [Fuj 00, Hoo 98,

Tan 00a, Tan 00b] e Baseadas em Grade (*Grid-Based*) [Abr 98, Ber 98, Azz 02, Azz 04, Mor 97]. A seguir são apresentadas algumas técnicas baseadas em região, grade e uma combinação dessas.

4.4.1. Gerenciamento de Distribuição de Dados Baseado em Regiões (*Region-Based*)

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

O problema é gerado pela ocorrência freqüente do teste de *matching* que deve ocorrer sempre que uma região de publicação/subscrição é modificada, pois esta necessita ser comparada com todas as regiões de subscrição/publicação. O mesmo ocorre quando uma região de subscrição/publicação é criada, ela deve ser comparada com todas as regiões de publicação/subscrição de diferentes federados pela comunicação *inter-host*, que necessita comparar regiões de diferentes federados.

Para possibilitar o funcionamento dessa técnica, o sistema DDM deve ter, como extensão da federação, um Coordenador_DDM, para o qual todos os 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 unirem-se ou deixar os grupos *multicast*.

A presença de um Coordenador_DDM e uma base de dados pode 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 [Tan 00b].

Na Figura 4 pode-se observar o funcionamento dessa 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.

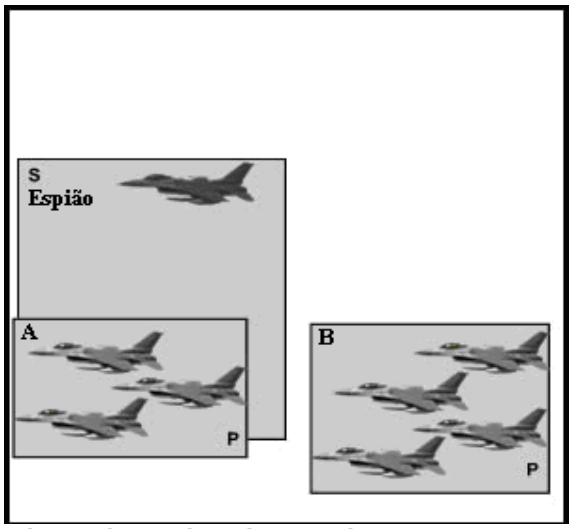


Figura 4 - Gerenciamento de Distribuição de Dados Baseado em Regiões (*Region-Based*), *matching* entre Esquadrão A e avião espião.

4.4.2. Gerenciamento de Distribuição de Dados Baseado 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 5 pode-se observar este mapeamento no qual 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 regiões-célula independentes 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, o 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 utilização 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.

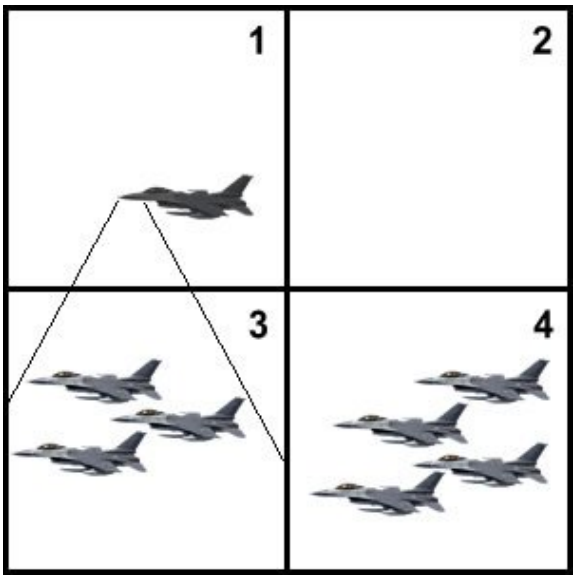


Figura 5 - Gerenciamento de Distribuição de Dados Baseado em Grade (*Grid-Based*), grade sobreposta para mapeamento de regiões-célula

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.

4.4.3. Gerenciamento de Distribuição de Dados Baseado em Grade Fixa (*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 é executado em cada *host* e 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 Baseada em Grade Fixa é que a intersecção das regiões é detectada apenas indiretamente, ou seja, sem o teste *matching*. Desta maneira, a comunicação inter-federados é minimizada drasticamente. Os dados são propriamente transferidos sempre que uma intersecção estiver presente porque tanto publicadores como subscritores terão se unido ao mesmo grupo *multicast*. Na Tabela 14 mostra-se o funcionamento desta técnica a partir do exemplo da Figura 5. 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.

Tabela 14 - Associação dos grupos *multicast* na técnica Fixa Baseada em Grade

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

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

4.4.4. Técnica Híbrida (*Hybrid Approach*) de Gerenciamento de Distribuição de Dados

A técnica híbrida une as técnicas Baseadas em Regiões e Grade, procurando reduzir os custos de suas desvantagens. Através do *matching* entre publicadores e subscritores que fazem parte de uma intersecção, ela procura reduzir o custo do *matching* excessivo associado à Baseada em Regiões e através do *matching* exato, procura reduzir o custo de atualização/mensagens desnecessárias da técnica Baseada em Grade [Tan 00b].

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 emparelhamento (*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 [Tan 00b]. 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.

4.4.5. Gerenciamento de Distribuição de Dados Baseado em Grade Dinâmica (*Dynamic Grid-Based*)

Como toda técnica Baseada em Grade, existe uma grade sobrepondo o terreno e definindo as células. Mas diferentemente da técnica Baseada em Grade Fixa, os grupos de *multicast* são alocados dinamicamente com base nas regiões de publicação e subscrição atuais 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 Baseada em Grade Dinâmica 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) os 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 [Bou 02].

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*. Isso pode ser visto como uma coleção de Coordenadores_DDM distribuídos, na qual cada um deles é responsável por 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 15 mostra-se o funcionamento dessa técnica, seguindo o exemplo da Figura 5. A 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*; isso porque, nessa técnica, apenas um grupo é criado, contra os quatro grupos criados na técnica Baseada em Grade Fixa.

Tabela 15 - Associação dos grupos *multicast* na técnica Dinâmica Baseada em Grade

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	-

4.4.6. Grade Filtrada Baseada em Regiões (*Grid-Filtered Region-Based*)

Na técnica de Grade Filtrada Baseada em Regiões o melhor das técnicas baseadas em regiões e grade é combinado 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 [Bou 05].

Essa 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 Regiões.

4.4.7. Comparação das Técnicas DDM

A técnica Baseada em Regiões, 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 Regiões, mas não resolve o problema do coordenador central. A Baseada em Grade Fixa não possui um coordenador central, 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* é utilizado quando comparado às outras técnicas.

Com a técnica Baseada em Grade Dinâmica esses problemas foram amenizados [Azz 02]. Para simulações em larga escala, com um número crescente de entidades, o número de grupos *multicast* é bem inferior em relação à técnica Baseada em Grade Fixa. Outra vantagem é que não são necessários 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 Baseada em Grade Dinâmica mais escalável que técnicas apoiadas na Baseada em Regiões. Mas o teste de intersecção, assim como na técnica Baseada em Grade Fixa, não é exato. Esse teste foi adicionado à Grade Filtrada Baseada em Regiões (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 Regiões.

Na Tabela 16 faz-se a comparação das técnicas de Gerenciamento de Distribuição de Dados, na qual as variantes são fatores desejáveis para que exista a detecção de intersecção de maneira exata e não exaustiva e sem um coordenador central.

Tabela 16 - 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

4.5. Considerações Finais

Neste capítulo foram apresentados os principais métodos de gerenciamento da distribuição de dados, a saber: baseado em regiões, baseado em grade, fixa e dinâmica e híbrida). Uma comparação entre os métodos foi realizada, mostrando que cada método possui vantagens e desvantagens. O baseado em região, por exemplo, possui maior exatidão na intersecção, mas pode gerar gargalo com seu coordenador central em sistemas com um número muito grande de federados. Já o mecanismo baseado em grade elimina o coordenador central mais não possui uma interseção muito precisa. Assim, foi escolhido o método Gerenciamento de Distribuição de Dados Baseado em Grade Dinâmica pois gera um número menor de grupos *multicast*, suportando um sistema de larga escala

O próximo capítulo tratará de algumas soluções de suporte para sistemas distribuídos, apresentando: CORBA, DCOM, JavaRMI, Gnutella, serviços *web* e JXTA, apontando suas características, protocolos utilizados, importância e forma de funcionamento.

5. Soluções de Suporte à Comunicação Distribuída

Este capítulo descreve as principais tecnologias de suporte a sistemas distribuídos, a saber: a arquitetura CORBA *(Common Object Request Broker Architecture)*, que foi desenvolvida pela OMG (*Object Management Group*), o DCOM *(Distributed component object model)* que é uma versão distribuída do modelo COM da Microsoft, JavaRMI da Sun, a versão do Java para programação em ambientes distribuídos; a rede Gnutella, desenvolvida no inicio de 2000 e, por fim, mas não menos importante, o JXTA que é uma plataforma de software aberto crida pela Sun em 2001.

5.1. CORBA (*Common Object Request Broker Architecture*)

A arquitetura CORBA (*Common Object Request Broker Architecture*) começou a ser definida pela OMG em 1989 [Omg 97]. A OMG (*Object Management Group*) é uma organização internacional suportada por centenas de membros, abrangendo um grande espectro de interesses, desde usuários até projetistas de sistemas. A carta de princípios da organização inclui o estabelecimento de diretrizes na indústria e especificações de gerenciamento de objetos para fornecer uma estrutura comum para desenvolvimento de aplicações. O objetivo primário é se alcançar sistemas baseados em objetos em ambientes distribuídos heterogêneos com características de reusabilidade, portabilidade e interoperabilidade.

Objetos clientes requisitam serviços às implementações de objetos através de um ORB. O ORB é responsável por todos os mecanismos requeridos para encontrar o objeto e preparar a implementação de objeto para receber e executar a requisição. O cliente vê a requisição de forma independente de onde o objeto está localizado, da linguagem de programação na qual ele foi implementado, ou qualquer outro aspecto que não está refletido na interface do objeto.

O CORBA utiliza a OMG IDL (*Interface Definition Language*) como uma forma de descrever interfaces, isto é, de especificar um contrato entre os objetos. A OMG IDL

é uma linguagem puramente declarativa baseada em C++. Isso garante que os componentes em CORBA sejam auto-documentáveis, permitindo que diferentes objetos, escritos em diferentes linguagens, possam inter-operar através das redes e de sistemas operacionais [Orf 96].

Uma definição de interface escrita em OMG IDL define completamente a interface e especifica cada parâmetro da operação. É importante ressaltar que os objetos não são escritos em OMG IDL, a qual é uma linguagem puramente descritiva. Eles são escritos em linguagens que possuem mapeamentos definidos dos conceitos existentes em OMG IDL. Em [Omg 95] são descritos mapeamentos para C, C++ e Smalltalk, e existem estudos para se adicionar outras linguagens nas futuras versões de CORBA.

O ORB (*Object Request Broker*), por si só, não executa todas as tarefas necessárias para os objetos inter-operarem. Ele só fornece os mecanismos básicos. Outros serviços necessários são oferecidos por objetos com interface IDL, o qual a OMG vem padronizando para os objetos de aplicação poder utilizar:

- Serviço de nomes.
- Serviço de controle de concorrência.
- Serviço de eventos.
- Serviço de tempo.

5.2. DCOM (*Distributed component object model*)

Em 1993, para resolver as deficiências da tecnologia OLE (*Object Linking and Embedding*), a Microsoft lançou uma nova tecnologia de encapsulamento de objetos designada COM (*Component Object Model*). O COM passa então a ser divulgado pela Microsoft como um modelo de programação orientado por objetos com suporte a integração de várias aplicações diferentes e desenhado para facilitar a interoperabilidade do software.

Um objeto no modelo COM é uma entidade funcional que obedece ao princípio de encapsulamento de orientação a objeto. Os clientes não manipulam os objetos diretamente. Ao invés disso, o objeto exporta para os seus clientes vários conjuntos de ponteiros de funções conhecidas como interfaces.

O DCOM é a extensão distribuída do COM que constrói uma camada de chamada de procedimentos de objetos remotos ORPC (*Object Remote Procedure Call*) no topo do DCE RPC (*Distributed Computing Environment Remote Procedure Call*) para

suportar objetos remotos. O COM define como os componentes de software e os seus clientes interagem. Essa interação é definida tal que o cliente e o componente possam se conectar sem a necessidade de qualquer recurso intermediário.

O DCOM define um padrão binário por plataforma sem a preocupação de definir ligações com linguagens de alto nível. Desta forma, os clientes e desenvolvedores podem integrar componentes gerados com ferramentas de diversos fabricantes e usá-los em diferentes implementações *runtime* do DCOM. Com o DCOM é possível distribuir uma única versão binária de um componente para uma dada plataforma que funciona com todos os outros componentes e biblioteca em tempo de execução. Qualquer linguagem que entenda este padrão pode criar e utilizar objetos DCOM. Com a distribuição do DCOM em todos os ambientes de desenvolvimentos da Microsoft (Visual C++, Visual Basic e Visual J++), ocasionou-se um aumento do número de linguagens e ferramentas que suportam esta tecnologia [Amo 04].

Com o DCOM, os detalhes sobre as questões de localização dos componentes não são especificados no código fonte, estando estes detalhes no mesmo processo ou em uma máquina em qualquer lugar do mundo. Em todos os casos, a forma como o cliente se conecta a um componente e chama os métodos do componente é idêntica. Da mesma forma que o DCOM não requer mudanças no código fonte, também não é necessária a recompilação do mesmo. A independência de localização do DCOM simplifica a tarefa de distribuição dos componentes de uma aplicação. No caso de uma aplicação possuir numerosos pequenos componentes, a carga na rede pode ser reduzida caso estes componentes venham a se localizar em um mesmo segmento de uma rede local, em uma mesma máquina ou em outra localização especifica. Por outro lado, se a aplicação for composta de um menor número de componentes maiores, estes poderão ser colocados em máquinas mais rápidas, sem a preocupação com a sua localização [Dco 07].

5.3. JavaRMI (*Java Remote Method Invocation*)

RMI é um *middleware* orientado por objetos que permite invocações de métodos entre JVMs (*Java Virtual Machine*) localizadas em nodos diferentes de uma rede. Java RMI é um pacote Java que permite criar objetos cujos métodos poderão ser invocados remotamente a partir de aplicações clientes, localizadas em JVMs remotas. Uma aplicação cliente em RMI tipicamente consulta o serviço de nomes do sistema – chamado RMI *Registry*

– a fim de obter uma referência de rede, isto é, uma referência para um objeto localizado em uma outra JVM. Essa referência pode ser então utilizada para invocar métodos do objeto remoto, com uma sintaxe equivalente àquela de chamadas locais.

Aplicações servidoras tipicamente são responsáveis por criar objetos remotos e, eventualmente, registrá-los no serviço de nomes do sistema. Em Java RMI a classe de um objeto remoto deve estender uma classe pré-definida, chamada *java.rmi.server.UnicastRemoteObject*. Além disso, essa classe deve implementar uma interface remota, isto é, uma interface que estende *java.rmi.Remote*. Uma interface remota define um contrato entre clientes e servidores, especificando métodos que os primeiros podem invocar nos segundos. Por fim, métodos de uma interface remota devem relacionar em sua cláusula *throws* exceções do tipo *java.rmi.RemoteException*. Em aplicações clientes, tais exceções são ativadas pela implementação de Java RMI para indicar uma falha de comunicação com a JVM servidora.

RMI *Registry*: A classe *java.rmi.Naming* oferece métodos estáticos para registrar (*bind*) e pesquisar (*lookup*) objetos remotos associados ao RMI *Registry*. Ao se registrar um objeto, deve-se informar um nome textual para o mesmo. A escolha deste nome deve ser acordada com as aplicações clientes, já que estas precisam informá-lo ao pesquisar por um objeto remoto associado ao *Registry* de um determinado nodo.

Arquitetura Interna: Internamente, a implementação de Java RMI se baseia em dois objetos principais, chamados de *stubs* e *skeletons*. Estes dois objetos são usados para implementar a abstração proporcionada por uma referência remota. Em uma aplicação cliente, uma referência de rede é, na realidade, uma referência para um objeto local, denominado *stub*. Esse objeto implementa a mesma interface remota do objeto servidor, sendo responsável por encapsular todos os detalhes de comunicação com o mesmo. Uma invocação remota é então "capturada" pelo *stub* do cliente, que converte seus parâmetros para uma representação seriada, a qual é enviada para um objeto da aplicação servidora chamado de *skeleton*. Esse objeto é responsável por recuperar os dados da chamada e invocar o método servidor. Após esse método ser executado, seu resultado é enviado de volta ao processo cliente, via objetos *skeleton* e *stub*. As classes de *stubs* e *skeletons* são geradas automaticamente a partir da classe do objeto servidor. Essa geração é realizada pela ferramenta rmic, a qual faz parte de ambientes de desenvolvimento de aplicações em Java [Val 03].

5.4. Serviços *Web* (*Web Services*)

Por volta dos anos 90 usuários começaram a se conectar a um site para baixar conteúdos, foi ai que a internet começou a se popularizar. O HTML *(Hyper Text Markup Language)* era o padrão "de fato" que permitia a apresentação das informações existentes na rede. Nos últimos anos, porém, novas tecnologias e *frameworks* de desenvolvimento estão surgindo, permitindo uma maior integração entre os diversos aplicativos e serviços disponíveis na Internet. Esse novo modelo em crescimento deve tratar tarefas complexas, como o gerenciamento de transações, através da disponibilidade de serviços distribuídos que utilizem interfaces de acesso simples e bem definidas. Esses serviços ou aplicativos distribuídos são conhecidos como Serviços *Web (web services)* [Web 06a].

A principal idéia do serviço *web* é considerar um software como um serviço. Esse conceito é baseado em um conjunto definido de tecnologias, suportadas pelos padrões industriais abertos, que trabalham juntos para facilitar a interoperabilidade entre sistemas heterogêneos. Em outras palavras, os serviços *web* permitem que aplicações se comuniquem com outras aplicações usando padrões abertos, o que potencializa a construção de pontes (ligações) entre sistemas. Serviços *web* podem se comunicar mesmo se estiverem rodando em diferentes sistemas operacionais e escritos em diferentes linguagens de programação. Com isso, os serviços *web* fornecem uma excelente forma para construir aplicações distribuídas que devem ser incorporadas por diversos sistemas sobre uma rede [Pul 04].

Serviços *web* são identificados por uma URI *(Unique Resource Identifier),* e são descritos e definidos usando XML. Um dos motivos que tornam serviços *web* atrativos é o fato de este modelo ser baseado em tecnologias padrão, em particular XML e HTTP. Serviços *web* são usados para disponibilizar serviços interativos na *web*, podendo ser acessados por outras aplicações. O SOAP *(Simple Object Access Protocol)* está se tornando padrão para a troca de mensagens entre aplicações e serviços *web*, já que é uma tecnologia construída com base em XML e HTTP [Web 06a].

Os serviços *web* são descritos como uma arquitetura baseada em serviços, também conhecida por SOA (*Service Oriented Architecture*), termo inicialmente cunhado pela IBM. Os componentes da arquitetura SOA representam uma coleção de serviços que se comunicam através da troca de mensagens XML.

Tecnicamente, serviços *web* são serviços distribuídos que processam mensagens SOAP codificadas em XML, enviadas através de HTTP e que são descritas através de *Web Service Description Language* (WSDL).

5.5. Gnutella

O Gnutella (neologismo proveniente da junção de GNU – *General Public License* – com Nutella, sobremesa da Ferrero feita de chocolate com avelãs) surgiu em março de 2000, criado por Justin Frankel e Tom Pepper, da Gnullsoft, também criadores do Winamp, e foi desenvolvido em apenas 14 dias como um experimento. Inicialmente tinha por objetivo compartilhar receitas culinárias. Porém, a AOL (America *Online*), que comprou a companhia, descontinuou o desenvolvimento do mesmo por seu envolvimento com a indústria fonográfica (o caso Napster já estava em discussão) [Gnu 07].

Foi então que desenvolvedores de software livre intervieram: Bryan Mayland realizou a engenharia reversa do programa e Ian Hall-Beyer e Nathan Moinvaziri criaram um site e um canal de IRC (*Internet Relay Chat*) com toda a documentação necessária, com o objetivo de buscar novos desenvolvedores.

Desde então, o Gnutella tem evoluído mais como uma rede do que um software propriamente, visto que existem atualmente vários softwares clientes que o utilizam, como o LimeWire e o Phex (com versões para Windows, Mac e Linux), o Morpheus e o BearShare (para Windows), o Mutella e o Qtella (para Linux). Só o Limewire contabiliza, em média, 1,5 milhão de usuários por dia.

O protocolo Gnutella restaura a simetria original da *web*, capacitando mesmo os computadores transitórios a participarem efetivamente como servidores. A relação entre Gnutella e a *web* é, portanto, bastante simples: embora transitórios, os *hosts* da Gnutella são *Websites*, e baixar um arquivo de um *host* Gnutella é tecnicamente equivalente a buscar um arquivo em um *Website*. A maioria das aplicações Gnutella combina a funcionalidade de servidores e clientes num pacote chamado *'servent'*. Mesmo os usuários que não disponibilizam arquivos para compartilhamento, e que apenas usam Gnutella para fazer buscas e baixar arquivos, disponibilizam *Websites* (vazios) quando rodam seus *servents*.

A rede Gnutella cria uma camada de aplicativo sobre a Internet e muda constantemente sua infra-estrutura. Seu funcionamento é descentralizado, e se baseia no princípio de que o nó se conecta a um *host* qualquer e a partir daí faz parte da rede. Além

disso, todo nó é cliente, servidor e a própria rede. Dessa forma, qualquer um que participe da rede contribui, mesmo que seja apenas com a banda passante para que os dados trafeguem de um nó para outro.

Diferentemente de outras redes em nível de aplicativo, o Gnutella não é baseado em circuitos e sim em mensagens. Ou seja, não se cria um circuito virtual persistente que manterá a conexão entre dois nós, dando uma topologia interconectada e redundante à rede.

Além disso, outra grande diferença é que, apesar de ser baseado em TCP, o Gnutella não faz *unicast*, mas *broadcast* para realizar uma comunicação. A junção dessas duas características gera o que é chamado de *Broadcasting* de Mensagens, que é útil em casos onde redes diferentes geram resultados válidos para a mesma busca.

Para evitar que essas mensagens fiquem eternamente na rede, elas recebem um UUID (*universal unique identifier*) de 128 bits que é memorizado quando a mensagem passa por um *host* e, caso ela passe por ele novamente não será retransmitida. Outra saída é o uso de TTL (*time-to-live*), uma espécie de cronômetro. Quando a mensagem é criada ela recebe TTL 7. A cada *host* que é transmitida, o TTL é decrementado e, quando atinge 0, o pedido não é mais retransmitido.

O *Broadcasting* de Mensagens é útil para buscas, mas não para respostas. Para isso, foi criado o roteamento dinâmico. Como já citado, ao passar por um *host* a mensagem tem seu UUID armazenado. Assim, quando um *host* tem a resposta necessária, ele vai enviar apenas para o *host* que lhe enviou (ou repassou o pedido), repetindo-se isso sucessivamente até a resposta chegar à origem do pedido. Para evitar conflitos, quando um nó sai da rede todas as informações referentes a ele, presentes em outros nós, são apagadas.

Um aspecto interessante da rede Gnutella é que, apesar de usar o TCP (que é normalmente confiável), ela não tem sempre essa característica. Quando o tráfego é maior do que a banda permite, os pacotes excedentes não são armazenados, fazendo com que haja perdas. Isso é feito para que a natureza de quase-tempo-real da rede não se perca. Essa característica gera também um problema quando nós muito lentos conectam-se a nós muito rápidos, fazendo com que ao receber mais informação do que sua capacidade, boa parte da informação se perca antes mesmo de ser entregue. Um lado positivo disso é a possibilidade de identificar se essa perda é grande constantemente, possibilitando reorganizar a rede, de forma a melhorar o tráfego.

Para evitar esse tipo de problema e os gargalos de baixa velocidade que ocorrem em uma rede desse tipo, a organização da rede Gnutella não é feita geograficamente,

mas de acordo com a velocidade da conexão. Ao se conectar pela primeira vez, um nó se encaixa em uma parte aleatória da rede. Após a constatação da velocidade média de conexão daquele nó, ele é movido para uma parte da rede mais adequada a sua velocidade. Assim, cria-se um *backbone* com nós extremamente rápidos (conexões a fibra óptica, por exemplo), localizados no centro da rede, e há a redução gradual das velocidades de conexão até se chegar às bordas, com conexões lentas (discadas, por exemplo) [Gnu 07].

5.6. JXTA (*juxtapose*)

A tecnologia JXTA (*juxtapose*) introduzida pela Sun Microsystems, Inc. é um conjunto de protocolos *peer-to-peer* de software aberto, que permite que diferentes tipos de dispositivos, de telefones celulares a servidores, possam se comunicar e colaborar [Lim 05]. Com o JXTA, aplicações e serviços distribuídos podem ser criados e acessados independentes de sua localização. JXTA permite escalabilidade e suporte em situações onde os recursos são altamente distribuídos. Devido a essas características, o JXTA foi escolhido como tecnologia para a implementação de nossa arquitetura de suporte a simulações distribuídas. O JXTA é descrito com mais detalhes no próximo capítulo.

5.7. Considerações Finais

Nesse capítulo foram apresentadas as principais tecnologias de suporte para sistemas distribuídos. Duas soluções em potencial foram identificadas para suporte a simulações distribuídas: serviços *web* e JXTA. A escolha dessas duas técnicas foi feita depois de um estudo detalhado que mostrou a plataforma JXTA como sendo muito flexível, além de sua independência de linguagens de programação. Os serviços *web* com seus padrões estáveis complementam o JXTA no suporte a simulações distribuídas. Essas duas tecnologias serão descritas em mais detalhes nos próximos capítulos.

6. JXTA (*JuXTApose*)

Este capítulo descreve com mais detalhes a tecnologia JXTA. JXTA (*juxtapose*) é uma arquitetura *open source* que define um conjunto de protocolos para as funcionalidades requeridas para uma rede P2P. É independente de sistema operacional, de linguagem de programação e do tipo de transporte empregado na rede. As especificações dos protocolos definem estados que os pontos da rede devem assumir para qualquer tipo de dispositivo. O projeto JXTA foi iniciado pela *Sun Microsystems* em abril de 2001, e de lá pra cá mais de 6 milhões de usuários tem feito o *download* dessa tecnologia [Lim 05].

Além das características citadas acima, o JXTA é uma grande aposta de padronização e oferece também os mecanismos necessários para a implementação da arquitetura proposta neste trabalho.

A tecnologia JXTA tem como principal objetivo fornecer uma plataforma com as funções básicas para o desenvolvimento de redes *peer-to-peer*. Além disso, pretende ter como principais características: interoperabilidade entre diferentes *peers* que fornecem serviços *peer-to-peer* e ubiqüidade, permitindo que qualquer dispositivo com um *heartbeat* digital possa ser acessível. Os conceitos JXTA incluem *peers*, *peergroups*, *pipes*, *messages*, *endpoints*, serviços, *advertisements*, *modules*, *rendezvous* e segurança. Um peer é qualquer dispositivo na rede capaz de fazer computações [Cor 05].

Um dos objetivos do projeto JXTA é oferecer uma plataforma com funções básicas necessárias para uma rede P2P. Além do mais, a tecnologia JXTA busca superar as faltas potencias de muitos sistemas P2P existentes, como se vê a seguir [Laf 03]:

- Interoperabilidade: A tecnologia JXTA é projetada para permitir que *peers* interconectados localize facilmente outros *peers*, comunique-se com outros *peers*, participe de atividades baseadas em comunidade e ofereçam serviços para os outros *peers* constantemente através de diferentes sistemas P2P e diferentes comunidades [Gon 01].
- Independência de plataforma: JXTA é independente de linguagens de programação (como C ou Java), sistemas operacionais (Windows ou Linux) e plataformas de rede (como TCP/IP ou *Bluetooth*)[Gon 01].

- Ubiqüidade: O JXTA é projetado para ser acessível a qualquer dispositivo e não apenas PCs ou plataformas especificas [Laf 03].

É importante ressaltar que JXTA não é uma aplicação, e não define também os tipos de aplicações que podem ser desenvolvidas. Os protocolos definidos no padrão, também não são definidos de uma forma rígida, pois suas funcionalidades podem ser estendidas de acordo com a necessidade. Essa é uma característica muito importante e um dos principais motivos da escolha de JXTA para a utilização nesse trabalho.

Em relação aos conceitos de P2P apresentados anteriormente, JXTA tenta encontrar um meio termo entre centralização e descentralização, tomando por conceito que alguns serviços numa rede P2P são mais efetivos se feitos por um limitado número de *peers*.

Com JXTA é possível criar uma rede virtual P2P em cima da rede física da Internet. Os *peers* podem trocar mensagens independentemente da infra-estrutura da rede e do protocolo de transporte. A Figura 6 mostra o mapeamento da rede virtual de JXTA para a rede que compõe a Internet [Lim 05]:

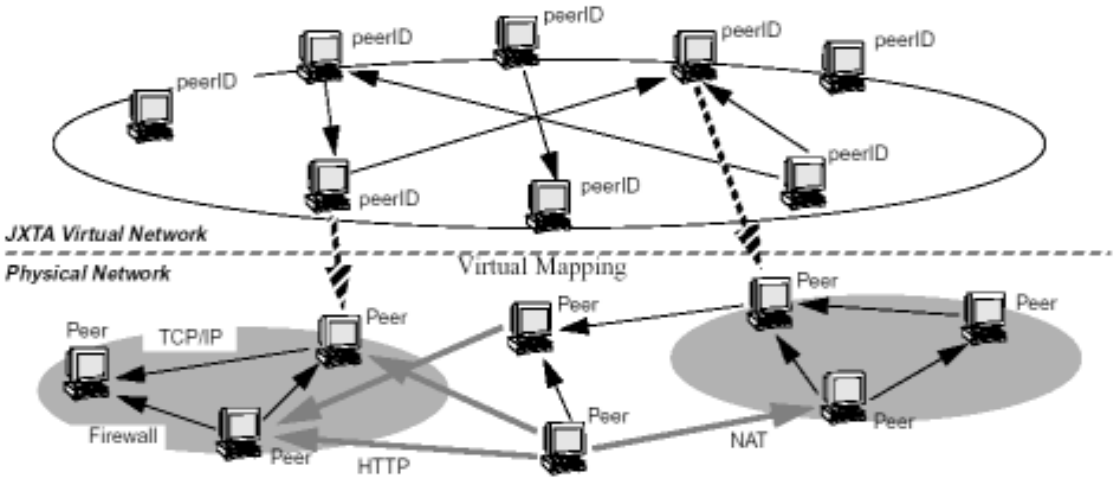


Figura 6 - Rede virtual de JXTA [Jxt 06a]

6.1. As Camadas Lógicas do JXTA

A plataforma JXTA pode ser dividida em três camadas mostradas a seguir:

- A camada *Core*: O Núcleo JXTA "*Core*" é responsável pelo nível básico de operação e comunicação. Essa camada inclui protocolos e

provê mecanismos de segurança. Ela equivale ao *'kernel'* de um sistema operacional e controla o grupamento de pares, os dutos de pares e a monitoração de pares. Além disso, ela é responsável pelos mecanismos de segurança [Int 06b].

- A camada JXTA *services*: Essa camada fornece funcionalidades de alto nível usando as funções primitivas da camada *Core* [Jxt 06b]. Os serviços providos por essa camada implementam funcionalidades que podem ser incorporados em diferentes aplicações P2P, como busca de recursos e de arquivos em um *peer*, ou realização de autenticação.
- A camada JXTA *Applications*: As aplicações JXTA são construídas usando serviços de *peers* e também a camada *Core*, para criar, por exemplo, aplicativos de mensagens instantâneas [Pro 06].

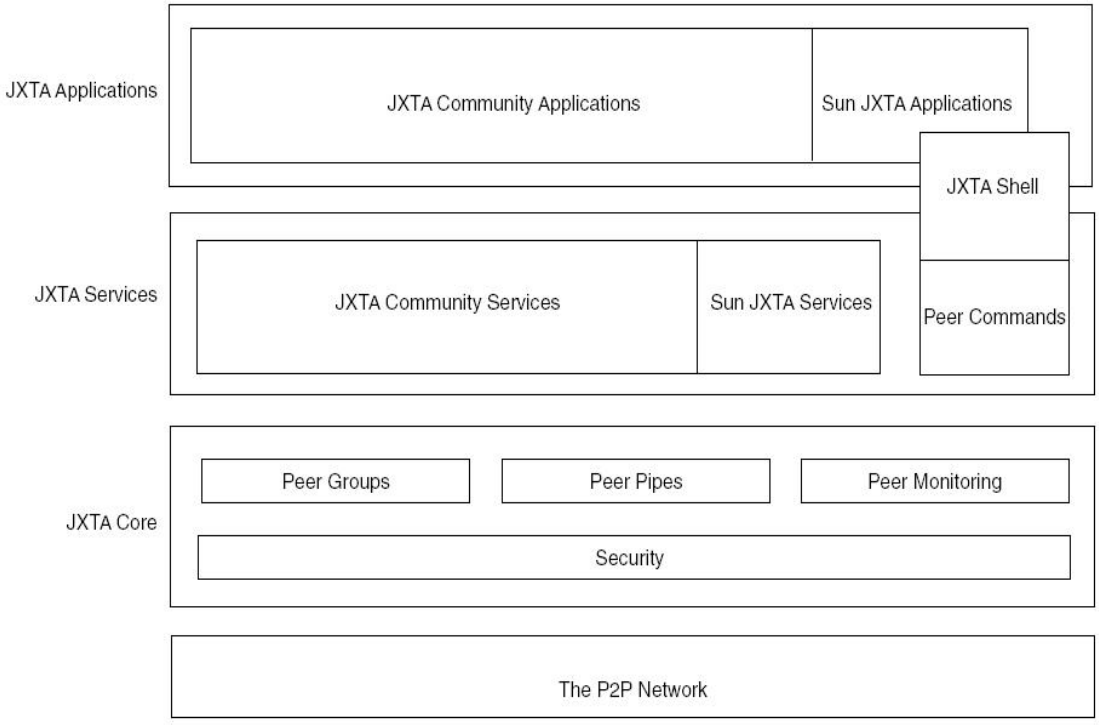


Figura 7 - As camadas lógicas do JXTA [Pee 06]

6.2. Elementos do JXTA

A seguir são apresentados os principais elementos do JXTA.

6.2.1. *Peers*

No JXTA existem quatro tipos de *peers*, como apresentados abaixo:

- *Minimal Peers*: Podem enviar e receber mensagens, mas não anunciar e rotear mensagens para outros *peers*. Dispositivos pequenos como PDAs ou celulares são considerados *Minimal Peers* [Pjt 06].
- *Simple Peers*: Podem enviar e receber mensagens e, normalmente, armazenam anúncios, mas não transferem qualquer pedido de descoberta. A maioria dos *peers* são *Simple Peers* [Sim 06c].
- *Rendezvous Peers*: Apresentam uma grande semelhança aos *Ultrapeers* da rede Gnutella [Gnu 06]. Qualquer *Simple Peer* pode configurar-se para se tornar um *Rendezvous Peer*. Eles possuem uma lista de outros *Rendezvous Peers*. Eles transferem os pedidos de descoberta para outros *Rendezvous* e para outros *peers* também [Pro 06a].
- *Relay Peers*: Provêem mecanismos de comunicação com *peers* separados por um *firewall* ou por máscaras de rede, ou seja, são capazes de rotear mensagens a outros *peers* da rede. Também fazem requisições e recebem respostas em nome de outro *peer* [Jac 06].

6.2.2. *PeerGroups*

Os *peers* se organizam em *Peer Groups*. Um grupo de *peers* é uma coleção de *peers* que tem um conjunto de interesses em comum. Cada grupo de *peers* é identificado por uma *PeerGroup Id*. Os protocolos JXTA definem quando, onde ou porque grupos de *peers* são criados. Os protocolos JXTA somente descrevem como *peers* podem publicar, descobrir, juntar e monitorar grupos de *peers* [Esp 06].

6.2.3. *Pipe*, *Endpont* e Mensagens

No protocolo de transporte de rede temos os conceitos de *endpoints*, *pipes* e mensagens. *Endpoints* são interfaces de rede que indicam a fonte/destinatário de qualquer informação trocada. *Pipes* (dutos) são canais virtuais de comunicação unidirecionais e

assíncronos entre dois *endpoints*. Mensagens são containeres para dados transmitidos num *pipe* de um *endpoint* para outro [Cor 05].

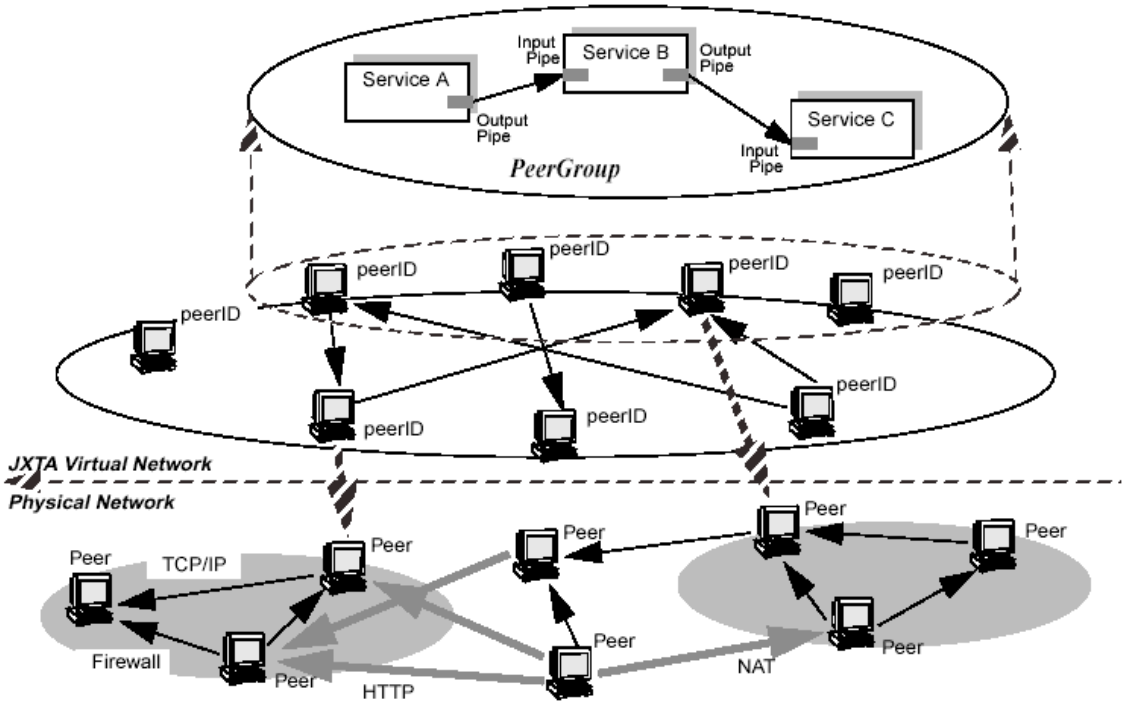


Figura 8 - Funcionamento dos *Pipes* [Dês 06]

6.2.4. *Advertisements*

Um *Advertisement* (anúncios) é um documento especial que publica a presença de recursos JXTA. Um recurso pode ser qualquer coisa usada por um *peer*, por exemplo, outro *peer* JXTA, *peergroups*, *pipes* e serviços fornecidos por *peers* JXTA [Ekl 04].

6.2.5. Serviços

Serviços são funcionalidades que um *peer* pode invocar remotamente para obter um resultado útil. Podemos dividir serviços JXTA em duas categorias: *peer services* e *group services*. *Peer services* são serviços associados exclusivamente a um *peer*. Quando esse *peer* sai da rede, os serviços desse *peer* deixam de estar disponíveis se nenhuma replicação explícita for feita. *Groups services* são funcionalidades que um grupo oferece a qualquer membro desse grupo. Essa funcionalidade pode ser disponibilizada por vários *peers*

permitindo redundância. Desde que um *peer* esteja num *peer group* os serviços desse grupo estão disponíveis [Cor 05].

6.2.6. *Module*

Um *Module* é um pedaço de código (funcionalidade) que pode ser carregado e instanciado num *peer* dinamicamente (em *runtime*). Os *modules* são publicados num *peer group* usando *advertisements* que definem o seu comportamento geral, uma especificação e uma implementação. Um *peer* pode carregar e instanciar uma implementação já existente em outro *peer*, ou diferentes implementações da mesma funcionalidade podem ser feitas em diferentes linguagens, em diferentes contextos [Cor 05].

6.2.7. Segurança

A segurança na tecnologia JXTA é fornecida pela possibilidade de encriptação de mensagens à saída de cada *endpoint*, permitindo integridade, autenticação e confidencialidade.

6.3. Protocolos do JXTA

Em um alto nível de abstração, a tecnologia JXTA é um conjunto de protocolos, ao todo seis protocolos. Cada protocolo é definido por uma ou mais mensagens trocadas entre os participantes. Cada mensagem tem uma forma pré-definida e pode incluir vários campos de dados [Gon 01]. A Figura 9 apresenta a pilha de protocolos do JXTA:

A descrição de cada protocolo é dada a seguir [Esp 06]:

- *Peer Resolver Protocol* (PRP): É um mecanismo pelo qual um *peer* pode enviar um pedido para um ou mais *peers* e receber uma resposta (ou várias respostas) do pedido. O PRP implementa um protocolo de pedido/resposta. A mensagem de resposta é combinada com o pedido através de uma *Id* única, incluída no corpo da mensagem. Pedidos podem ser direcionados para todo grupo ou para *peers* específicos.

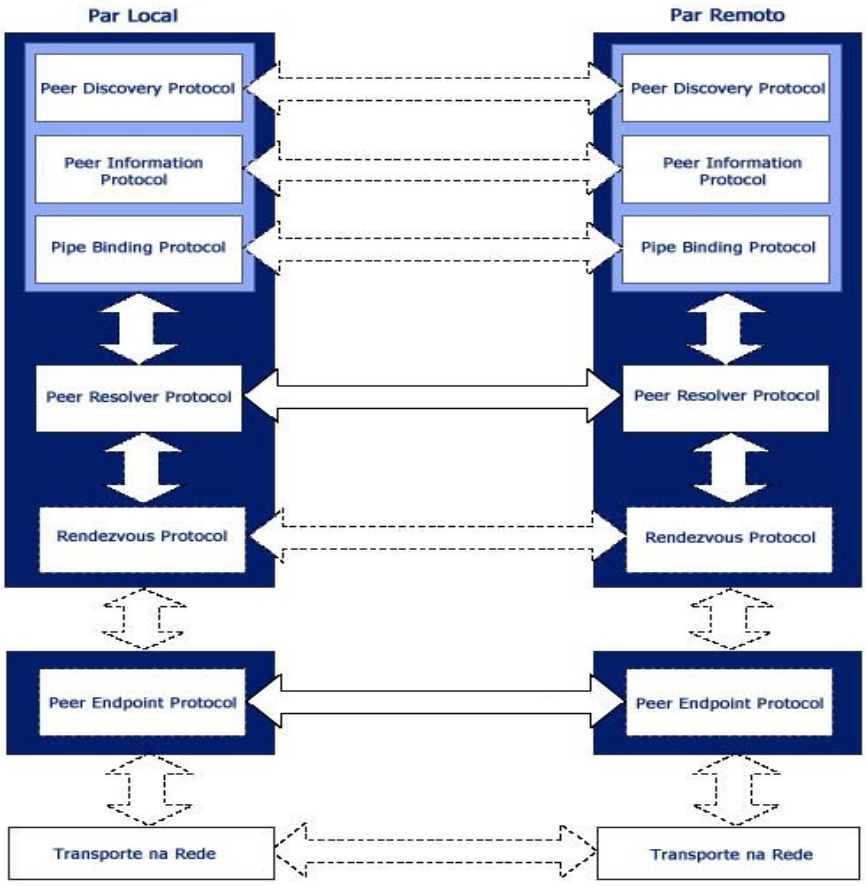


Figura 9 - Pilha de protocolos do JXTA [Esp 06].

- *Peer Discovery Protocol* (PDP): É o mecanismo pelo qual um *peer* pode anunciar seus recursos e descobrir os recursos de outros *peers* (grupo de *peers*, serviços, *pipes* e *peers* adicionais). Todo recurso de *peer* é descrito e publicado usando um anúncio.
- *Peer Information Protocol* (PIP): É o mecanismo pelo qual um *peer* pode obter informações de estado sobre outros *peers*. Isso pode incluir estado, tempo de vida, capacidade e outras informações.
- *Pipe Binding Protocol* (PBP): É o mecanismo pelo qual um *peer* pode estabelecer um canal de comunicação virtual ou *pipe* entre um ou mais *peers*. O PBP é usado por um *peer* para ligar dois ou mais pontos de conexão (*pipe endpoints*). *Pipes* fornecem o mecanismo de comunicação básica entre *peers*.
- *Endpoint Routing Protocol* (ERP): É o mecanismo pelo qual um *peer* pode descobrir uma rota (seqüência de *hops*) usada para enviar uma mensagem para outro *peer*. Se um *peer* A quer enviar uma mensagem

para o *peer* C e não existe uma rota direta conhecida entre A e C, então o *peer* A precisa determinar a rota de informações.

- *Rendezvous Protocol* (RVP): É o mecanismo pelo qual um *peer* pode subscrever ou ser um subscritor para uma propagação de um serviço. Dentro de um grupo de *peers*, os *peers* podem ser também *peer rendezvous* ou *peers* que estão "ouvindo" *peer rendezvous*. O RVP deixa um *peer* enviar mensagens para todos aqueles que estão ouvindo o serviço.

6.4. Considerações Finais

A grande vantagem da arquitetura JXTA sobre outras tecnologias existentes é, principalmente, prover uma abstração para a comunicação entre *peers*, que permite suporte a uma grande variedade de serviços, dispositivos e formas de transporte na rede.

Entretanto, existem problemas na definição de JXTA, principalmente na questão da invocação de serviços. Atualmente existem padrões que definem como um serviço é invocado, entre eles o Web *Services Description Language* (WSDL) amplamente utilizado na invocação de serviços de *web services*, mas nenhum desses padrões é usado pela arquitetura JXTA. Ou seja, o JXTA provê suporte para a troca de informação entre *peers*, mas não tem nenhum mecanismo para trocar informações necessárias para a descrição de serviços [Lim 05]. De modo a superar essa limitação, utilizamos, neste trabalho, a integração de *web services* com a JXTA.

A questão do uso de XML também tem que ser avaliado pois, apesar das vantagens conhecidas, o XML apresenta como principal desvantagem um alto *overhead* causado pela troca de mensagens. Dependendo do tipo de aplicação, pode ser um grande problema, principalmente se uma aplicação não tem o objetivo de aproveitar as capacidades do JXTA de incorporar outros serviços P2P [lim 05]. Mas o JXTA não restringe a troca de mensagens entre os *peers* com o uso de XML, deixando a plataforma mais flexível, sendo esse um dos principais motivos para a escolha dessa plataforma.

O próximo capítulo apresenta o conceito de serviços web, que será integrado à arquitetura JXTA neste trabalho, com mais detalhes.

7. Serviços web (*Web Services*)

Os serviços *web* são identificados por uma URI *(Unique Resource Identifier)*, e são descritos e definidos usando XML. Um dos motivos que tornam os serviços *web* atrativos é o fato deste modelo ser baseado em tecnologias padrão, em particular XML e HTTP. Os serviços *web* são usados para disponibilizar serviços interativos na WEB, podendo ser acessados por outras aplicações. O SOAP *(Simple Object Access Protocol)* está se tornando padrão para a troca de mensagens entre aplicações e serviços *web*, já que é uma tecnologia construída com base em XML e HTTP [Web 06a].

7.1. Arquitetura Orientada a Serviço – SOA (*Service Oriented Architecture*)

Os serviços *web* são descritos como uma arquitetura baseada em serviços, também conhecida por SOA *(Service Oriented Architecture)*, termo inicialmente cunhado pela IBM. Os componentes da arquitetura SOA representam uma coleção de serviços que se comunicam através da troca de mensagens XML. Nessa arquitetura estão definidos três papéis que interagem entre si. Os papéis são [Ser 06a]:

- Provedor de serviço – responsável pela descrição e publicação de um determinado serviço *web* no registro dos serviços. O provedor também é responsável por descrever as informações de ligação do serviço usadas para sua chamada. As informações estão representadas em um documento XML escrito na linguagem padrão WSDL *(Web Service Description Language).*
- Consumidor do serviço – responsável por descobrir um serviço, obter a sua descrição e, usá-lo para se ligar a um provedor a fim de invocar um serviço *web.*
- Registro dos serviços - mantém um diretório com informações sobre serviços, como por exemplo, nome, provedor e categoria. O padrão

adotado na SOA para registro é o UDDI *(Universal Description, Discovery and Integration).*

A interação entre os três papéis envolve a publicação da informação sobre um determinado serviço, a descoberta dos serviços disponíveis e a ligação entre esses serviços [Ser 06a].

7.2. Tecnologias para Serviços *web*

Algumas das tecnologias mais usadas para serviços *web* são:

- *Extensible Markup Language* (XML) - é um formato de texto simples, muito flexível, derivado do Padrão Internacional de Escrita de Arquivos Hipertexto na Internet *(Standard Generalized Markup Language)* SGML (ISO 8879). Originalmente projetada para encontrar meios de publicar em grande escala, facilitando o compartilhamento de informações através da Internet, o XML está assumindo um papel cada vez mais importante na troca de dados na *web* [Ext 06].
- *Simple Object Access Protocol* (SOAP) - protocolo para intercâmbio de mensagens entre programas de computador. O SOAP é um dos protocolos usados na criação de serviços *web*. Geralmente os servidores SOAP são implementados utilizando-se servidores HTTP pré-existentes, embora isso não seja uma restrição para o funcionamento do protocolo. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo orgão W3C [Soa 06]. Toda mensagem SOAP deve conter o envelope (que é o elemento raiz do documento XML) e um cabeçalho, que é opcional e carrega informações adicionais (como por exemplo, se a mensagem deve ser processada por um determinado nó intermediário). O corpo, que é obrigatório, contém a informação a ser transportada para o seu destino final [Web 06a].
- *Universal Distribution Discovery and Interoperability* (UDDI) - é uma especificação que define um serviço de registro para serviços web. Um serviço de registro UDDI é um Serviço web que gerencia informação

sobre provedores, implementações e metadados de serviços, o qual suporta processos de registro para a composição automática dos serviços [Mor 04]. É uma especificação industrial para publicação e localização de informações sobre os serviços *web* [Des 06].

- *Web Service Description Language* (WSDL) - padrão baseado em XML para descrever o serviço, no qual ele descreve os métodos do serviço *web*, suportando a definição de funções de um serviço *web* [Mor 04]. É uma nova especificação para descrever serviços baseados em XML conectados a rede. É a principal parte do UDDI para fornecer diretórios e descrições de tais serviços on-line para negócios eletrônicos, como URL para acessar o serviço e nome do serviço [Usi 06].
- *Hypertext Transport Protocol* (HTTP) - protocolo para comunicações entre computadores utilizados, majoritariamente, sobre a Internet e intranetes. Esse protocolo serve de base à *World Wide Web* (WWW), permitindo-nos visualizar as páginas *web* ao estabelecer a ligação entre os clientes WWW e os servidores *web* [Int 06a].

Tecnicamente, serviços *web* são serviços distribuídos que processam mensagens SOAP codificadas em XML, enviadas através de HTTP e que são descritas através de *Web Service Description Language* (WSDL).

Na Figura 10 deve ser observado que a aplicação cliente (definida em JAVA, C++, ou outra linguagem) está enviando chamadas para um serviço no servidor. A aplicação cliente, após localizar o serviço remoto (definido por um documento WSDL) em um repositório UDDI, invoca os seus serviços, através de RPC. O serviço *web* recebe e processa a chamada e envia uma resposta. É válido lembrar que ambos, cliente e serviço *web*, interagem usando SOAP sobre HTTP.

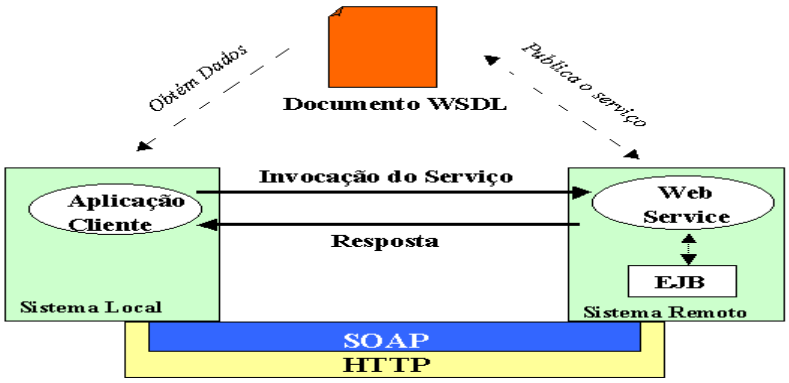


Figura 10 - Aplicação cliente acessando diretamente um serviço *web* [Web 06].

7.3. Especificações dos Serviços *web*

A arquitetura de serviços *web* é uma plataforma que tem por objetivo integrar aplicações na *web*. A plataforma de serviços *web* é classificada em quatro conjuntos de especificações que têm em comum o uso da linguagem XML. Esses conjuntos são [Cur 01]:

- Descrição de serviços - utilizada para definir as operações, mensagens e os tipos de dados de um serviço, além de manter as informações sobre como acessar os serviços.
- Publicação e descoberta de serviços - contém os protocolos que possibilitam a localização da descrição dos serviços. Conceitualmente o protocolo UDDI apresenta três papéis: a) Páginas Brancas - contém identificadores sobre o contato técnico do serviço oferecido, b) Páginas Amarelas - contém informações genéricas sobre os tipos e localização dos serviços disponíveis e c) Páginas Verdes - contém informações técnicas sobre um determinado serviço *web*.
- Descrição de composição de serviços - contém os modelos e linguagens utilizadas para descrever como se dará a interação dos serviços.
- Protocolos de comunicação - utilizados para definir, estabelecer e manter a comunicação entre as aplicações, além de conter a descrição dos formatos das mensagens utilizadas no estabelecimento da comunicação entre aplicações.

7.4. Considerações Finais

Nesse capítulo foram apresentados os conceitos principais dos serviços *web*, como a arquitetura orientada a serviços (SOA), as tecnologias para serviços *web* (XML, SOAP, WSDL, UDDI e HTTP) e por fim as especificações dos serviços *web*. No próximo capítulo será apresentada a proposta desse trabalho, que utiliza os serviços *web*, integrados à arquitetura JXTA, para implementar uma estrutura de suporte aos serviços de Gerenciamento da Distribuição de Dados (WS-DDM) em simulações distribuídas, em conformidade com o padrão HLA.

8. WS-DDM: Uma arquitetura de suporte ao Gerenciamento da Distribuição de Dados Baseada em Serviços Web para Simulação Distribuída

Neste capítulo é apresentada a arquitetura WS-DDM *(Web Service-Based Data Distribution Management*) de suporte ao Gerenciamento da Distribuição de Dados Baseada em Serviços Web para Simulação Distribuída. O objetivo principal dessa arquitetura é permitir que vários participantes (federados) entrem e saiam da simulação em tempo de execução sem a necessidade de parada desta, permitindo a mudança na configuração da simulação em tempo real, sem precisar esperar que a simulação termine para alterá-la. A Figura 11 apresenta a visão geral da arquitetura WS-DDM.

Na WS-DDM, cada *peer* é estruturado conforme mostra a Figura 12, contendo: o visualizador/controlador que mostra para o participante os outros objetos da simulação que podem ser vistos naquele momento, além dos serviços da HLA e serviços extras como o gerenciamento do envio de dados na rede e a sincronização do tempo, os protocolos do JXTA que definem como a comunicação é feita de forma descentralizada em uma rede *peer-to-peer* e por fim um cliente do serviço *web* que permite a entrada e saída da simulação de forma centralizada.

Cada *peer* possui: um cliente *web*, o espaço de roteamento da federação, um visualizador, a lista dos federados que integram a federação, suas coordenadas no espaço de roteamento, a identificação das células que envolvem sua área de interesse e os serviços da interface de especificação da HLA.

A WS-DDM contém também o endereço do repositório da lista de servidores de federação que fornece os endereços dos mesmos através de *web services*. O cliente *web* em cada participante é utilizado para acessar o servidor de federação, escolher uma sessão (federação) que deseja participar e cadastrar seus dados para iniciar sua participação recebendo a lista de cadastros com o endereço dos outros participantes. Essa é a primeira troca de dados feita pelo federado, através de serviço *web*, depois de ter encontrado o endereço dos servidores de federação que se encontra em um repositório.

No visualizador, o participante pode enxergar o que está dentro do seu campo de visão para tomar suas próprias decisões, além de possibilitar a troca de informações.

Com o algoritmo DDM sendo executado em cada participante, é feita a verificação entre os demais participantes para apresentar ou não sua área de interesse no visualizador e evitar mensagens irrelevantes na rede.

Nos servidores de federação são armazenados os IPs, identificação dos federados e o espaço de roteamento da simulação que serão acessados pelos participantes, tendo neles o serviço responsável pelo cadastro de cada federado.

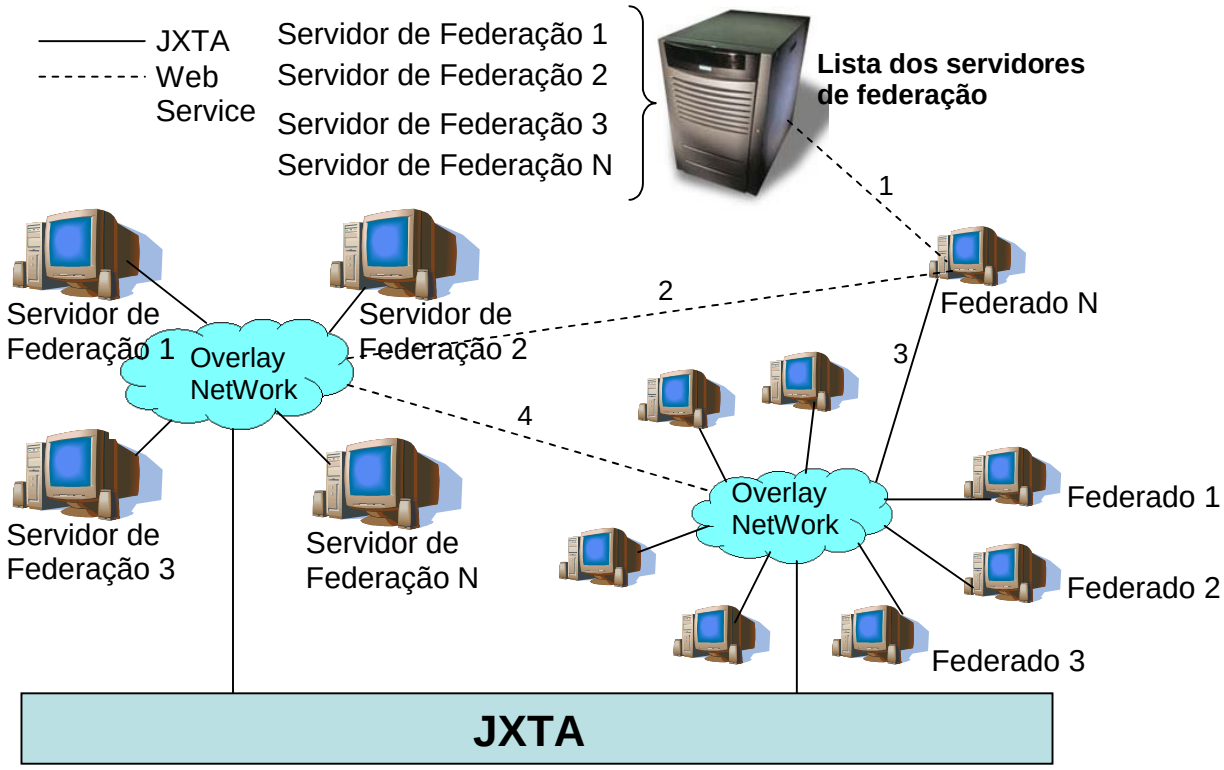


Figura 11 - Arquitetura WS-DDM

Os passos para um federado entrar, participar e sair da simulação são os seguintes. Dado o federado N, este primeiro busca, em um servidor UDDI, o endereço dos servidores de federação e escolhe um para então fazer seu cadastro (IP e identificação) e acessá-lo para pegar todas informações de que precisa, tais como: tamanho do espaço de roteamento e os IPs dos federados já cadastrados. A partir daí, o federado N já pode se comunicar com os outros federados de forma descentralizada e interagir com todos eles. Quando o federado N quiser deixar a simulação, envia uma mensagem para todos os federados para ser excluído e acessa o servidor de federação para excluir seu cadastro.

8.1. Descrição dos *Peers*

Conforme descrito no início deste capítulo, cada *peer* possui: um cliente *web*, o espaço de roteamento da federação, um visualizador, a lista dos federados integrantes na federação, suas coordenadas no espaço de roteamento, a identificação das células que envolvem sua área de interesse e os serviços da interface de especificação da HLA para interagir com a simulação. A Figura 4 mostra as camadas que constituem os *peers*.

A arquitetura JXTA é uma boa opção para redes P2P, pois possui um conjunto de protocolos que permite criar redes descentralizadas e suportar um número crescente de *peers* em uma simulação. Uma das limitações da JXTA é a falta de suporte à invocação de serviços e por isso o serviço *web* é utilizado para complementar o JXTA, mostrando que essas tecnologias podem trabalhar juntas. Observe que a comunicação pode ser feita por rede centralizada, quando é usada a tecnologia de *web services* e por rede P2P, quando são usados os protocolos do JXTA para comunicação entre os *peers*.

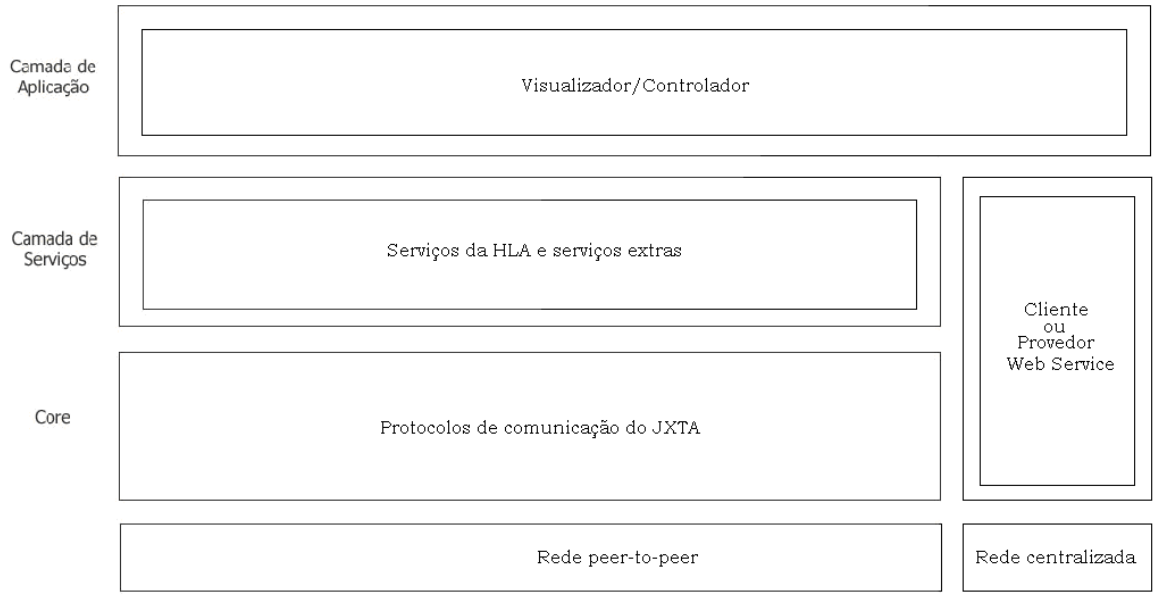


Figura 12 - Camadas dos peers

Cada participante possui um Visualizador/Controlador (implementado no *Microsoft Visual Studio* 2005 utilizando a linguagem VB.Net) na camada de aplicação, conforme mostra a Figura 12. Possui também um conjunto de serviços que estão definidos como Serviços da HLA, mas que podem incorporar serviços extras, tais como a técnica de

dead-reckoning e um servidor controlador para que pequenos dispositivos de baixo poder de processamento como *Palm tops* e celulares possam controlar um computador remotamente, possuindo apenas o Visualizador/Controlador para executar o controle. A Figura 13 apresenta a interface gráfica do visualizador que cada participante possui para controlar e/ou visualizar o estado da simulação.

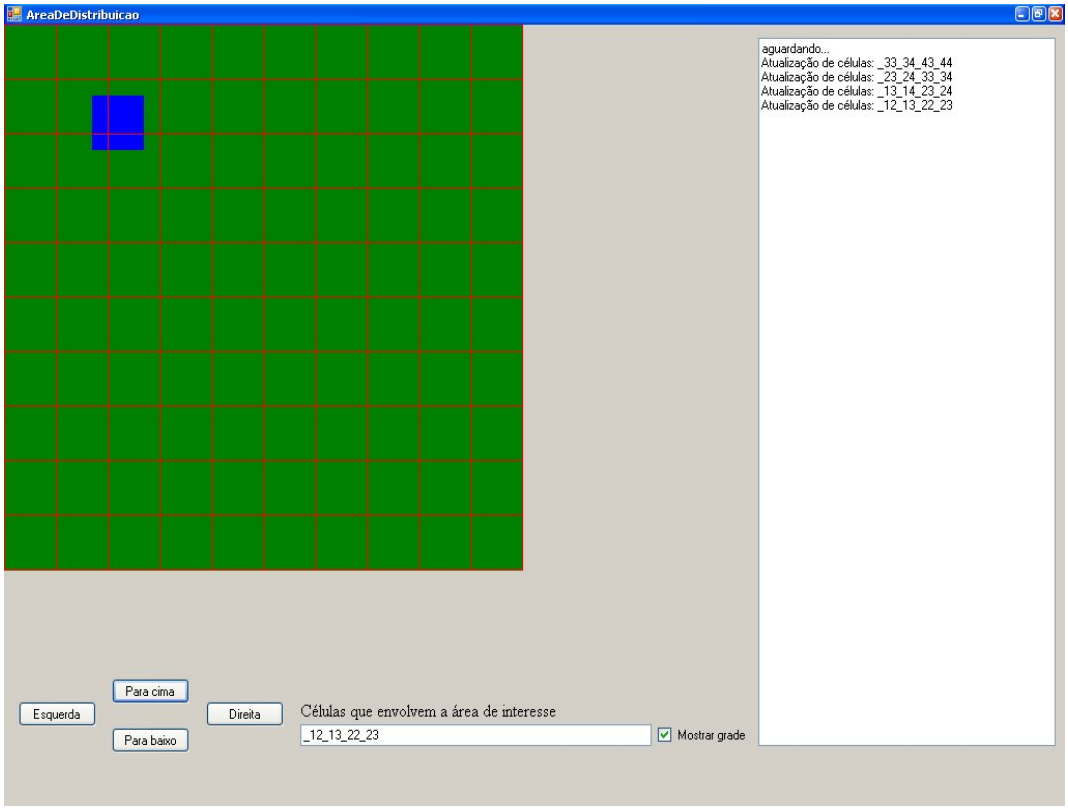


Figura 13 - Visualizador/Controlador

8.2. Descrição do Algoritmo DDM Implementado

Nesta sessão será explicado o funcionamento do algoritmo DDM implementado neste trabalho de mestrado. O algoritmo DDM utilizado foi o baseado em grade dinâmica, que tem como objetivo, reduzir o número de grupos *multicast* evitando o envio de mensagens irrelevantes na rede. Os algoritmos DDM baseado em grade são técnicas que sobrepõem uma grade sobre o espaço de roteamento, para que sejam identificadas as células que envolvem a área de interesse de um federado. O espaço de roteamento é a área que acontece a simulação e a área de interesse pode ser entendida como sendo a área de visão de um federado.

O algoritmo implementado funciona da seguinte forma. Para se conseguir identificar as células que envolvem a área de interesse de um federado, é preciso conhecer o tamanho do espaço de roteamento que é representado pelas variáveis x e y, além também das coordenadas da área de interesse do federado que são as variáveis x1, x2, y1, e y2. Deste modo, é criada uma grade sobre o espaço de roteamento, sendo que cada célula possui o mesmo tamanho, por exemplo, 50x50 pixels e são numeradas da esquerda para a direita de cima para baixo. Com essas células numeradas é criado um conjunto de número de células

que envolvem a área de interesse do federado. O conjunto definido é enviado para os outros federados da simulação para que eles possam analisar se existe alguma intersecção com outro federado. Se esta existir, o federado remoto que identificou a intersecção pede as coordenadas exatas para o federado local para que seja visualizado no visualizador remoto. O mesmo pode acontecer no federado local, pois quando o conjunto de células é identificado, é feita uma análise local com os conjuntos de células dos federados remotos para identificar intersecções. Se estas existirem é solicitada a posição exata do federado remoto para ser apresentada no visualizador.

Para que o algoritmo DDM implementado encontre o conjunto de células, primeiro são encontradas as células das quinas da área de interesse, que são armazenadas em um vetor. Depois disso, as células das quinas são armazenadas junto com as células do meio que foram encontradas em outro vetor e assim é definido o conjunto dos números de cada célula no espaço de roteamento. O algoritmo implementado é descrito no Apêndice A.

8.3. Descrição de uma Simulação utilizando o WS-DDM *(Web Service-Based Data Distribution Management*)

Para entendermos melhor como a arquitetura funciona durante uma simulação, chamaremos de P1 o primeiro participante, P2 o segundo participante e assim por diante. Observe a Figura 14, que mostra cinco instantes diferentes (T1, T2, T3, T4 e T5). Em T1 o P1 faz o acesso ao serviço *web* e depois espera outro participante entrar para iniciar uma comunicação através do JXTA no instante T2. Logo em seguida P2 faz no instante T3 o mesmo que P1 fez no instante T1 e no tempo T4 junta-se com P1. Desta forma, P1 e P2 trocam informações e P2 resolve sair no instante T5 usando serviços *web.*

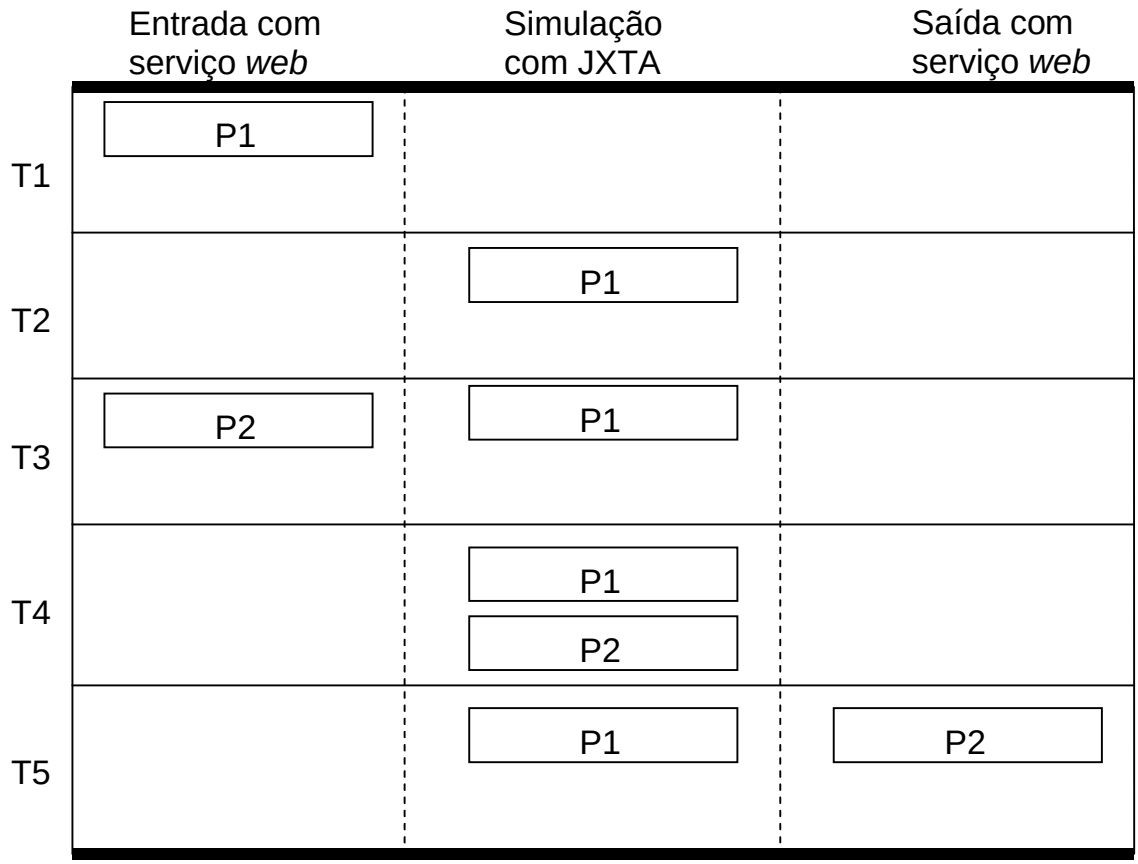


Figura 14 – Diagrama de P1 e P2 entrando e saindo da simulação

Quando P1 desejar entrar na simulação, ele fará sua primeira troca de dados através de um serviço *web* com algum servidor de federação (depois de ter consultado o repositório da lista de servidores de federação e escolhido qual servidor de federação vai usar, como mostra a Figura 15), sendo que, utilizando o cliente *web*, P1 verifica se existe uma sessão existente e decide entrar em uma sessão que já foi criada (se houver) ou criar uma nova sessão (federação) passando seus dados, endereço IP e identificação. Se P1 criar uma nova federação, o servidor de federação manda um aviso para que seja definida a área de distribuição, assim P1 envia as coordenadas da área de distribuição e depois define suas coordenadas (permanecendo localmente na máquina do participante). A partir desse momento, o servidor de federação já sabe a área de distribuição e o IP de P1. Desta forma, quando P2 entrar na mesma federação, fará o mesmo que P1 fez, enviando endereço IP, identificação e definindo suas coordenadas localmente.

É importante mencionar que estamos assumindo que os servidores de federação são sincronizados com auxílio do serviço de gerenciamento de tempo da HLA. Assim, quando um servidor de federação adiciona ou exclui um participante, o mesmo

comunica o fato aos outros servidores de federação para manter a rede atualizada. Depois do cadastro, P2 recebe a área de distribuição e uma lista do servidor de federação com todos os participantes da federação. Até aqui, as trocas foram feitas somente por *web services.* No caso de P1, este estabelece uma conexão via *socket* com os membros da lista, passando seus dados (coordenadas e IP) para cada um e recebendo suas coordenadas. A partir daqui, as trocas de dados são feitas sobre a plataforma JXTA, usando seus protocolos, como o *Peer Information Protocol* (PIP), *Pipe Binding Protocol* (PBP), *Peer Resolver Protocol* (PRP) e *Rendezvous Protocol* (RP). Os *web services* são usados novamente somente quando um participante deixar a simulação ou houver alguma correção de falha que será tratada na próxima subseção. Quando houver mais de um integrante na federação, é iniciada a execução do algoritmo DDM para definir os grupos *multicast* e assim apresentar o campo de visão de cada participante em seu visualizador local.

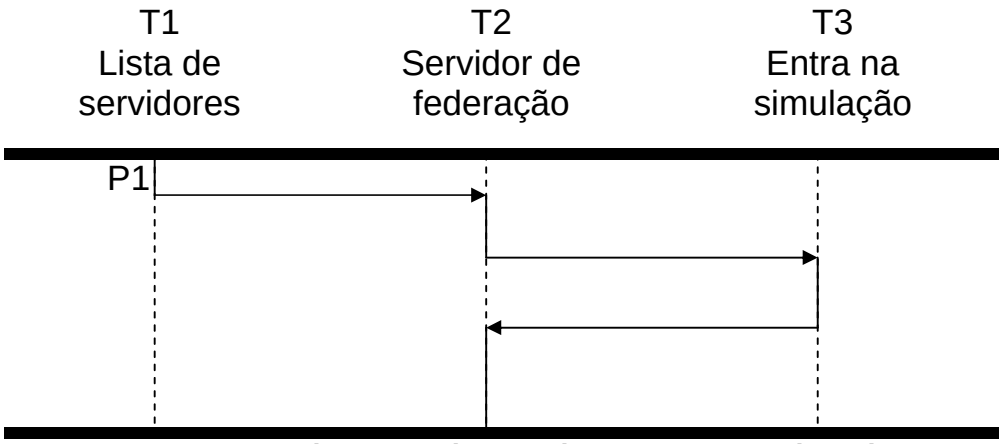


Figura 15 – Diagrama de P1 acessando os servidores para entrar e sair da simulação

Quando um participante desejar sair da simulação, ele deve enviar um pedido de exclusão para o servidor de federação, excluindo seu cadastro da lista de participantes e, em seguida, fazer a mesma operação para todos os integrantes da simulação via *socket.* A próxima seção trata de exceções.

Observe que, conforme mais participantes ingressam na federação, é formada uma rede *peer-to-peer* descentralizada, pois nenhum nodo/*peer* (participante) mantém algum tipo de conexão com o servidor de federação, fazendo apenas duas comunicações com o mesmo: quando entra e quando sai da federação.

Com a formação de uma descentralizada surge um problema clássico nas redes *peer-to-peer*, que é a complexidade de gerenciamento. Por outro lado, ganha-se na escala do sistema, já que a limitação devido à criação de gargalos em servidores da rede é superada,

pois os acessos a servidores são realizados apenas na entrada e na saída de cada federado da federação - cada *peer* possui todos os serviços necessários localmente.

8.4. Tratamento de saída anormal de federados no WS-DDM

Um dos propósitos das simulações é prever o que poderá ocorrer em um sistema real, sem ter que utilizar um. Com isso, é importante que uma simulação forneça dados confiáveis. Assim, em nossa arquitetura, uma questão importante que deve ser tratada é a saída anormal de federados durante a simulação, para que a execução prossiga sem ter que parar ou ficar em estado inconsistente devido à saídas abruptas dos participantes. Desta forma, *peers* gerenciadores são definidos ao iniciar a simulação, de forma que a área de distribuição seja toda coberta pelas áreas de subscrição desses gerentes. Esses gerenciadores são necessários, pois, para que cada *peer* seja monitorado por algum sinal na rede, o mesmo precisa enviar uma mensagem em um intervalo de tempo para todos os outros *peers* participantes e evidentemente que os outros precisam fazer o mesmo, isso resulta em várias mensagens recebidas ao mesmo tempo e congestionado os participantes. O uso dos gerenciadores é para que isso não ocorra, pois eles são computadores preparados para esse tipo de serviço.

A Figura 16 mostra que o *peer* gerenciador, ou apenas gerenciador, possui apenas área de subscrição e não de publicação como um federado comum, já que ele não precisa enviar dados, tais como coordenadas ou células de posicionamento. O tamanho da área de subscrição de cada gerenciador é definido de forma livre, atentando-se para o fato de que a soma das áreas dos gerenciadores seja igual à área de distribuição.

De modo a considerar o federado pertencente a uma área de cobertura de um gerenciador, as posições de x1 e y1 deste precisam estar dentro da área que pertence a um gerenciador, visto que a área do federado pode estar em uma ou mais áreas ao mesmo tempo. Considerando-se que em uma simulação tenhamos P1, P2 e P3, na qual P1 e P2 estão sendo gerenciados pelo gerenciador G1 e P3 pelo gerenciador G2 como mostra a Figura 17.

Suponhamos que P1 tenha um problema e com isso perca a conexão com a rede, conseqüentemente, perdendo a conexão com os membros da simulação. Neste caso, P1 permanecerá na lista de participantes do servidor de federação e dos integrantes até que G1 identifique essa ausência e mande uma mensagem para todos os gerenciadores. Dessa forma, cada gerenciador manda uma mensagem para os federados que estão em sua área de

subscrição (no caso, G1 para P1, P2 e G2 para P3) e para um servidor de federação para que eles fiquem atualizados, pedindo a exclusão de P1. A mensagem que é enviada para os federados possui a identificação do federado que deve ser excluído. Essa mensagem deve ser repassada para que todos os federados da simulação sejam avisados, assim P1 é finalmente eliminado da simulação sem que a execução seja interrompida.

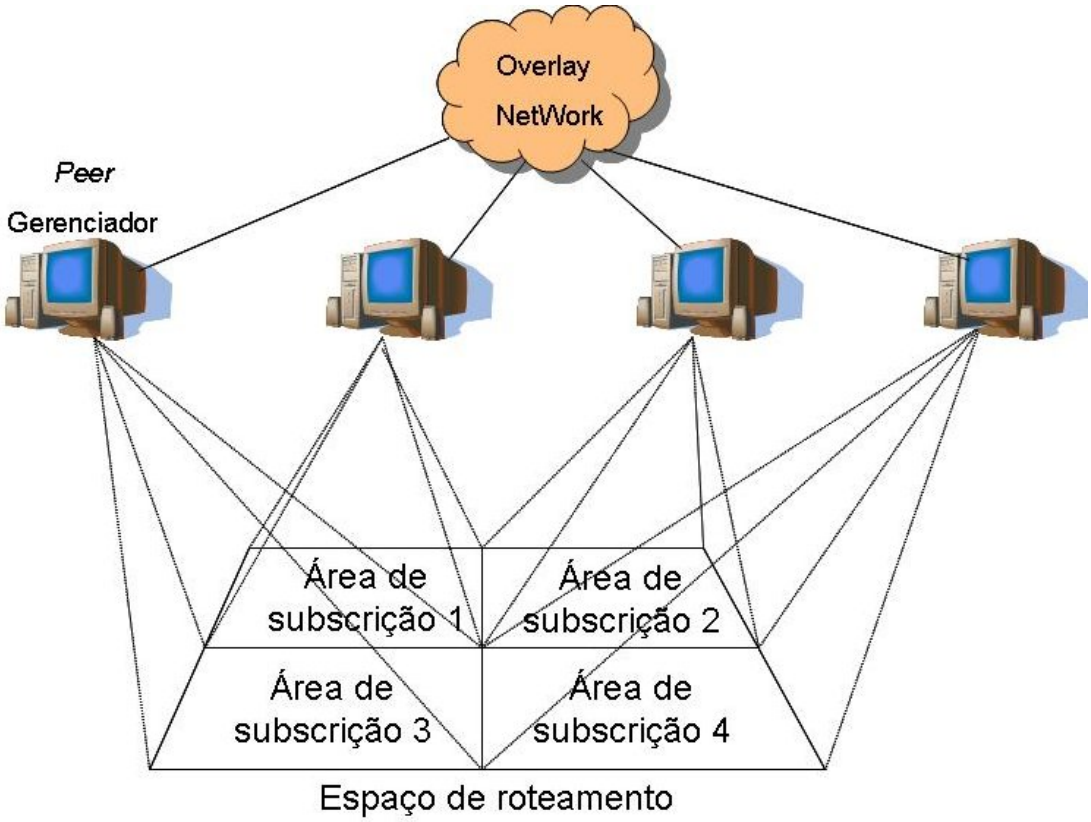


Figura 16 – Gerenciadores cobrindo o espaço de roteamento total.

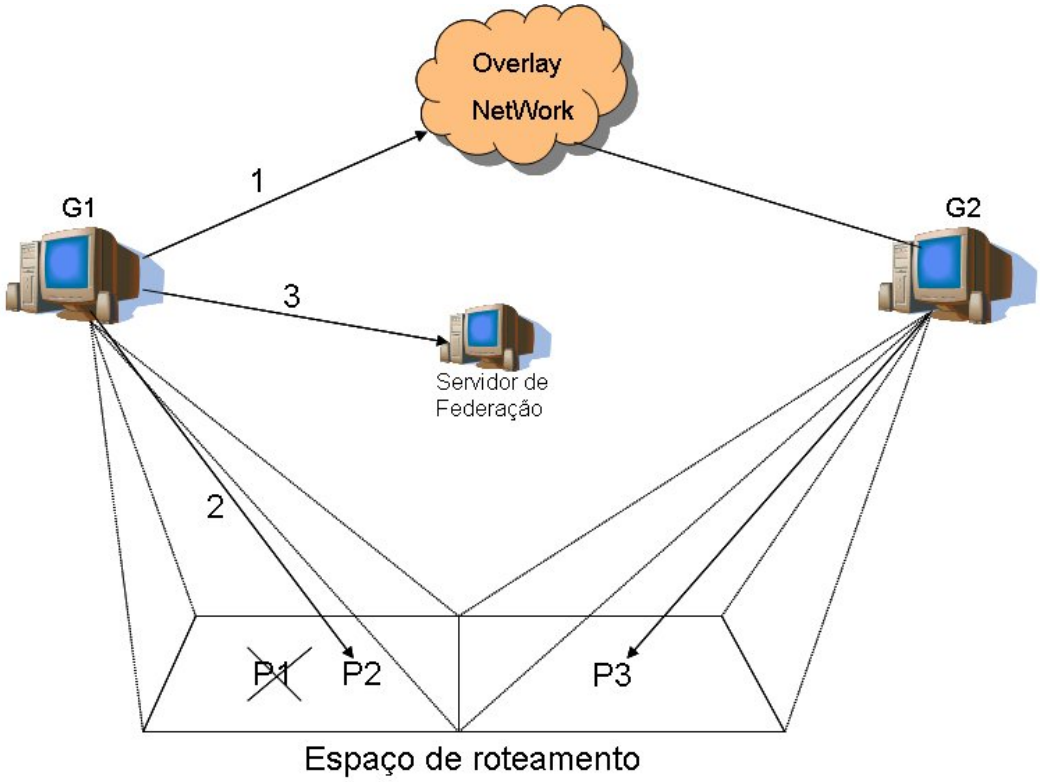


Figura 17 – Gerenciador detectando P1 com falha no espaço de roteamento.

8.5. Considerações Finais

O capítulo 8 apresentou a arquitetura WS-DDM de suporte ao serviço de gerenciamento da distribuição de dados de simulações distribuídas, em conformidade com a arquitetura HLA. A WS-DDM utiliza as tecnologias de serviços web e a plataforma JXTA de forma integrada. Foram apresentadas as camadas que compõem cada um dos *peers*, bem como o Visualizador/Controlador, além do algoritmo de distribuição de dados (DDM) que foi implementado como um dos serviços da HLA. Foi descrito também o mecanismo de tratamento de saída anormal de federados, um aspecto limitante importante das implementações tradicionais da arquitetura RTI. No próximo capítulo uma análise de desempenho da arquitetura WS-DDM é realizada.

9. Ilustração da Arquitetura WS-DDM *(Web Service-Based Data Distribution Management*)

Este capítulo apresenta uma avaliação da arquitetura WS-DDM descrita no capítulo 8, que foi proposta e implementada como parte dessa dissertação de mestrado.

9.1. Número de mensagens necessárias para atualização da simulação

Nesta seção, a arquitetura WS-DDM é comparada à arquitetura WSIM *(Web Services Internet Management)* [PUL 04], compatível com a arquitetura HLA, em termos de número de mensagens necessárias para a manutenção da consistência da simulação. Na arquitetura WSIM os federados são máquinas controladas remotamente por clientes que possuem um visualizador/controlador. Esses clientes podem controlar um ou mais federados de uma vez. Na WSIM, o serviço de gerenciamento de interesse é dividido em três funções: RBAC *(Role Based Access Control)* que especifica que tipo de informação um usuário particular pode acessar; AOIM *(Area of Interest Management)* que especifica o raio de visão do usuário; e a AGIM *(Aggregation Interest Management)* que gerencia como os dados agregados serão apresentados. Na WSIM, os federados são máquinas que podem ser controladas por usuários remotamente através de um visualizador/controlador, sendo que cada usuário pode controlar mais de uma máquina.

Na arquitetura WSIM os federados são maquinas controladas por clientes que possuem um visualizador/controlador, o que não acontece na arquitetura WS-DDM, onde os federados não são controlados por ninguém, pois os federados são os próprios clientes.

Definindo que toda a comunicação, tanto na arquitetura WSIM como na WS-DDM seja feita por *sockets* e a mensagem seja de 5 Bytes, observe o tráfego gerado pelas duas arquiteturas no pior caso:

Se na arquitetura WS-DDM um federado provocar uma ação e mandar para todos os outros federados da simulação, o tráfego será de 3 * 5 Bytes = 15 Bytes de tráfego na rede. Já na arquitetura WSIM se um federado fizer o mesmo que um federado fez na WS-

DDM, o tráfego será de (3 * 5 Bytes) + 5 Bytes = 20 Bytes, pois como WSIM possui um controlar, exige no mínimo mais uma mensagem do controlador pra o federado controlado.

Assim pode-se prever o tráfego da rede nessas situações como:

Considerando-se as seguintes métricas:

WS-DDM: NM = (NF - 1) * NA

WSIM: NM = ((NF - 1) * NA) + NA

Onde:

NF - Número de federados

NM - Número de mensagens

NA - Número de ações

O *NA* a mais, considerado na arquitetura WSIM significa que esta precisa enviar uma mensagem a mais do que a WS-DDM (isto porque no WSIM o federado é controlado remotamente por um cliente o qual, quando gera um evento, precisa enviar este evento para o federado que ele controla e depois para os outros federados da simulação).

A figura 18 mostra o número de mensagens trocadas pelas duas arquiteturas para manutenção da consistência da simulação.

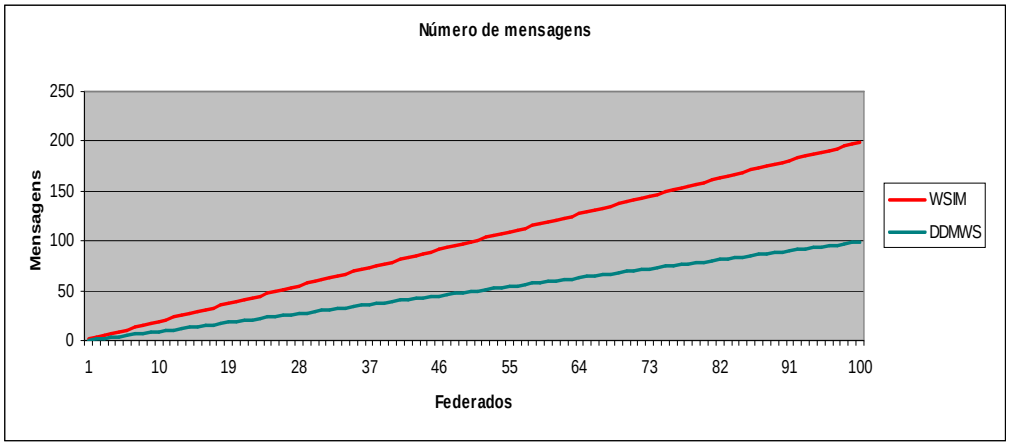


Figura 18 – Gráfico do número de mensagens de WSIM e WS-DDM.

9.2. Avaliação do tamanho da célula

Uma questão importante a ser considerada na implementação do serviço de gerenciamento da distribuição de dados é a definição do tamanho da célula. Assim, foi feita uma demonstração cinco federados para ilustrar a importância da definição do tamanho da

célula. Para que os federados A, B, C e D possam visualizar o federado E, eles precisam caminhar aproximadamente até que suas áreas de interesses se sobreponham a célula que o federado E está. Se a distância percorrida for menor, o erro de intersecção tende a ser maior e o número de mensagens de atualização tende a ser menor. Esse é um problema clássico para a definição do tamanho da célula. Supondo uma área de distribuição de 10 X 10 km, podemos variar o tamanho da célula para encontrar o tamanho ideal para esse exemplo.

Considerando que os federados A, B, C e D irão caminhar até visualizarem o federado E, e que ambos possuem 0,5 X 0,5 (Km) de área de interesse, e estão a diferentes distâncias do federado E. Considerando também que o tamanho máximo da célula seja 5 X 5 (Km), podemos observar o comportamento do número de mensagens transmitidas na rede e o erro de intersecção que é causado em cada caso.

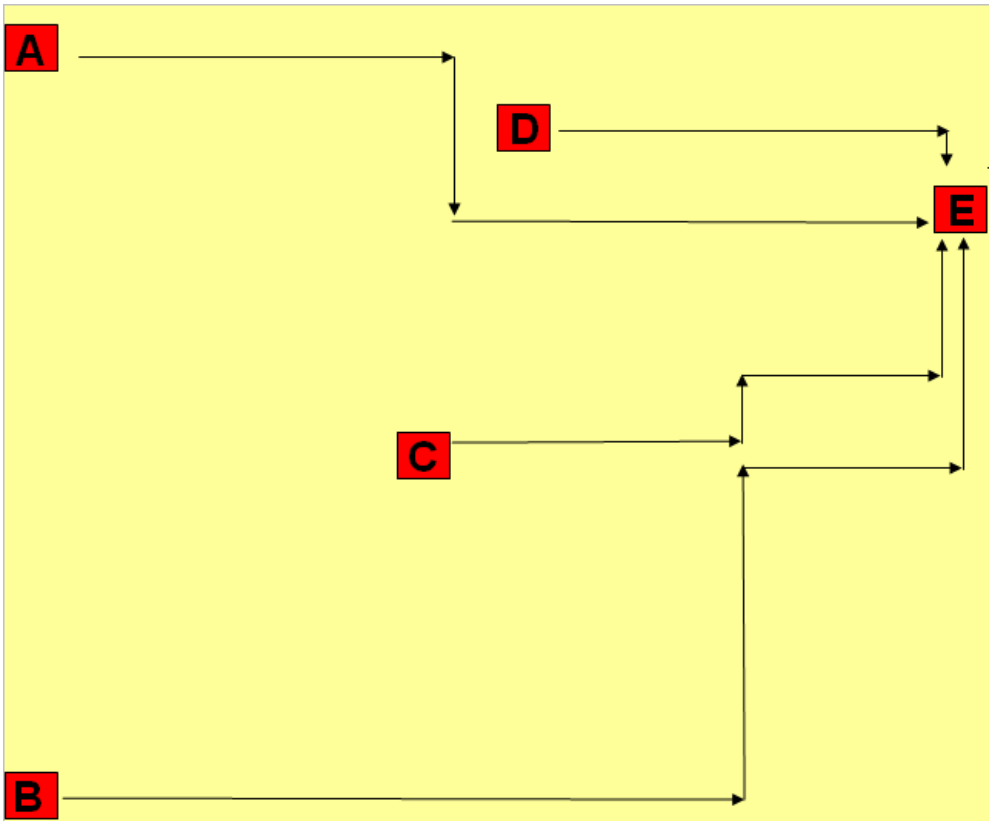
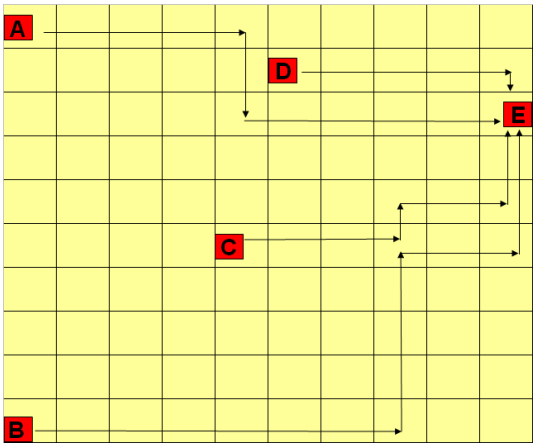
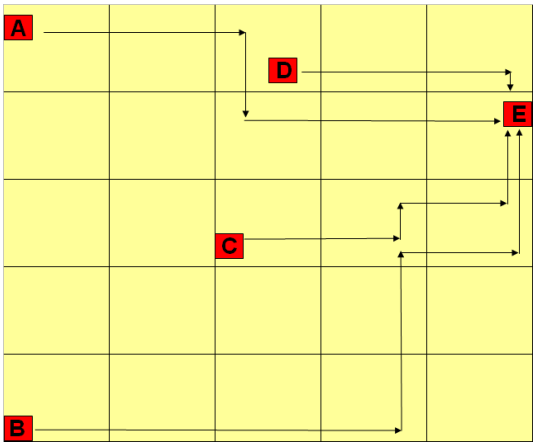


Figura 19 – Área que envolve os federados A, B, C, D e E de 10 x 10 Km.

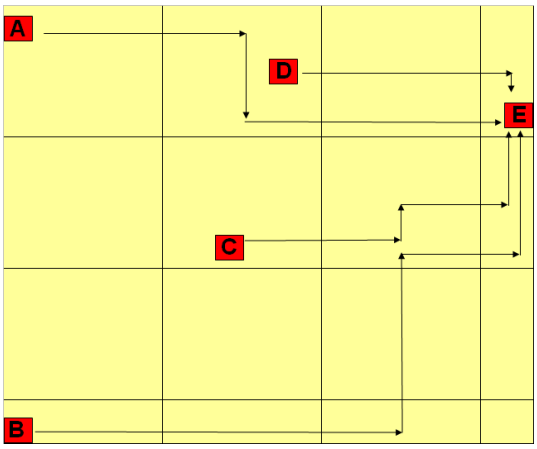
Observe a Figura 20, onde estão os cinco tamanhos de células, a posição dos federados A, B, C, D e E, com linhas que representam a trajetória de cada objeto até visualizarem o federado E.



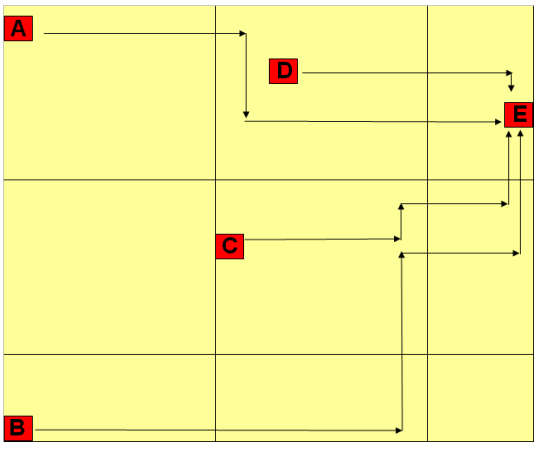
Célula de tamanho 1x1



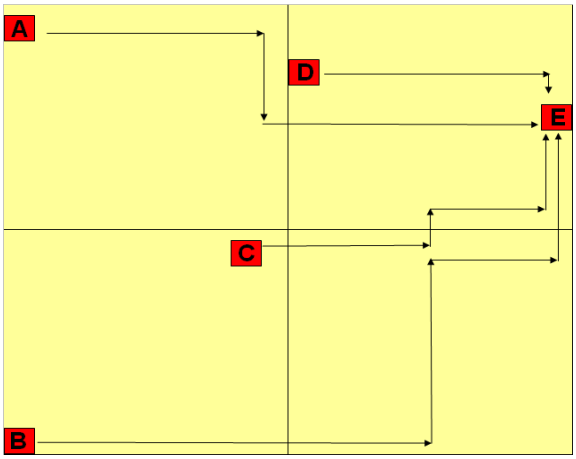
Célula de tamanho 2x2



Célula de tamanho 3x3



Célula de tamanho 4x4



Célula de tamanho 5x5

Figura 20 – Áreas com tamanhos de células diferentes.

A tabela abaixo apresenta o tamanho da célula e o número de mensagens que serão transmitidas até os federados A, B, C e D avistarem o federado E. A tabela apresenta também o erro gerado para cada caso pela técnica em grade.

Tabela 17 - Comparação entre os tamanhos das células

Tamanho da célula em Km²	Número de mensagens para cada objeto				Erro gerado para cada objeto			
	A	B	C	D	A	B	C	D
1 x 1	21	31	15	9	0,5	0,25	0,25	0,25
2 x 2	9	13	5	5	1,5	1,25	1,25	0,25
3 x 3	5	11	5	3	0,5	0,25	0,25	1
4 x 4	3	7	3	1	1,5	1,25	1,25	2
5 x 5	1	3	3	0	4,5	2,25	3,5	4

A Figura 20 mostra graficamente os resultados da Tabela 17. Conforme o tamanho da célula aumenta, o número de mensagens necessárias para manter a consistência da simulação diminui.

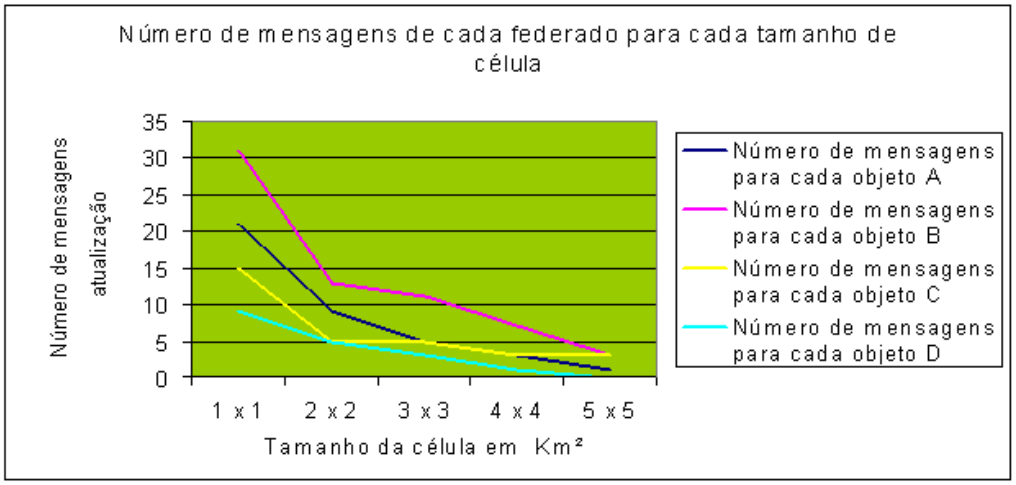


Figura 201 – Gráfico do número de mensagens para cada tamanho de célula.

A Tabela 17 mostra o erro gerado na técnica em grade. Este erro representa a distância entre as áreas de interesse dos federados. A figura 21 mostra que nessa situação, o comprimento da célula mais interessante é igual a 3, pois possui os valores dos erros mais baixos e um número de mensagens enviadas pequeno.

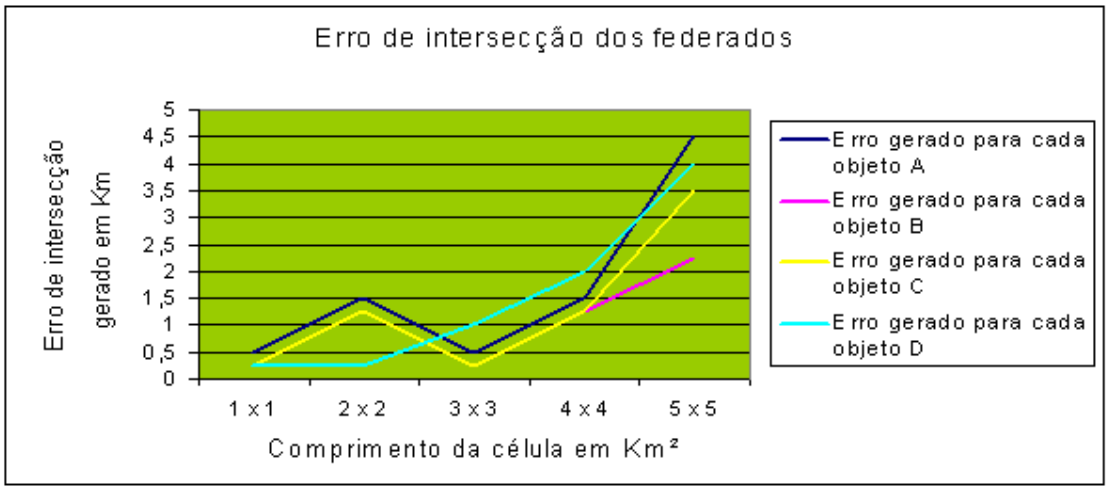


Figura 22 – Gráfico do erro de intersecção de A e B

No cenário descrito acima, um tamanho a ser considerado para o tamanho da célula é 3, pois este valor gera um número de mensagens de atualização aceitável e o erro mais baixo. É importante dizer que o problema da definição do tamanho da célula depende da aplicação. Por exemplo, para uma simulação de vôo, como um avião tem uma área de interesse grande e sua velocidade também é bastante alta, não podemos definir uma célula pequena pois causaria um número de atualização muito grande, dando um fluxo de mensagens na rede muito intenso, assim, definir células maiores neste caso é mais indicado para evitar esse tipo de problema.

9.3. Envio/recepção de mensagens pelo visualizador/controlador

A Figura 23 mostra a capacidade que o visualizador tem para enviar e receber mensagens por segundo, que foi de 50 mensagens enviadas e 30 mensagens recebidas. Foram consideradas três medidas de tempo para chegar a esses resultados: T1 = 0,02 segundos, que representa o tempo para calcular as coordenadas e decidir enviar ou não a atualização de células; T2 = 0,30, que representa o tempo que leva para empacotar, enviar, receber e desempacotar a mensagem de atualização; e T3 = 0,03, que é o tempo que o visualizador leva para verificar se existe intersecção ou se é um pedido de coordenadas. Para estes valores, o máximo de mensagens enviadas foi de 50 e o máximo de mensagens recebidas foi de 33 mensagens. De modo a aumentar o número de mensagens recebidas por segundo, *threads* deverão ser implementados.

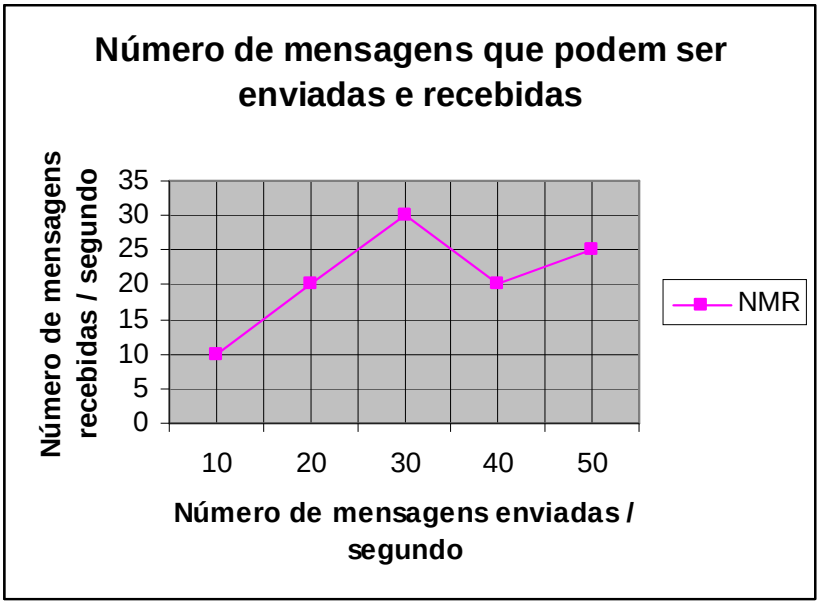


Figura 23 – Número de mensagens recebidas/enviadas.pelo visualizador

9.4. Considerações Finais

Esta seção apresentou uma breve comparação entre as arquiteturas WSIM e WS-DDM em relação ao número de mensagens que cada uma gera na atualização do estado da simulação. Foi discutido também o tamanho da célula que deve ser utilizado, uma questão clássica na técnica baseada em grade, e que depende da aplicação. Finalmente, o Visualizador/Controlador foi avaliado em termos de sua capacidade de transmissão e recepção de mensagens. Melhor eficiência pode ser obtida implementando-se threads. O próximo capítulo apresenta as conclusões.

10. Conclusões

Neste trabalho de dissertação de mestrado foi projetada e implementada uma arquitetura de suporte a simulações distribuídas em conformidade com a HLA, através da integração das tecnologias JXTA e serviços web.

O JXTA é uma tecnologia *Peer-to-Peer* que focaliza no uso eficiente dos recursos da rede para suporte a sistemas de larga escala. Entretanto, o JXTA é limitado em relação à invocação de serviços, ou seja, o JXTA provê troca de informação entre *peers*, mas não tem nenhum mecanismo para trocar informações necessárias para a invocação de serviços Já os serviços web definem formas de como serviços podem ser invocados. Assim, o uso integrado das tecnologias serviços web e JXTA no projeto da arquitetura WS-DDM aqui descrita, mostra ser promissor no suporte a simulações distribuídas. Ambas as tecnologias adotam protocolos abertos e são independentes de linguagens de programação e sistemas operacionais.

Este capítulo descreve as contribuições geradas com a realização deste trabalho, bem como trabalhos futuros e conclusões finais.

10.1. Contribuições Geradas

O projeto e implementação da arquitetura WS-DDM para suporte ao gerenciamento da distribuição de dados, geraram contribuições relevantes principalmente para os membros do Laboratório de Realidade Virtual (LRVNet) do DC da UFSCar, em seus trabalhos de implementação de outros serviços da HLA. Este trabalho contribui na medida que esta arquitetura servirá como base para outros trabalhos em redes *peer-to-peer* em conformidade com o padrão HLA, pois atualmente o RTI não é gratuito como era antes, a medida os outros serviços da interface de especificação forem implementados a arquitetura WS-DDM se fortalecerá, continuando gratuita e aberta para os acadêmicos. O trabalho contribuiu também para superar as limitações que o RTI do DC, permitindo:

- Interação com o usuário;
- Visualização gráfica;

- Entrada e saída de federados durante a simulação.

10.2. Trabalhos futuros

Como continuidade ao trabalho aqui iniciado, as seguintes atividades ainda deverão ser realizadas:

- Aprimoramento do visualizador/controlador para suporte a um número crescente de federados;
- A implementação dos outros serviços da interface da HLA;
- Uso de *peers* diferentes como *palms* e celulares, para participar de uma simulação como federados;
- Implementação da técnica de *Dead-Reckoning* para auxiliar na eficiência da WS-DDM;
- A possibilidade de um participante controlar mais de um federado e, em cada federado, poder criar e controlar vários objetos.

10.3. Conclusões finais

Neste trabalho, o uso integrado das tecnologias JXTA e serviços web tiveram o objetivo de potencializar as vantagens de cada uma dessas tecnologias no suporte a simulações distribuídas de larga escala. Com a arquitetura proposta aqui, limitações foram superadas da versão 1.3 da Infra-estrutura de Tempo de execução (RTI), instalada no cluster do DC da UFSCar, tais como: suporte a um número crescente de objetos/federados; suporte a diferentes tipos de federados (sendo eles diferenciados pelas áreas de interesse); suporte ao *man-in-the-loop;* e suporte ao controle da simulação com a saída/entrada dos federados em tempo de execução, sem que a simulação tenha que ser interrompida.

A integração das tecnologias Peer-to-peer e serviços web mostra ser uma solução em potencial para o suporte a simulações distribuídas de larga escala. Este trabalho pretende ser uma contribuição nesta direção.

11. Referências Bibliográficas

[Abr 98] ABRAMS, H.; WATSEN, K.; ZYDA, M. Three-Tiered Interest Management for Large-Scale Virtual Environments. ACM Symposium on Virtual Reality Software and Technology, 1998.

[Amo 04] AMORIM, S., A Tecnologia Web Services e sua Aplicação num Sistema de Gerência de Telecomunicações, Tese de Mestrado, Universidade Estadual de Campinas, Campinas-SP, 2004.

[Ant 07] ANTES do Second Life e do GoogleEarth: A utilização da simulação baseada na web para o estudo de processos operacionais: Disponível em: http://api.adm.br/artigos/?m=200710. Acesso em 02/11/2007.

[Azz 01] AZZEDINE B.; CARON D., Performance Comparison of Data Distribution Management Strategies, Proceedings of the Fifth IEEE International Workshop on Distributed Simulation and Real-Time Applications, p.67, August 13-15, 2001.

[Azz 02] AZZEDINE B.; ROY, A. Dynamic Grid-Based Approach to Data Distribution Management. Journal of Parallel and Distributed Computing, v. 62, p. 366 – 392, 2002.

[Azz 03] AZZEDINE B.; CARON D.: Dynamic Grid-Based vs. Region-Based Data Distribution Management Strategies for Large-Scale Distributed Systems. IPDPS 2003: 280.

[Azz 04] AZZEDINE B.; A et al. 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.

[Ber 98] AZZEDINE B. Alternative Approaches to Multicast Group Allocation in HLA Data Distribution Management. 98S-SIW-198. Spring Simulation Interoperability Workshop (SIW), March 1998.

[Cor 05] CORREIA, M.; CARDOSO, J. Aumento da resiliência dos Web services com uma Infra-estrutura Peer-to-Peer", 6ª Conferência da Associação Portuguesa de Sistemas de Informação (CAPSI 2005), 26-28, October 2005. Bragança, Portugal.

[Cur 01] CURBERA, F.; MUKHI N.; WEERAWARANA, S. On the emergence of a web services component model. In: Budapest. Proceedings of the 6th International Workshop on Component-Oeiented Programming Budapest: 2001.

[Dah 97] DAHMANN, J.; FUJIMOTO, R.; WEATHERLY, R.. The Department of Defense High Level Architecture, Winter Simulation Conference, Dezembro 1997.

[Dco 07] DCOM Technical Overview.
Disponível em:
http://msdn2.microsoft.com/en-us/library/ms809340.aspx. Acesso em: 10/04/2007

[Dep 06] ESTADOS UNIDOS DA AMERICA. Departamento de Estado. A Internet em constante evolução. Disponível em: <http://usinfo.state.gov/journals/itgic/1103/ijgp/ijgp1103.htm>. Acesso em: 24/03/06.

[Dep 98] ESTADOS UNIDOS DA AMERICA. Departamento de Estado. High Level Architecture Interface Specification, Version 1.3, Defense Modeling and

Simulation Office – DMSO, 1998. Disponível em: http://hla.dmso.mil. Acesso em: 20/01/2006.

[Des 06] DESCRIZIONE del progetto JP². Disponível em: http://dsg.ce.unipr.it/research/JXT Arvp/pag2.html. Acesso em: 08/12/2006.

[Dês 06] DESCRIÇÃO geral dos Serviços. Disponível em: UDDI http://www.microsoft.com/t echnet/prodtechnol/windowsserver2003/pt-pt/library/ServerHelp/d59585 4e-7bba-44f4-a65a-f7a79f23664f.mspx?mfr=true. Acesso em: 10/05/2006.

[Ekl 04] EKLOF, M. et al. Peer-to-peer-based resource management in support of HLA: based distributed simulations. Simulation, v. 80, p. 181 – 190, 2004.

[Esp 06] JXTA. Protocols specification. Disponível em: http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.html. Acesso em: 08/12/2006.

[Ext 06] EXTENSIBLE Markup Language (XML). Disponível em: http://www.w3.org/XML. Acesso em: 11/05/2006.

[Fuj 00] FUJOMOTO, R. et al. Design of high performance RTI Software.: IEEE, 2000. IEEE Distributed Interactive Simulation and Real-Time Applications, DS-RT'2000.

[Gnu 06] GNUTELLA. Disponível em: http://en.wikipedia.org/wiki/Gnutella. Acesso em: 08/12/2006.

[Gnu 07] GNUTELLA. Disponível em: http://www.gta.ufrj.br/grad/06_1/p2p/gnutella.html. Acesso em: 12/03/2007

[Gon 01] GONG L., Project JXTA: A Technology Overview, Sun Microsystems Inc., 2001.

[Hig 06] HIGH Level Architecture - Conceitos, FEDEP e Ambientes. Disponível em: http://www.inf.ufrgs.br/~wildt/cmp157/tf_hla/cmp157_tf_wildt.htm#A_1,24/. Acesso em: 03/2006.

[Hla 06] HLA MODULE 1, Part 3 - The Object Model Template. Disponível em: http://www.ecst.csuchico.edu/~hla/LectureNotes/HLA_1.3NG_M1_P3.pdf. Acesso em: 04/04/2006

[Hoo 98] HOOK, D.J.V.; CALVIN, J.O. Data distribution management in RTI 1.3. 98S-SIW-206, Spring SIW, 1998.

[Iee 00a] Institute of Electrical and Electronics Engineers - IEEE. 1516.2-2000: IEEE Standard for Modeling and Simulation (M&S) High-Level Architecture (HLA) - Framework and Rules.

[Iee 00b] Institute of Electrical and Electronics Engineers - IEEE. 1516.2-2000: IEEE Standard for Modeling and Simulation (M&S) HIGH-LEVEL Architecture (HLA). Object Model Template (OMT) Specification.

[Iee 00c] Institute of Electrical and Electronics Engineers - IEEE. 1516.2-2000: IEEE Standard for Modeling and Simulation (M&S) High-Level Architecture (HLA). Federate Interface Specification.

[Int 06] INTRODUCTION to the High Level Architecture. Disponível em: http://www.ecst .csuchico.edu/~hla/LectureNotes/HLA_1516_M1_P1.pdf. Acesso em: 24/03/2006.

[Int 06a] INTRODUÇÃO à Internet. Disponível em: http://www.educare.pt/DicioWeb/DicioWe bGuiaNet_Cap1E.asp#http. Acesso em: 13/05/2006.

[Int 06b] INTRODUÇÃO às redes peer-to-peer (P2P). Disponível em: http://www.gta.ufrj.br/se minarios/semin2003_1/william/Tecnologia.htm. Acesso em: 01/12/2006.

[Jac 06] JACORB sobre JXTA. Disponível em: http://www.linux.ime.usp.br/~cef/mac499-04/monografias/ivanneto/apresentacao.ppt#280,8,Tipos de Peers. Acesso em: 06/12/2006.

[Jer 99] BANKS, J. Introduction to simulation. In: WINTER SIMULATION CONFERENCE. 1999. Proceedings… Local: Instituição , 1999.

[Jxt 06a] JXTA: JXTA virtual network. Disponível em: http://www.soi.wide.ad.jp/class/ 200 40003/slides/02/69.html. Acesso em: 07/12/2006.

[Jxt 06b] JXTA – Architecture. Disponível em: http://wiki.cs.uiuc.edu/cs427/JXTA+-+Architecture. Acesso em: 03/12/2006.

[Kuh 00] KUHL, F; WEATHERLY, R.; DAHMANN, J. Creating Computer Simulation Systems An Introduction to the High Level Architecture. The MITRE Corporation, Prentice Hall PTR, 2000. 212 p.

[Laf 03] LAFFRANCHI, M. M., Uma solução peer-to-peer para a implementação de jogos multiusuário baseado no padrão emergente MPEG-4 UM, dissertação de mestrado da UFSCar, Departamento de Computação, 2003.

[Lim 05] LIMA, A. V. F. P2P, LBS e Comunidades Virtuais: os ingredientes para aplicações inovadoras em sistemas 3G. Pernambuco: UFPE/Centro de Informática, 2005. Trabalho de graduação.

[Mac 95a] MACEDONIA, M. A Network Software Architecture for Large Scale Virtual Environments. Naval Postgraduate School, June 1995. Tese de doutorado

[Mac 95b] MACEDONIA, M.; BRUTZMAN D.; ZYDA, M. et. all. NPSNET: A Multi-Player 3D Virtual Environment over the Internet. Proceedings of the ACM 1995 Symposium on Interactive 3D Graphics, 9-12 April 1995.

[Mil 95] MILLER, D. C.; THORPE, J. A. SIMNET:the advent of simulator networking. Proceedings of the IEEE, v. 8, p. 1114-1123, Aug. 1995.

[Mor 04] MORSE, K. L. et.al. An Architecture for web services Based Interest Management in Real Time Distributed Simulation, Distributed Simulation and Real-Time Applications, 2004, pp. 108-115.

[Mor 97] MORSE, K.L.; STEINMAN, J.S. Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering, Proceedings of SIW, Spring 1997.

[Omg 95] Object Management Group - OMG. The Common Object Request Broker: Architecture and Specification, Revision 2.0, Jul. 1995.

[Omg 97] Object Management Group - OMG 97. Disponível em: http://www.omg.org/. Acesso em: 08/04/2007.

[Orf 96] ORFALI, R.; HANKEY, D.; EDWARDS, J. The Essential Distributeds Objects — Survival Guide, John Wiley & Sons, Inc. 1996.

[Pee 06] PEER-TO-PEER Networking (P2P). Disponível em: http://www.daimi.au.dk/~mariu s/p2p-course/lectures/01/talk.html. Acesso em: 08/12/2006.

[Pjt 06] PROJECT JXTA. Disponível em: http://www.jxta.org/LinuxWorld/JXTA_LinuxW orld02.pdf. Acesso em: 05/12/2006.

[Pop 89] POPE, A. R. The SIMNET network and protocols. Cambridge: BBN Systems and Technologies, 1989. Technical Report 7102.

[Pro 06] PROJECT JXTA: An Open, Innovative Collaboration. Disponível em: http://www.j xta.org. Acesso em: 05/12/2006.

[Pro 06a] PROJECT JXTA. Disponível em: http://www.srdc.metu.edu.tr/webpage/seminars/p 2p/Project%20JXTA.ppt#286,50, Network Organization. Acesso em: 05/12/2006.

[Pul 04] PULLEN, J. MARK, et.al., Using web Services to integrate heterogeneous simulations in a grid environment, Proceedings of the International Conference on. Computational Science, 2004.

[Rac 05] RACZY, C.; TAN, G.; YU, J. A Sort-based DDM Matching Algorithm for HLA, Transactions on Modeling and Simulation, Vol. 15, No. 1, January 2005, Pages 14–38, ACM.

[Rei 00] REID, M., An Evaluation of the High Level Architecture (HLA) as a Framework for NASA Modeling and Simulation. In Proceedings of the 25th NASA Software Engineering Workshop, Goddard Space Flight Center, Maryland, November 2000.

[Ric 03] RICHARD M., Parallel and distributed simulation systems, proceedings of the 35th conference on Winter simulation: driving innovation New Orleans, Louisiana, SESSION: Advanced tutorials, pages: 124 - 134, 2003.

[Ryc 05] RYCERZ, K.; BUBAK, M.T.; MALAWSKI, M.; P.M.A. Sloot: A Framework for HLA-Based Interactive Simulations on the Grid, Simulation: Transactions of The Society for Modeling and Simulation International, (Special Issue on Applications of Parallel and Distributed Simulation) vol. 81, nr 1 pp. 67-76. April 2005. Special Issue.

[Ser 06] SERVIÇOS web – uma breve introdução. Disponível em: http://www.nce.ufrj .br/conceito/artigos/2005/01p1-1.htm. Acesso em: 24/032006.

[Ser 06a] SERVIÇOS Web – Uma breve introdução (Parte I). Disponível em: http://www.nce. ufrj.br/conceito/artigos/2005/01p1-1.htm. Acesso em: 05/05/2006.

[Sim 06a] SIMULAÇÃO de Processos. Disponível em: http://www.agais.com/simula.htm#T5. Acesso em: 22/08/2006.

[Sim 06b] SIMULAÇÃO a eventos discretos. Disponível em: http://www.engprod.ufjf.br/fernando/epd042/Simulacao.pdf. Acesso em: 25/08/2006.

[Sim 06c] SIMPLE JXTA. Disponível em: http://dit.unitn.it/~ilya/Download/Publications/J XTA_05.07.02.ppt#280,24,JXTA Network Organization - 2. Acesso em: 05/12/2006.

[Sis 06] SISTEMAS e modelos. Disponível em: http://paginas.fe.up.pt/~feliz/newSim_cap1_PI.pdf. Acesso em: 20/08/2006.

[Soa 06] SOAP. Disponível em: http://pt.wikipedia.org/wiki/SOAP. Acesso em: 15/05/2006.

[Tan 00a] 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.

[Tan 00b] 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.

[Tan 05] TAN, G; PERSSON, A.; AYANI, R, HLA Federate Migration, in proceedings of 38th IEEE Annual Simulation Symposium, San Diego, California, USA, April 2005.

[Tay 05] TAYLOR; et al; Integrating Heterogeneous Distributed COTS Discrete-Event Simulation Packages: An Emerging Standards-Based Approach, IEEE Transactions on Systems, Man and Cybernetics. 2005

[Usi 06] USING WSDL in SOAP applications. Disponível em: http://www-128.ibm.com/developerworks/webservices/library/ws-soap/index.html?dwzone=ws. Acesso em: 13/05/2006.

[Val 03] VALENTE, M. T. O. ; SANTOS, J. P. ; COUTO, C. F. M. . Interceptação de Métodos Remotos em Java RMI. In: VII Simpósio Brasileiro de Linguagens de Programação, 2003, Ouro Preto, 2003. p. 50-63.

[Vas 01] COSTA, V.; BAPTISTA, H.; PEREIRA, J. Jogos de Guerra e Paz, 10º Encontro Português de Computação Gráfica, ISCTE, Lisboa, Outubro de 2001.

[Web 06] SERVIÇOS web, SOAP e Aplicações Web. Disponível em: http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html. Acesso em: 23/03/2006.

[Web 06a] SERVIÇOS web, SOAP e Aplicações Web. Disponível em: http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html. Acesso em: 07/05/2006.

[Wyt 03] WYTZISK, A.; SIMONIS, I.; KLEIN, U. Integration of HLA Simulation Models into a Standardized Web Service World. Proceedings of: Euro-SIW, Stockholm, Sweden. 2003.

Kat 04 Rycerz, K.; et al; Monitoring of HLA Grid Application Federates with OCM-G. DS-RT, 125-132, 2004.

Apêndice A

O código a seguir é uma rotina que faz o gerenciamento da distribuição de dados (DDM) para verificar quais células envolvem uma região de interesse. Este algoritmo foi implementado no visualizador/controlador deste trabalho de mestrado, observe os comentários para entender como é feito o processamento.

```
Public Sub DDM(ByVal x, ByVal y, ByVal x1, ByVal x2, ByVal y1, ByVal y2)
        ax = x
        ay = y
        ax1 = x1
        ax2 = x2
        ay1 = y1
        ay2 = y2
        k = 1
        i = 1
        j = 1
        cont = 0
        a = 0
        'esta rotina encontra as quatro celulas das quinas que envolvem a
area de interesse, com a celula 50x50
        While k <= 4

            Select Case k

                Case 1
                    i = 50
                    j = 50
                    While j <= y
                        While i <= x
                            cont = cont + 1
                            If ((i - 50) <= x1 And x1 <= (i)) And ((j - 50)
<= y1 And y1 <= (j)) Then
                                v(a) = cont
                                a = a + 1
                                i = x + 50
                                j = y + 50
                                'End
                            End If
                            i = i + 50
                        End While
                        j = j + 50
                        i = 50
                    End While

                Case 2
                    i = 50
                    j = 50
                    cont = 0
                    While j <= y
                        While i <= x
```

```
                            cont = cont + 1
                            If ((i - 50) <= x2 And x2 <= (i)) And ((j - 50)
<= y1 And y1 <= (j)) Then
                                v(a) = cont
                                a = a + 1
                                i = x + 50
                                j = y + 50
                            End If
                            i = i + 50
                        End While
                        j = j + 50
                        i = 50
                    End While

                Case 3
                    i = 50
                    j = 50
                    cont = 0
                    While j <= y
                        While i <= x
                            cont = cont + 1
                            If ((i - 50) <= x1 And x1 <= (i)) And ((j - 50)
<= y2 And y2 <= (j)) Then
                                v(a) = cont
                                a = a + 1
                                i = x + 50
                                j = y + 50
                            End If
                            i = i + 50
                        End While
                        j = j + 50
                        i = 50
                    End While
                Case 4
                    i = 50
                    j = 50
                    cont = 0
                    While j <= y
                        While i <= x
                            cont = cont + 1
                            If ((i - 50) <= x2 And x2 <= (i)) And ((j - 50)
<= y2 And y2 <= (j)) Then
                                v(a) = cont
                                a = a + 1
                                i = x + 50
                                j = y + 50
                            End If
                            i = i + 50
                        End While
                        j = j + 50
                        i = 50
                    End While
            End Select
            k = k + 1
        End While


        ' essa parte encontra as outras celulas que envolvem a area de
interesse mas que nao faz
        'parte das quatro celulas das quinas
```

```
        c1 = v(0) 'c1, c2, c3, c4 celulas das quinas (da esquerda pra
direita, de baixo pra cima)
        c2 = v(1)
        c3 = v(2)
        c4 = v(3)
        i = 0
        l = 0

        'para areas colunas e quadradas ou retangulares

        If c1 <> c3 Then
            If c1 <> c2 Then
                While c1 <= c3
                    While c1 <= c2
                        avc(i) = c1 'vetor que guarda todas as celulas da
area de interesse
                        c1 = c1 + 1
                        i = i + 1
                    End While
                    l = l + 1
                    c1 = v(0) + ((x / 50) * l)
                    c2 = c2 + (x / 50)
                End While
            End If
        Else
            While c1 <= c3
                avc(i) = v(0)
                i = i + 1
                l = l + 1
                c1 = v(0) + (x / 50 * l)
            End While
        End If

        ' parte para area linha

        c1 = v(0)
        c2 = v(1)
        c3 = v(2)
        c4 = v(3)
        i = 0
        l = 0

        If (c1 <> c2) And (c1 = c3) Then
            While c1 <= c2
                avc(i) = c1
                c1 = c1 + 1
                i = i + 1
            End While
        End If

        ' parte para area com uma celula

        c1 = v(0)
        c2 = v(1)
        c3 = v(2)
        c4 = v(3)
        i = 0
        l = 0

        If (c1 = c2 = c3 = c4) Then
```

```
            avc(i) = v(0)
        End If

        txtBoxCelulas.Text = ""
        t = 0

        'while que mostra o conjunto de celulas em um textBOx
        While (t <= 100 And avc(t) <> 0)
            txtBoxCelulas.Text = txtBoxCelulas.Text + "_" +
(avc(t).ToString)
            t = t + 1
        End While

    End Sub
```

Apêndice B

Com as células encontradas no algoritmo DDM do apêndice A, toda informação de conjunto de células dos outros federados será analisada por esse procedimento para verificar se existe intersecção entre as áreas de interesse, se houver o protocolo de informação de peer (PIP) será executado para pedir as coordenadas exatas do federado que contém ao menos uma célula igual ao federado local.

```
'loop que verifica se existe intereseccao com as celulas recebidas
                    While (frmAreaDeDistribuicao.avc(i) <> 0)
                        While (VetCelRec(j) <> 0) ‘celulas recebidas
                            If frmAreaDeDistribuicao.avc(i) = VetCelRec(j)
Then

                                'pede as coordenadas
                                frmAreaDeDistribuicao.PedeCoordenadas()
                                i = 99 'faz sair dos dois whiles
                                j = 99
                                MessageBox.Show("Percebe intersecção e cria
grupo multicast")
                            End If
                            j = j + 1
                        End While
                        i = i + 1
                        j = 0
                    End While
```

Assim o federado remoto recebe uma mensagem pedindo as coordenadas e envia para federado local

```
'verifica se está pedindo as coordenadas para envia-las
            If StrRec = "pedindo coordenadas/" Then ‘string recebida
                            frmAreaDeDistribuicao.EnviaCoordenadas()
```