Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

"*Hybrid Ad Hoc Grid*: Uma Grade Computacional Entre-Pares Auto-Organizável"

Rodrigo Lopes da Silva

Campina Grande, Paraíba, Brasil
Agosto, 2011

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

"*Hybrid Ad Hoc Grid*: Uma Grade Computacional Entre-Pares Auto-Organizável"

Rodrigo Lopes da Silva

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande – Campus I como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Marco Aurélio Spohn
Orientador

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S586h Silva, Rodrigo Lopes da.

Hybrid ad hoc grid: uma grade computacional entre-pares auto-organizável / Rodrigo Lopes da Silva. — Campina Grande, 2011.

47 f. : il. col.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Referências.

Orientador: Prof. Ph.D. Marco Aurélio Spohn.

1. Rede de Computadores. 2. Grades Computacionais. 3. *ad hoc*. 4. Sistemas Distribuídos. 5. Auto-organização. I. Título.

CDU – 004.7 (043)

"HYBRID AD HOC GRID: UMA GRADE COMPUTACIONAL ENTRE-PARES AUTO-ORGANIZÁVEL"

RODRIGO LOPES DA SILVA

DISSERTAÇÃO APROVADA EM 26.08.2011

MARCO AURELIO SPOHN, Ph.D
Orientador(a)

REINALDO CÉZAR DE MORAIS GOMES, Dr.
Examinador(a)

FABIANO SALVADORI, D.Sc
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Grades computacionais consistem em sistemas que permitem o compartilhamento de uma quantidade de recursos computacionais sem precedentes, possibilitam grande capacidade de expansão para aplicações paralelas e com um custo bastante reduzido em relação aos supercomputadores paralelos. Nos últimos anos, novas abordagens de grades computacionais têm sido exploradas, entre elas as grades esporádicas ou *ad hoc* que apresentam como características poucas atividades administrativas e rápida adaptação às mudanças na comunidade. O *Hybrid Ad Hoc Grid* é uma grade computacional auto-organizável que apresenta instalação simplificada, atividades administrativas reduzidas e recuperação a falhas. Essa grade foi desenvolvida com o objetivo de resolver problemas de implantação e administração de grades, reduzir custos com sua instalação e possibilitar a ampliação do uso de grades computacionais.

Abstract

Grid computing consists of systems that allow share a number of unprecedented computational resources, provide scalability for large parallel applications and with a reduced cost compared to parallel supercomputers. In recent years, new approaches to grid computing have been explored, including the sporadic or ad hoc grids that have few administrative activities and quickly adapt to changes in the community. The Hybrid Ad Hoc Grid is a self-organizing computational grid which has simplified installation, reduced administrative activities and fault recovery. This grid was developed to solve problems of deployment and management of grids, reduce installation costs and enable increased use of grid computing.

Agradecimentos

Primeiramente, agradeço a Deus por tudo em minha vida. Depois, agradecimento ao meu pai e minha mãe que me ajudaram e apoiaram durante, sem os quais não teria conseguido alcançar tudo que alcancei ao longo de minha vida.

Agradeço também as minhas irmãs Lúcia e Cidinha que sempre me apoiaram. Em especial, agradeço a minha irmã Marília, minha melhor amiga, que sempre paciente me ajudou ao longo de todo o mestrado. Agradeço também ao meu cunhado Ricardo pelo seu apoio.

Um agradecimento especial ao Marco Spohn, meu orientador por ter me orientado, acompanhado e acreditado em mim.

Agradeço também ao professor Fubica pelas contribuições e pelo disponibilizar o LSD. Aprendi muito nesse ambiente e obtive condições para prosseguir com esse trabalho.

Agradeço aos meus amigos Thiago, Alan e Athayde que me auxiliaram durante essa jornada.

Por último, agradeço a todos que de alguma forma contribuíram para a conclusão desse trabalho.

Conteúdo

Lista de Abreviaturas e Siglas

API - Application Programming Interface

ARPANET - Advanced Research Projects Agency Network

BoT - Bag-of-Tasks

BSD - Berkeley Software Distribution

GDF - Grid Description File

GPL - General Public License

IP - Internet Protocol

JDBC - Java Database Connectivity

LDAP - Lightweight Directory Access Protocol

MBPS - Megabit por segundo

NOF - Network of Favors

ODBC - Open Database Connectivity

QoS - Quality of Service

RMI - Remote Method Invocation

RUNES - Reconfigurable Ubiquitous Networked Embedded Systems

SDF - Site Description File

VCard - Versitcard

XDB - XML Database

XML - Extensible Markup Language

XMPP - Extensible Messaging and Presence Protocol

Lista de Ilustrações

Listas de Quadros

1 Introdução

Nos anos oitenta e noventa do século passado, a computação em larga escala começou a ser utilizada para resolver problemas. Nesse período, as aplicações necessitavam de recursos computacionais que alcançavam os limites dos mais rápidos computadores paralelos existentes. Com isso, iniciou-se a exploração da ideia de compartilhamento de recursos dos supercomputadores, resultando nas primeiras grades computacionais.

Grades computacionais consistem em uma infraestrutura que permite o compartilhamento de recursos computacionais de forma sem precedente, apresentando uma capacidade de expansão para aplicações paralelas, com um custo reduzido em relação aos supercomputadores paralelos.

Nos últimos anos, várias pesquisas têm sido realizadas sobre grades computacionais, sobre diferentes abordagens, focando em pervasividade e auto-organização. De acordo com Cirne *et al.* (2003), as grades nas abordagens tradicionais são criadas e gerenciadas pelos proprietários dos recursos compartilhados. Dessa forma, os benefícios que as aplicações podem obter das grades computacionais são limitados. Com o objetivo de superar essas limitações, surgiram as grades *ad hoc* que, segundo Kurdi *et al.* (2008), consistem em uma formação espontânea de computadores heterogêneos, em uma comunidade sem uma estrutura fixa e pré-definida. A principal motivação para essa abordagem é simplificar a instalação, manutenção e utilização de grades computacionais.

Nesse trabalho apresentamos uma grade computacional *ad hoc*, o *Hybrid Ad Hoc Grid*, desenvolvida a partir da grade *OurGrid*[1] que apresenta como características: auto-organização, recuperação de falhas e instalação simplificada. Essa grade é híbrida, pois apesar de consistir em uma grade *ad hoc*, apresenta um componente com localização fixa e predefinida. Assim, espera-se que o *Hybrid Ad Hoc Grid* possa ampliar a utilização de grades computacionais, permitindo a criação de grades computacionais com um extenso número de membros.

1.1 Motivação

Existem três fatos que motivaram a pesquisa e o desenvolvimento do *Hybrid Ad Hoc Grid*. O primeiro está relacionado aos diferentes tipos de comunidades em grades. As grades computacionais tradicionais apresentam como objetivo unir o poder computacional de

[1] http://www.ourgrid.org

grandes empresas ou instituições. Normalmente, essas comunidades apresentam uma infraestrutura fixa e com máquinas dedicadas. Porém, existem comunidades cujos membros mudam constantemente, tanto do ponto de vista físico quanto do ponto de vista lógico. Como exemplo de mudança física, temos o número de máquinas na grade. Como exemplo de mudança lógica, temos as políticas de segurança. As grades que atendem aos requisitos dessa última comunidade são geralmente classificadas como grades esporádicas ou *ad hoc*.

O segundo fato está relacionado à instalação e a manutenção. Foram realizadas pesquisas em 2003 e 2005 (Duarte *et al.*, 2006) com usuários de grades computacionais espalhados pelo mundo. O resultado dessas pesquisas apresentou, como principal causa de falhas, erros na configuração dos ambientes das grades. Assim, simplificar e reduzir as configurações da grade deve resultar na redução dessas falhas facilitando a sua implantação.

Por último, temos a utilização de grades. Existem diversos tipos de aplicações que podem obter proveito dos benefícios das grades computacionais, entre essas as *Bag-of-Tasks* (BoT). Segundo Cirne *et al.* (2003), essas são aplicações cujas tarefas são independentes entre si e que são utilizadas em diversos cenários como, por exemplo, mineração de dados e simulações. Apesar do potencial das grades computacionais para a resolução de problemas, existem poucos usuários de aplicação BoT que utilizam grades devido a vários fatores, entre eles a criação da grade e acesso a novos recursos. Para esse cenário, grades auto-organizáveis possibilitam a criação simplificada de grades com muitos membros cujos novos recursos serão disponibilizados de forma automatizada.

1.2 Objetivos

O objetivo dessa dissertação é o desenvolvimento de uma grade computacional auto-organizável entre pares. Essa grade deverá atender a comunidades esporádicas e apresentar instalação e acesso aos recursos simplificados. Com isso, espera-se possibilitar a ampliação do uso de grades computacionais.

1.3 Estrutura da Dissertação

No capítulo dois, será feita uma revisão da fundamentação teórica sobre grades computacionais e será apresentada uma introdução sobre a grade *OurGrid*, a partir da qual, será desenvolvida a grade objeto dessa dissertação. No capítulo três, são introduzidos

trabalhos relacionados. Enquanto que no capítulo quatro é abordado o *Hybrid Ad Hoc Grid*, sua estrutura e seu funcionamento. No capítulo cinco, é demonstrada a avaliação realizada na grade desenvolvida. Por último, no capítulo seis serão apresentadas as conclusões dessa dissertação e as possibilidades para trabalhos futuros.

2 Grades Computacionais

Uma grade computacional consiste em um complexo sistema de *software* e serviços que permitem o compartilhamento de recursos computacionais distribuídos, de forma eficiente e estável, através de uma interface amigável. Existem várias grades clássicas citadas na literatura, entre essas temos *Globus*[2], *Condor*[3] e *Mars*(*Gehrinf* e *ReinFeld*, 1996).

Conforme Minoli (2005), o conceito de grade computacional pode ser compreendido como uma forma transparente de integrar, racionalizar e compartilhar recursos em um único sistema. Esses recursos podem ser estações de trabalho, servidores, sistemas de armazenamento, dados e a rede de computadores. Esse sistema é criado de forma a agregar eficiência e desempenho de processamento às aplicações.

Segundo Berman *et al.* (2003), nos anos 80 do século passado a computação em larga-escala começou a ser utilizada como ferramenta para resolver problemas de grande escala, em busca de novas descobertas científicas. Nesse período, com a necessidade de computadores mais velozes ocorreram extensas pesquisas sobre paralelismo. A multidisciplinaridade dos problemas e a distribuição geográfica dos colaborados forneceram aos pesquisadores experiência sobre coordenação e distribuição. Esses conceitos seriam utilizados posteriormente para o desenvolvimento das grades.

A ideia de grades computacionais foi desenvolvida nas últimas décadas, porém suas raízes vêem de muito antes, ainda no desenvolvimento da *Advanced Research Projects Agency Network* (ARPANET). Nessa época, as redes computacionais eram vistas por J.C.R. Lickerlider como um sistema rápido e automatizado de apoio à tomada de decisão humana. Devido ao desenvolvimento das redes de computadores, hoje as grades computacionais se tornaram globais, interligando desde laboratórios e *data center*, até países e continentes.

Berman *et al.* (2003) afirmam que as grades têm convergido para um modelo em camada, possibilitando a integração de serviços e software que compõem os recursos das grades. Esse modelo de camadas pode ser observado na FIGURA 1.

[2] http://www.globus.org
[3] http://www.cs.wisc.edu/condor

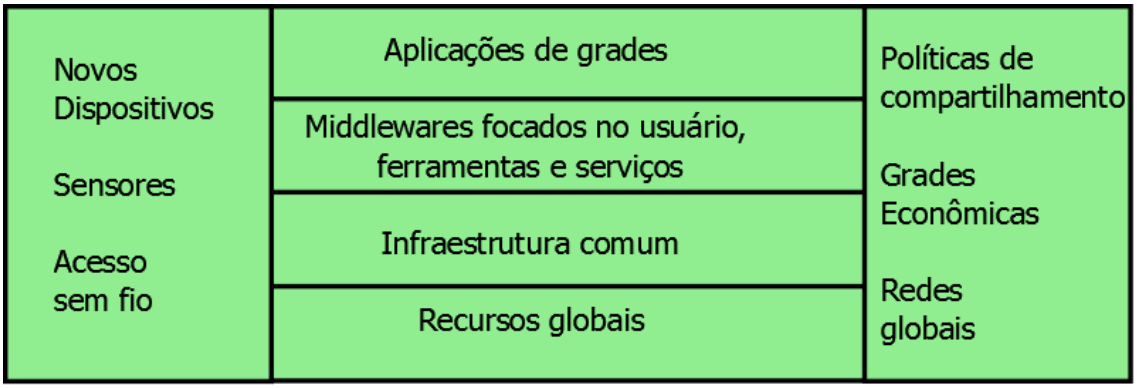

FIGURA 1: Arquitetura em camadas de um modelo de grades

A camada Recursos globais é constituída pelo *hardware* da grade, ou seja, computadores, rede, dispositivos virtualizados, etc. É uma camada dinâmica, tendo em vista as constantes substituições de componentes antigos. Além disso, apresenta dispositivos distribuídos e com diferentes desempenhos.

A camada Infraestrutura comum consiste em serviços e sistemas que virtualizam a grade. Berman *et al.* (2003) acreditam que o surgimento de um *software* como padrão possibilitará a criação de uma plataforma virtual unificada. Essa plataforma possibilitaria que o desenvolvimento de *software* e aplicações para grade receba maior atenção.

A camada *Middlewares* focados no usuário, ferramentas e serviços compreende portais, escalonamento de aplicações e afins. A função dos componentes dessa camada é realizar atividades complexas, como autenticação e transferência de arquivo, para possibilitar que as aplicações utilizem os recursos da grade de forma mais produtiva.

A camada Aplicações de grades corresponde às aplicações e usuários. Conforme Berman *et al.* (2003), o sucesso das grades depende dos usuários da comunidade e das camadas horizontais assegurarem uma plataforma robusta e estável para o compartilhamento de recursos.

A camada vertical compreende os desafios dos próximos anos: integração com novos dispositivos, como PDA e sensores, e considerações sobre políticas de compartilhamento e uso de recursos. Com a globalização das grades computacionais, essas políticas necessitam de maior atenção.

2.1 Redes de Computadores

As redes de computadores são componentes fundamentais em grades, pois o seu desempenho determina o sucesso no uso de recursos distribuídos. Conforme Berman *et al.* (2003), existem três características de redes importantes para as grades computacionais: largura de banda, latência e qualidade de serviço (QoS).

Nos últimos anos, foram realizadas melhorias na largura de banda, resultando no surgimento de redes de alto desempenho como a Internet2 *Abilene Network*[4]. Porém, a latência e a qualidade de serviço não apresentaram melhorias tão significativas quanto à largura de banda. As redes sem fio também têm apresentado melhorias contínuas, possibilitando que dispositivos móveis possam obter proveito dos recursos das grades.

2.2 Grades Emergentes

Novas abordagens de grades têm explorado características como auto-organização e pervasividade, sendo denominadas por Kurdi *et al.* (2008) como grades emergentes. A pervasividade é composta por quatro características: acessibilidade, interatividade, centrada ao usuário e gerenciamento. Nas próximas seções, são detalhadas as quatro categorias de grades emergentes, de acordo com a classificação de Kurdi *et al.* (2008).

2.2.1 Acessibilidade

As grades focadas em acessibilidade devem possibilitar o acesso aos recursos independente da localização geográfica e do acesso físico à grade. Esse tipo de grade é constituído por membros fixos ou móveis, podendo apresentar estrutura pré-definida ou *ad hoc*. A principal característica desse tipo de grade é a sua estrutura dinâmica originada pelas constantes mudanças dos membros. Existem três subcategorias de grades de acessibilidade:

- Grades *ad hoc (*FIGURA 02*)*: ao contrário das grades tradicionais, não apresentam estrutura pré-definida, com novos usuários ingressando na grade ao localizar um dos membros. Outra característica dessas grades consiste em apresentar reduzidas atividades administrativas. De acordo com Amin, Laszewski e Mikler (2004), grades *ad hoc* suportam modalidades de utilização esporádicas e *ad hoc*, e apresentam

[4] http://noc.net.internet2.edu

independência da arquitetura, tecnologia e controle. Exemplos dessa subcategoria de grade são o *OurGrid* e o *MyGrid*[5].

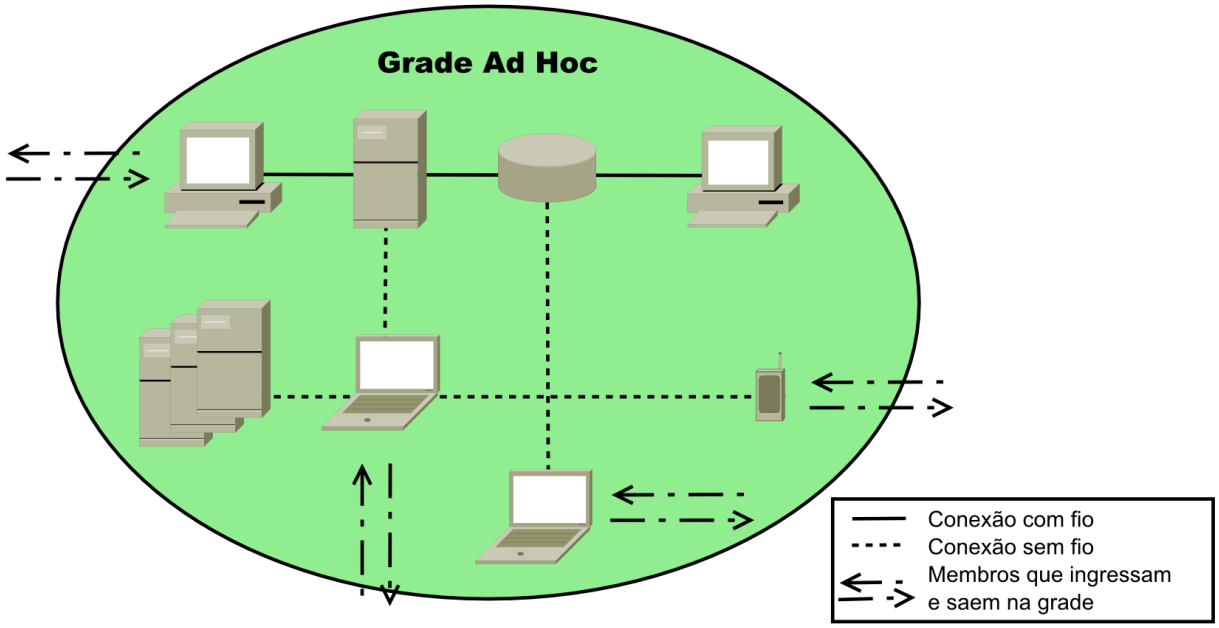

FIGURA 2: Modelo de grade *ad hoc*

- Grades sem fio: são constituídas por diferentes tipos de dispositivos como sensores, telefones celulares e *notebooks*, apresentando alguns de seus membros conectados sem fio. As grades sem fio podem ser divididas em duas subcategorias:
    - Grades sem fio (FIGURA 03): os membros sem fio utilizam os recursos dos outros membros e realizam processamento e armazenamento de dados;
    - Grades sem fio de acesso (FIGURA 04): os membros com conexão sem fio utilizam os recursos dos outros membros, porém não realizam processamento ou armazenamento de dados.

[5] http://www.mygrid.org.uk

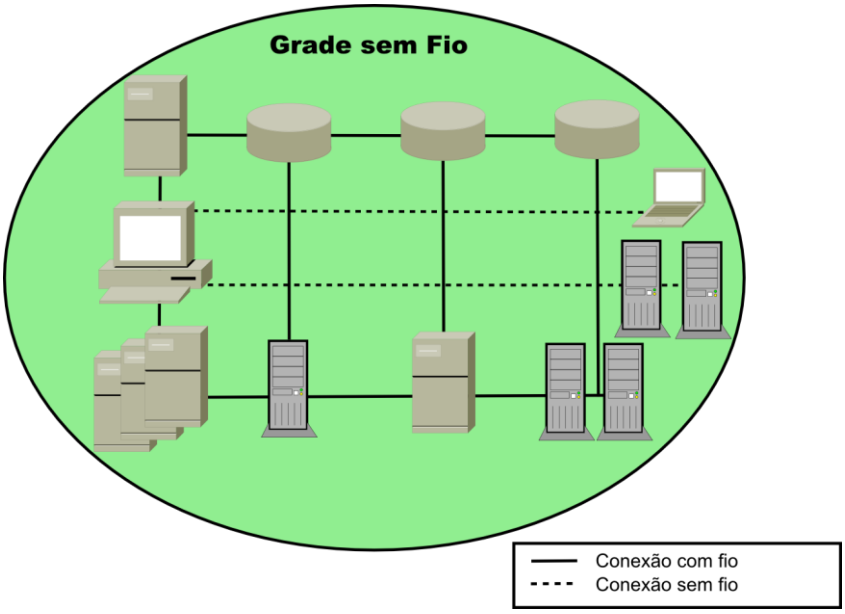

FIGURA 3: Modelo de grade sem fio

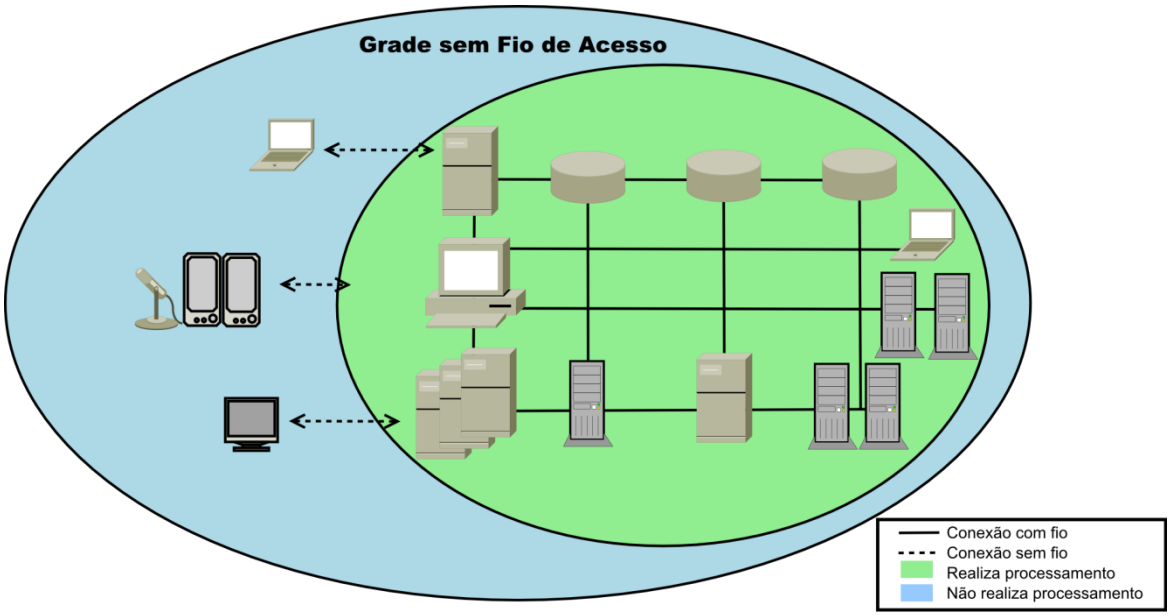

FIGURA 4: Modelo de grade sem fio de acesso

- Grades móveis: diferentemente das grades *ad hoc* cuja característica principal é a natureza esporádica dos membros, as grades móveis são caracterizadas pela mobilidade dos membros. Como exemplo dessa categoria tem o *Akogrimo*[6]. Existem duas subcategorias de grade móvel:
    - Grade móvel (FIGURA 5): categoria formada por membros móveis que acessam os recursos da grade e um membro móvel que executa o processamento e armazenamento de dados;

[6] http://www.mobilegrids.org

- Grade móvel de acesso (FIGURA 6): categoria formada por membros móveis que apenas acessam os recursos de uma grade fixa.

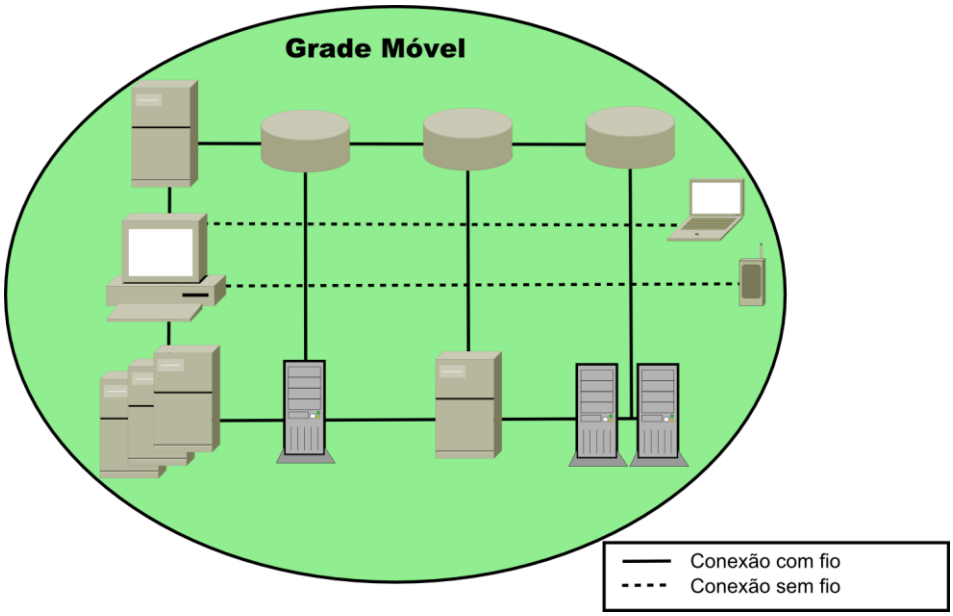

FIGURA 5: Modelo de grade móvel

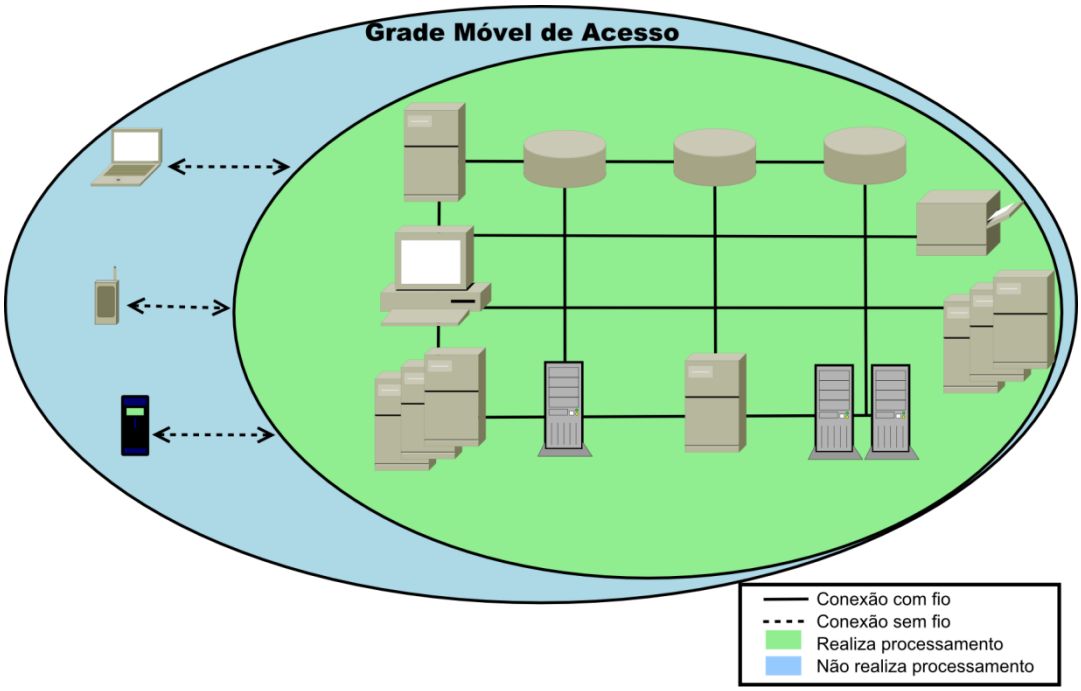

FIGURA 6: Modelo de grade móvel de acesso

2.2.2 Interatividade

As grades tradicionais utilizam o modelo cliente/servidor de comunicação. Diferentemente dessas, as grades de interatividade apresentam um modelo de comunicação que permite a interação em tempo real. Normalmente, essa categoria apresenta uma arquitetura em duas camadas: uma camada formada por um portal *web* que permite ao usuário submeter aplicações e uma camada com um *middleware* que executa as aplicações.

2.2.3 Centradas nos Usuários

As grades tradicionais são desenvolvidas para atender usuários de áreas de pesquisa. Como normalmente outros tipos de usuários enfrentam dificuldade na sua utilização, foram desenvolvidas as grades centradas nos usuários para atender a essa demanda. Essa última é caracterizada por apresentar personalização e adaptação às necessidades dos usuários. Como exemplo dessas grades, podemos citar *Akogrimo* e *MyGrid*.

2.2.4 Gerenciamento

Gerenciar uma grade é uma atividade complexa que requer grande conhecimento do sistema. As grades tradicionais apresentam um esquema de gerenciamento central, ao contrário das grades distribuídas, como *peer-to-peer* (P2P), que apresentam gerenciamento distribuído.

As grades de gerenciamento apresentam a capacidade de gerenciar o uso de seus recursos, organizar e recuperar as falhas de forma automatizada. Devido a essas características, essas grades apresentam instalação, configuração e administração simplificada. Isso resulta em redução de custos com a administração e aumento da escalabilidade. Existem quatro abordagens de grades nessa área:

- Grades autônomas consistem em sistemas de computação autônoma capaz de controlar a grade sem a intervenção do usuário. Alguns exemplos dessa subcategoria são IBM *OptimalGrid*[7] e *AutoMAGI* (Sajjad *et al.*, 2005);
- Grades de conhecimento registram metadados semânticos nos dados, recursos e serviços de forma a serem compreendidos por máquinas e seres humanos. O objetivo

[7] http://www.mobilegrids.org

desses metadados é possibilitar a criação de uma infraestrutura de gestão do conhecimento generalizado. Pode ser citado como exemplo o *InteliGrid*[8];

- Grades orgânicas consistem em um design que conta com uma abordagem *peer-to-peer* descentralizada, um esquema de escalonamento de processos distribuídos e agentes móveis. Um exemplo de grade orgânica é a *Organic Grid* (Chakravarti, Baumgartner e Lauria, 2005) que se utiliza da ideia de colônias de formigas para o seu modelo de autogerenciamento;
- Por último, existem abordagens híbridas que combinam mais de uma das abordagens citadas anteriormente.

2.3 OurGrid

Conforme Cirne *et al.* (2006), o *OurGrid* é uma grade computacional aberta, que se encontra em produção desde 2004. O mesmo é baseado em uma rede de compartilhamento de recursos entre pares (*peer-to-peer*) no qual os laboratórios doam seus recursos excedentes, obtendo em troca os recursos excedentes doados de outros laboratórios. Com isso, o objetivo do *OurGrid* é melhorar a capacidade computacional dos laboratórios, podendo tanto ser utilizado em *desktops* interativos, quanto em *clusters* dedicados. Além disso, uma de suas características é priorizar o uso local dos recursos, de forma a assegurar que o desempenho local não seja prejudicado.

O foco dessa grade são as aplicações *Bag-of-Task* (BoT). Segundo Cirne *et al.* (2003), BoT consistem em aplicações paralelas cujas tarefas são independentes umas das outras, portanto podendo potencialmente se beneficiar do uso de um grande número de processadores.

Uma grade típica criada com o *OurGrid* é apresentada na FIGURA 7. Nesse modelo podemos visualizar a estrutura básica da grade.

[8] http://www.inteligrid.com

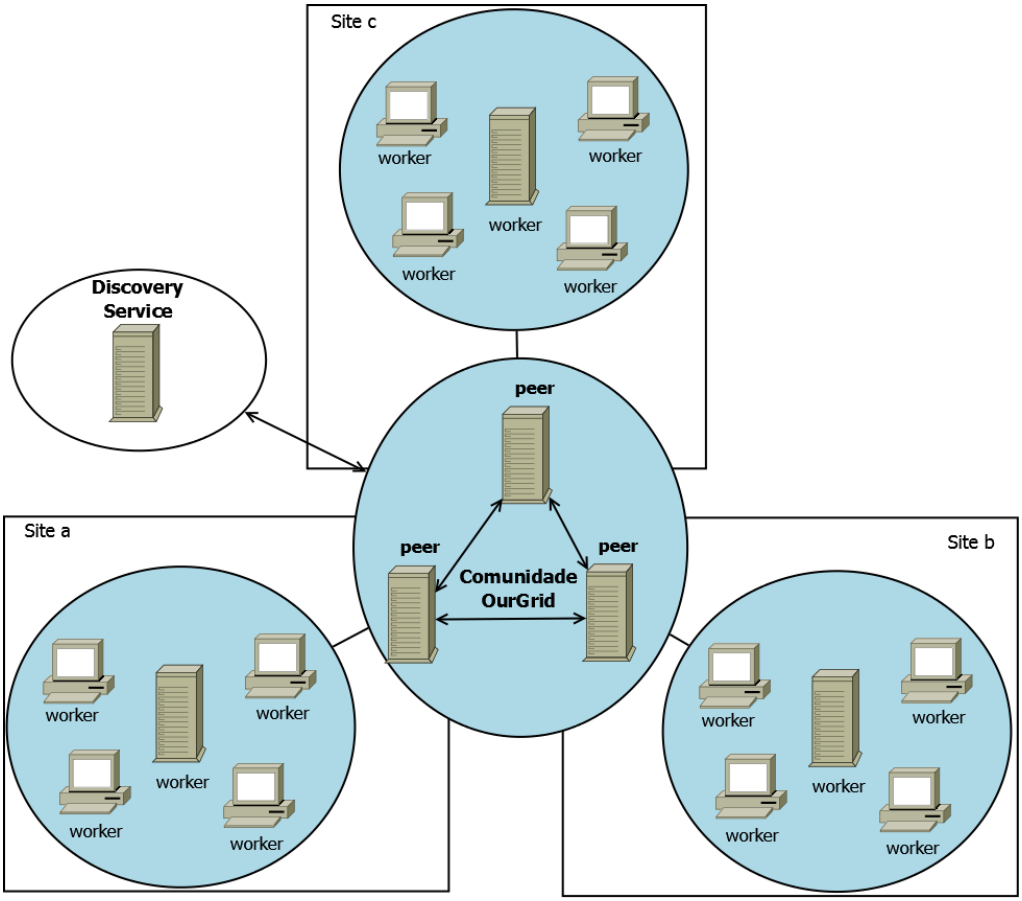

FIGURA 7: Modelo de grade *OurGrid*

Conforme a figura, uma grade típica é constituída por *peers* que se comunicam e compartilham recursos diretamente. Cada conjunto de recursos conectados a um mesmo *peer* constitui um *site*. Ao ingressar em uma grade, um *peer* localiza os demais *peers* através do *discovery service*.

O OurGrid utiliza dois arquivos de configuração: SDF (*Site Description File*) e GDF (*Grid Description File*). O primeiro arquivo define os *workers* no *peer*, sempre que um *worker* não presente no arquivo for inicializado, esse arquivo deve ser alterado e enviado ao *peer*. O segundo arquivo define a localização do *peer* para o *broker*, dessa forma, sempre que a localização do *peer* em um *site* for alterada, cada usuário deverá editar o arquivo antes de iniciar o *broker*. Além disso, em cada *site*, novos usuários devem ser adicionados no *peer* para que possam enviar suas aplicações. A adição de novos usuários é realizada manualmente pelo administrador do *peer*.

2.3.1 Rede de Favores

Segundo Andrade *et al.* (2007), em grades *peer-to-peer* os participantes não necessariamente se conhecem, portanto, não é possível garantir que eles realmente compartilharão seus recursos. Dessa forma, alguns participantes conhecidos como *free rider* podem utilizar os recursos cedidos por outros participantes, sem compartilhar seus próprios recursos. Para evitar que isso ocorra é necessário um mecanismo de incentivo à colaboração.

O *OurGrid* possui um mecanismo de incentivo denominado rede de favores (*Network of Favors*). Esse mecanismo é baseado em um histórico local de recursos recebidos de outros *peers.* Nesse mecanismo, quando mais de um *peer* solicitar um mesmo recurso, será priorizado o *peer* que mais contribuiu para a grade.

Conforme Andrade *et al.* (2007), mecanismos de incentivos baseados em histórico local precisam de interações frequentes entre *peers*. Isso torna esse mecanismo ineficiente para uso em alguns casos. No entanto, essas interações não constituem problema para grade P2P, pois suas características promovem interações frequentes entre os *peers*, mesmo quando o número de participantes for elevado.

De acordo com as características da rede de favores, podemos afirmar que o *OurGrid* consiste em uma grade *ad hoc* que apresenta independência de controle, ou seja, as políticas de compartilhamento de recursos são realizadas de forma automatizada. Isso permite que a grade possa crescer mais rapidamente que as grades tradicionais cujos recursos são gerenciados por humanos.

2.3.2 Estrutura do *OurGrid*

O *OurGrid* é constituído por cinco componentes, com diferentes responsabilidades na grade. A troca de mensagens é realizada através de XMPP, utilizando criptografia assimétrica (par de chaves: pública e privada). Os componentes são listados abaixo:

- O *Peer* [9] é o gerenciador do *site*, responsável por distribuir os recursos disponíveis, bem como manter a rede de favores;
- O *Broker*[10] é a interface através da qual os usuários da grade podem acessar a grade e submeter aplicações;

[9] http://www.ourgrid.org/index.php?option=com_content&view=article&id=52&Itemid=1
[10] http://www.ourgrid.org/index.php?option=com_content&view=article&id=52&Itemid=1

- O *Worker*[11] é o componente que executa as tarefas das aplicações. Essas tarefas são realizadas de forma protegida de outros usuários e sem prejudicar o desempenho local através de uma *sandbox* com acesso a disco restrito e sem acesso a rede;
- O *Discovery Service*[12] é o componente responsável por divulgar os *peers* para a comunidade;
- O *Aggregator*[13] é o componente responsável por armazenar dados estatísticos relacionados à execução de tarefas na comunidade.

Em qualquer grade criada com o *OurGrid*, os três primeiros componentes precisam ser utilizados. Porém, o *aggregator* e o *discovery service* não precisam ser instalados em todas as grades. O primeiro deverá ser instalado apenas se existir interesse em coletar dados estatísticos sobre a grade. O segundo apenas deverá ser instalado em grades que apresentam mais de um *site*, pois nesse há a necessidade de divulgar os *peers* na comunidade.

2.3.3 Mecanismo de Comunicação

A partir da versão quatro do *OurGrid*, foi substituído *Remote Method Invocation Protocol*[14] (RMI) por uma nova infraestrutura de comunicação que utiliza o *middleware Commune*[15] para manipulação remota de objetos. O *Commune* disponibiliza uma *application programming interface* (API) desenvolvida para sistemas distribuídos e orientada a objeto para comunicação assíncrona, que é intermediada pelo *Extensible Messaging and Presence Protocol*[16](XMPP).

Tendo em vista que o *OurGrid* utiliza o *Commune* no seu mecanismo de comunicação, o primeiro necessita de um servidor XMPP para o seu funcionamento. Na página da internet do *OurGrid* consta que qualquer servidor XMPP deve funcionar corretamente, porém apenas o servidor *Openfire*[17] é certificado para os componentes do *OurGrid*.

[11] http://www.ourgrid.org/index.php?option=com_content&view=article&id=55&Itemid=1
[12] http://www.ourgrid.org/index.php?option=com_content&view=article&id=74&Itemid=1
[13] http://www.ourgrid.org/index.php?option=com_content&view=article&id=74&Itemid=1
[14] http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp
[15] http://commune.lsd.ufcg.edu.br/
[16] http://xmpp.org/
[17] http://www.igniterealtime.org/projects/openfire

3 Trabalhos Relacionados

Entre as grades computacionais *ad hoc* existentes, como o OurGrid, o MyGrid, a que apresenta maior similaridade com o *Hybrid Ad Hoc Grid* é o *Ad Hoc Grid* (Tibúrcio, 2009). Ambas consistem em grades P2P, apresentando características autônomas em relação à estrutura e ao controle. Assim, nessa seção será apresentada a grade desenvolvida por Tibúrcio.

3.1 *Ad Hoc Grid*

O *Ad Hoc Grid* é uma grade computacional P2P auto-organizável desenvolvida por Tibúrcio (2009), a partir do *OurGrid* versão 3.3. Esta apresenta um modelo de comunicação baseado em grupos *multicast* e a manipulação remota de objetos realizada utilizando RMI.

O *Ad Hoc Grid* é constituído por três componentes: *Peer, MyGrid* e *Worker*. O *MyGrid* é o equivalente ao *Broker* das versões atuais do *OurGrid*. O *Peer* e o *Worker* do *Ad Hoc Grid* apresentam as mesmas responsabilidades que o *Peer* e *Worker* do *OurGrid*.

Na grade de Tibúrcio (2009), o *peer* não possui localização fixa, sendo inicializado automaticamente em uma das máquinas que possuir um *worker* instalado. A escolha da localização do *peer* é baseada na máquina do grupo local que possuir o maior valor de endereço IP (*Internet Protocol*).

A comunicação no *Ad Hoc Grid* utiliza dois grupos *multicast*: o *LocalMulticastGroup* que consiste em um grupo utilizado para comunicação entre os componentes na rede local e o *P2PMulticastGroup* que consiste em um grupo utilizado para comunicação entre os *peers* da grade. Esse último possui um endereço fixo e conhecido por todos os *peers*.

Como qualquer *worker* de um grupo *multicast* local pode se tornar também um *peer*, a rede de favores é replicada na rede local sempre que ocorrerem alterações em seus dados. Tal ação é realizada com o objetivo de manter a consistência da rede de favores, quando um novo *peer* for instanciado. A FIGURA 8 ilustra um modelo de grade típica criada com o *Ad Hoc Grid*.

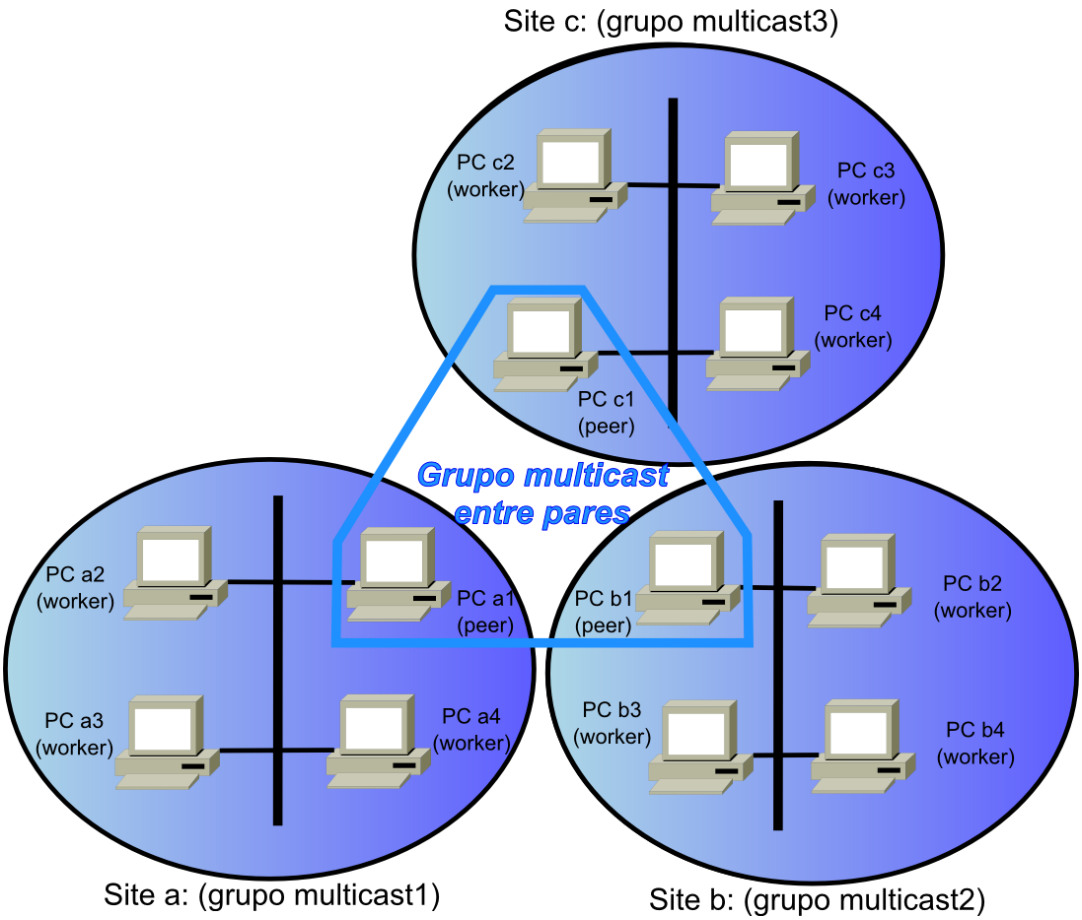

FIGURA 8: Modelo de grade *Ad Hoc Grid*

Conforme a figura ilustra, a grade é constituída por *sites*. Cada *site* utiliza um grupo *multicast* local para comunicação entre os componentes. A comunicação entre os *peers* também utiliza um grupo *multicast*, dessa forma é dispensado o uso de um componente equivalente ao *discovery service* do *OurGrid*.

3.1.1 Limitações

O *Ad Hoc Grid* apresenta algumas limitações que podem inviabilizar a sua utilização. A primeira limitação está relacionada à complexidade em sua instalação. O modelo de comunicação adotado para a organização da grade (grupos *multicast)* apresenta dificuldades operacionais. Segundo Tibúrcio (2009), apesar da comunicação *multicast* estar consolidada, a configuração de equipamentos de interconexão da rede, para uso com o *Ad Hoc Grid*, é uma tarefa com custos e demorada. Por esse motivo, a instalação dessa grade necessita de investimento inicial com equipamentos e administradores capacitados para configurá-la.

A outra limitação está relacionada ao roteamento *multicast*. A internet não oferece suporte a esse tipo de roteamento, o que dificulta a utilização dessa grade em diferentes redes.

Por último, temos o problema de atualização da grade. O *Ad Hoc Grid* foi desenvolvido a partir do *OurGrid* na versão 3.3. Tendo em vista que essa versão utiliza RMI e que as novas versões do *OurGrid* utilizam o *Middleware Commune*, torna-se pouco provável o lançamento de novas versões do *Ad Hoc Grid*.

4 *Hybrid Ad Hoc Grid*

Nesse capítulo, será apresentado o *Hybrid Ad Hoc Grid*, uma grade *ad hoc* que apresenta autonomia em relação ao seu controle e a sua estrutura. O mesmo foi desenvolvido com o objetivo de atender a comunidades esporádicas e simplificar sua instalação e utilização. O *Hybrid Ad Hoc Grid* foi desenvolvido a partir da versão 4.2.1 do *OurGrid* e utiliza *multicast* e XMPP como mecanismos de comunicação.

O *Hybrid Ad Hoc Grid* é uma solução híbrida porque, apesar de possuir uma estrutura auto-organizável, possui um componente com localização fixa que é responsável por divulgar os membros na grade. Como foi comentando anteriormente, uma das limitações do *Ad Hoc Grid* é a utilização de *multicast* para a divulgação. A vantagem da solução híbrida consiste em possibilitar a utilização na internet, pois como o endereço do membro fixo é conhecido, não é necessário o uso de *multicast* entre os *sites*.

4.1 Estrutura do *Hybrid Ad Hoc Grid*

O *Hybrid Ad Hoc Grid* é constituído por cinco componentes: *Peer*, *Worker*, *Broker*, *Discovery Service* e *AdHoc*. Os três primeiros são variações dos componentes do *OurGrid*, modificados para se adaptarem ao comportamento *ad hoc* da grade. Esses componentes serão detalhados nas próximas subseções. O Componente Aggregator do OurGrid não foi adotado no *Hybrid Ad Hoc Grid*, pois manter um banco de dados *ad hoc* poderia consumir muitos recursos e também por não existir necessariamente um administrador, os dados estatísticos coletados pelo *aggregator* podem não ter muita utilidade. A FIGURA 9 abaixo ilustra uma grade *Hybrid Ad Hoc Grid* típica.

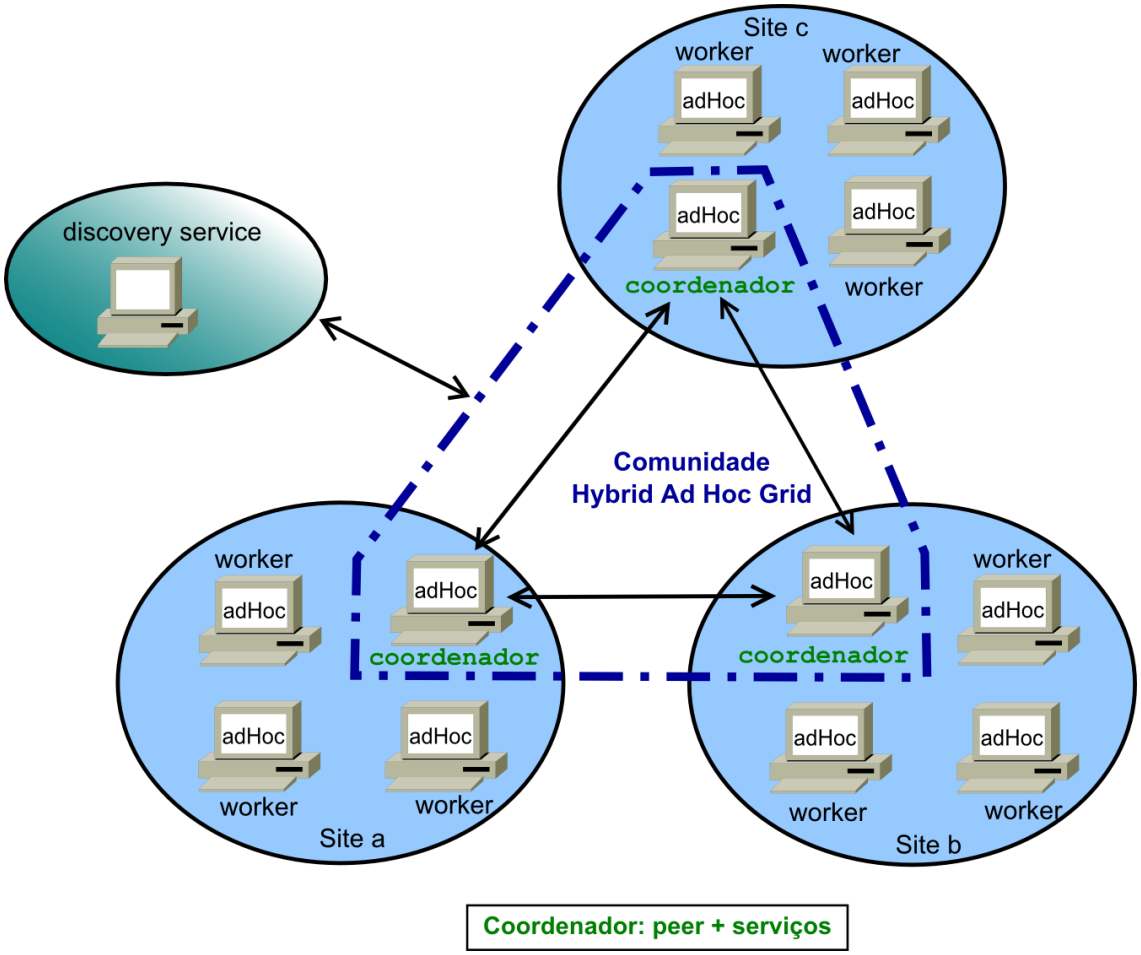

FIGURA 9: Modelo de grade *Hybrid Ad Hoc Grid*

Conforme a figura, uma grade é constituída por *sites*. Para cada *site*, um membro é eleito coordenador, sendo então responsável por administrar um *peer* e manter os serviços necessários à grade. O Coordenador é um membro esporádico e pode sair da grade a qualquer momento. Nesse caso, um novo coordenador é eleito através de um algoritmo de eleição.

Uma comunidade é constituída pelo conjunto de *peers* da grade. O *discovery service* consiste em um componente com localização fixa e conhecida por todos os membros da grade. Sua função é divulgar os membros da comunidade, permitindo assim que os *peers* se comuniquem diretamente.

4.1.1 Componente *AdHoc*

O *AdHoc* é o componente responsável por administrar os *sites*, adaptando-se às constantes mudanças características de grades *ad hoc*. Para isso, o *adHoc* é executado como um serviço em cada membro da grade. Em cada *site*, um *adHoc* é eleito coordenador através

de um algoritmo de eleição. Todo componente *adHoc* possui responsabilidades que variam de acordo com a sua função na grade (coordenador ou não-coordenador).

Os não-coordenadores possuem quatro responsabilidades. A primeira consiste em registrar as máquinas no coordenador. Dessa forma, essas máquinas tornam-se aptas a executar um *worker* ou uma interface de acesso à grade (*broker*). O registro é realizado em dois momentos: quando um não-coordenador localiza o coordenador e quando o algoritmo de eleição é executado. Esse registro é necessário para permitir ao *peer* conhecer os recursos que serão compartilhados e os usuários que podem acessá-lo.

A segunda responsabilidade dos não-coordenadores consiste em determinar falha do atual coordenador. Constantemente o coordenador informa para os membros que se encontra ativo. A falha é determinada quando o coordenador para de realizar esse informe. Quando isso ocorre, os não-coordenadores iniciam a eleição de um novo coordenador.

A terceira responsabilidade consiste em executar a eleição de um novo coordenador. O algoritmo de eleição é detalhado na seção 4.2.

Por último, os não-coordenadores são responsáveis por manter réplica do mecanismo de incentivo (rede de favores). Periodicamente o coordenador envia uma réplica da rede de favores para os outros membros. Cada réplica possui um número de sequência que a identifica. Quando essas são recebidas, o seu número de sequência é comparado ao valor local. Caso o valor local seja maior que o da réplica recebida, uma notificação é enviada para todos os outros *adHoc*, junto com a réplica com maior número de sequência. Em caso contrário, o valor do número de sequência e a réplica local são atualizados.

O coordenador é responsável por registrar os *workers* e os usuários do *site*, replicar as chaves públicas e privadas do *peer* e os registros do mecanismo de incentivo (rede de favores), e manter um *peer* e um servidor XMPP ativos. A réplica das chaves é necessária para assegurar que toda nova instância de um coordenador no mesmo *site* seja reconhecida como o mesmo coordenador, garantindo a consistência dos dados da rede de favores. O Servidor XMPP consiste em um serviço necessário para o funcionamento da grade, pois a comunicação entre os componentes utiliza XMPP.

4.1.2 *Peer*

O *peer* é o componente responsável por distribuir os recursos da grade para os usuários. Uma comunidade é constituída por vários *peers* que compartilham recursos entre si.

Para que um *peer* ingresse em uma comunidade, o mesmo deve conhecer a localização do *discovery service* da comunidade de que tenha interesse. Essa localização é obtida através do arquivo de configuração peer.properties que deve ser editado, no primeiro componente *AdHoc* coordenador inicializado em um *site*. Os demais *AdHoc* inicializados obtêm a localização do *discovery service* a partir do primeiro *AdHoc* inicializado no *site*.

Da mesma forma que no *OurGrid*, no *Hybrid Ad Hoc Grid* os componentes são identificados por suas chaves públicas. De acordo com o comportamento esporádico do *Hybrid Ad Hoc Grid*, é esperado que um novo coordenador seja eleito frequentemente. Sempre que isso ocorrer, o novo coordenador deve utilizar as mesmas chaves utilizadas pelo anterior. Com isso, fica assegurado que as novas instâncias não serão identificadas como *sites* diferentes. Essas chaves são criadas quando o primeiro coordenador é instanciado e são replicadas quando um *adHoc* não-coordenador se comunica pela primeira vez com o coordenador.

4.1.3 *Worker* e *Broker*

O *worker* e o *broker* do *OurGrid* foram desenvolvidos para *sites* de estrutura fixa. Cada alteração na estrutura de um *site* requer que novas configurações manuais sejam realizadas em todos os componentes. Além disso, toda nova instância de *worker* que ingressar em um *site* do *OurGrid* precisa ser registrado manualmente no *peer*. O mesmo ocorre para cada novo usuário que desejar utilizar a grade.

O *Hybrid Ad Hoc Grid* foi desenvolvido para suportar comunidades esporádicas e que requerem reduzidas atividades administrativas. Assim, o *worker* e o *broker* obtêm do coordenador do *site* os parâmetros necessários à sua inicialização. Além disso, nessa grade o registro dos *workers* e dos usuários é realizado de forma automática pelo *adHoc*, possibilitando que ambos se adaptem de forma automática às mudanças na grade.

Essa adaptação automática tornou desnecessária a utilização dos arquivos SDF (*Site Description File*) e GDF (*Grid Description File*) comentados na seção 2.3 simplificando a instalação do *Hybrid Ad Hoc Grid* e automatizando a adaptação do *peer* a mudanças dos membros do *site*.

4.1.4 *Discovery Service*

O componente *Discovery Service* do *OurGrid* foi mantido inalterado no *Hybrid Ad Hoc Grid*, tendo em vista que o primeiro não apresenta comportamento *ad hoc* na grade.

4.2 Coordenação do *Site*

Em uma grade *Hybrid Ad Hoc Grid* típica existem vários componentes *AdHoc* em execução, sendo uma instância por membro da grade. Para cada *site* há um coordenador. Quando um *site* fica sem coordenador, o mesmo fica indisponível para a comunidade e os resultados das tarefas em execução são perdidos. Assim, é escolhido um novo coordenador através de um algoritmo de eleição discutido a seguir.

4.2.1 Algoritmos de Eleição

O objetivo de um algoritmo de eleição é obter um consenso sobre a escolha de um entre vários processos em um sistema distribuído, para que esse realize a função de coordenador dos demais. Segundo Garcia-Molina (1982), a eleição de um coordenador pode ser tratada como um problema de sincronização de processos paralelos. Para que o algoritmo seja executado corretamente, deve ser garantido que em algum momento um coordenador seja escolhido e que seja escolhido apenas um por eleição. Além disso, um bom algoritmo também deve tratar possíveis falhas ocorridas durante a eleição.

No *Hybrid Ad Hoc Grid*, o processo de eleição se torna necessário sempre que um *site* não apresentar um coordenador. Isso ocorre quando um novo *site* é criado e quando o coordenador falha ou sai da grade. De modo geral, qualquer instância do *adHoc* pode determinar a falha do atual coordenador e iniciar o processo de eleição. Além disso, todos os processos participantes da eleição precisam ser informados e concordar com o resultado da eleição. Nas próximas seções serão discutidos alguns algoritmos clássicos de eleição.

4.2.1.1 Algoritmo em Anel

O Algoritmo baseado em anel foi proposto por Chang e Robert (1979) para processos interligados em estrutura de anel, com as mensagens circulando em apenas um sentido. Nesse

algoritmo, cada processo tem um identificador único e a eleição se inicia quando um processo P_i envia uma mensagem de eleição contendo o seu identificador para o processo $P_{(i+1 \bmod N)}$. Quando o processo $P_{(i+1 \bmod N)}$ recebe essa mensagem, o mesmo calcula o máximo entre o seu próprio identificador e o identificador da mensagem recebida. O resultado é enviado através de uma mensagem de eleição para o próximo processo. A eleição se encerra quando um processo recebe uma mensagem de eleição contendo seu próprio identificador. Uma vez encerrada, uma mensagem é enviada para todos os processos informando o identificador do processo eleito.

4.2.1.2 Algoritmo *Bully*

O Algoritmo *Bully* foi proposto por Garcia-Molina (1982) para sistemas síncronos e é baseado na difusão de mensagens. Nesse algoritmo, supõe-se que os membros permaneçam sempre ativos, executam o mesmo algoritmo de eleição e possuam um identificador único. Quando a eleição é iniciada, ocorrem trocas de mensagens entre os membros e o processo com maior identificador será eleito coordenador dos demais processos.

Existem dois casos que iniciam um processo de eleição:

- O primeiro caso é iniciado pelo coordenador. O mesmo verifica periodicamente os membros ativos, comparando-os com os membros envolvidos na última eleição. Caso ocorram mudanças, uma nova eleição é iniciada. Dessa forma, o algoritmo assegura que, em qualquer momento, o membro com maior identificador será o coordenador;
- O segundo caso é iniciado pelos demais membros. Os mesmos verificam periodicamente o coordenador e, caso um desses verifique falha do coordenador, um novo processo de eleição é iniciado.

4.2.2.3 Algoritmo de *Bully* Modificado

O *Hybrid Ad Hoc Grid* precisa eleger um componente *adHoc* como coordenador para o *site*. Para isso, foi desenvolvida uma variação do algoritmo de *Bully*. Optou-se por esse algoritmo, pois o mesmo possibilita eleger o coordenador de forma simples e sem nenhum conhecimento prévio da estrutura da grade. Sua utilização será restrita ao nível do *site*, portanto não apresentando problemas na difusão das mensagens.

Ao contrário dos componentes *adHoc* não-coordenadores, o coordenador consome muitos recursos locais, pois mantém um *peer* e os serviços da grade ativos. O Algoritmo de *Bully* foi modificado para adequá-lo à grade e reduzir o consumo de recursos. As alterações realizadas no algoritmo de *Bully* foram:

- Como cada componente *AdHoc* possui uma chave pública, essa foi utilizada como identificador no processo na eleição.
- Seguindo a ideia do algoritmo em anel, o algoritmo Bully modificado assegura que o membro com maior identificador, no momento da eleição, é escolhido coordenador. Ao contrário do algoritmo original, a entrada de membros com maiores identificadores que o coordenador não causa o início de uma nova eleição. Com isso, é reduzida a frequência de realização de eleição, evitando mudanças desnecessárias na estrutura dos *sites*.
- Ao contrário do algoritmo de *Bully* original no qual os membros verificam se o coordenador está ativo, nessa versão modificada o coordenador divulga periodicamente que está ativo. Dessa forma, reduz-se a troca de mensagens entre coordenador e os participantes. Essa redução resulta em um impacto positivo no desempenho do coordenador, principalmente quando há um grande número de participantes.
- A detecção de falhas consiste em aguardar que o coordenador não envie mensagens por um período de tempo de número de tentativas X *timeout*. Caso isso ocorra, um novo processo de eleição é iniciado.

O algoritmo também é capaz de se detectar falhas ocorridas após a escolha do coordenador e antes que o mesmo se divulgue e inicie suas atividades. Para isso, quando o membro com maior identificador envia uma mensagem com o seu identificador, os demais processos aguardam por um período de tempo que esse membro inicie suas atividades. Caso após esse período de tempo não sejam enviadas mensagens pelo coordenador, um novo processo de eleição é iniciado imediatamente. Esse período de tempo é definido pelo triplo do valor de *timeout*. Esse valor é obtido no arquivo adhoc.properties, sendo o valor padrão de 2000 ms.

No novo algoritmo, eleições são iniciadas apenas quando não existir um coordenador ativo. Durante o desenvolvimento foram atribuídos valores fixos para *timeout* e número de tentativas (3). O valor do *timeout* pode ser configurado no arquivo de configuração adhoc.properties (APÊNDICE C - Adhoc.properties). Esses valores foram escolhidos com o

objetivo de evitar a inicialização de eleições desnecessárias. Em trabalhos futuros, podem ser exploradas alternativas como determinar esses valores a partir da análise do tráfego na rede.

4.3 Servidor XMPP

O coordenador de um *site* é escolhido através de um algoritmo de eleição. Para garantir o correto funcionamento do *site*, um servidor XMPP é inicializado em cada componente *adHoc* que se tornar coordenador. Após a inicialização, sua localização é divulgada via *multicast* pelo coordenador para os demais membros. Com isso, é assegurado que os membros da grade sempre obtenham a localização atualizada do servidor XMPP.

Existem vários servidores XMPP disponíveis atualmente. O QUADRO 1 a seguir mostra uma breve comparação entre vários servidores *open-source*.

QUADRO 1
Comparação entre servidores XMPP *open-source*

Características	**Wildfire (Openfire)**	**Ejabberd**	**OpenIM**	**WPJabber**
Situação	Atualizado	Atualizado	Sem atualizações recentes	Sem atualizações recentes
Licença	Código aberto (GPL)	Código aberto (GPL)	Código aberto (BSD)	Código aberto (GPL)
Plataforma	Qualquer plataforma capaz de executar Java	Qualquer plataforma capaz de executar Erlang: variações de Unix (Linux, Solaris, BSD), Windows e Mac OS X	Qualquer plataforma capaz de executar Java 1.4	Somente Linux. Sendo necessário aplicar um pacote de correções para Linux 2.4.x e 2.6.x
Estabilidade	Alta estabilidade, existem várias implementações e o código vem sendo desenvolvido há vários anos	Alta estabilidade, funcionamento estável em grandes hospedagens e o código está em desenvolvimento há vários anos	Desde janeiro de 2004, utilizado em produção com sucesso	Alta estabilidade, funcionamento estável em várias grandes hospedagens
Desenvolvimento	Ativo, com o desenvolvimento de melhorias e correções falhas em aberto	Ativo, com o desenvolvimento de melhorias e correções falhas em aberto	Inativo, com correções falhas em aberto	Inativo, sem alterações há muito tempo

Número de desenvolvedores	Dois desenvolvedores principais (Gaston Dombiak, Matt Tucker). Alguns colaboradores da Jive Software e da Ryan Graham trabalhando em plugins e correções de falhas. Demais colaboradores trabalhando em *plugins*	Um desenvolvedor principal (Alexey Shchepin) e vários outros contribuidores.	Dois (Justin Kirby e Stephen Marquard)	Um (Lukas Karwacki)
Suporte	Site na internet, fórum, base de conhecimento e suporte comercial	Lista de *e-mail*, sala de discussão, Fórum e suporte comercial	Lista de *e-mail*	Lista de *e-mail* e comentários no site
Armazenamento de dados	Armazenamento, autenticação e informações sobre usuários podem ser armazenadas em banco de dados. Armazenamento pode ser realizado em: MySQL, Postgresql, Oracle, SQL Server, DB2, Berkeley DB. Autenticação e informações sobre o usuário podem ser realizadas em qualquer meio de armazenamento, incluindo LDAP e *Active Directory*	Podem ser utilizado VCards e LDAP, porém esse último pode ser utilizado apenas para autenticação. O Armazenamento de dados padrão é o Mnesia, porém oferece suporte a ODBC e PostgreSQL	Armazenamento e dados de autenticação podem mantidos em diferentes repositórios. São suportados: arquivos, LDAP, banco de dados via JDBC ou Hibernate (experimental)	Armazenamento e dados de autenticação podem mantidos em diferentes repositórios. Ambos podem ser manipulados por XDB, portanto qualquer modulo XDB pode ser utilizado

Fonte: SAINT-ANDRE, 2010.

O servidor escolhido para ser utilizado no *Hybrid Ad Hoc Grid* foi o *Openfire*, que consiste em um servidor XMPP desenvolvido pela mesma equipe do *Wildfire*. Essa escolha foi baseada nos seguintes argumentos: Esse servidor é certificado para os componentes do

OurGrid, apresenta alta estabilidade, possibilita o seu uso com banco de dados embarcado e pode ser utilizado em diversos sistemas operacionais.

4.4 Rede de Favores

A Rede de favores (*Network of Favor* – NOF) consiste em um mecanismo de incentivo para o compartilhamento de recursos na comunidade. No *Hybrid Ad Hoc Grid*, devido a sua natureza *ad hoc*, o *peer* pode estar localizado em qualquer membro do *site* e esse membro pode mudar a qualquer momento. Para garantir a consistência da rede de favores, o coordenador do *site* a replica periodicamente nos membros não-coordenadores. Essa replicação da NOF pode gerar muito tráfego na rede, se a grade possuir muitos membros. Para evitar isso, a replicação é realizada utilizando *multicast*, por se tratar de uma forma eficiente de difusão de mensagens.

4.5 Mecanismo de Comunicação XMPP

No desenvolvimento do *Hybrid Ad Hoc Grid*, foi mantida a arquitetura de comunicação utilizada no *OurGrid* 4.2. Na FIGURA 10, ilustramos as classes utilizadas para acesso e manipulação remota de objetos no componente *adHoc*.

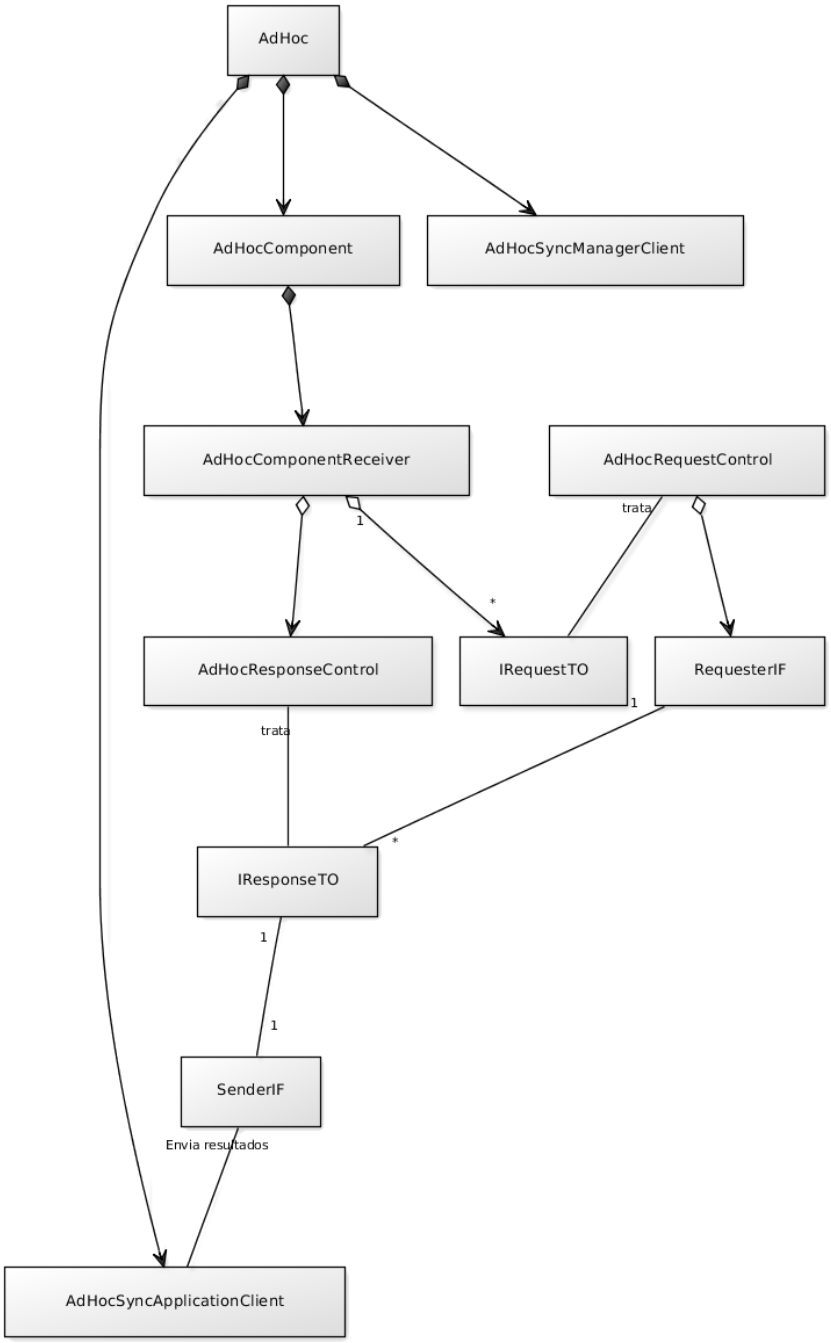

FIGURA 10: Diagrama do mecanismo de comunicação XMPP

O *AdHoc* possui três classes utilizadas para comunicação remota entre objetos: o *AdHocComponent*, o *AdHocSyncManagerClient* e o *AdHocSyncApplicationClient*.

Para realizar a manipulação remota da classe *AdHocComponent* é necessário registrar interesse através do *middleware Commune*. No momento do registro de interesse, é associada uma classe que será responsável por tratar as respostas dos métodos remotos. O *AdHocComponent* cria um *AdHocComponentReceiver* que possui os métodos que podem ser requisitados remotamente. Para cada um desses métodos existe uma instância de uma implementação da interface *IRequestTO* no *AdHocRequestControl*. Esse último associa para

cada implementação do *IRequestTO* a uma implementação correspondente da interface *RequesterIF*.

As implementações da interface *RequesterIF* são responsáveis por efetivamente executar as requisições. Como resultado dessa execução, é produzida uma lista com implementações da interface *IResponseTO*.

AdHocResponseControl é responsável por enviar as respostas produzidas na execução de um *RequesterIF*. Para isso, existe uma associação entre cada implementação de *IResponseTO* a uma implementação da interface *SenderIF*. O resultado do tratamento da lista de *IResponseTO* resulta em uma lista de *SenderIF*. Os resultados armazenados no *SenderIF* são então enviados através do *AdHocSyncApplicationClient*.

Na instância remota que solicitou algum método, as respostas da execução são recebidas através da classe *AdHocSyncManagerClient* e tratadas através de uma classe associada no momento do registro do interesse.

4.6 Mecanismo de Comunicação *Multicast*

Como o *Hybrid Ad Hoc Grid* não apresenta um coordenador fixo, o servidor XMPP utilizado também não apresenta um endereço fixo. Dessa forma, quando um novo coordenador precisar ser escolhido, não existe um servidor XMPP ativo para intermediar a comunicação. Com isso, se torna necessário outro mecanismo de comunicação que possa ser utilizado independentemente de existir um servidor XMPP ativo no *site*.

Com o objetivo de atender e viabilizar a comunicação nesse cenário, um segundo mecanismo de comunicação foi desenvolvido para o *Hybrid Ad Hoc Grid*. O novo mecanismo utiliza *multicast* porque consiste em uma estratégia eficiente para distribuir mensagens para vários destinatários e, ao contrário da grade de Tibúrcio (2009), é utilizado apenas na rede local. Esse mecanismo é utilizado para:

- Trocas de mensagens na execução do algoritmo de eleição do coordenador, tendo em vista que nesse momento a utilização do protocolo XMPP não é possível;
- Divulgação do coordenador, pois cada novo membro deve ser capaz de obter do coordenador a localização do *peer* e do servidor XMPP para poder ingressar na grade;
- Replicação da rede de favores, pois dessa forma é possível reduzir tráfego de rede que seria causado pelo envio de várias réplicas;

- Envio dos parâmetros de configuração necessários para a inicialização dos componentes b*roker* e w*orker*.

  A FIGURA 11 apresenta o diagrama do mecanismo de comunicação *multicast*.

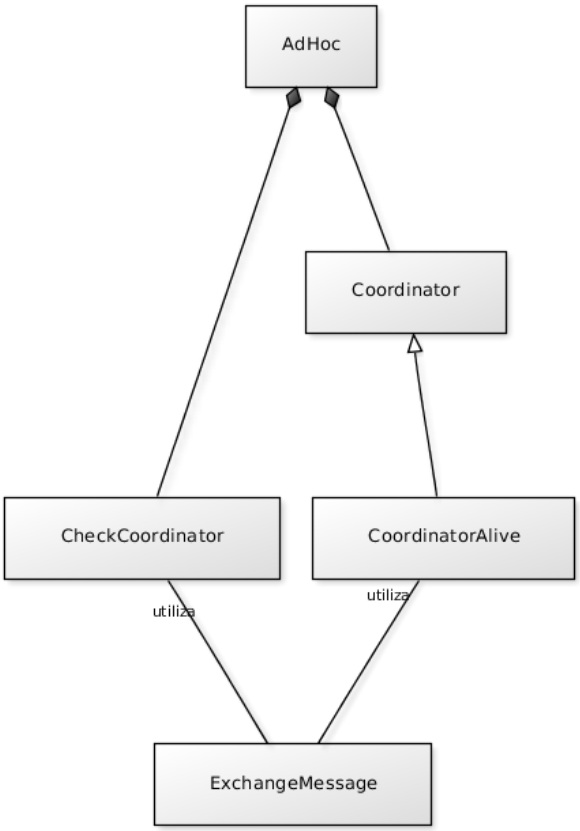

FIGURA 11: Diagrama do mecanismo de comunicação *multicast*

Cada componente *AdHoc* coordenador mantém uma instância da classe *Coordinator*. Essa última é responsável por divulgar réplicas da rede de favores e manter uma instância do *CoordinatorAlive*.

O *CoordinatorAlive* é responsável por divulgar as localizações do coordenador e do servidor XMPP do *site*, e por enviar os parâmetros de configuração necessários para o *broker* e *worker*.

Os componentes *AdHoc* não-coordenadores do *site* apenas mantêm uma instância de *CheckCoordinator*, que é responsável por verificar a presença e determinar falhas do coordenador. Esse último também é responsável por iniciar o algoritmo de eleição quando for determinada falha do coordenador.

ExchangeMessage é responsável pelas trocas de mensagens implementando um protocolo definido para essa finalidade. As mensagens são transmitidas por *multicast* utilizando o grupo 239.10.10.10.

5 Avaliação

Nessa seção é apresentada a metodologia de avaliação realizada no *Hybrid Ad Hoc Grid*, para determinar seu desempenho em quatro cenários. Na seção 5.1, avalia-se um cenário de criação de um *site* em uma grade. Na seção 5.2, avalia-se um cenário típico, no qual novos recursos são disponibilizados na grade. Na seção 5.3, analisa-se um cenário no qual um novo coordenador precisa ser eleito. Na seção 5.4, avalia-se o desempenho da grade para a recuperação à falha ocorrida durante a eleição de um coordenador.

O *Hybrid Ad Hoc Grid* foi instalado nos 26 computadores utilizados na avaliação. Os procedimentos realizados na instalação podem ser vistos no APÊNDICE A - Instalação do *Hybrid Ad Hoc Grid*. Todos os experimentos descritos nessa seção foram realizados em uma rede *Ethernet* com trinta máquinas, largura de banda de 100 Mbps e com um servidor DHCP ativo. Os computadores utilizados nos experimentos apresentam características de desempenho heterogêneas. Durante os experimentos existiam usuários distintos compartilhando os recursos.

5.1 Criação de um *Site*

Na criação de uma grade *Hybrid Ad Hoc Grid*, é instalado um *discovery service* que possibilita que os *peers* possam formar uma comunidade. Em seguida, são instalados os *sites* da grade. Quando apenas um componente *adHoc* for instalado e inicializado em um *site*, ocorre um processo de eleição e o mesmo se elegerá coordenador do *site*, assumindo as responsabilidades de coordenador e divulgando o *site* no *discovery service*.

Nesse cenário, foi avaliado o tempo necessário para que o primeiro *adHoc* determine a inexistência de um coordenador no *site* e se eleja novo coordenador. Foram realizados dez experimentos de criação de *sites*, sendo então registrados os tempos obtidos através de arquivos de log. Os resultados podem ser vistos no GRÁFICO 1.

GRÁFICO 1: Tempos do experimento de criação de *site*

O tempo médio para o *adHoc* determinar a inexistência de um coordenador no *site* foi em média 6332 ms, com desvio padrão de 261 ms. A partir desses dados, pode-se afirmar, com um nível de confiança de 95%, que o intervalo de confiança para que o primeiro coordenador de um *site* seja eleito é entre 6171 ms e 6494 ms. As fórmulas utilizadas para a obtenção dos resultados constam no APÊNDICE F.

5.2 Ingresso de um Novo Membro na Grade

Em uma grade *Hybrid Ad Hoc Grid*, os novos membros de um *site* ingressam de forma automática na mesma. Em um cenário típico de um *site*, existe um coordenador ativo na grade, portanto, todos os serviços necessários se encontram disponíveis. Para que um novo membro ingresse na grade, é necessária a instalação de um componente *adHoc* local, o qual administrará outros componentes (*worker* e *broker*) que podem ser instalados localmente.

Nessa seção mostra-se a avaliação do tempo necessário para que um componente *adHoc* localize o coordenador do *site*. Foram realizados dez experimentos em um *site* com o coordenador e todos os seus serviços ativos. Os tempos para descobrir o coordenador podem ser visualizados no GRÁFICO 2.

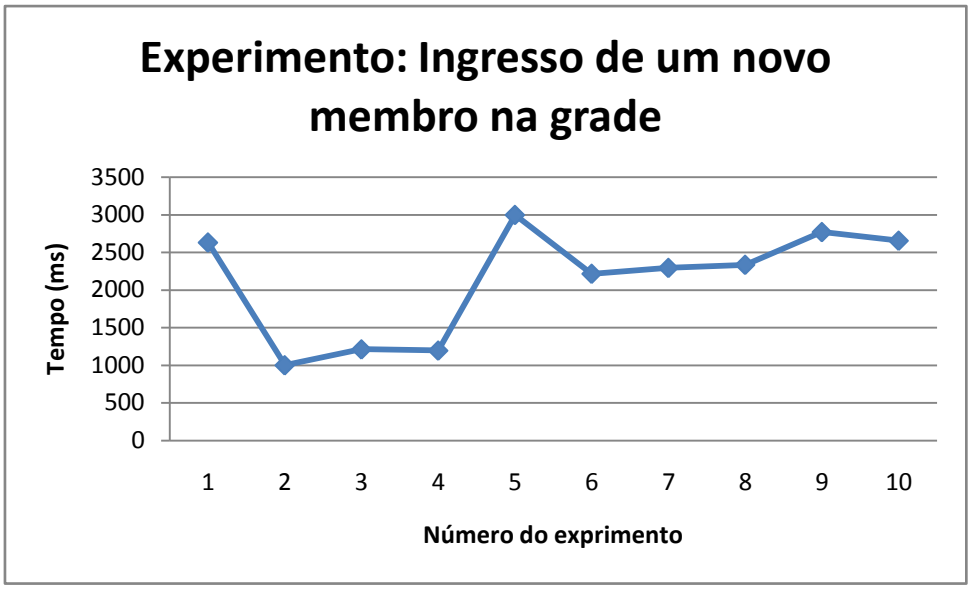

GRÁFICO 2: Tempos do experimento de ingresso de um novo membro na grade

O coordenador do *site* foi localizado com o tempo médio de 2129 ms, com desvio padrão de 726 ms. Assim, pode-se afirmar, com um nível de confiança de 95%, que o intervalo de confiança para o tempo de descoberta do coordenador é entre 1679 ms e 2589 ms.

5.3 Eleição de um Novo Coordenador

Os membros de uma grade *ad hoc* podem sair da grade a qualquer momento. Como no *Hybrid Ad Hoc Grid* qualquer membro da grade pode ser eleito coordenador, esse último também pode sair da mesma a qualquer momento. Além disso, um coordenador também pode apresentar falha durante o seu funcionamento. Nesses dois casos, um novo coordenador precisa ser eleito e isso é realizado através do algoritmo de eleição comentado anteriormente.

Nessa seção, analisa-se o impacto da escolha de um novo coordenador para a grade. Para isso foram realizados dez experimentos, instanciando e monitorados componentes *adHoc* em 26 máquinas. Após estabelecimento do *site*, o coordenador foi encerrado manualmente. Dessa forma, reproduziu-se um cenário de início de eleição. O tempo necessário para a escolha de um novo coordenador pode ser visualizado no GRÁFICO 3.

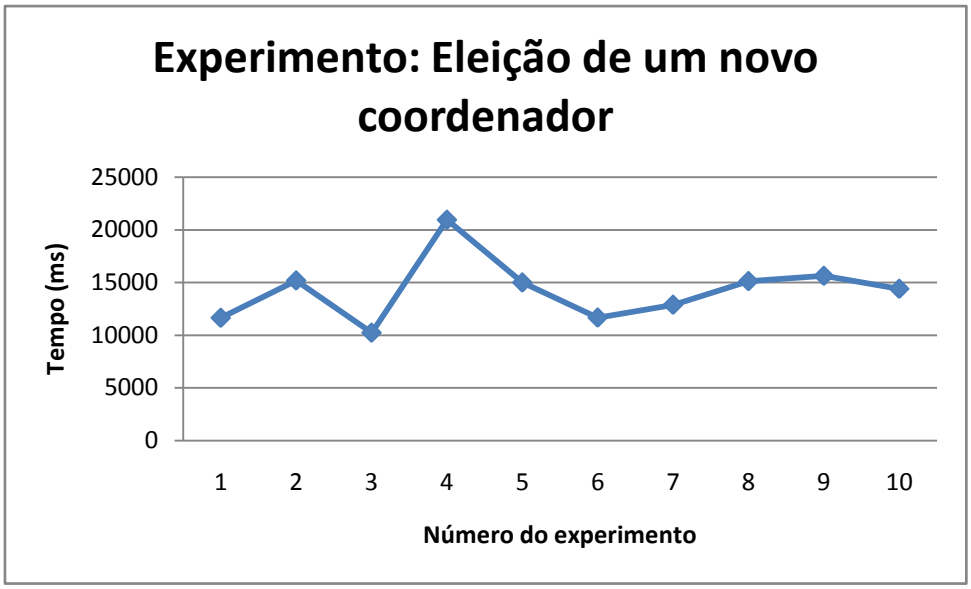

GRÁFICO 3: Tempos do experimento de eleição de um novo coordenador

Um novo coordenador foi eleito com o tempo médio em 14275 ms, com desvio padrão de 2994 ms. A partir desses dados, pode-se afirmar, com um nível de confiança de 95%, que o tempo para a escolha do coordenador apresenta um intervalo de confiança entre 12420 ms e 16131 ms.

5.4 Falha Durante a Eleição do Coordenador

Na execução do algoritmo de eleição, um componente *adHoc* A_1 envia uma mensagem com seu identificador para os demais *adHoc* participantes da eleição. Os *adHoc* com identificador maior que A_1 respondem à mensagem com seus próprios identificadores. Quando o *adHoc* A_n com o maior identificador dentre os participantes responder a mensagem, os demais aguardam, por certo período de tempo, que A_n se declare coordenador. Se A_n não se declarar coordenador, assume-se a falha do mesmo e uma nova eleição é iniciada.

Nessa seção, foi analisado o desempenho da grade nesse tipo de cenário. Para isso, foram realizados dez experimentos, monitorando 26 componentes *adHocs* no processo de eleição. Quando o *adHoc* com maior identificador respondia ao processo de eleição, o mesmo era encerrado manualmente. Os tempos para determinação de falha na eleição podem ser visualizados no GRÁFICO 4.

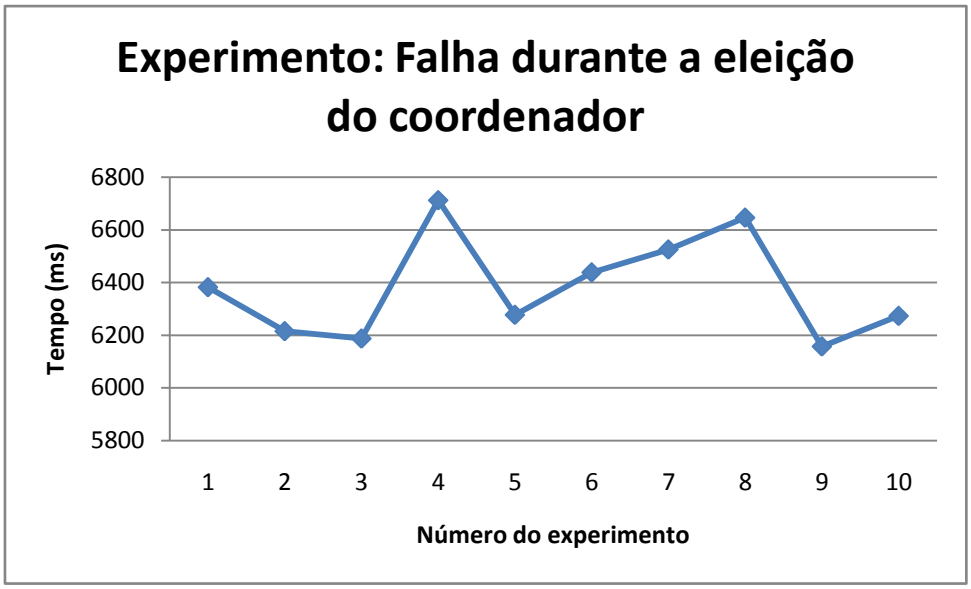

GRÁFICO 4: Tempos do experimento falha durante a eleição do coordenador

O tempo médio para determinar falha na eleição foi de 6359 ms, com desvio padrão de 186 ms. Com isso, afirma-se, a um nível de confiança de 95%, que o intervalo de confiança para determinação de falha na eleição é entre 6244 ms e 6475 ms.

5.5 Discussão dos Resultados

O *Hybrid Ad Hoc Grid* foi desenvolvido a partir do *OurGrid*. Esse último encontra-se em produção desde 2004, sendo utilizado por diversos usuários. Após a formação da comunidade, o *Hybrid Ad Hoc Grid* funciona de forma similar ao *OurGrid*. Isso pode ser confirmado através da execução de aplicações nas duas grades. Porém não é o objetivo dessa avaliação analisar todos os aspectos da grade, mas apenas as características de auto-organização.

O *Ad Hoc Grid* é a grade que apresenta maior similaridade com o *Hybrid Ad Hoc Grid*, devido aos fatores discutidos na seção sobre trabalhos relacionados. Infelizmente, não há informações sobre o desempenho dessa primeira grade para comparação. Além disso, instalar, obter e configurar os equipamentos necessários para o seu funcionamento e testar o *Ad Hoc Grid* para obter informações de desempenho exige tempo e conhecimento que fugiria ao escopo dessa dissertação.

O *Hybrid Ad Hoc Grid* foi desenvolvido de forma que a grade fique o menor tempo possível sem coordenador e que novos recursos sejam disponibilizados tão rapidamente

quanto possível. Com isso, se reduz os impactos causados pelas mudanças na estrutura da grade.

De acordo com os experimentos, pode-se dizer que os tempos obtidos na criação do *site* e no ingresso de um novo membro são aceitáveis. O tempo para eleição de um novo coordenador apresentou tempos um pouco elevados. Isso ocorre devido à troca de mensagens entre os participantes na eleição e ao tempo que o coordenador aguarda para se declarar eleito e iniciar suas atividades de coordenador.

Durante o processo de eleição, um coordenador pode falhar antes de iniciar suas atividades. Os experimentos realizados no quarto cenário avaliam essa situação. Esse tempo isoladamente pode ser aceitável, porém torna-se elevado quando somado ao tempo necessário para o novo processo de eleição resultante da falha.

6 Conclusão

Nesse trabalho, foi desenvolvida uma grade computacional *ad hoc* entre pares, o *Hybrid Ad Hoc Grid*. As grades *ad hoc* podem apresentar independência de controle, estrutura e tecnologia. Essa grade foi desenvolvida a partir da grade OurGrid, que apresenta independência de controle devido as políticas de compartilhamento de recursos serem realizadas de forma automatizada, através do seu mecanismo de incentivo, denominada rede de favores. O *Hybrid Ad Hoc Grid* apresenta independência de controle, herdada do OurGrid, e independência de estrutura. O estabelecimento da estrutura da grade e a descoberta de novos recursos são realizados de forma automatizada, através do componente *adHoc* introduzido para essa finalidade.

O Hybrid Ad Hoc Grid consiste em uma grade *ad hoc* híbrida, pois possui o componente *discovery service*, responsável pela divulgação dos *sites* para a comunidade, que não apresenta comportamento auto-organizável. Essa solução foi adotada como forma de simplificar a divulgação dos *sites* para a comunidade, pois soluções alternativas para divulgação exigiriam serviços de terceiros ou seriam complexos de serem mantidos, como no caso do *Ad Hoc Grid.*

O Hybrid Ad Hoc Grid foi desenvolvido de forma a apresentar uma instalação simplificada, pois necessita de poucas configurações para a sua utilização. Além disso, por apresentar independência de controle, essa grade se adapta rapidamente as mudanças em sua estrutura, dispensando a necessidade de um administrador. Devido a essas características, essa grade possibilita uma rápida expansão do seu número de membros.

Em uma grade tradicional, o administrador pode demorar muito tempo para se adaptar a mudanças em sua estrutura e recuperar falhas. Já no *Hybrid Ad Hoc Grid*, essas atividades são realizadas de forma automatizada e, conforme demonstram os experimentos realizados, demorando apenas alguns poucos segundos.

Como trabalho futuro, existem várias possibilidades que podem ser exploradas:

- Explorar a troca de chaves públicas: O coordenador troca a chave pública e privada com os demais componentes *adHoc* no mesmo *site*. Isso é necessário para assegurar que toda nova instância do coordenador não seja reconhecida como um coordenador diferente. Essa troca precisa utilizar algum mecanismo de segurança para assegurar que as chaves não sejam obtidas e utilizadas de forma mal-intencionada;

- Avaliar o comportamento da grade em outros cenários, como redes sem fio, para determinar o impacto que essas podem causar no algoritmo de eleição;
- Avaliar as possibilidades dos componentes *adHoc* controlarem o número de *workers* disponíveis no *site* de acordo com a demanda de recursos;
- Explorar a possibilidade de colocar os membros ociosos para hibernar, como forma de economia de energia;
- O servidor *Openfire* utilizado no *Hybrid Ad Hoc Grid* utiliza as portas 5222, 5223 para comunicação com os clientes e 5269 para comunicação entre os servidores. Para isso é necessário liberar essas portas no *firewall*. Uma solução futura seria eliminar essas configurações, por exemplo, utilizando tunelamento, de forma a simplificar a instalação e resolver possíveis problemas de configuração.

Bibliografia

Abilene Network and Control Center. Disponível em: <http://noc.net.internet2.edu>. Acesso em 15 Jan. 2010.

Akogrimo. Disponível em: <http://www.mobilegrids.org>. Acesso em 15 Jan. 2010.

AMIN, Kaizar; LASZEWSKI, Gregor von; MIKLER, Armin R. Toward an architecture for ad hoc grids. 12th International Conference on Advanced Computing and Communications, 2004.

ANDRADE, Nazareno; BRASILEIRO, Francisco; CIRNE, Walfredo; MOWBRAY, Miranda. Automatic grid assembly by promoting collaboration in peer-to-peer grids. Journal of Parallel and Distributed Computing. 2007.

BERMAN, Fran; GEOFFREY, Fox; HEY, Tony. Grid Computing – Making the Global Infrastructure a Reality. John Wiley & Sons Ltd. Chichester, Inglaterra. 2003.

CHAKRAVARTI, Arjav J.; BAUMGARTNER, Gerald; LAURIA, Mario. The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network. IEEE Trans. Systems, Man, and Cybernetics Part A: Systems and Humans, 2005.

CHANG, Ernest; ROBERTS, Rosemary An improved algorithm for decentralized extrema-finding in circular configurations of processes. Communications of the ACM (ACM) 22 (5): 281–283. 1979.

CIRNE, Walfredo; BRASILEIRO, Francisco; ANDRADE, Nazareno, COSTA, Lauro; ANDRADE, Alisson; NOVAES, Reynaldo; MOWBRAY, Miranda. Labs of the world, unite!!!. Journal of Grid Computing, 4:225–246. 2006.

CIRNE, Walfredo; BRASILEIRO, Francisco; SAUVÉ, Jacques; ANDRADE, Nazareno; PARANHOS, Daniel; SANTOS-NETO, Elizeu; MEDEIROS, Raissa. Grid computing for Bag-of-Tasks applications in Proc. I3E2003, 2003.

Commune, Disponível em: <http://commune.lsd.ufcg.edu.br/>. Acesso em 16 Abr. 2010.

Creating a private OurGrid Community. Disponível em: <http://www.ourgrid.org/index.php?option=com_content&view=article&id=74&Itemid=1>. Acesso em 08 Abr. 2010

DUARTE, Alexandre; BRASILEIRO, Francisco; CIRNE, Walfredo; FILHO, José. Collaborative fault diagnosis in grids through automated tests (Proceeding AINA '06 Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 01 IEEE Computer Society Washington, DC, USA ©2006).

Extensible Messaging and Presence Protocol (XMPP). Disponível em: <http://xmpp.org/>. Acesso em: 12 Nov. 2009.

FRANÇA, Júnia Lessa; VASCONCELLOS, Ana Cristina de; colaboração: MAGALHÃES, Maria Helena de Andrade; BORGES, Stella Maris. Manual para normalização de publicações técnico-científicas. Belo Horizonte: Ed. UFMG, 2007.

GARCIA-MOLINA, Hector. Elections in a Distributed Computing System, IEEE Trans. On Computers, Vol C-31. 1982.

GEHRINF, Jörn; REINFELD, Alexander. Mars - a framework for minimizing the job execution time in a metacomputing environment. Proceedings of Future general Computer Systems, 1996.

Installing and configuring an OurGrid site. Disponível em: <http://www.ourgrid.org/index.php?option=com_content&view=article&id=52&Itemid=1>. Acesso em 08 Abr. 2010.

Installing and configuring Workers. Disponível em: <http://www.ourgrid.org/index.php?option=com_content&view=article&id=55&Itemid=1>. Acesso em 08 Abr. 2010.

InteliGrid. Disponível em: <http://www.inteligrid.com>. Acesso em 15 Jan. 2010.

JAIN, Raj. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. New York: John Wiley & Sons, 1991.

KURDI, Heba; LI, Maozhen; AL-RAWESHIDY, Hamed. A Classification of Emerging and Traditional Grid Systems. IEEE distributed systems online, 9(3):1–1, 2008.

LEHMAN, Tobin J.; KAUFMAN, James H. OptimalGrid: Middleware for Automatic Deployment of Distributed FEM Problems on an Internet-Based Computing Grid. Fifth IEEE International Conference on Cluster Computing (CLUSTER'03), 2003.

MINOLI, Daniel. A Networking Approach to Grid Computing. Published by John Wiley & Sons, Inc., Hoboken, New Jersey. 2005.

MyGrid. Disponível em: <http://www.mygrid.org.uk>. Acesso em 12 Dez. 2010.

Openfire. Disponível em: <http://www.igniterealtime.org/projects/openfire>. Acesso em 08 Abr. 2010.

OurGrid. Disponível em: <http://www.ourgrid.org/>. Acesso em: 04 Dez. 2009.

Remote Method Invocation (RMI). Disponível em: <http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>. Acesso em: 11 Nov. 2009.

SAINT-ANDRE, Peter. Open-source XMPP server comparison chart. Disponível em: <http://www.saint-andre.com/jabber/jsc/>. Acesso em 01 Set. 2010.

SAINT-ANDRE, Peter; SMITH, Kevin; TRONÇOM, Remko. XMPP: The Definitive Guide. O'Reilly Media, 2009.

SAJJAD, Ali; JAMEEL, Hassan; KALIM, Umar; HAN, Sang Man; LEE, Young-Ko; LEE, Sungyoun. AutoMAGI—An Autonomic Middleware for Enabling Mobile Access to Grid Infrastructure. Proc. Joint Int'l Conf. Autonomic and Autonomous Systems and Int'l Conf. Networking and Services, 2005.

The Condor Project, Disponível em: <http://www.cs.wisc.edu/condor>. Acesso em 07 Dez. 2010.

The Globus Project Web Site, Disponível em: <http://www.globus.org>. Acesso em 07 Dez. 2010.

TIBÚRCIO, Pablo Gustavo Soares. Ad Hoc Grid: Uma Grade Computacional entre-pares auto-organizável. 2009. 70 f. Dissertação (Mestrado em Informática) – Curso de Pós-Graduação em Informática, Universidade Federal de Campina Grande, Campina Grande, 2009.

APÊNDICE A – Instalação do *Hybrid Ad Hoc Grid*

O *Hybrid Ad Hoc Grid* pode ser obtido em: http://sourceforge.net/projects/hybridadhocgrid.

Pré-Requisitos para Instalação

Antes de iniciar a instalação, verifique se a máquina possui Java *Runtime Environment* (JRE) versão 6 ou superior instalado. Adicionalmente, verifique se o diretório de instalação da JRE encontra-se no *path* do sistema. Finalmente, crie uma variável de ambiente denominada JAVA_HOME que referencie o local de instalação da JRE.

Caso seja necessário, libere as portas 5222, 5223, 5269 e 6777 no *firewall*. As três primeiras são utilizadas pelo servidor XMPP e a última é utilizado pelo componente *adHoc* apenas no rede local.

O *Hybrid Ad Hoc Grid* é distribuído em um arquivo compactado contendo todos os componentes da grade. Realize a extração para um diretório de sua escolha.

Caso esteja sendo criada uma comunidade particular, siga os procedimentos de instalação do *discovery service* que constam no APÊNDICE D - Criação de uma comunidade particular. Em caso contrário, siga os procedimentos a seguir para continuar com a instalação.

Instalação do *AdHoc*

Em cada membro da grade, deve ser instalado um componente *adHoc* que deve ser iniciado juntamente com o sistema operacional de cada máquina. Antes de iniciá-lo, edite o arquivo peer.properties (APÊNDICE B - Peer.properties) adicionando o endereço do *discovery service*. Para inicializar o componente apenas execute o arquivo adhoc (Linux) ou o arquivo adhoc.bat (Windows). Não é necessária nenhuma configuração adicional para esse componente.

Instalação do *Worker* e *Broker*

Antes de iniciar o *worker* ou o *broker* é necessário verificar se a máquina possui um *adHoc* instalado e em execução. Em caso afirmativo, execute o arquivo broker (Linux) ou broker.bat (Windows) para iniciar o *broker* ou execute o arquivo worker (Linux) ou worker.bat (Windows) para iniciar o *worker*. Não são necessárias configurações dos componentes.

APÊNDICE B - Peer.properties

Abaixo é apresentado o conteúdo do arquivo de configuração do *peer*: peer.properties. Esse arquivo encontra-se no diretório de instalação do *Hybrid Ad Hoc Grid*. É recomendável que a linha 28 seja alterada para o endereço do *discovery service* previamente instalado para a grade.

```
1  commune.privatekey=
2  peer.accounting.cpuunit=1
3  peer.description=peer ad hoc
4  commune.certification.file.mycertificatefilepath=certification/mycertificate/m
   ycertificate.cer
5  commune.publickey=
6  peer.longitude=0
7  commune.fd.wan.detectiontime=300
8  peer.usevomsauth=no
9  commune.fd.localhost.heartbeatdelay=5
10 commune.fd.lan.detectiontime=180
11 commune.xmpp.username=micro05
12 confdir=/home/rodrigol/workspace/hybridadhocgrid
13 commune.filetransfer.timeout=600
14 commune.fd.localhost.detectiontime=60
15 peer.joincommunity=yes
16 commune.fd.wan.heartbeatdelay=60
17 peer.latitude=0
18 peer.ds.update=60
19 commune.fd.lan.heartbeatdelay=15
20 commune.xmpp.servername=micro05
21 peer.interval.saving=1800000
22 peer.accounting.dataunit=1
23 commune.filetransfer.max.simultaneous=10
24 peer.repeatrequest=120
25 commune.xmpp.serversecureport=5223
26 peer.label=micro05
27 #Esse parâmetro deverá ser configurado com o endereço do discovery service
28 peer.ds.network=lsd-ds@xmpp.ourgrid.org
29 commune.xmpp.password=xmpp-password
30 peer.rankingfile=/home/rodrigol/workspace/hybridadhocgrid/rankings.dat
31 commune.usemonitor=no
32 peer.vomsurl=voms.eela.ufrj.br\:8443/voms/oper.vo.eu-eela.eu
33 commune.filetransfer.notify.progress=no
34 peer.default.sdf=./example.sdf
35 commune.xmpp.serverport=5222
36 peer.email=no e-mail available
```

APÊNDICE C - Adhoc.properties

Abaixo é apresentado o conteúdo do arquivo de configuração do *adhoc*: adho.properties. Esse arquivo encontra-se no diretório de instalação do *Hybrid Ad Hoc Grid*. Através desse arquivo é possível alterar o *timeout* para entrega de mensagens na grade e o endereço e porta *multicast*.

```
1  commune.privatekey=
   commune.certification.file.mycertificatefilepath=certification/mycertificate/mycertificate.cer
2  commune.publickey=
3  commune.fd.wan.detectiontime=300
4  adhoc.peerAdress=peer@micro05
5  commune.fd.localhost.heartbeatdelay=5
6  #Esse parâmetro deverá ser alterado de acordo com o tráfego da rede
7  adhoc.timeout=2000
8  commune.fd.lan.detectiontime=180
9  commune.xmpp.username=micro05adhoc
10 confdir=/home/rodrigol/workspace/hybridadhocgrid
11 commune.filetransfer.timeout=600
12 commune.fd.localhost.detectiontime=60
13 commune.fd.wan.heartbeatdelay=60
14 commune.fd.lan.heartbeatdelay=15
15 commune.xmpp.servername=micro05
16 commune.filetransfer.max.simultaneous=10
17 adhoc.multicast.addressGroup=239.0.0.10
18 commune.xmpp.serversecureport=5223
19 adhoc.multicast.port=6777
20 adhoc.sequencenumber=0
21 commune.xmpp.password=xmpp-password
22 commune.usemonitor=no
23 commune.filetransfer.notify.progress=no
24 commune.xmpp.serverport=5222
```

APÊNDICE D – Criação de uma Comunidade Privada

Para criar uma comunidade privada, é necessário instalar um *discovery service* na comunidade. A instalação deve ser realizada em um local conhecido e acessível por todos os membros. O *discovery service* é distribuído juntamente com o *Hybrid Ad Hoc Grid*. Verifique os pré-requisitos de instalação no APÊNDICE A - Instalação do *Hybrid Ad Hoc Grid*.

Após atender aos pré-requisitos, realize a instalação de um servidor XMPP em um endereço fixo, por exemplo, xmpp.seudominio.com. Em seguida, deve ser registrado um usuário nesse servidor.

Obtenha uma cópia do *Hybrid Ad Hoc Grid* e extraia seu conteúdo para um diretório de sua preferência. Em seguida edite o arquivo ds.properties (APÊNDICE E - ds.properties), adicionando o endereço do servidor XMPP e nome de usuário e senha criados na etapa anterior.

Para que a comunidade utilize esse *discovery service*, os arquivos peer.properties devem ter o endereço do *discovery service* alterado. Por exemplo, se o nome registrado no servidor XMPP for discovery e o endereço do mesmo for xmpp.seudominio.com, então o novo endereço será discovery@xmpp.seudominio.com.

APENDICE E - Ds.properties

```
1  #=====================================================
2  #
3  #        ds.properties        (Discovery Service configuration properties)
4  #
5  #=====================================================
6  # No specific properties are defined for the Discovery Service
7  #=====================================================
8  # Commune Properties
9  #
10 # In this section of the configuration file you'll define
11 # your settings for the Jabber server and user
12 #
13 #=====================================================
14 # Your Jabber user name
15 commune.xmpp.username=
16 # Your Jabber password
17 commune.xmpp.password=
18 # Address used by the Jabber Server (mandatory)
19 commune.xmpp.servername=
20 # Your private key
21 commune.privatekey=
22 # Your public key
23 commune.publickey=
```

APENDICE F – Fórmulas Utilizadas para a Obtenção do Intervalo de Confiança

$$Média\ \mu = E(x) = \sum_{i=1}^{n} pi \quad x_i$$

$$Variância\ \sigma^2 = Var(x) = E[(x-\mu)\quad^2] = \sum_{i=1}^{n} p_i(x_i-\mu)\quad^2$$

$$Desvio\ Padrão\ \sigma = \sqrt{\sigma^2}$$

$$Tamanho\ da\ amostra\ n$$

$$Para\ o\ nível\ de\ confiança\ 0.95,$$

$$Z_n = 1.960\ -\ Obtido\ da\ tabela\ distribuição\ normal\ (Jain, 1991)$$

Intervalo de confiança $= \mu \pm Z_n \frac{\sigma}{\sqrt{n}}$