FABIO PIOLA NAVARRO

A ESPECIFICAÇÃO DE UM MIDDLEWARE PARA DISPOSITIVOS MÓVEIS UTILIZANDO GRADES DE COMPUTADORES

FLORIANÓPOLIS
2006

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Fabio Piola Navarro

A ESPECIFICAÇÃO DE UM MIDDLEWARE PARA DISPOSITIVOS MÓVEIS UTILIZANDO GRADES DE COMPUTADORES

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Dr. CARLOS BECKER WESTPHALL

Co-Orientadores: MSc. FERNANDO KOCH e MSc. MARCOS ASSUNÇÃO

Florianópolis, Fevereiro de 2006

A ESPECIFICAÇÃO DE UM MIDDLEWARE PARA DISPOSITIVOS MÓVEIS UTILIZANDO GRADES DE COMPUTADORES

Fabio Piola Navarro

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Raul S. Wazlawick
Coordenador do PPGCC

Banca Examinadora

_____

Prof. Dr. Carlos Becker Westphall
Orientador

_____

Prof. Dr. Bruno R. Schulze

_____

Profa. Dra. Carla Merkle Westphall

_____

Prof. Dr. Mário A. Ribeiro Dantas

AGRADECIMENTOS

Agradeço a Deus por todas as oportunidades, pela saúde, por tudo!

Agradeço aos meus Pais e irmãos pela sincera ajuda dispensada, seja monetariamente, seja psicologicamente, meu muito obrigado pela oportunidade que me deram de me oferecer um caminho, uma profissão. Muito Obrigado!!!

Agradeço a minha Namorada Carolina, que prestou grande favor em me escutar nos devaneios de pesquisa, nas horas de certa dúvida no êxito, nos finais de semana sem passeio, nas férias que não podemos viajar...enfim Muito Obrigado por ficar do meu lado também nestas horas

Meu sincero Agradecimento aos Mestres, no sentido de ensino e não de titulação, Professor Doutor Carlos Becker Westphall, Doutorando Fernando Koch e Doutorando Marcos Assunção pela paciência, pelos ensinamentos, pelas conversas e esclarecimentos. Professores muito obrigado pela formação científica e profissional que me prestaram!!!

Agradeço também aos amigos que fiz durante o mestrado, Westphall, Koch, Marcos, Schulter, Rafael, Júlio, Felipe e Professores das disciplinas que cursei. Muito Obrigado!!!

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API: Application Programming Interface

CPU: Central Processing Unit

GIS: Geographic Information System

GRAM: Globus Resource Allocation Manager

HTML: Hyper-Text Markup Language

HTTP: Hyper-Text Transfer Protocol

J2ME: Java 2 Micro Edition

J2SE: Java 2 Standard Edition

P2P: Peer to Peer

PDA: Personal Digital Assistant

PC: Personal Computer

W3C: World Wide Web Consortium

XML: Extensible Markup Language

MEMS: Micro EletroMechanical Systems

RESUMO

Grades de computadores (grids) têm como característica principal prover uma distribuição de processamento e fornecer integração entre os dispositivos da grade. A computação móvel tem por objetivo o fornecimento de serviços móveis, isto é, entrega de informação a qualquer hora e em qualquer lugar, através de dispositivos móveis. Baseado neste contexto, esta dissertação motivada pela capacidade das grades de computadores de homogeneização de dispositivos e alto poder de processamento apresenta uma arquitetura para integrar os serviços da computação móvel às características da computação em grade através de um middleware.

Este middleware é responsável pela transparência de acesso aos recursos, dispositivos móveis e não móveis, por parte dos usuários. Com isto pretende-se resolver alguns problemas relacionados a computação móvel, como falta de poder de processamento, baixa capacidade de memória entre outros descritos ao longo da dissertação pelo uso de grades de computadores.

Esta dissertação contribui com uma nova abordagem para a resolução de problemas móveis ao integrar computação em grade com computação móvel.

***Palavras-chaves**: Computação em grid, grids, grade de computadores, computação móvel, redes de sensores, middlewares.*

ABSTRACT

Computational grids are characterized mainly as providers of distributed processing and integration of its devices. Mobile computing aims for providing mobile services, that is, delivering information anytime anywhere through mobile devices. In this context, this work presents an architecture for integrating mobile computing services with grid computing characteristics with the help of a middleware which will homogenize the grid's devices and will provide a uniform environment for the device interactions and application development.

***Keywords**: Grid Computing, Mobile Computing, Sensor Networks, Middleware.*

1. INTRODUÇÃO

> *Integrating sensor networks and grid computing in sensor-grid computing is like giving 'eyes' and 'ears' to the computational grid.*
>
> Chen-Khong Tham and Rajkumar Buyya

Este capítulo apresenta uma breve descrição de um cenário que possibilita vislumbrar as tecnologias do contexto da computação móvel e de grades de computadores. Posteriormente são apresentados os objetivos e justificativa deste trabalho, e por fim é descrito a estrutura desta dissertação.

1.1 MOTIVAÇÃO E CONTEXTUALIZAÇÃO

1.1.1 MOTIVAÇÃO

A tecnologia de grades de computadores vem crescendo rapidamente, por sua inerente condição em agregar dispositivos e fazer com que estes sejam transparentes ao usuário, esta pode prover uma miríade de capacidades, como armazenar dados, compartilhar recursos, alto poder de processamento entre outras. Estes dispositivos podem ser utilizados sem uma preocupação ou conhecimento prévio por parte do usuário. Uma comparação clássica referente ao uso de grades de computadores é com o uso da energia elétrica atualmente, onde conectamos diversos aparelhos e estes utilizam do serviço da rede elétrica, sem nos darmos conta de como este serviço está sendo provido.

No entanto, o poder de uma grade de computadores não está somente em prover capacidade de armazenamento, alto poder de processamento ou compartilhamento de recursos, mas sim na homogeneização ou uniformização de ambientes. Desta forma, o uso de grades de computadores traz alguns benefícios, pois quando utilizada em um ambiente no qual cada dispositivo precise interagir com os demais e todos terem acesso

a informações corporativas, grades de computadores se torna uma tecnologia que habilita estas características.

Sensores, segundo [THAM 2005] são sistemas micro-eletromecânicos (MEMS) que podem ser atuadores como *Buzzers,* coletores/medidores do ambiente. O conjunto de vários sensores (nodos) forma uma rede de sensores (*Network Sensors*). Estes sensores monitoram o ambiente e possuem a capacidade de coletar e enviar informações para um determinado lugar. A solução de redes de sensores deve resolver o problema de coleta e transmissão de dados, enquanto o acesso de dados pelos usuários do sistema deve ser tratado por uma solução de integração de dispositivos móveis em uma grade (grid) de comunicação e integração dos equipamentos.

A computação móvel ou computação ubíqua foi citada primariamente por Mark Weiser em 1991 quando este, em seu artigo "The Computer for the 21st Century" [WEISER 1991] explana sobre uma computação do futuro na qual não seria necessário uso de fios e o computador seria intrínseco ao meio. Trazendo esses conceitos para atualidade vemos que Weiser tinha razão, pois o uso de sensores e dispositivos móveis de comunicação como celulares ou computadores de mão (*handhelds*) possibilitam o acesso a informações a qualquer hora e em qualquer lugar, conceitos básicos da computação móvel.

Um middleware é uma camada de software que visa intermediar outras duas camadas de software, ou prover uma interface ou até mesmo um framework que possibilite interações com outros aplicativos. O Globus [FOSTER 2005] é um exemplo de um middleware que possibilita a criação de aplicações e sistemas para grids.

A integração dos elementos e tecnologias acima descritos forma a base para o desenvolvimento desta pesquisa, já que, a proposta deste trabalho é utilizar um middleware para integração e gerenciamento dos dispositivos, sejam móveis ou estáticos como os sensores. Particularmente neste trabalho aplicaremos a arquitetura em um ambiente hospitalar, no qual os sensores estão ligados a um paciente, e estes sensores enviam dados como taxa de oxigênio, taxa de glicose, batimentos cardíacos

entre outras informações vitais para uma central que faz o armazenamento e possível publicação destas informações.

Acreditamos que o uso de grades de computadores (*grid computing*) possa prover esta infra-estrutura, fornecendo os serviços de comunicação de dados, diretório e distribuição de serviços, segurança, autenticação, descoberta de recursos e as necessárias interfaces de programação. A vantagem do uso de grades de computadores está no re-uso das especificações e recursos providos por essas arquiteturas que facilitam o trabalho do desenvolvedor, fornecendo a orientação para a implementação desses serviços.

1.2 OBJETIVO GERAL

O objetivo geral deste trabalho é propor uma arquitetura de um middleware para grade de dispositivos móveis através da integração das tecnologias de grades de computadores (*grid computing*) e computação móvel (*mobile computing*).

1.3 OBJETIVOS ESPECÍFICOS

No intuito de alcançar o objetivo geral citado acima, são definidos os seguintes objetivos específicos:

- Evidenciar o conceito de grades de dispositivos móveis e seu estado da arte.
- Estudo das tecnologias relacionadas tais como: redes de sensores, grades de computadores, middlewares e computação móvel.
- A partir deste conhecimento, elaborar a arquitetura do middleware que fará a integração das tecnologias de grades de computadores e computação móvel.
- Validar a arquitetura através de implementações e simulações.
- Analisar as vantagens e desvantagens da arquitetura proposta, comparando-a com os trabalhos da área.
- Analisar as limitações e trabalhos futuros que podem utilizar a arquitetura proposta.

1.4 JUSTIFICATIVA

Alguns ambientes necessitam de certo nível de automação para resolução de problemas. Baseado nesta premissa e tomando como cenário problema um ambiente hospitalar onde médicos possam ter acesso aos dados dos pacientes, que são enviados por sensores.

Este acesso deve se dar por meio de dispositivos móveis para garantir mobilidade e entrega de informação a qualquer hora e em qualquer lugar, este ambiente é característico por necessitar de homogeneização dos dispositivos, gerenciamento dos dispositivos e principalmente entrega de serviços móveis.

Contudo, o desenvolvimento de serviços móveis envolve dois domínios de problemas: (i) infra-estrutura computacional para a implementação dos serviços, ou seja, estrutura de hardware e software que facilite a conexão dos dispositivos móveis, tais como protocolos para transporte de dados e segurança, e serviços de base como identificação da localização dos dispositivos, tolerância à falhas e outros [HENRICKSEN 2001]; (ii) implementação do serviço móvel em si, onde o desafio é integrar e fazer uso desta infra-estrutura computacional e criar uma solução que atenda às necessidades dos usuários. Como descrito em [SATYANARAYANAN 2001], a computação móvel impõe uma complexidade adicional no desenvolvimento desta infra-estrutura em virtude dos ambientes dinâmicos, mobilidade, limitações de recursos computacionais, latência e instabilidades na transferência de dados, limitação do suprimento de energia, e limitações das interfaces de entrada e saída.

Vários estudos vêm sendo realizados no intuito de diminuir esta complexidade e solucionar alguns destes problemas. Por exemplo, temos percebido avanços nas áreas de (a) comunicação de dados, onde redes de transferência de dados mais rápidas e confiáveis para dispositivos móveis têm sido apresentadas [USKELA 2003]; (b) softwares destinados a estes recursos têm obtido melhoras em virtude do aprimoramento do poder computacional destes equipamentos [DAS 2003], e; (c) engenharia de software, com a criação de metodologias de desenvolvimento e aplicações que adaptam

seu comportamento diante das limitações dos recursos [GUIGERE 2001]. Porém, ainda é difícil reunir as tecnologias propostas e ter uma solução completa para o desenvolvimento de serviços móveis. Isto tem levado ao uso de soluções ad hoc específicas para cada cenário sendo tratado.

Desta forma, por não haver uma infra-estrutura básica para o desenvolvimento de serviços móveis, esta falta de re-usabilidade e homogeneização acarreta custos extras no desenvolvimento de tais aplicações. Portanto, a questão que se está tentando responder neste trabalho é:

> ***Como prover uma solução integrada e homogênea para um ambiente de computação móvel que permita a comunicação, integração e gerenciamento das aplicações sendo executadas nesses dispositivos?***

Além disso, como garantir a re-usabilidade desta solução para que não seja necessário implementar a infra-estrutura novamente, sempre que o desenvolvimento de um serviço móvel seja necessário.

No intuito de responder a esta pergunta, este trabalho apresenta a integração de grades de computadores (*grid computing*) e computação móvel (*mobile computing*), através da criação de um middleware grid para o desenvolvimento e gerenciamento de serviços móveis. O grid middleware descrito neste trabalho oferece serviços como comunicação de dados, diretório de serviços além de descoberta e segurança, desta forma provendo ao desenvolvedor um conjunto de recursos reutilizáveis e homogêneos.

1.5 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está organizado em 6 capítulos, para mostrar a pesquisa e possíveis contribuições deste trabalho, e estão assim organizados:

- Neste primeiro capítulo, foi apresentada uma contextualização para situar o leitor sobre o tema da pesquisa, posteriormente tem-se os objetivos

gerais e específicos e então a justificativa para proporcionar ao leitor o escopo desta pesquisa.

- No capítulo dois são apresentados os conceitos de grades (*grids*) e computação móvel, bem como conceitos de redes de sensores e middlewares, este capitulo fornecerá a base para a compreensão dos capítulos posteriores.
- No terceiro capítulo são abordados os conceitos de grades de dispositivos móveis, cenários de aplicações e os trabalhos na área.
- No capítulo quatro é apresentada a arquitetura proposta, além dos métodos propostos para utilização da arquitetura.
- O quinto capítulo apresenta os resultados e conclusões baseadas em casos de uso.
- O sexto capítulo descreve as considerações finais e trabalhos futuros.

2. CONCEITOS DE GRIDS, COMPUTAÇÃO MÓVEL, REDES DE SENSORES E MIDDLEWARE

2.1 INTRODUÇÃO

Neste capítulo serão apresentados, de forma mais abrangente, os principais conceitos das tecnologias relacionadas ao desenvolvimento do middleware proposto para grade de dispositivos móveis.

2.2 CONCEITOS SOBRE GRADES DE COMPUTADORES

A tecnologia de Grades de computadores ou Grids (pode ser encontrado neste trabalho a palavra Grids ou *grid Computing*, em vez de Grades de computadores por estas também possuírem grande difusão no meio). Segundo [ASSUNÇÃO 2004] começou a ser difundida no final da década de 90 com a descrição de uma arquitetura que possibilita um conjunto de recursos geograficamente distribuídos serem utilizados para processar grandes volumes de dados.

Em 1999, Foster e Kesselman [FOSTER 1999] definiram Grids como sendo "*uma infraestrutura de hardware e software que fornece um acesso pervasivo, consistente, acessível e confiável a capacidades computacionais de alto desempenho*".

Em um trabalho mais recente Foster em seu artigo "The anatomy of the grid " [FOSTER 2001] define Grid como sendo "*compartilhamento coordenado de recursos e solucionador de problemas em instituições ou organizações virtuais dinâmicas*"

Em 2002 Foster [FOSTER 2002] estende o conceito de Grid ao descrever três pontos importantes e enfatiza que Grid é um sistema que:

- Coordena recursos que não estão sujeitos a um controle centralizado, ou seja, recursos em diferentes unidades administrativas são coordenados e integrados por meio de uma Grid.
- Usa de protocolos e interfaces de padrão abertos e para propósito geral, ou seja, uma Grid é construída e gerida por protocolos e interfaces de propósito geral que atendem a questões fundamentais como autenticação, autorização e descobrimento e acesso aos recursos.
- Entrega qualidade de serviço não triviais, ou seja, uma Grid permite por meio de seus recursos combinados entregar diferentes qualidades de serviço, como tempo de resposta, disponibilidade, banda passante, segurança entre outros.

Baseado nesta evolução das definições de *grid computing*, presentemente Grid possui definições bem próximas às anteriores, sendo uma delas, "*Grid Computing* habilita organizações a trocar e compartilhar informações e recursos computacionais entre departamentos e organizações de forma segura e eficiente" [GRIDFORUM 2005].

No entanto, grades de computadores possui uma conceito intrínseco que permeia nosso trabalho, pois além de possibilitar troca de informações e compartilhamento de recursos, grades de computadores possui uma característica essencial que é a homogeneização de dispositivos, ou seja, todos os componentes da grade de computadores são nodos, seja um dispositivo móvel como um PDA, seja um dispositivo de armazenamento (*storage device)* todos são nodos da grade e a comunicação entre estes nodos e o compartilhamento de recursos torna o uso da tecnologia de grades de computadores fundamental em nosso trabalho.

2.2.1 GRADES DE COMPUTADORES – SERVIÇOS E PADRONIZAÇÃO DE NOMES

Em seu recente artigo nomeado "*The Different Faces of IT as Service* " Ian Foster e Steven Tuecke [FOSTER 2005], descrevem que estamos em uma era parecida com o início da Internet, onde não havia padronização de nomenclaturas. É realizado até uma comparação com uma fábula dos homens cegos que encontram um elefante pela primeira vez e cada um dá uma descrição da parte que encontrou do elefante. Esta comparação é feita para apresentar essa confusão de termos como Grids, computação sobre demanda, virtualização, arquitetura orientada a serviços entre outros que tentam representar essa nova era da computação distribuída.

No entanto, o termo Grid é o que compreende melhor esse novo paradigma de serviços em TI (Tecnologia da Informação), que tende a substituir a verticalização, paradigma onde as aplicações eram suportadas por vários servidores e estes quando uma nova aplicação necessitava de mais poder de processamento seria então necessárias novas aquisições, fazendo uma alusão a adicionar, colocar em cima, mais recursos ou seja, novos servidores para as novas aplicações.

Atualmente tem-se um novo paradigma, a integração horizontal dos recursos, onde os recursos de hardware, como os servidores entre outros, são integrados e gerenciados por meio de um sistema Grid que faz alocação de recursos para determinada aplicação, gerenciamento desta aplicação e possui interfaces padronizadas para alocação de outras aplicações, gerando um menor custo e uma maior otimização dos recursos disponíveis.

O termo Grid além destas características acima descritas ainda suplanta termos como "o*n-demand*", um termo muito usado para denotar sistemas e tecnologias que permitem usuários ou aplicações adquirirem recursos adicionais para satisfazer mudanças de requisitos. E termos como "*Computing Utility*" que se refere a uma separação entre provedores de serviços e consumidores e uma capacidade para negociar níveis de qualidade de serviços, como ilustrado na figura 2.2.

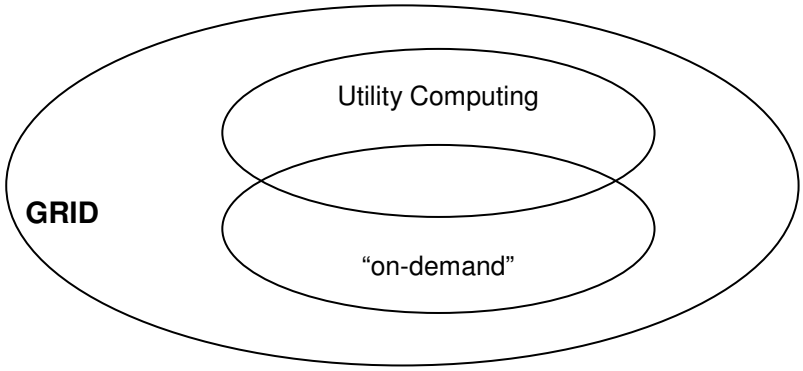

Figura 2.2. Visão da relação entre Grid, Utility e on-demand Computing – adaptado de [FOSTER 2005]

2.2.2 PROJETOS GRID

Nesta subseção serão apresentados alguns dos projetos pioneiros utilizando a tecnologia de grids seguidos dos projetos atuais em grids.

Pesquisas contra o câncer, o anthrax entre outras doenças utilizam do poder computacional dos PC (*Personal Computer*) para pesquisar em milhões de combinações de proteínas genéticas aquela que seria a cura contra tais doenças. Este sistema que utiliza processamento dos PC e outros dispositivos é controlado e gerenciado por uma Grid, que faz o controle e alocação dos recursos. Além de projetos relacionados à pesquisa com medicamentos, existem pesquisas em outras direções como o projeto SETI@HOME [SETI 2005] que tem a finalidade de procurar vida extra-terrestre analisando informações processadas em um computador de um usuário comum e então enviadas à central na forma de arquivos para serem analisados.

Estes são projetos pioneiros na computação grid, no entanto, na próxima seção serão apresentados os projetos atuais em grid de computadores.

2.2.3 GRIDS – PROJETOS E ESTADO DA ARTE

Atualmente existem vários projetos envolvendo grids, seja na área cientifica, comercial ou saúde, muitas iniciativas estão sendo desenvolvidas. Segue uma tabela envolvendo os principais projetos grids baseado no Fórum Global Grid [GGF 2005]:

Tabela 2.1 - Projetos atuais envolvendo grids Fonte: [GGF 2005]

ASTROGRID	http://www.astrogrid.org/
BIOGRID	http://www.biogrid.jp/
BIOINFORMATICS	http://www.bbsrc.ac.uk/science/initiatives/bep.html
BIRN	http://www.nbirn.net/
CONDOR	http://www.cs.wisc.edu/condor/
CROSSGRID	http://www.crossgrid.org/
DAMIEN	http://www.hlrs.de/organization/pds/projects/damien/
DATAGRID	http://datatag.web.cern.ch/datatag/
DATATAG	http://datatag.web.cern.ch/datatag/
DOE SciDAC	http://www.osti.gov/scidac/
DOE SCIENCE GRID	http://doesciencegrid.org/
Earth System Grid II	http://www.earthsystemgrid.org/
ECOGRID	http://www.buyya.com/ecogrid
EGEE	http://public.eu-egee.org/
ESnet	http://www.es.net/
EUROGRID	http://www.eurogrid.org/
FUSIONGRID	http://www.fusiongrid.org/
GÉANT	http://www.dante.net/geant/
GLOBUS	http://www.globus.org/
GRACE	http://www.grace-ist.org/
GRIDBUS	http://www.gridbus.org/
GriDis	http://web.datagrid.cnr.it/GriDis/jsp/index.jsp
GRIDPP	http://www.gridpp.ac.uk/
GRIP	http://www.grid-interoperability.org/
GriPhyN	http://www.griphyn.org/index.php
Inca	http://inca.sdsc.edu/
IPG	http://www.ipg.nasa.gov/
jvDgl	http://www.ivdgl.org/
LEAD	http://lead.ou.edu/
LEGION	http://www.cs.virginia.edu/~legion/
MCNC	http://www.mcnc.org/gcns2/
myGrid	http://www.mygrid.org.uk/
NAREGI	http://www.naregi.org/
National Grid Service (UK)	http://www.ngs.ac.uk/
NEES grid	http://www.neesgrid.org/
NextGrid	http://www.nextgrid.org/
NORDUGRID	http://www.nordugrid.org/
nsf middleware initiative	http://www.nsf-middleware.org/
Open Science Grid Consortium	http://www.opensciencegrid.org/
PPDG	http://www.ppdg.net/
RealityGrid	http://www.realitygrid.org/
SimDat	http://www.simdat.org/
TERAGRID	http://www.teragrid.org/

UNICORE Plus	http://www.fz-juelich.de/unicoreplus/index.html
WestGrid	http://www.westgrid.ca/home.html

Segundo o site do Fórum Global Grid [GGF 2005] existem várias áreas onde a tecnologia de grids está sendo aplicada, entre elas pode-se citar, áreas de infra-estrutura computacional, gerenciamento, segurança, de pesquisas nas áreas biológica, saúde, simulações de mercado entre outras.

Entretanto dentro destas áreas, a área de inovação tecnológica tem relevância fundamental neste trabalho, pois nesta área estão novos projetos usando da tecnologia grid junto a computação móvel, destaque para o grupo de pesquisa ACE-RG (*Advanced Collaborative Enviroments – Research Group*) [ACE 2005] que possui domínios de pesquisa em computação ubíqua, ou mais precisamente colaboração entre dispositivos móveis.

2.3 CONCEITOS SOBRE COMPUTAÇÃO MÓVEL

Segundo [LITKE 2004] computação móvel é um termo genérico que incorpora aplicações para dispositivos de pequeno porte, portáveis com comunicação e computação sem fio. Estes dispositivos podem ser notebooks, com tecnologia de comunicação sem fio, smartphones, celulares e PDAs (*Personal Digital Assistants*).

Existem outras definições de computação móvel tais como, uma integração de dispositivos móveis e redes sem fio, ou ainda, uma combinação de computadores portáteis, modems e telefonia. Também existem outros termos que denotam a computação móvel como, computação ubíqua, computação pervasiva ou ainda computação nômade, com pequenas diferenças entre estes.

Como citado na introdução deste trabalho, a computação móvel surge por volta do inicio da década de 90 com Mark Weiser, que em sua frase “As tecnologias mais profundas são aquelas que desaparecem”, faz referência a um ambiente onde os seres humanos não tenham que entrar no mundo virtual para desempenhar uma atividade, mas

sim a tecnologia estar em todos os lugares e a todo momento, dando a idéia de desaparecimento da tecnologia, pois o ambiente das máquinas e dos humanos seria o mesmo.

O objetivo da computação móvel é prover informação, acesso e serviços a qualquer hora e em qualquer lugar, porém os meios físicos reais para se conseguir isso nem sempre são triviais, necessitando do desenvolvimento e criação de estruturas de comunicação e modificação das redes, sistemas operacionais e aplicações [GAI 2005].

O emprego da computação móvel implica em contornar algumas restrições tais como as limitações dos dispositivos móveis utilizados pelos usuários. Restrições como limite de armazenamento de dados, baixo poder de processamento, tempo de uso limitado pela capacidade das baterias devem ser consideradas ao desenvolver aplicações que tenham a característica de mobilidade.

2.4 CONCEITOS SOBRE REDES DE SENSORES

Sensores são compreendidos como um sistema micro-eletromecânico (MEMS) que possuem uma unidade central de processamento CPU (*central processing unit*) memória e um transmissor sem fio (*wireless transceiver*). Tais sensores podem monitorar um ambiente e até mesmo interagir com este ambiente propiciando aplicações nas áreas da saúde, tráfego, segurança, bélico entre outros.

Redes de sensores (THAM 2005) são coleções de sensores espalhados por um ambiente provendo a este uma maior visibilidade do mundo real para um sistema computacional.

Aplicações de redes de sensores possuem grande disseminação tanto no âmbito acadêmico quanto comercial. Isto é devido em grande parte pela capacidade de vincular o mundo real com o mundo virtual (computacional). Na área de desenvolvimento bélico já são vistos sistemas em que forças aéreas, terrestres e marítimas se intercomunicam e

têm visualização de posicionamento real das unidades através de uma rede de sensores, onde cada unidade possui um sensor que recebe e envia informações.

Uma outra área de aplicação é vista no monitoramento de ambientes, seja este ambiente interno ou externo. Um exemplo de aplicação para o ambiente externo está no monitoramento de vulcões que podem entrar em erupção, ou seja, segundo uma determinada medição verificada pelo sensor este pode transmitir informações continuas deste ambiente e por meio destas informações pode-se programar avisos sobre determinado cenário, ou seja, quando verificada certa medição tomar alguma decisão.

Um exemplo na área da saúde e este totalmente ligado ao nosso trabalho é o monitoramento de pacientes. Um paciente em um ambiente hospitalar tem constantemente todos os seus sinais vitais monitorados, sinais como nível de oxigênio, nível de gás carbônico, taxa de glicose, batimentos cardíacos entre outros precisam ser medidos ininterruptamente.

Estes sinais vitais são medidos por meio de sensores ligados ao paciente, e os sensores enviam geralmente direto para um monitor estes dados referentes aos sinais. Na arquitetura proposta estes dados são enviados para uma central de recebimento onde será efetuado o tratamento e refinamento destes dados.

2.5 CONCEITOS SOBRE MIDDLEWARE

Pelo site *ObjectWeb* [OBJWEB 2005] e com a permissão do autor do termo descrita no site, middleware é definido em um ambiente de computação distribuída como uma camada de software que faz a mediação entre um sistema operacional e uma aplicação em cada local do sistema. Ainda segundo citação acima existe uma variedade de sistemas onde a tecnologia middleware pode ser empregada, tais como, componentes e objetos distribuídos, comunicação orientada a mensagens e ainda aplicações com suporte móvel.

Segundo [CAMPBELL 1999] middleware é um software que é usado para mover informações de um programa para um ou mais programas em um ambiente

distribuído, ocultando do desenvolvedor as dependências de protocolos de comunicação, sistemas operacionais e plataformas diferentes.

Existem algumas áreas onde a utilização de middlewares é inerente. Sistemas legados muitas vezes, possuem interfaces especificas e o custo para mudança de interface para comunicação de novos sistemas torna-se um fator negativo. No entanto o uso de um middleware pode resolver este problema ao integrar o sistema legado aos novos sistemas com diferentes interfaces.

De acordo com [OBJWEB 2005] as principais funções de um middleware são, mascarar o ambiente distribuído, parecendo ao usuário ou ao desenvolvedor um sistema único; ocultar a heterogeneidade de dispositivos de hardware, de sistemas operacionais e de protocolos de comunicação; prover ao desenvolvedor interfaces padronizadas para que as aplicações desenvolvidas possam ser portáveis, reusáveis, possuam interoperação e sejam facilmente construídas.

No entanto, o conceito de middleware em nosso trabalho e no meio acadêmico e empresarial utiliza o termo middleware como um software que faz a interconectividade entre dois sistemas, seja em níveis de hardware ou software. Particularmente em nossa pesquisa o termo middleware faz referência ao software que faz a interconexão entre a aplicação sendo executada no dispositivo móvel do usuário e a grade de computadores ou a grid, abrangendo as características acima citadas.

2.6 CONSIDERAÇÕES E DISCUSSÃO DO CAPÍTULO

Este capítulo apresentou os conceitos essenciais referentes ao desenvolvimento do middleware para dispositivos móveis, tais como definições de grids ou grades de computadores, definições de computação móvel, de redes de sensores e de middlewares.

Com relação a grids foi realizado um levantamento de diferentes definições e autores oferecendo um histórico recente do conceito e apresentado um resumo da tentativa de padronização de nomes existente nos dias atuais.

A computação móvel foi abordada desde os primórdios com as definições de Mark Weiser [WEISER 1991], de forma a definir um panorama do passado e presente da mobilidade. Já em relação ao conceito de middleware foram mostradas as principais aplicações e funções atuais e a subseção de redes de sensores explicitou seu conceito e mostrou suas diferentes utilizações.

Estes conceitos são importantes para a compreensão dos capítulos posteriores, pois serão abordados de forma natural e apresentado ao leitor como se o mesmo tivesse o prévio conhecimento.

3. GRIDS DE DISPOSITIVOS MÓVEIS, ESTADO DA ARTE E TRABALHOS CORRELATOS

Neste capítulo será apresentada a evolução e os conceitos relativos à grids de dispositivos móveis e seu estado da arte, também serão descritos os principais trabalhos correlatos, fazendo assim uma comparação com este trabalho.

3.1. O QUE É O GRID DE DISPOSITIVOS MÓVEIS

Como descrito no capítulo anterior a computação grid possui várias áreas de aplicações, no entanto, inicialmente sua principal função foi a integração de recursos e o processamento de grandes volumes de dados. Entretanto, grids vem evoluindo e não está ligada somente a este tipo de grid, ou seja, de alto poder de processamento. As grids estão se tornando pervasivas e ubíquas devido ao avanço dos dispositivos móveis e das redes de sensores. Neste sentido, surge então um novo paradigma de computação grid, as grades de dispositivos móveis ou grids móveis ou ainda grids de dispositivos móveis.

Como descrito em [LITKE 2004] **grids de dispositivos móveis** é uma herança de grid computing com características adicionais da computação móvel, ou seja, a capacidade de suportar usuários móveis de um modo transparente, seguro e eficiente.

O aparecimento de grids móveis surge por volta do ano de 2002 quando da união de duas outras tecnologias, a computação móvel subárea dentro da computação

distribuída e fortemente ligada às redes de usuários p2p com conceitos de nós (nodos) que trabalham como cliente e como servidor ao mesmo tempo, e com a outra grande área também recente que é a área de grid computing, que como descrito acima, assumiu papel inicial de solução para problemas de processamento de grande volume de dados.

Na figura 3.1 pode-se ter idéia das tecnologias ligadas a grids móveis, ou grids de dispositivos móveis. Observa-se a coligação de outras áreas da ciência da computação, como *web services*, redes *ad-hoc*, redes p2p entre outras.

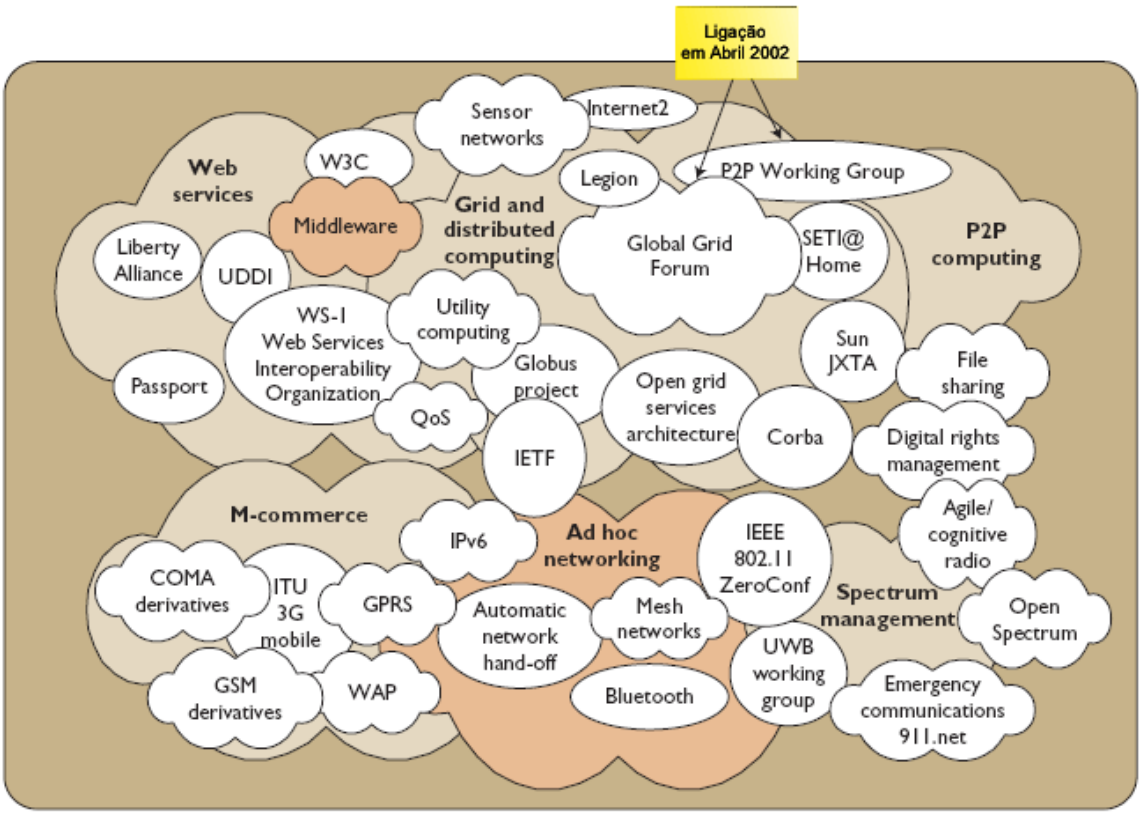

Figura 3.1. Tecnologias e Organizações associadas à grids de dispositivos móveis (*mobile grid computing*) adaptado de [MCKNIGHT 2004]

3.2 DISPOSITIVOS MÓVEIS

Dispositivos móveis são aparelhos eletrônicos portáteis como celulares, *handhelds*, PDA (*personal digital assistants*) , que permitem comunicação e ao mesmo tempo mobilidade para seus usuários. No nosso trabalho a utilização de dispositivos

móveis está vinculada à necessidade de acesso a informações pelos médicos e enfermeiras aos dados de determinado paciente, permitindo assim mobilidade ao corpo médico e acesso a qualquer hora e de qualquer lugar.

É importante considerar as limitações atuais impostas pelos dispositivos móveis como baixo poder de processamento, baixa autonomia da bateria e baixa capacidade de armazenamento. E baseado nestas limitações, as aplicações e interfaces para estes dispositivos devem merecer atenção, pois o desenvolvedor deve levar em conta as capacidades destes dispositivos.

A evolução dos dispositivos móveis proporcionou uma maior segurança, velocidade e versatilidade destes dispositivos, permitindo assim um maior desenvolvimento nesta área e levando à implementação de novas soluções baseadas na portabilidade, mobilidade e comunicação que tais dispositivos oferecem.

3.3 GRIDS DE INFORMAÇÃO

Segundo [FERREIRA 2004] grid de informação é uma estrutura que permite que usuários e aplicações possam acessar e trocar informações sem se importar como estas informações estão armazenadas, pois pela definição básica de grids, estas estão mais ligadas a poder de processamento.

A utilização de grids de informação está diretamente relacionada com este trabalho, pois a utilização de grids não está voltado objetivamente para o processamento de dados, mas sim em utilizar a grid como um fator de homogeneização dos dispositivos não transparecendo ao usuário a infra-estrutura de comunicação entre os nodos da grid, como ilustrado na figura 3.2.

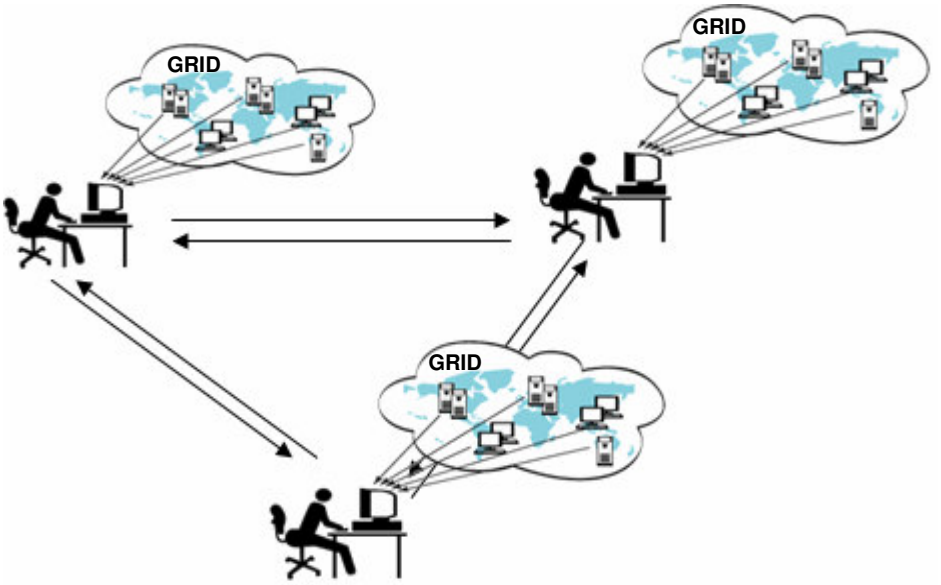

Figura 3.2. Visão da grid pelo usuário, transparência - adaptado de [FERREIRA 2004]

3.4 AMBIENTES DE APLICAÇÃO DE GRIDS DE DISPOSITIVOS MÓVEIS

Aplicações móveis típicas incluem essencialmente acesso remoto a dados, colaboração, mobilidade entre outras características. Dentre os principais serviços que uma grid de dispositivos móveis pode prover, um deles é o acesso a informações de qualquer lugar e a qualquer momento.

Recentemente a resolução de problemas de ambientes que envolvem mobilidade e entrega de informação a qualquer hora e em qualquer lugar não era de simples implementação. A utilização de uma única tecnologia, tal como J2ME (*Java 2 Micro Edition*) [J2ME 2005] não resolve completamente tais problemas, necessitando então da união de novas tecnologias tais como, redes sem fio para comunicação dos dispositivos, algum processamento quando necessário e homogeneização dos dispositivos.

Também não é considerado que grids de dispositivos móveis resolvem todos os problemas, mas a utilização da tecnologia de grids de dispositivos móveis agrega poder computacional quando necessário e mantém a comunicação e mobilidade entre os

usuários e principalmente, pela utilização de um middleware pode-se prover uma reutilização de códigos.

Neste contexto, algumas áreas de negócio, vislumbrando o potencial desta tecnologia, proporcionam o uso de dispositivos com capacidade de acesso e envio de informações a qualquer hora e em qualquer lugar. O mercado corporativo é um dos que mais investe nesse segmento por motivos como, obter total conhecimento de seus colaboradores, aumentar ainda mais a possibilidade de trabalho destes mesmo em horários fora do expediente, total interação entre colaboradores, padronização de comunicação, entre outros fatores. A seguir são citadas algumas áreas de aplicação de grids de dispositivos móveis.

3.4.1 GERENCIAMENTO DE FLUXO DE TRABALHO

Uma área que faz uso de grids de dispositivos móveis é o de gerenciamento de fluxo de trabalho ou (management workflow) a chamada *grid worflow* que pode ser entendida como um conjunto de tarefas a serem processadas em recursos distribuídos com uma ordem bem definida para atingir um objetivo específico [BUYYA 2005]. O gerenciamento de fluxo de trabalho baseado em um ambiente grid possui algumas particularidades e vantagens como [SPOONER 2004]:

- Habilidade para construir aplicações dinâmicas gerenciadas por recursos distribuídos.
- Utilização de recursos alocados em diferentes domínios aumentando *throughput* e redução de custos.
- Busca por domínios administrativos diferentes para encontrar capacidades especificas de processamento.

Estas capacidades aplicadas em um ambiente móvel pode aumentar ainda mais o poder de gerenciamento de fluxos de trabalhos ao prover habilidades para adaptar dinamicamente estruturas organizacionais através da capacidade de troca de recursos móveis e sensibilidade ao contexto.

3.4.2 MOBILE GIS

Sistemas de Informação Geográficos (*Geographic Information System-GIS*) são sistemas que utilizam do auxílio de tecnologias de hardware, software e geoprocessamento para facilitar a análise, gestão e representação do espaço e dos fenômenos que nele ocorrem.

Sistemas móveis de informação geográfica (*Mobile GIS*) é uma das áreas de aplicação de grids de dispositivos móveis, utilizando da capacidade de mobilidade dos dispositivos móveis aliado ao poder computacional de uma grid.

No artigo (*Analysis of Remotely Sensed Images on the Grid Platform from Mobile Handheld Devices – A Trial*) [YANGUANG 2004] é abordada uma utilização de grids de dispositivos móveis para análise de imagens obtidas em campo por PDA como, palmtops, smartphones e afins. Estas imagens são enviadas para serem analisadas por uma grade de computadores, esta grade é formada por 4 computadores e é utilizado o middleware CONDOR [CASTANO 2004] para gerenciar esta grade. Depois de analisadas as imagens são então enviadas novamente ao dispositivo móvel, porém agora já processadas.

3.4.3 GRIDS DE DISPOSITIVOS MÓVEIS COMO UM SISTEMA DE APOIO A DECISÃO

Em um recente artigo publicado [NAVARRO 2005] apresenta um middleware para grid de dispositivos móveis o qual oferece suporte para este ser aplicado em um sistema de apoio a decisão.

O cenário é o de um hospital no qual os médicos por meio de seus PDA's tem acesso a informações dos pacientes, estas informações por sua vez são passadas por meio de sensores para uma central grid que se encarrega de disponibilizar tais

informações. Este acesso pode ser feito por vários outros médicos que podem estar até mesmo fora do ambiente hospitalar, por exemplo, em outro país e por meio desta disponibilidade das informações e o acesso de outros médicos este pode ser definido como um sistema de apoio a decisão, pois inferências para um possível diagnóstico podem ser feitas por vários médicos de vários locais, auxiliando assim a tomada de uma decisão.

Nesta próxima subseção serão apresentados os principais trabalhos correlatos, evidenciando os middlewares mais difundidos na comunidade.

3.5 TRABALHOS CORRELATOS

Existem alguns trabalhos de grande relevância na área. [BHATTI 2005] mostra uma aplicação no campo da saúde, descrevendo uma arquitetura na qual, por meio de IDC (Internet Data Center) seja possível armazenar, gerenciar e disponibilizar informações multimídia de forma segura e eficiente de cada paciente para cada hospital e compartilhar estas informações de tal forma que outros hospitais possam ter acesso.

Neste trabalho encontramos dois aspectos afins ao nosso, primeiro, o uso de grids como tecnologia que homogeneiza e integra os dispositivos e, segundo, no aspecto de que esta arquitetura disponibiliza aos médicos, acesso às informações através de PDA, inclusive fora do ambiente do hospital (fator semelhante ao nosso trabalho, evidenciando a mobilidade).

Este trabalho complementa o nosso ao propiciar uma das propriedades da computação móvel, ou seja, sensibilidade ao contexto (*context awareness*) ao localizar os centros de tratamento mais próximos baseado no cadastro do sistema de saúde de cada paciente, e nosso trabalho fornece colaboração (*colaboration*) outra propriedade da computação móvel e distribuída.

Em [GAYNOR 2004] são apresentadas duas aplicações, uma na área da saúde e outra na área de cadeia de suprimentos (*supply chain*), onde o uso de grade de sensores automatiza e faz o gerenciamento destes ambientes. O sistema ligado à área da saúde apresenta uma aplicação que monitora os pacientes através de sensores que enviam as informações para uma central que pode enviar alarmes quando determinado nível é atingido. No outro ambiente é apresentada uma cadeia de suprimentos que pode ser monitorada por sensores e otimizar o gerenciamento dos pedidos e o armazenamento das mercadorias.

Ainda em [GAYNOR 2004] encontramos citação ao padrão OGSA (Open Grid Services Architecture), descrito no capítulo 2 subseção 2.2 Grids, o qual define um conjunto de interfaces para desenvolvimento de sistemas grid, padronizando e viabilizando compatibilidade. Este artigo é afim ao nosso trabalho por propor uma rede de sensores em uma grid de computadores, possibilitando gerenciamento centralizado, porém não oferece um middleware que faz agregação de recursos, segurança, gerenciamento de dispositivos, comunicação entre dispositivos como nosso middleware proposto.

Em [YANGUANG 2004] temos uma aplicação relacionada a análise de imagens. É utilizada uma grid de computadores baseada no projeto CONDOR [CASTANO 2004], que oferece processamento em larga escala, mecanismos de escalonamento de tarefas e gerenciamento dos recursos. Nesta aplicação o middleware tem o papel de gerenciar as tarefas enviadas ao pool de máquinas CONDOR (grid), que além de suportar envio e recebimento de tarefas de dispositivos móveis (semelhante ao nosso trabalho), também faz o processamento das imagens para serem enviadas aos dispositivos.

Este trabalho é o que mais se aproxima do nosso, por prover um middleware que faz a gerência do sistema grid por meio de comandos provindos dos dispositivos móveis. Porém, este trabalho não prove colaboração entre os dispositivos e nem sensibilidade ao contexto.

3.5.1 PRINCIPAIS MIDDLEWARES

A tecnologia de grid computing oferece o suporte para requisitos para a implementação de serviços móveis para suporte a decisão. Podemos encontrar vários pacotes para a implementação de serviços de grades de computadores. Nessa subseção estaremos apresentando aqueles de maior difusão na comunidade e discutindo seu suporte para a criação de serviços móveis.

Como descrito em [KOCH 2005], o desenvolvimento de serviços móveis para apoio a decisão necessita suportar: colaboração, interface com usuário, interface com elementos do ambiente (*context-awareness)* e um processo de inferência que permita o desenvolvimento de sistemas além do puramente reativo. Neste ambiente com recursos limitados, aplicações devem suportar as restrições de recursos computacionais, rede de comunicação intermitente e não confiáveis, limitação no fornecimento de energia, mobilidade, ambientes dinâmicos, interfaces com usuário e outros dispositivos reduzidas.

[SATYANARAYANAN 2001] propõe que essas limitações são inerentes ao ambiente e devem ser amenizadas, mas não suplantadas, por desenvolvimentos tecnológicos futuros. Baseado nos requisitos descritos anteriormente, serão avaliados neste artigo os middlewares mais difundidos na comunidade em termos de (α) suporte a colaboração; (β) suporte a sensibilidade ao contexto; (γ) suporte a alocação de recursos; (δ) suporte a ambientes dinâmicos; (ε) suporte a execução em dispositivos móveis.

O GLOBUS [FOSTER 2005] é um software de código aberto, desenvolvido pela *Globus Alliance*, que oferece um kit de ferramentas para desenvolver aplicações e sistemas de computação em grid. Sua arquitetura pode ser encontrada na figura 3.3 onde seu suporte para as características necessárias em aplicações para serviços móveis acima descritas é encontrado em (α) suporte a colaboração que por meio dos protocolos da camada de recursos (GRAM, GRIP, GridFTP) pode obter e receber informações e ainda controlar as tarefas, promovendo assim colaboração pela distribuição aos recursos. Também existe o (γ) suporte a alocação de recursos, provido pelo gerenciador de

recursos (GRAM - Globus Resource Allocation Manager) que fornece uma interface para envio e monitoramento de tarefas.

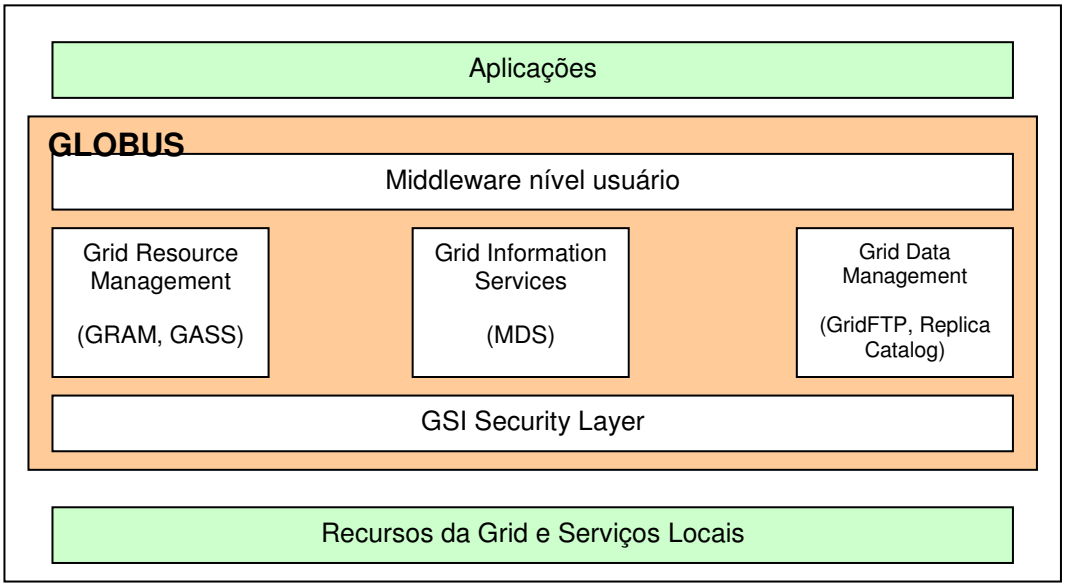

Figura 3.3 Arquitetura do middleware Globus, adaptado de [FERREIRA 2004]

O GRIDBUS [GRIDBUS 2005] do laboratório de pesquisa e desenvolvimento de software para computação em grade e sistemas distribuídos (GRIDS) da universidade de Melbourne na Austrália, é um pacote de código aberto utilizado para arquiteturas e ferramentas para implementação de grades de computadores para (eScience e eBusiness applications). Para isso, faz uso de diversos outros middlewares como: Globus, Unicore, Alchemi entre outros, como pode ser visto em sua arquitetura na figura 3.4.

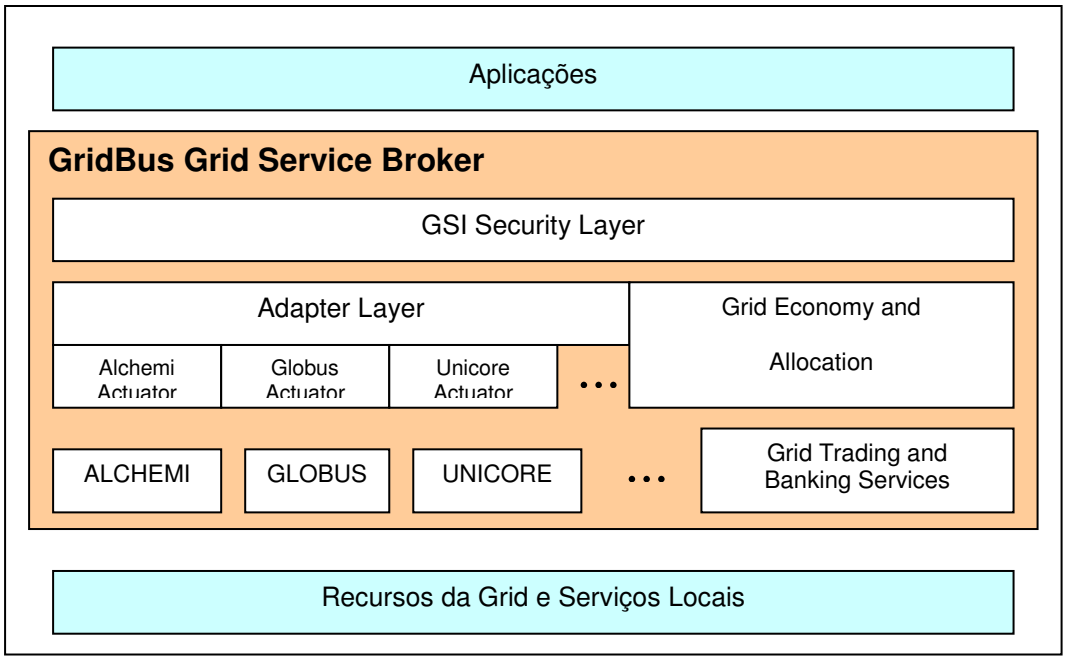

Figura 3.4 Arquitetura do middleware GridBus, adaptado de [FERREIRA 2004]

Seu suporte para as características necessárias em aplicações para serviços móveis acima descritas é encontrado em (α) suporte a colaboração, (γ) suporte a alocação de recursos e (δ) suporte a ambientes dinâmicos , providos por middlewares de baixo nível ou core middlewares, como os acima descritos e também usados no desenvolvimento de aplicações.

O Legion [LEGION 2005] é um middleware desenvolvido por um projeto da universidade da Virginia, definido como um meta-sistema baseado em objetos (recursos) com bilhões de hosts e trilhões de objetos vinculados por links de alta velocidade conectando redes, estações de trabalho e supercomputadores em um sistema que pode agregar diferentes arquiteturas, sistemas operacionais e localizações físicas.

O suporte provido pelo Legion às características inerentes a aplicações para serviços móveis, como as acima descritas, é visto em (α) suporte a colaboração, onde por meio do sistema de *binding* é possível colaboração através de tuplas como <LOID, LOA, timeout> que fornecem, endereçamento aos objetos (LOID) e gerenciamento

provido por (LOA) como pode ser visto por sua arquitetura na camada acima da infra-estrutura na figura 3.5 Existe também (γ) suporte a alocação de recursos, que é realizada pelo LOA (Legion Object Addresses) que incorpora um endereço físico como o IP e pode distribuir estes recursos como multicast ou comunicação entre grupos de objetos.

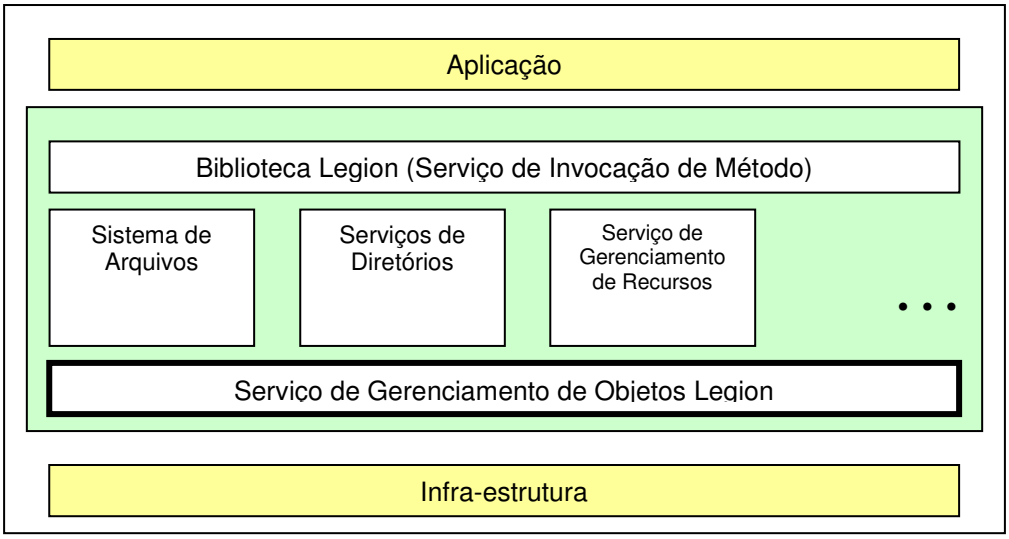

Figura 3.5 Arquitetura do middleware Legion, fonte: [LEGION 2005]

O UNICORE (UNIform Interface to COmputer REsources) [UNICORE 2005] é um middleware que integra os recursos da computação em grade por meio de uma interface gráfica desenvolvida na linguagem JAVA. Seu suporte às características para aplicações de serviços móveis, pode ser encontrado em (α) suporte a colaboração, provido pelos servidores UNICORE depois de autenticação do cliente e usuário. A colaboração é realizada pelos servidores que enviam os jobs (tarefas) a serem executadas para os Peer Unicore gateways, que executam e devolvem ao servidor. O suporte a (γ) distribuição de recursos, é feito pelo AJO (Abstract Job Object) que é uma classe/biblioteca que controla a comunicação, envio e recebimento dos jobs e faz a distribuição dos recursos.

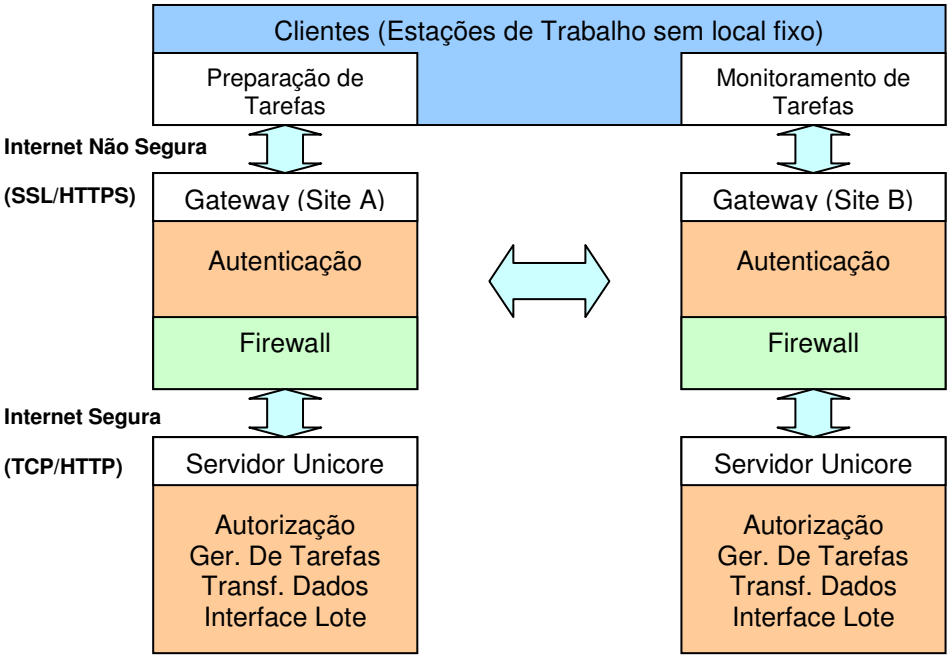

Figura 3.6 Arquitetura do Middleware Unicore – [UNICORE 2005]

Como pode ser visto na arquitetura do middleware Unicore na figura 3.6 o cliente pode acessar mesmo em um ambiente sem segurança, pois esta é fornecida pelo middleware Unicore quando é feito o envio de uma tarefa (*job*) esta mesmo sendo provinda de uma área sem segurança, ao entrar no servidor Unicore este requer autorização para processamento desta tarefa.

Tabela 3.1 Comparação dos principais middlewares em relação as características para aplicações em serviços móveis

	GLOBUS	GRIDBUS	LEGION	UNICORE
Suporte a colaboração **(α)**	Sim	Sim	Sim	Sim
Suporte a sensibilidade ao contexto **(β)**	Não	Não	Não	Não
Suporte a alocação de recursos **(γ)**	Sim (GRAM)	Sim	Sim (LOA)	Sim (AJO)
Suporte a ambientes dinâmicos **(δ)**	Não	Não	Não	Não

Suporte a execução em dispositivos móveis **(ε)**	Não	Sim	Não	Não

A Tabela 3.1 sumariza o suporte provido por esses pacotes, no aspecto relacionado ao desenvolvimento de serviços para computação móvel. Da análise dos trabalhos relacionados concluímos que os pacotes para desenvolvimento de *grades de computadores* ou grid computing existentes no campo não suprem as necessidades para a criação de serviços móveis, tais como suporte a colaboração, suporte a sensibilidade ao contexto, suporte a alocação de recursos, suporte a ambientes dinâmicos e suporte a execução em dispositivos móveis.

Sendo assim, identificamos a possibilidade de colaboração aos desenvolvimentos existentes na área de computação em grade ou grid computing para dispositivos móveis através da criação de um *Middleware para Dispositivos Móveis em um ambiente Grid* que forneça a funcionalidade necessária para suportar as características acima descritas, como S={α,β,γ,δ,ε}.

3.6 PROJETOS EM MIDDLEWARES PARA GRIDS DE DISPOSITIVOS MÓVEIS

3.6. 1 O PROJETO GRIDLAB

O projeto GridLab [GRIDLAB 2005] é um dos maiores projetos europeus de pesquisa relacionado com o desenvolvimento de ferramentas de aplicações e middlewares para ambientes grid. O GridLab possui uma série de aplicações orientadas a serviços Grid e ferramentas como gerenciamento de recursos, monitoramento, gerenciamento de dados, segurança entre outros. Tais serviços podem ser acessados por meio do GAT API (*Grid Application Toolkit*) que pode ser observado na figura 3.7 abaixo e que fornece uma estrutura para usuários finais que precisam desenvolver aplicações, baseado nesta estrutura os usuários desenvolvedores não necessitam de ter um conhecimento dos detalhes sobre o ambiente de execução. Ainda neste espaço chamado de espaço do usuário existe o Gridsphere Portal que permite o desenvolvimento de aplicações por meio de seu framework.

Abaixo está localizada a camada do middleware, que cobre uma gama de serviços grid que usuários, aplicações e desenvolvedores necessitam, tais como **GRMS** (*Grid Resource Management and Brokering Service*), **Data Access and Management** (*Grid Services for data management and access*), **GAS** (*Grid Authorization Service*), **iGrid** (*GridLab Information Services*), **Delphoi** (*Grid Network Monitoring & Performance Prediction Service*), **Mercury** (*Grid Monitoring infrastructure*), **Visualization** (*Grid Data and Visualization Services*), **Mobile Services** (*Grid Services supporting wireless technologies*).

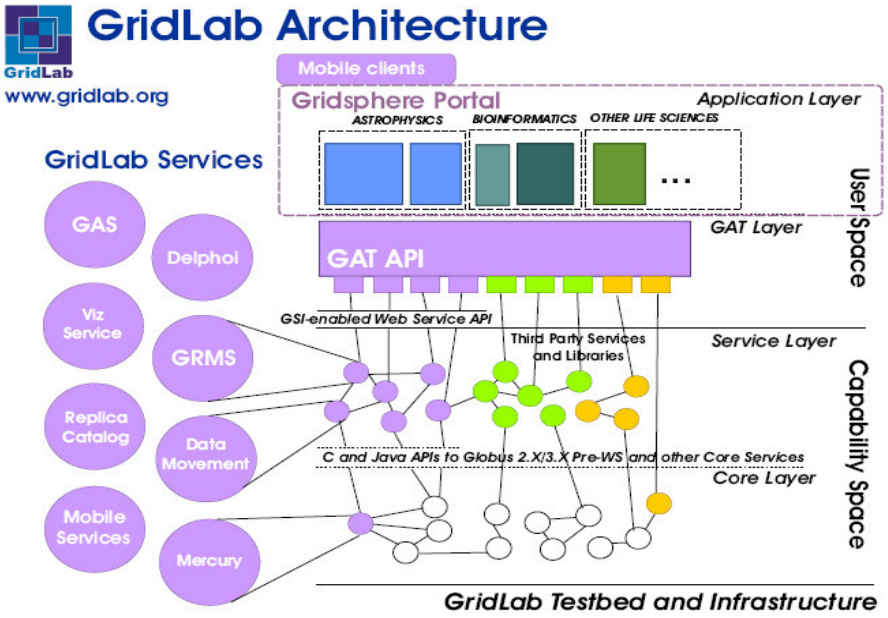

Figura 3.7 Arquitetura do GriLab, visão geral - fonte:[GRIDLAB 2005]

Dentre estes serviços, o que temos foco é o Grid Mobile Services, que possui serviços grid para dispositivos móveis, o qual iremos explorar mais a fundo.

3.6.2 O GRID MOBILE SERVICES DO PROJETO GRIDLAB

O projeto gridLab oferece um pacote de serviços para usuários de dispositivos móveis, neste pacote existem algumas ressalvas quanto aos recursos acessados por estes dispositivos tais como interfaces e conteúdos compatíveis com a banda do aparelho.

O objetivo do Grid Mobile Service é prover ao usuário da Grid a possibilidade de acessar suas aplicações e recursos de qualquer lugar e a qualquer hora pelo uso dos

dispositivos móveis. De acordo com o projeto GridLab estes dispositivos não serão incorporados à Grid como *Peers*, ou seja, como um nodo que tem funções de cliente e servidor ao mesmo tempo, hora servindo informações ou hora requisitando informações, mas sim como somente clientes, ou seja, com a função de requisitar informações. Isto porque, segundo [GRIDLAB 2005] tais dispositivos possuem sérias limitações e assim não poderiam fazer a função de servidores.

Na figura 3.8 temos a visão de como os dispositivos móveis conseguem interagir com com os serviços Grid. Esta possibilidade de comunicação é dada por meio do Gateway de comunicação, este gateway mapeia as características do cliente móvel para os plug-ins (softwares que extendem as funções de uma determinada aplicação) do gateway que então se comunicam com os serviços Grid em nome do dispositivo móvel.

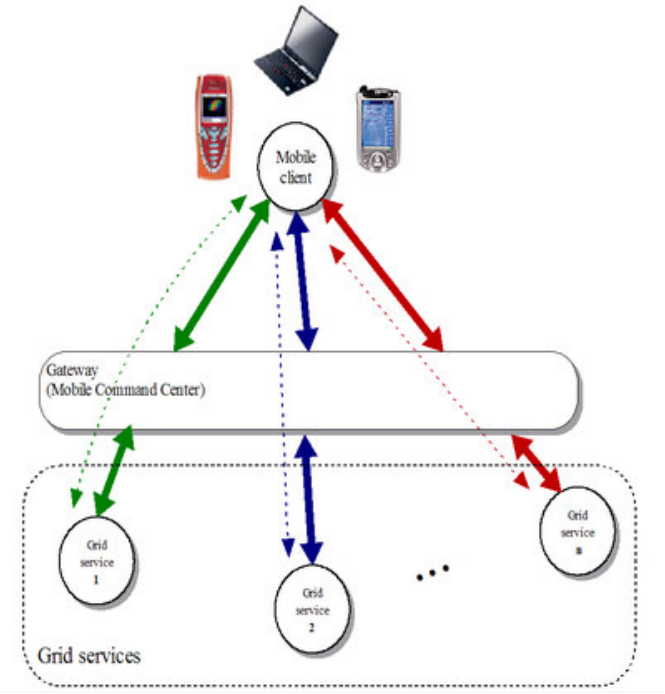

Figura 3.8 Comunicação dispositivos móveis e serviços Grid através de um Gateway [GRIDLAB 2005]

3.7 PORQUE DO USO DE GRIDS

Neste contexto, é oportuno salientar a necessidade ou o porquê do uso de grids como tecnologia que faz a integração e homogeneização de todos os dispositivos. Pois pelo uso de grids pode-se agregar qualquer dispositivo e este se tornará parte da grid, ou seja, um nodo podendo então ser acessado e tendo acesso a qualquer outro dispositivo, tal como uma **peer grid**, na qual cada dispositivo trabalha como um nodo, funcionando como servidor e cliente ao mesmo tempo, requisitando informações (cliente) ou servindo informações (servidor).

Diferente do projeto GridLab, citado no capítulo 3, no qual cada dispositivo não funciona como um peer, e sim como um cliente dos serviços da grid, por possuir sérias limitações de processamento, armazenamento e interfaces.

3.8 RESUMO E DISCUSSÃO DO CAPÍTULO

Neste capítulo foi apresentado o conceito de grids de dispositivos móveis mostrando a união de duas tecnologias (grades de computadores e computação móvel), foi mostrado também uma breve introdução sobre dispositivos móveis.

A subseção sobre grids de informação mostrou a importância da classificação dos tipos de grids, evidenciando neste trabalho a utilização de grids como tecnologia para a homogeneização dos dispositivos agregados a grid e não como tecnologia para processamento de grandes volumes de dados.

Em seguida foram apresentadas algumas áreas de aplicação de grids de dispositivos móveis tais como, gerenciamento de fluxo de trabalho, sistemas móveis para informação de geoprocessamento com destaque para o artigo sobre analise de imagens por uma grid baseada nas informações enviadas pelos dispositivos móveis e finalmente um sistemas de apoio a decisão onde as informações sobre dados vitais coletadas por nodos eram então enviadas a grid que disponibiliza então estes dados em um local público para inferências conjuntas.

Na subseção trabalhos correlatos foram analisados alguns middlewares e suas respectivas características para o fornecimento de serviços móveis seguindo de outra subseção sobre projetos existentes em grids de dispositivos móveis onde foi apresentado o projeto GridLab.

Por fim na subseção 3.8 foi observado o porquê então do uso da tecnologia de grids de computadores junto a tecnologia da computação móvel, explicitando que esta prove a homogeneização dos dispositivos a ela agregados.

4. PROPOSTA - UM MIDDLEWARE PARA GRIDS DE DISPOSITIVOS MÓVEIS

Neste capítulo será apresentada a proposta de um middleware para grids de dispositivos móveis, será também descrita sua arquitetura assim como os serviços prestados por esta arquitetura e as respectivas interações.

Os serviços que um ambiente de computação móvel ao qual esta arquitetura é baseada deve prover são descritos na tabela 4.

Tabela 4.1 Serviços prestados pela arquitetura do middleware para grids de dispositivos móveis – baseado em [HENRICKSEN 2001]

Descrição dos Serviços
1. Estrutura de hardware e software que facilite a conexão dos dispositivos móveis, tais como protocolos para transporte de dados e segurança.
2. Serviços de base como identificação da localização dos dispositivos.
3. Suporte a mobilidade de dispositivos
4. Suporte a heterogeneidade de dispositivos
5. Suporte a mobilidade de usuários
6. Suporte para interfaces universais
7. Suporte a descobrimento de recursos

Para o suporte destes serviços é apresentada a seguir a arquitetura proposta, a qual cada componente desta arquitetura deve suplantar as necessidades do provimento de cada serviço listado na tabela anterior.

4.1 ARQUITETURA GERAL E SERVIÇOS ABRANGIDOS

A arquitetura proposta é composta por módulos, cada um direcionado a uma atividade relacionada aos serviços que devem ser prestados, logo abaixo é descrito cada um desses módulos.

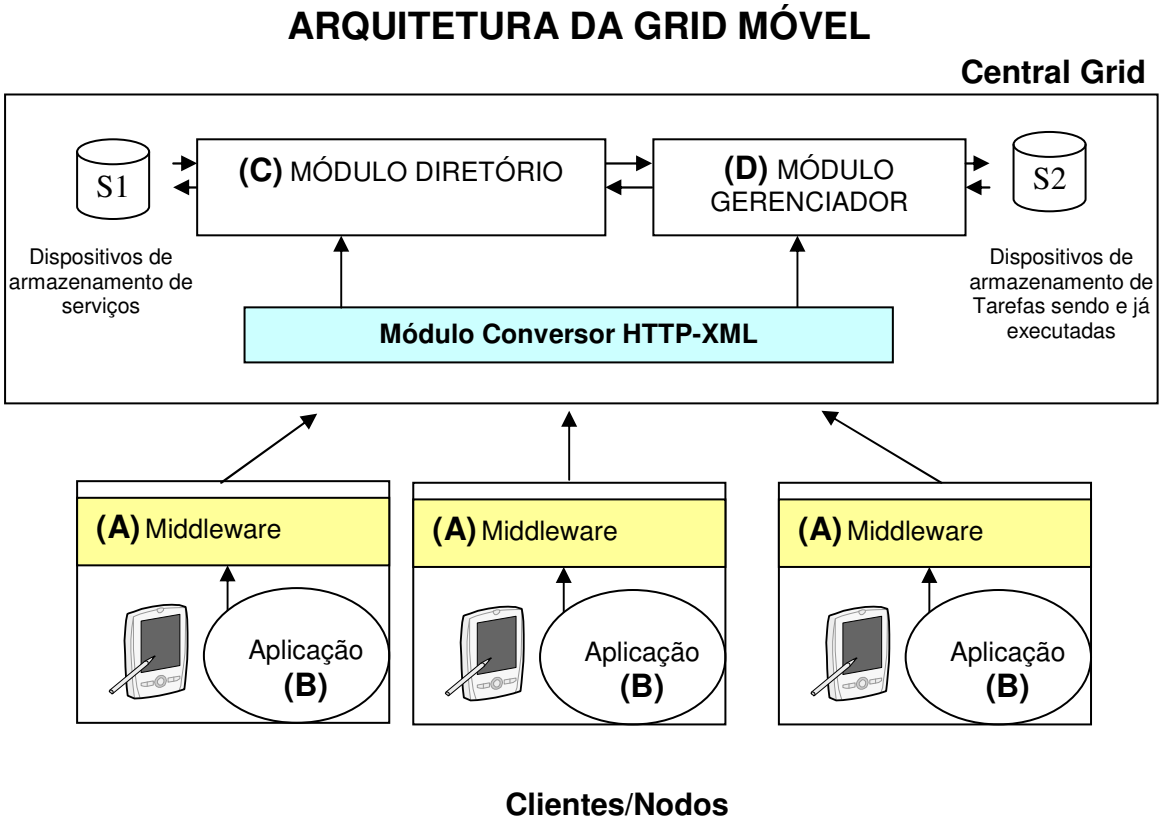

Figura 4.1 Arquitetura do Sistema – Visão Geral

Na figura 4.1 é mostrada a arquitetura proposta. O **(A) middleware** funciona como uma ponte entre as **(B) aplicações** rodando nos dispositivos móveis e, em **(C, D)** central de gerenciamento da grid móvel, estão localizados o **módulo de diretório** e o

módulo gerenciador. Nesta central de serviços, o (C) módulo diretório armazena os serviços disponíveis pela grid no banco de dados de serviços **(S1)**. O (D) módulo gerenciador administra os recursos, agenda as tarefas e monitora a execução destas tarefas armazenando estes dados e os resultados das tarefas enviadas no banco de dados de tarefas **(S2)**.

Baseado na arquitetura acima apresentada e nos serviços que devem ser providos por um ambiente de computação pervasiva à qual esta arquitetura está atrelada, será apresentado uma tabela relacionando os componentes da arquitetura e os serviços resolvidos por aquele componente.

Tabela 4.2. Relação entre componentes da arquitetura e serviços prestados.

Componente da arquitetura	**Serviço resolvido**
1. Estrutura de hardware e software que facilite a conexão dos dispositivos móveis, tais como protocolos para transporte de dados e segurança.	Para satisfazer a conexão de dispositivos é utilizado o protocolo HTTP (HyperText Transfer Protocol) garantindo assim compatibilidade entre dispositivos, no caso, quando da necessidade de segurança pode ser utilizado o protocolo HTTPS (HyperText Transfer Protocol Secure) e métodos de criptografia e autenticação no registro de novos nodos
2. Serviços de base como identificação e localização dos dispositivos.	Para localização e identificação dos dispositivos agregados a grid, o módulo gerenciador (D) mantém uma lista dos dispositivos conectados e sua localização.
3. Suporte a mobilidade de dispositivos	O suporte à mobilidade dos dispositivos está intrinsecamente ligado a capacidade dos dispositivos móveis de se moverem e se manterem conectados a uma central, nesse caso a central grid, a qual por meio do módulo gerenciador (D) mantém um banco de dados dos dispositivos conectados.
4. Suporte a heterogeneidade de dispositivos	Para garantir heterogeneidade de dispositivos é utilizado o módulo conversor HTTP-XML o qual garante independência dos dispositivos por meio

	da comunicação baseada em arquivos XML, o qual permite troca de informações sem dependência de dispositivos garantindo heterogeneidade dos dispositivos à grid agregados.
5. Suporte a mobilidade de usuários	Também garantida pela utilização pelos usuários dos dispositivos móveis, tais como, handhelds, PDA, smartphones.
6. Suporte para interfaces universais	Garantida pela troca de informações através do middleware baseado em protocolos XML.
7. Suporte a descobrimento de recursos e execução e manutenção de tarefas	O descobrimento de recursos é garantido pelo módulo gerenciador (D) acima citado, já o suporte à tarefas é mantido pelo módulo diretório (C) que mantém um banco de dados das tarefas e serviços que podem ser prestados pela grid, serviços como processamento, armazenamento entre outros.

Neste sentido, na próxima subseção serão apresentadas as interações existentes para o suporte dos serviços prestados.

4.2 INTERAÇÕES BASEADAS NOS SERVIÇOS PRESTADOS

Para suportar os serviços que devem ser prestados, as interações entre dispositivos e central grid necessárias são descritas abaixo:

Para o **serviço 1** (Estrutura de hardware e software que facilite a conexão dos dispositivos móveis, tais como protocolos para transporte de dados e segurança) listado na tabela 4.2 acima segue sua descrição baseada em um arquivo XML (eXtensible Markup Language).

Para este serviço existe a interação de “Registro de um NODO” que é apresentada da seguinte forma, primeiro é descrita em pseuso-código, posteriormente em um diagrama UML (*Unified Modeling Language)*, figura 4.2 que descreve os passos de registro do nodo e finalmente mostrando seu respectivo arquivo XML que faz a interação propriamente dita através do middleware.

Pseudo-código:

```
(I.a) RegisterPdaRule() {
        registerNode(PDA1,pda) -> r1
        if(waitConfirmation(r1, 10))
          OK
        else
          RETRY.. whatever!
         fi }
```

Diagrama UML:

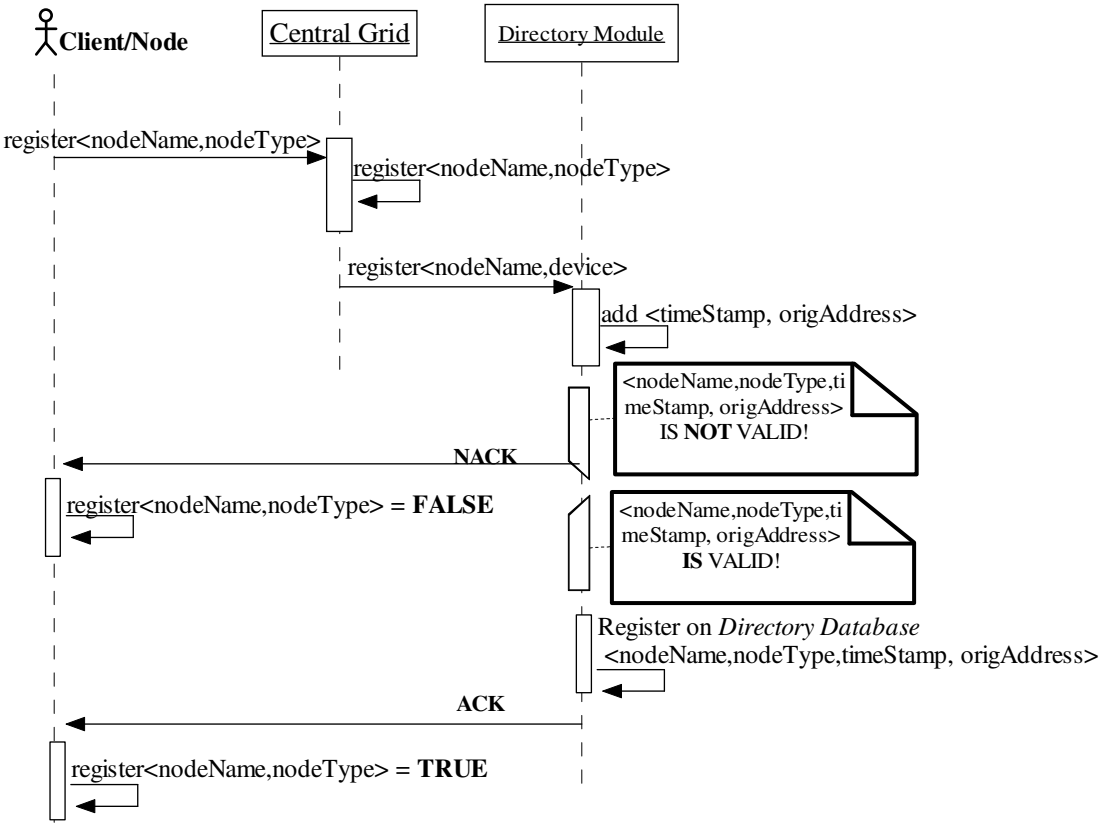

Figura 4.2. Diagrama UML que representa o registro de um nodo

Arquivo XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<register_node>
<node_name> PDA_1 </node_name>
</register_node>
```

Nessa interação foi descrito o registro de um PDA, processo e estrutura que também é valida e semelhante para o registro de um sensor, sendo portanto generalizada para o registro com prévia autenticação de qualquer dispositivo à grade.

De forma semelhante a estrutura da operação de retirar um nodo da grid, somente o arquivo XML, baseado em um comando do usuário, é mudado fazendo a comunicação à central grid da remoção do PDA ou de um Sensor.

Arquivo XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Unregister_node>
<node_name> PDA_1 </node_name>
</Unregister_node>
```

Baseado no serviço 2 (Serviços de base como identificação e localização dos dispositivos) o módulo gerenciador (D) mantém um banco de dados com os dispositivos e sua localização, como:

Tabela 4.3 – BD com os dispositivos móveis e sua localização

Dispositivo	**Local**
PDA_1	Sala Y
Sensor_3	Sala X
PDA_5	XXX.XXX.XXX.XXX IP fora do ambiente local, acesso pela Internet
Sensor_2	Sala_Z

Como a tecnologia utilizada para intercomunicação da rede é sem fio (ex. Wi-Fi) os dispositivos móveis estarão periodicamente se comunicando com a central grid ou com algum ponto de acesso (*access point*) permitindo assim uma localização destes dispositivos no ambiente ou fora dele.

Os **serviços 3 e 5** são semelhantes, respectivamente (Suporte a mobilidade de dispositivos) e (Suporte a mobilidade de usuários) os quais permitem um dos serviços essenciais em um ambiente que requer entrega de informação a qualquer hora e em

qualquer lugar, a mobilidade. Nestes não é necessário um componente da arquitetura para prove-los já que estes são intrínsecos às características dos dispositivos móveis, ou seja, mobilidade tanto do usuário quanto do próprio dispositivo móvel é inerente ao próprio.

Os **serviços 4 e 6**, respectivamente (Suporte a heterogeneidade de dispositivos) e (Suporte para interfaces universais) estão relacionados ao módulo conversor HTTP-XML o qual padroniza a comunicação entre dispositivos e central grid.

Como todos os dispositivos da grid se comunicam pelo intermédio do middleware, este por meio do módulo HTTP-XML faz a troca de informações através de arquivos XML o que garante o suporte a heterogeneidade de dispositivos e o suporte a interfaces universais.

Na figura 4.3 são descritos diferentes dispositivos móveis (tipos 1,2 e 3) evidenciando o serviço 4, suporte a heterogeneidade de dispositivos. No entanto, todos estão executando diferentes tarefas, mas pelo middleware todos se comunicam através de comandos que seguem uma interface de comunicação universal, ou seja, existe somente um modo de registrar um nodo, somente um modo de enviar uma tarefa. Portanto o serviço 6, suporte a interfaces universais está garantido pela utilização do middleware que faz sua comunicação através de arquivos XML pré-definidos e universais.

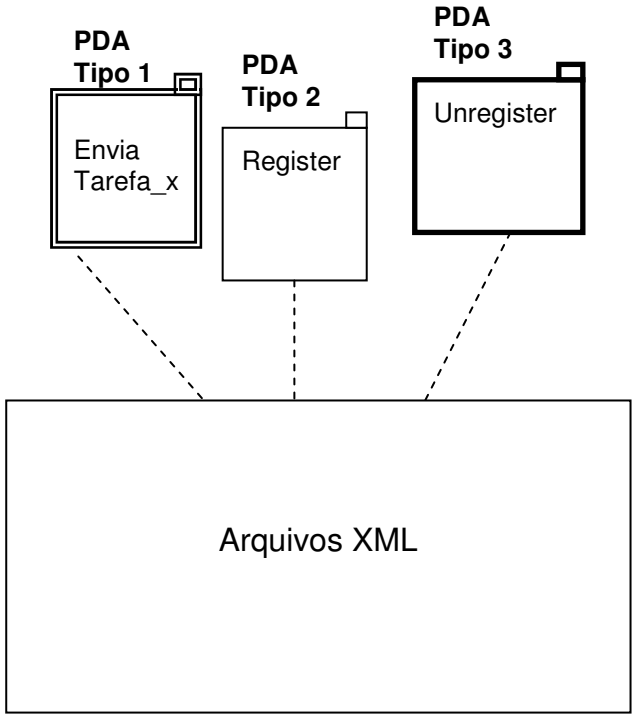

Figura 4.3 Garantia dos serviços de suporte a interfaces universais e heterogeneidade de dispositivos.

O **serviço 7** (Suporte a descobrimento de recursos e execução e manutenção de tarefas) é provido pelo componente da arquitetura, módulo diretório (C) que mantém um banco de dados das tarefas enviadas à grid.

Tabela 4.4 – Relação entre tarefa e seu estado

Tarefa	**Estado**
Tarefa_1	Processando
Tarefa_2	Em fila
Tarefa_3	Pronta
. . .	. . .

Pois ao enviar uma tarefa para grid seu pseudo-código seria este:

(II.b) sendTasks{

```
createTask(store,  data) -> Task
sendTask(Task) -> t1
 if(waitConfirmation(t1, 10))
    sendMessage(PDA1, Message) -> MessageID
 else
    RETRY.. whatever!
 fi}
```

O arquivo XML que é enviado é desta forma:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<sendtask>
<task_name> Tarefa_1 </task_name>
<task_func> store </task_func>
<task_data> Dados </task_data>
</sendtask>
```

Seu Diagrama UML onde estão descritos os passos para submissão de uma tarefa está ilustrado na figura 4.4, abaixo.

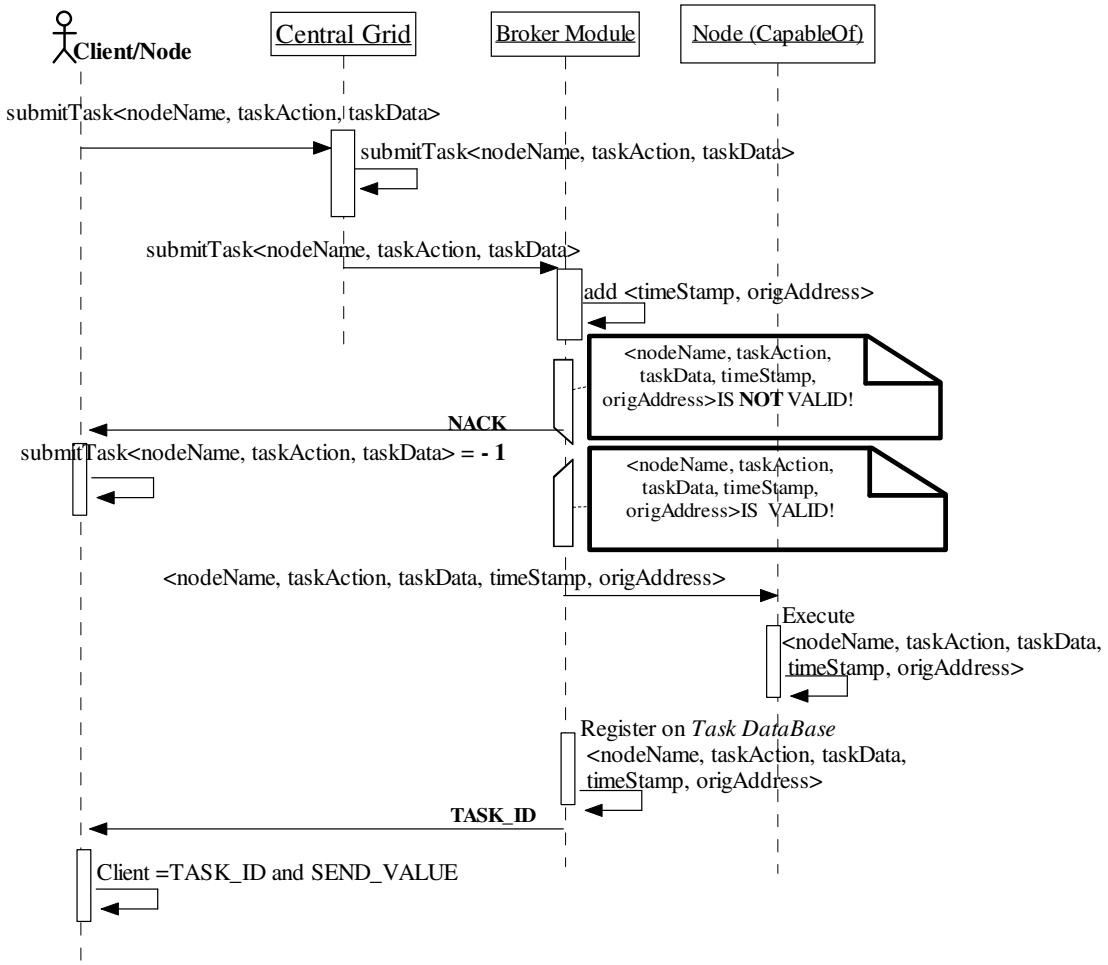

Figura 4.4 Diagrama UML que representa o envio de uma tarefa

Baseado nas descrições acima, o suporte aos serviços prestados foi garantido pelos componentes da arquitetura que está sendo proposta. Com isso cobrimos todos os serviços prestados e neste contexto, a seguir será apresentado um cenário real que servirá de exemplo para ser comprovado a necessidade destes serviços.

Consideremos o cenário problema abaixo:

Uma instituição hospitalar quer disponibilizar um sistema para o seu corpo médico onde os dados vitais dos pacientes internados ficam disponíveis aos médicos em tempo real de ***qualquer lugar e a qualquer hora****. Este sistema poderá servir como um sistema de apoio à decisão, pois estes dados poderão ser publicados para análises e inferências.*

Nesse hospital existem coletores de dados vitais (sensores) interligados aos aparelhos que coletam os dados dos pacientes em tempo real. Esses sensores transmitem os dados para uma central via uma rede de computadores local sem fio (e.g., usando tecnologia Wi-Fi ou semelhante). Este sistema será responsável pela catalogação e armazenamento dos dados recebidos dos sensores e também pelo gerenciamento dos nodos (sensores e atuadores). Sensores são dispositivos eletrônicos que coletam ou recebem informação, os que recebem informação são chamados de atuadores, pois recebem uma informação e podem atuar ou influenciar um ambiente.

Um dos requisitos nesse ambiente é ter um padrão rígido de segurança, evitando perda de dados ou acessos indevidos. Completando os personagens deste sistema, o corpo médico do hospital, será equipado com Personal Digital Assistants (PDA) carregados com software que permita o acesso desses aos dados que estão sendo coletados em tempo real dos pacientes.

Os médicos podem visualizar os dados de um ou vários pacientes ao mesmo tempo e também programar seus assistentes pessoais para monitorar estes dados e

gerar alarmes quando certas condições são atingidas. O PDA deve também permitir acesso a outros dados da instituição e a comunicação entre médicos e enfermeiros, constituindo-se assim um sistema que fornece colaboração.

4.3 ANÁLISE DE REQUISITOS

Dentro do cenário proposto acima, será analisado as principais interações necessárias ao funcionamento de um sistema que sirva de solução para este tipo de ambiente e baseado nestas interações será considerado os requerimentos do middleware que suportará tais interações.

Consideramos então alguns dos requisitos para o desenvolvimento do middleware:

- Ser computacionalmente leve, visando ser integrado aos dispositivos de computação móvel, tendo como base dispositivos com processadores de 20Mhz e capacidade de memória de 512Kbytes ou maiores.

- Apresentar uma interface genérica para que através de várias plataformas e dispositivos esta seja um ponto único de contato entre a o solução e as tecnologias de base (independente do dispositivos onde estiver executando).

- Ser compatível com as plataformas Java J2SE [J2SE 2005] e Java J2ME [J2ME 2005], garantido que as soluções possam usar o mesmo middleware para integrar componentes entre servidores, computadores de mesa e dispositivo móveis, tais como assistentes digitais pessoais (PDA) e telefone celulares.

A tabela 4.5 relaciona e evidencia os serviços necessários para o cenário real apresentado acima.

Tabela 4.5 Comprovação da utilização dos serviços, baseado em um ambiente real

Necessidade no cenário proposto	**Serviço que soluciona**
Comunicação entre os dispositivos e os sensores **(a) métodos de comunicação de dados**	Serviço 1
Controle de interferências externas **(b) métodos de segurança**	
Para adição de recursos à grid e controle de entrada e saída de dispositivos **(c) métodos de registro de recursos**	Serviço 2
Controle das tarefas enviadas **(d) gerenciamento de tarefas enviadas a grid**	Serviço 7
(e) interfaces amigáveis e padronizadas que possibilitem uma melhor interação	Serviço 6

Adicionalmente, o suporte para estes requisitos está além do escopo das linguagens de programação que possam ser executadas em vários dispositivos (e.g. Java na sua versão Java 2 Micro Edition [J2ME 2005]) e requerem uma solução de software de base que suporte a integração de dispositivos, homogeneização do desenvolvimento, re-uso do software, coordenação, busca de recursos, resolução de problemas de endereçamento, segurança e outras características.

Acreditamos que este suporte seja oferecido pela integração das tecnologias de Grid Computing e Computação Móvel [THAM 2005], e que realizaremos através de nosso middleware que será apresentado com maiores detalhes na seção seguinte.

4.4 O MIDDLEWARE

Nessa subseção apresentaremos a arquitetura do sistema do ponto de vista do middleware em dois níveis de abstração permitindo ilustrar as interações entre os dispositivos da grid e para uma maior organização o middleware será divido em módulos que permitem a prestação dos serviços relacionados.

4.4.1 NÍVEL DE ABSTRAÇÃO 1

A Figura 4.5 apresenta, a localização e relação do middleware com os outros dispositivos (sensores, PDAs, dispositivos de armazenamento, entre outros) e os possíveis atuadores na grid (usuários, desenvolvedores ou gerentes). Através da figura pode-se verificar que o middleware atua como gerenciador/mediador das possíveis ações na grid, ou seja, o registro de um sensor, ou o registro de um PDA na grid deve necessariamente passar pelo middleware.

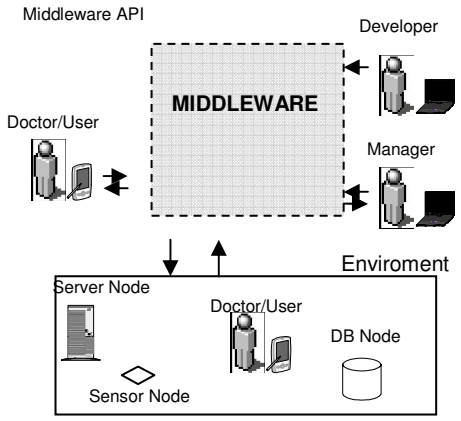

Figura 4.5 Arquitetura do Sistema com nível de abstração 1 – API e Interações

4.4.2 NÍVEL DE ABSTRAÇÃO 2 E MÓDULOS PERTENCENTES AO MIDDLEWARE

A figura 4.6 mostra, em um nível de detalhes maior os módulos, que estão numerados em algarismos romanos, para posterior classificação das regras/métodos de cada módulo. Estes módulos estão acoplados ao middleware que é responsável pelo controle das possíveis ações nesta arquitetura e, por conseguinte a prestação dos serviços. A seguir será descrito cada módulo e os métodos (regras) que o regem.

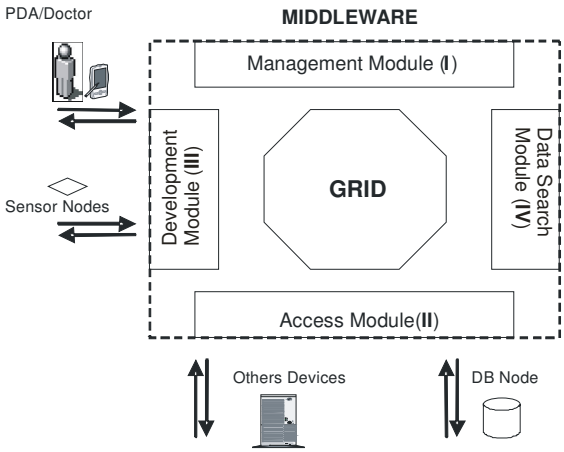

Figura 4.6 Arquitetura do Sistema com nível de abstração 2 – Módulos de Gerenciamento

4.5 REGRAS E MÉTODOS DA ARQUITETURA PROPOSTA

Nesta subseção é descrito as regras e métodos de cada módulo da arquitetura proposta. Cada método está descrito através de uma pseudo-codificação.

O módulo I (Management Module), tem a função de gerenciar os dispositivos agregados à grid, possuindo uma lista de controle dos dispositivos que entram (se conectam) e saem (desconectam) da grid.

Este módulo contempla-se de algumas regras/métodos que implementam o seu funcionamento, tais como:

Faz o registro de um Sensor

```
(I.b) RegisterSensorRule() {
 registerNode(Sensor1,sensor) -> r2
        if(waitConfirmation(r2, 10))
          OK
        else
          RETRY.. whatever!
         fi }
```

Retira um PDA da Grid

```
(I.c) UnregisterPdaRule() {
        unregisterNode(PDA1,pda) -> r1
        if(waitConfirmation(r1, 10))
          OK
        else
        RETRY.. whatever!
        Fi}
```

Retira um Sensor da Grid

```
(I.d) UnregisterSensorRule(){
       unregisterNode(sensor1,sensor) -> r2
        if(waitConfirmation(r2, 10))
          OK
       else
         RETRY.. whatever!
       fi}
```

No entanto, deve-se prever que seja necessário adicionar um novo dispositivo à grid, como por exemplo, um dispositivo de armazenamento (*Storage Device*), ou um dispositivo de processamento (*Processor Device*), portanto, é definido no módulo III **Development Module**, ou seja, módulo de desenvolvimento pelo qual será possível criar novos dispositivos, por meio de regras/métodos como:

Criação de Novos dispositivos na grid

(III.a) CreateNewDevice (deviceType, deviceName)

Módulo II, (*Access Module*) módulo de acesso aos dados, este módulo permite que, por exemplo, médicos possam consultar as condições vitais de seus pacientes, através de seus PDA's e enviar dados (tarefas) para a central grid. Para isto temos as regras/ métodos:

Acesso a dados, por exemplo, doutor requisita informações de determinado paciente.

Pseudo-código:

(II.a) getPacientStatus{

```
createTask(getPatientStatus,  PatientName) -> Task
sendTask(Task) -> t1
 if(waitConfirmation(t1, 10))
    sendMessage(PDA1, Message) -> MessageID
 else
    RETRY.. whatever!
 fi}
```

A regra seguinte, permite que personagens autorizados como um doutor que possui seu PDA cadastrado no sistema, requisite informações de determinado paciente o qual sua identificação é passada como parâmetro na chamada. No entanto, este módulo deve se comunicar com o **módulo IV** (*Data Search Module*), módulo de procura de informações, o qual coleta as informações de cada sensor e armazena em um dispositivo de armazenamento (storage Device). Sua principal regra/método é:

Coleta e armazenamento de informações de um determinado sensor.

(IV.a) collectRule{

```
        Sensor.collect () -> DataSet
         createTask(store, DataSet) -> Task
         sendTask(Task) -> t1
 if(waitConfirmation(t1, 10))
    OK
 else
    RETRY.. whatever!
 fi}
```

Baseado nestas regras/métodos tem-se a possibilidade de executar as principais funções de qualquer grid como: adicionar novos nodos, retirar nodos, enviar tarefas a serem processadas e armazenar informações.

Ainda por meio destas regras pode-se proporcionar uma característica importante da computação móvel, a **colaboração**, pois dispositivos móveis (nodos) podem ser adicionados à grid e outros médicos (PDA/nodos) podem comunicar-se entre si e mais, ter acesso a uma área comum, onde dados de um determinado paciente podem ser publicados e então possibilitar aos doutores fazerem inferências para um determinado diagnóstico.

4.6 CONSIDERAÇÕES E DISCUSSÃO DO CAPÍTULO

Neste capítulo foi apresentada a proposta do middleware para grades de dispositivos móveis, foi descrita sua arquitetura, interações e serviços prestados necessários para um ambiente de grades de dispositivos móveis.

No começo do capitulo foram apresentados os serviços que devem ser prestados, posteriormente foi apresentada a arquitetura geral, onde os componentes da arquitetura fecham com os serviços necessários, ou seja, os componentes da arquitetura solucionam os serviços.

Na seção seguinte foram apresentadas em pseudocódigo e diagramas UML as interações baseadas nos serviços e mostrados os arquivos XML que regem cada serviço.

Este capítulo é o centro de nossa pesquisa, pois propõe uma arquitetura baseada em um middleware que é responsável pelas interações entre as tecnologias de grades de computadores e computação móvel. Tentamos demonstrar os principais serviços que um ambiente móvel deve ter e as interações que regem estes serviços.

No próximo capítulo serão apresentados dois estudos de caso para aplicação dos componentes da arquitetura, assim como das regras propostas. Os resultados serão demonstrados por meio dos estudos de caso mapeados sobre a arquitetura proposta e assim verificar se o que esta sendo proposto é viável, ou seja, a validação.

5. RESULTADOS EXPERIMENTAIS

Nessa seção serão apresentadas soluções através de casos de uso, onde serão utilizadas as funcionalidades da arquitetura e interações propostas na seção anterior. Almeja-se que essa estrutura de software facilite a criação de serviços móveis distribuídos e forneça ao desenvolvedor uma plataforma que integre as soluções da área da computação distribuída com os recursos e necessidades da computação móvel.

5.1 POSICIONAMENTO DA SOLUÇÃO

Na Figura 5.1 é apresentado o posicionamento do middleware proposto. Esse deve integrar os serviços oferecidos pela computação em grade, i.e., comunicação de dados, serviços de diretório e outros, a serviços comuns na área de computação móvel, tais como sensibilidade ao contexto e interfaces dos dispositivos móveis.

O middleware deve oferecer uma interface de programação unificada, que possibilite ao desenvolvedor da solução utilizar um único middleware tanto para o acesso de recursos da computação em grade (e.g., estabelecimento de canais de comunicação de dados, transferência de dados, descoberta de recursos) como dos serviços do dispositivo móvel (e.g., apresentação da informação na interface dos dispositivos e sensibilidade ao contexto, tal como localização do dispositivo).

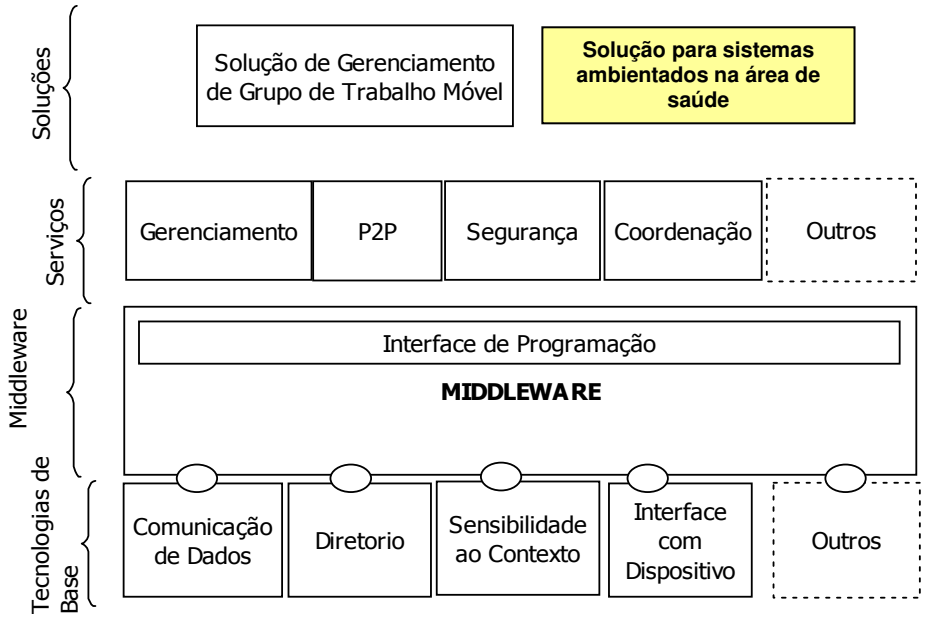

Figura 5.1 Posicionamento do Middleware para Computação em Grade em Dispositivos Móveis

Sobre essa interface de programação poderemos desenvolver os serviços integrados, tais como serviços de gerenciamento, comunicação peer-to-peer (P2P), segurança, coordenação e outros. Esses serviços integrados funcionarão como blocos de construção das soluções para serviço móvel, tais como soluções para o gerenciamento de grupo de trabalho.

5.2 ESTUDOS DE CASOS

Nesta subseção serão apresentados 2 (dois) estudos de casos onde serão mostradas as vantagens do uso da arquitetura proposta para o desenvolvimento da solução. Serão aplicadas as regras/métodos do middleware bem como a interação entre os componentes da arquitetura proposta para demonstrar tais vantagens.

5.2.1 ESTUDO DE CASO 1 – CENÁRIO APLICADO A ÁREA DA SAÚDE (*HEALTHCARE SCENARIO*)

Consideremos o seguinte cenário problema ambientado em um hospital:

Um paciente possui seus sinais vitais sendo obtidos periodicamente por um sensor. Estes sinais, após serem coletados são enviados para uma central onde serão processados e então armazenados. Um médico, através de seu PDA, requisita tais dados de forma que possa acompanhar a situação, em tempo real, desse paciente. Após observar os dados o médico decide terminar, encerrando suas requisições.

Para que este cenário seja possível e estas ações acontecerem os seguintes passos ou interações devem ser seguidos:

i) Registro do PDA do doutor na grid;

ii) Prévio registro do Sensor na grid;

iii) Obtenção dos dados vitais e posterior envio para grid

iv) Confirmação que os dados foram recebidos

v) Requisição dos dados pelo doutor;

vi) Procura do paciente e consulta/apresentação dos sensores a ele ligados.

vii) Envio dos dados ao médico;

viii) Médico sai do ambiente desfazendo seu registro do grid

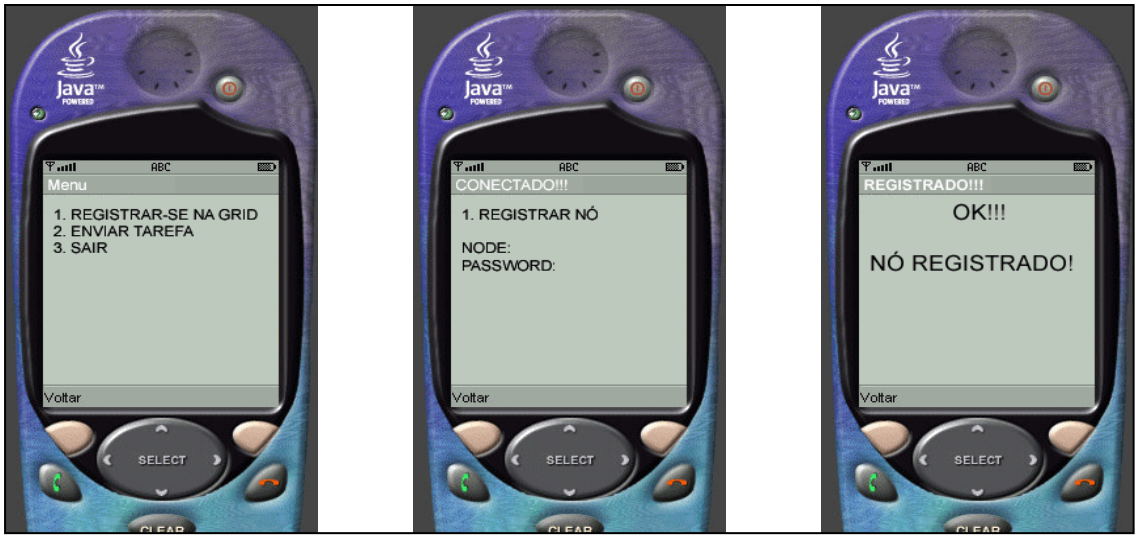

Figura 5.2 Interfaces para registro de um nodo na grid – exemplificando o primeiro passo

Na figura 5.2 é apresentado uma interface de um dispositivo móvel, pelo exemplo, um celular que está sendo registrado à grid. Pode-se notar os passos para o registro de um dispositivo na grid, ou seja, por meio de um menu do sistema sendo executado no dispositivo é possível acessar o comando de registro de tal dispositivo,

posteriormente é solicitado uma senha e um nome de usuário para controle e segurança e logo depois é enviada resposta ao nodo/dispositivo agora conectado à grid.

E para que estes passos ocorram, as seguintes regras/métodos serão utilizados:

- Para as ações (i) e (ii) são utilizadas as regras (I.a) e (I.b) respectivamente, do módulo I (*Management Module*) ou módulo de gerência, para registro do PDA e do Sensor na grid.

(I.a) RegisterPdaRule() {
registerNode(PDA1,pda) -> r1
if(waitConfirmation(r1, 10))
OK
else
RETRY.. whatever!
fi }

(I.b) RegisterSensorRule() {
registerNode(Sensor1,sensor) -> r2
if(waitConfirmation(r2, 10))
OK
else
RETRY.. whatever!
fi }

- Para as ações (iii) e (iv) é utilizada a regra (IV.a) do módulo IV (*Data Search Module*) ou módulo de procura de dados, que coleta as informações dos sensores e envia para o módulo diretório para armazenar em S1 os dados de um determinado paciente, aguardando uma confirmação.

(IV.a) collectRule{
Sensor.collect () -> DataSet
createTask(store, DataSet) -> Task
sendTask(Task) -> t1
if(waitConfirmation(t1, 10))
OK
else
RETRY.. whatever!
fi}

- Para as ações (v), (vi) e (vii) é utilizada a regra (II.a) do módulo II (*Access Module*) ou módulo de acesso, que permite o acesso, às informações do paciente requisitado.

(II.a) getPacientStatus{

```
createTask(getPatientStatus,  PatientName) -> Task
sendTask(Task) -> t1
 if(waitConfirmation(t1, 10))
    sendMessage(PDA1, Message) -> MessageID
 else
    RETRY.. whatever!
 fi}
```

Neste momento os dados requisitados pelo doutor são buscados em um arquivo XML (*Extensible Markup Language*) que dependendo da consulta realizada pelo doutor, são então retornados. Este cenário está explicitado na figura 5.1.2 onde um arquivo exemplo XML é construído a partir da coleta dos sinais vitais que são obtidos por meio de um sensor que ininterruptamente verifica estes sinais de um paciente.

Este arquivo sofrerá um processo de varredura nos seus dados para proporcionar os dados pesquisados e enviar estes dados para o dispositivo de armazenamento S2 da arquitetura proposta.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<paciente>
<nome>João</nome>
<idade>97</idade>


<tx-oxi>90%</tx-oxi>

<tx-co2>5%</tx-co2>

<bt-card>75</bt-card>

<glicose>80</glicose>

</paciente>
```

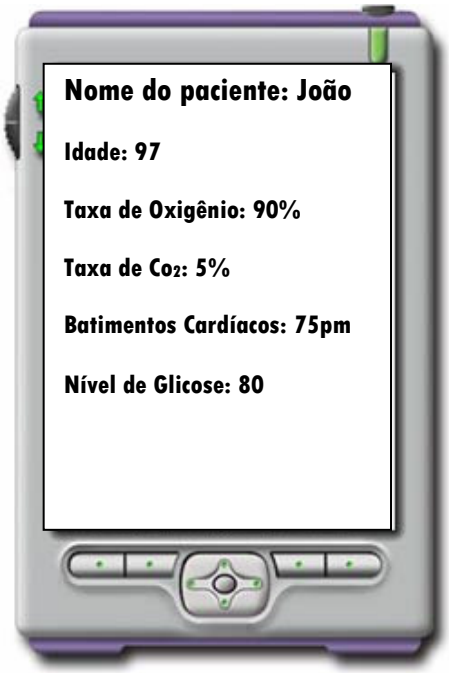

Figura 5.3 Dados provenientes de sensores, mapeados em um arquivo XML o qual fornece informações sobre determinado paciente

Como pode ser visto na figura 5.3 acima, os dados referentes a um determinado paciente são apresentados no dispositivo móvel. É necessário haver uma verificação da capacidade de cada dispositivo móvel, pois nem todos possuem uma interface de grande porte, isto é verificado quando o dispositivo móvel se cadastra na grid e baseado nestas informações são apresentadas as informações.

- Para a ação (viii) é utilizada regra (I.c) que retira o PDA do médico da grid, após este encerrar suas requisições.

```
(I.c) UnregisterPdaRule() {
        unregisterNode(PDA1,pda) -> r1
        if(waitConfirmation(r1, 10))
          OK
        else
        RETRY.. whatever!
        Fi}
```

Com este estudo de caso pode-se verificar que a arquitetura proposta e os métodos/regras suprem a solução para problemas inerentes a estes tipos de ambientes, nos quais é necessário mobilidade, acesso de qualquer lugar e a qualquer momento e quando necessário processamento.

A seguir um outro estudo de caso, no qual serão utilizados os métodos e regras baseado na arquitetura proposta.

5.2.2 ESTUDO DE CASO 2 – CENÁRIO APLICADO A ÁREA DE GERENCIAMENTO DE FORÇA DE TRABALHO (*MOBILE WORKFORCE MANAGEMENT*)

Consideremos o seguinte cenário problema:

Uma empresa precisa prover a seus funcionários um ambiente de trabalho no qual seja essencial a troca e visualização de informações a qualquer hora e em

qualquer lugar por todos da força de trabalho. Além disso seus funcionários devem acessar dados para ajuda no fechamento de transações e podem fazer e enviar coletas de informações em campo, para essas informações serem disponibilizadas para se tornarem apoio à decisões.

De forma semelhante ao caso anterior, para que este cenário seja possível e estas ações acontecerem os seguintes passos ou interações devem ser seguidos:

i) Registro do PDA do funcionário na grid;
ii) Obtenção dos dados em campo e posterior envio para grid
iii) Confirmação que os dados foram recebidos
iv) Requisição dos dados por outros funcionários;
v) Procura dos dados nos nodos específicos de armazenamento;
vi) Envio dos dados ao funcionário;
vii) Funcionário sai do ambiente desfazendo seu registro do grid

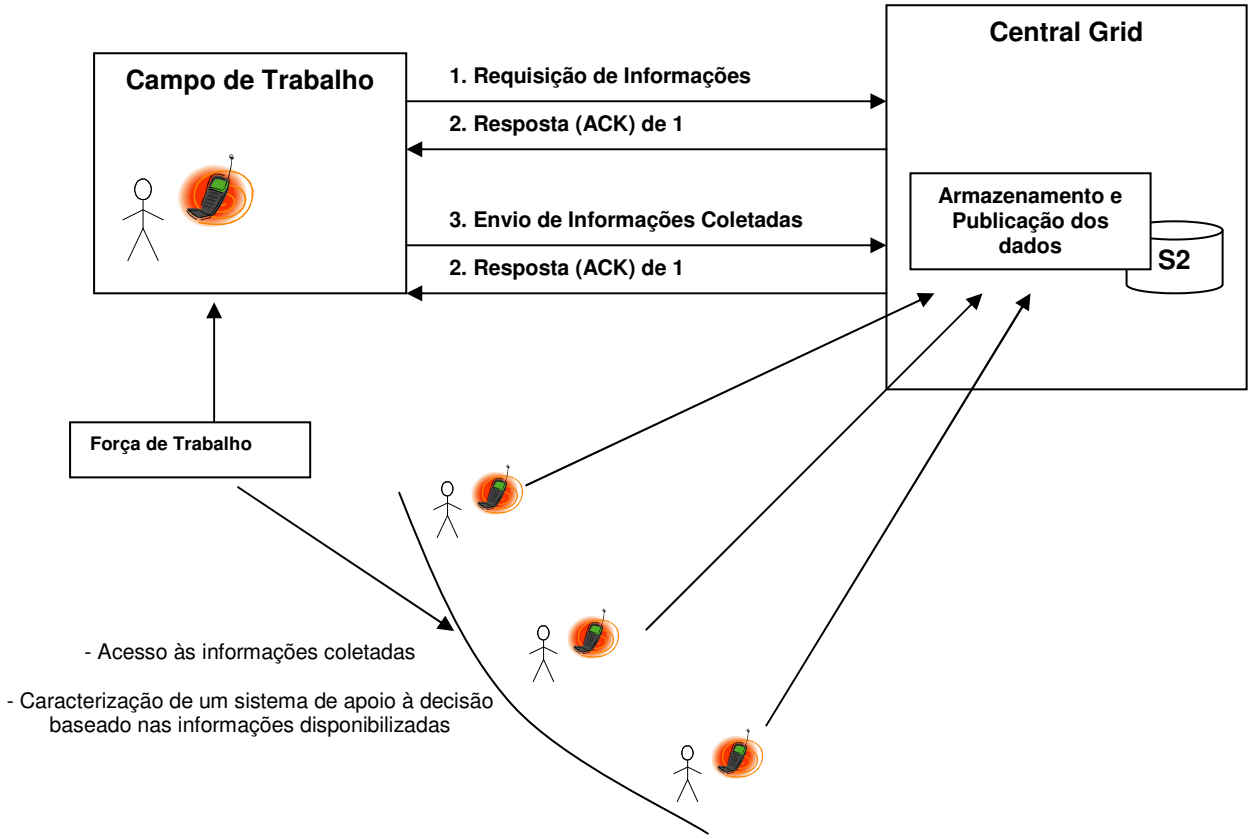

Figura 5.4 Gerenciamento em um cenário de força de trabalho, onde os usuários podem enviar, receber, armazenar e publicar informações.

Na figura 5.4 acima, é apresentado um cenário geral onde forças de trabalho utilizam da infra-estrutura de grids de dispositivos móveis ao utilizar PDA, smartphones ou afins para comunicar-se, enviar dados e visualizar estes dados. Estes dados são enviados para uma central grid que armazena e publica estes dados podendo então caracterizar-se como um sistema de apoio a decisão.

E para que estes passos ocorram, as seguintes regras/métodos serão utilizados:

- Para a ação (i) é utilizada a regra (I.a), do módulo I (*Management Module*) ou módulo de gerência, para registro do PDA na grid.

```
(I.a) RegisterPdaRule() {
        registerNode(PDA1,pda) -> r1
        if(waitConfirmation(r1, 10))
          OK
        else
          RETRY.. whatever!
        fi }
```

- Para as ações (ii) e (iii) pode ser utilizado um sistema de envio de tarefas para grid para serem ou armazenadas ou processadas.

```
(II.b) sendTasks{
      createTask(store,  data) -> Task
      sendTask(Task) -> t1
       if(waitConfirmation(t1, 10))
          sendMessage(PDA1, Message) ->
MessageID
       else
          RETRY.. whatever!
       fi}
```

- Para as ações (iv), (v) e (vi) é utilizada a regra (II.a) do módulo II (*Access Module*) ou módulo de acesso, que permite o acesso, às informações armazenadas na grid.

(II.a) getData{

```
createTask(getData,  dataID) -> Task
sendTask(Task) -> t1
 if(waitConfirmation(t1, 10))
    sendMessage(PDA1, Message) -> MessageID
 else
    RETRY.. whatever!
 fi}
```

- Para a ação (vii) é utilizada regra (I.c) que retira o PDA da grid, após este encerrar suas requisições.

(I.c) UnregisterPdaRule() {

```
unregisterNode(PDA1,pda) -> r1
 if(waitConfirmation(r1, 10))
    OK
 else
 RETRY.. whatever!
 Fi}
```

5.3 VANTAGENS DA ARQUITETURA PROPOSTA

Com base nas ações tomadas segundo as regras e arquitetura propostas pretende-se demonstrar a vantagem do uso desta abordagem defendida por este trabalho. Pois para cada ação tomada um determinado método ou regra fornece a capacidade para cada uma, e através destes enumeraremos as vantagens:

1. *Reusabilidade*, com os métodos propostos é possível reutilização de código, dispensando re-trabalho do desenvolvedor, pois o registro ou a retirada de um dispositivo da grid, ou até mesmo uma consulta geral não requer re-programação de novos métodos, podendo ser reusado para diferentes aplicações.

2. *Portabilidade,* como tais regras estão localizadas no middleware e este é compatível com as plataformas Java J2SE [J2SE] e Java J2ME [J2ME], existe a garantia que as soluções desenvolvidas para o middleware possam ser executadas em vários tipos de dispositivos tais como assistentes digitais pessoais (PDA), telefone celulares entre outros.

3. *Mobilidade*, pois através da arquitetura proposta a comunicação entre o dispositivo móvel e a grid que proverá os recursos tem a quebra de comunicação minimizada, pois os recursos são previamente alocados para cada aplicação e assim quando a conexão cair outros recursos estarão alocados pela gerenciador de recursos da grid para aquela aplicação, provendo portanto mobilidade ao usuário do dispositivo móvel.

4. *Colaboração*, os nodos conectados a grid tem acesso a uma área comum de dados onde cada dispositivo móvel pode tanto visualizar estas informações, como também pode sem comunicar com os outros dispositivos para, por exemplo, inferirem um certo diagnóstico ou discutir sobre um determinado assunto.

5. *Possibilidade de utilização como um sistema de apoio a decisão,* pois com a arquitetura proposta pode-se fazer a publicação de dados em um determinado site. Este site pode permitir que usuários com prévia autorização possam ter acesso a estes dados de qualquer lugar, por exemplo, somente pelo uso de um navegador de internet e baseado nestas informações seja possível então de fazer inferências sobre determinado assunto seja um diagnóstico seja uma decisão de venda de modo colaborativo.

5.4 ALGUMAS DESVANTAGENS/RESTRIÇÕES

Esta arquitetura não está mapeada para outros middlewares, mesmo que estes possuam alguma infra-estrutura para dispositivos móveis.

Número limitado de métodos/regras, mas que podem ser criados.

Foi aplicado em poucos ambientes, no entanto, pode ser utilizado para qualquer ambiente que necessite de mobilidade e homogeneização de dispositivos.

5.5 CONSIDERAÇÕES E DISCUSSÃO DO CAPÍTULO

Neste capitulo foram realizados dois estudos de caso para validação da arquitetura e do middleware propostos.

No primeiro estudo de caso, o ambiente escolhido e objetivo de aplicação prática deste trabalho foi um cenário em um hospital no qual um paciente está ligado a sensores e estes estão ligados a grid. Estes sensores enviam informações para a central grid que armazena e quando necessário realiza a publicação destes dados para visualização de outros médicos, caracterizando assim também um sistema de apoio a decisão.

O segundo estudo de caso foi em um cenário de força de trabalho móvel, na qual os funcionários de uma empresa podem enviar e receber informações em seus dispositivos móveis a qualquer hora e em qualquer lugar, aumentando assim a interação.

Nos dois casos a arquitetura proposta suplantou as necessidades de resolução dos problemas intrínsecos a estes tipos de ambientes, que necessitam de mobilidade e entrega de informação a qualquer momento e lugar.

No próximo capítulo serão apresentados um protótipo teste e o aprimoramento da especificação do middleware para dispositivos móveis utilizando grades de computadores.

6. PROTÓTIPO E APRIMORAMENTO DA ESPECIFICAÇÃO

A arquitetura do middleware foi especificada em dois níveis: o do software relacionado ao nodo e o da estrutura da grid.

Nessa seção é apresentada a implementação de um protótipo, que foi utilizado para testar a especificação proposta. Este protótipo possui fase inicial já que como um dos itens de trabalhos futuros, esta dissertação está sendo continuada por outros alunos do Programa de Pós Graduação em Ciência da Computação com uma implementação mais completa e com maior enfoque.

6.1 DETALHAMENTO DA ARQUITETURA

A arquitetura do middleware foi especificada em dois níveis: o do software relacionado ao nodo e o da estrutura da grid, esta estrutura será detalhada ao longo deste capítulo, no parágrafo seguinte temos a explicação de um código escrito na linguagem C que faz a conexão com um servidor e troca informações no formato XML, e representa o início dos testes, pois este servidor enviará mensagens para os nodos da grid.

Esta rotina faz a comunicação por meio de arquivos XML entre o servidor e, por exemplo, um dispositivo móvel. Este código não contempla nenhuma adição sobre grades de computadores, no entanto, comprova a base para a comunicação entre dispositivos móveis e um nodo da grade, ou seja, a troca de informações por meio de arquivos XML, e que serve como teste, já que o servidor utilizado poderia ser um nodo da grade de computadores.

A partir disto, qualquer troca de informação via XML é possível mostrando assim que o envio e recebimento de mensagens entre um nodo da grade de computadores e um dispositivo móvel seja concreto.

Connect.C

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
void error(char *msg)
{
    perror(msg);
    exit(0);
}
int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[256];
    if (argc < 3) {
       fprintf(stderr,"usage %s hostname port\n", argv[0]);
       exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
         (char *)&serv_addr.sin_addr.s_addr,
         server->h_length);
    serv_addr.sin_port = htons(portno);
 if (connect(sockfd,&serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR connecting");
   /* Montando XML */
    strcpy (buffer,"POST /testeXML/TesteXML.do HTTP/1.0\n\n");
    strcat (buffer, "Host: www.serverteste.com\n");
    strcat (buffer, "Content-Type: application/xml; charset=iso-8859-1\n");
    strcat (buffer,"Content-Length: 128\n");
    strcat (buffer,"xml=");
    strcat (buffer,"\n");
    strcat (buffer,"<?xml version=1.0 encoding=iso-8859-1?>\n");
    strcat (buffer,"<data>");
    strcat (buffer,"<temperature>36</temperature>");
    strcat (buffer,"</data>");
    strcat (buffer,"\n\n");
        n = write(sockfd,buffer,strlen(buffer));
    if (n < 0)
         error("ERROR writing to socket");
    bzero(buffer,256);
    n = read(sockfd,buffer,255);
    if (n < 0)
         error("ERROR reading from socket");
    printf("%s\n",buffer);
    return 0;
}
```

Figura 6.1 Código para conexão com servidor e envio de dados em XML

A partir desta premissa, a de que é possível a troca de informações entre um nodo da grade e um dispositivo móvel, foi aprimorado um conjunto de estruturas e arquivos XML para troca de informações assim como o detalhamento da arquitetura, que estão descritos no decorrer deste capítulo.

6.1.1 ARQUITETURA DO MIDDLEWARE EM RELAÇÃO AO SOFTWARE DO NODO

Nodos são orientados a comunicação, qualquer operação no middleware é iniciada por uma comunicação feita pela interface de programação de aplicações através de um *request* ou pelo escalonador de tarefas ou ainda pelo canal de comunicação.

Na figura 6.1 é apresentada de forma mais detalhada a arquitetura do software middleware que representa um guia para a implementação, ou seja, um modelo de referência e uma documentação para futuras implementações. Segue a descrição de cada módulo da arquitetura.

(a) Módulo de Comunicação: este módulo contém as subrotinas encarregadas da intercomunicação entre nodos. Como dito anteriormente o protocolo de comunicação escolhido foi o HTTP sobre o TCP/IP. Estes nodos devem receber os pacotes através de portas HTTP (por meio de portas padrão do middleware grid) ou receber os pacotes previamente armazenados em um servidor. Este módulo reserva as mensagens para então serem convertidas em XML para então enviá-las ao módulo (b) o emissor de mensagens.
(b) Módulo Emissor de Mensagens: contém as subrotinas para emitir as mensagens recebidas para os serviços, por exemplo executados por um plug-in, ou seja, um novo serviço que resolve determinada tarefa. Serviços de Profile são importantes para todos os nodos. Novos serviços (plug-ins) podem ser adicionados através da interface de programação.
(c) Módulo Executor de Serviços: neste módulo são executados os serviços e também são adicionados novos serviços implementados pela interface de programação.
(d) Módulo agendador (*scheduler*) de Requests: permite agendar verificações para determinadas tarefas, como verificar se determinada tarefa já está pronta em períodos de tempo.
(e) Módulo de Interface de Programação de Aplicações: permite a conexão de aplicações externas para resolução de tarefas enviadas pelos nodos.
(f) Aplicações: as aplicações estão situadas em uma camada acima da camada do middleware, estas aplicações resolvem por meio do executor de serviços

as chamadas dos nodos e podem ser desenvolvidas por meio da interface de programação.

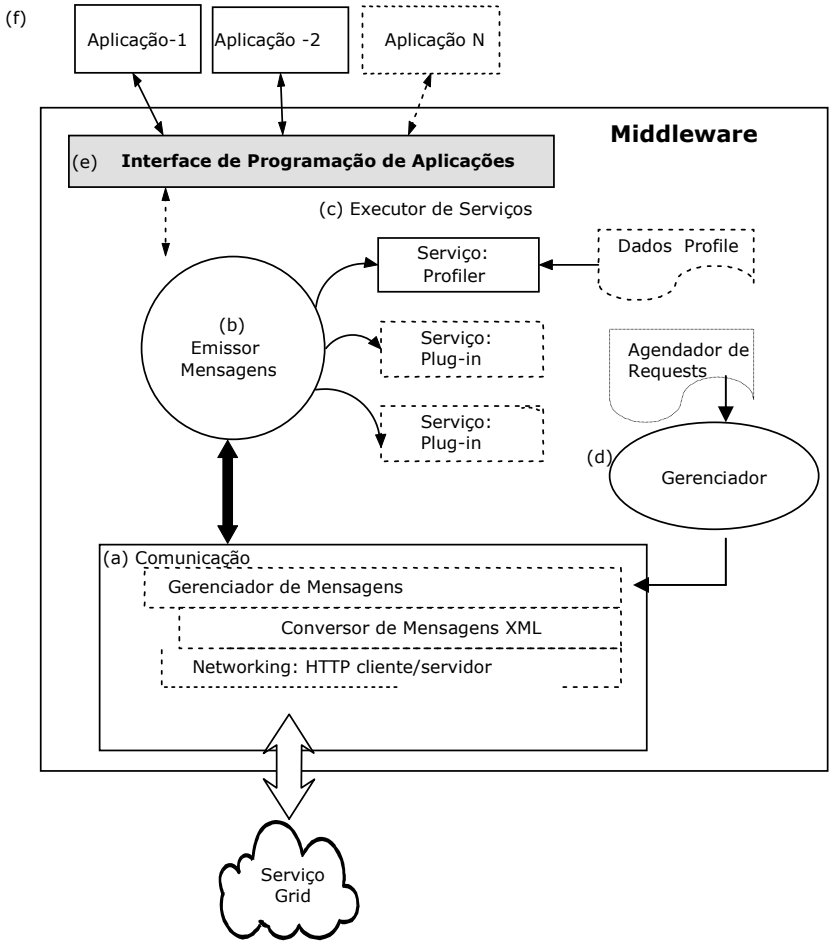

Figura 6.2 Modelo de referência para arquitetura do software middleware

6.1.2 ARQUITETURA DA ESTRUTURA GRID

Especificamente em relação à arquitetura da grid, seu aprimoramento é representado na figura 6.2, a qual apresenta a relação entre os nodos da grid e os serviços por ela providos.

Os nodos da grid representados na figura 6.2 pela letra "a" e os serviços "b" estão interconectados por meio da estrutura de comunicação que é a mesma para todos os nodos da grid. O serviço grid é composto por:

(a) Os nodos da grid, já descritos anteriormente.

(b) Os serviços grid, que são os provedores das respostas para as chamadas de cada nodo.

(c) Serviço de diretórios, que é uma estrutura onde os serviços estão registrados e correlacionados aos nodos capazes de implementá-los.

(d) Gerenciador de Tarefas, é a estrutura que controla os *requests* para os serviços.

(e) Roteador de mensagens faz o roteamento das mensagens, ou seja, alguns nodos podem não pertencer a rede TCP/IP então este módulo usa de técnicas como *store-and-forward* (armazena e envia) entre outras para realizar a comunicação de forma concisa.

(f) Gerenciador da grid, esta estrutura conecta outros serviços que possam ser prestados por outros middlewares, como por exemplo o Globus.

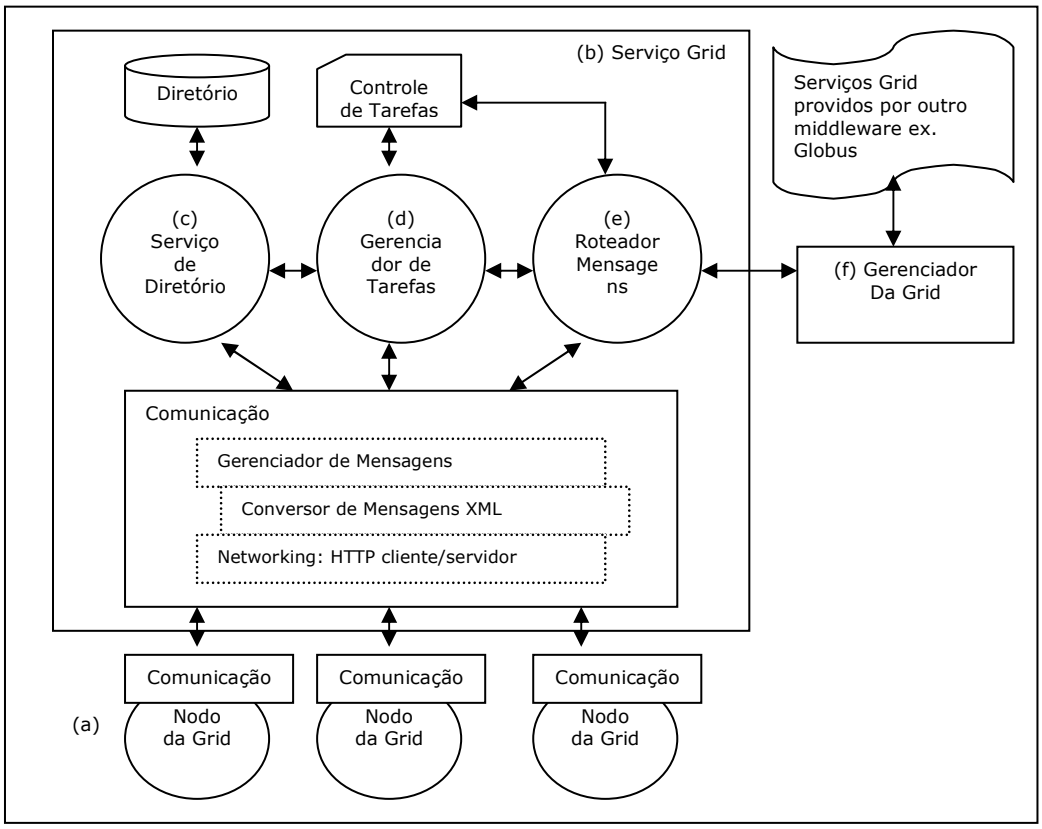

Figura 6.3 Visão Abstrata da Arquitetura de Serviço da Grid

6.2 GUIA PARA OPERAÇÕES ENTRE NODOS

Em um ambiente onde a confiança não é inerente e por outro lado é característica principal, como em um ambiente de computação móvel, a execução de tarefas de modo distribuído é o modo mais aceitável para um correto funcionamento [THAM 2005].

Neste tipo de ambiente de processamento, os nodos enviam tarefas, *Tasks* (que são estruturas de dados) para serem executadas por outros nodos e receber o resultado da tarefa enviada como um *TaskResult*. Aqui cabe uma observação, a de que os comandos e estruturas estão em inglês para permitir uma maior padronização.

Uma estrutura de dados de uma tarefa (TASK) contém:

SERVICE-REQUEST
TASK-ID
TIMESTAMP
TTL
PARAMETER-1
PARAMETER-2
…
PARAMETER-N

Onde:

SERVICE-REQUEST é o serviço requisitado, que está associado ao Executor de Serviços que entrega a um nodo capaz de executar este serviço requisitado.
TASK-ID é a identificação da tarefa.
TIMESTAMP: timestamp associada a tarefa
TTL: time to live, é o máximo de tempo para execução de uma tarefa, e quem gerencia este tempo é o serviço grid Gerenciador de Tarefas.
PARAMETER-N: são os parâmetros da tarefa e podem ser do tipo STRING, INTEGER ou LIST.

A estrutura de dado para o resultado de uma tarefa (TASK-RESULT) contém:

RESPONSE-CODE
TASK-ID
[PART]

[NUMBER-PARTS]
TIMESTAMP
PAYLOAD

RESPONSE-CODE: é o código de resposta da tarefa executada e pode ser: OK, ERROR, NOT-UNDERSTOOD, REFUSED
TASK-ID é a identificação da tarefa em execução e esta é utilizada para o item seguinte.
PART e NUMBER-PARTS denota o número da tarefa ou o número que identifica as várias partes de uma tarefa, já que esta pode ser quebrada em partes para ser enviada pela rede.
TIMESTAMP quando a resposta foi composta.
PAYLOAD é a descrição da resposta.

Exemplo de tarefa (task) e resultado desta tarefa (task-result) de forma abstrata:

MULTIPLY
:param(integer) 128
:param(integer) 2

OK
:result(integer) 256

6.3 ESTRUTURAS DAS MENSAGENS

A estrutura para mensagens em uma comunicação entre nodos segue o padrão definido pela "especificação da estrutura da mensagem FIPA XC00061 FIPA ACL" [FIPA 2006]. Este documento contém especificações para os elementos da mensagem incluindo: a lista de elementos atuais da mensagem, os procedimentos para a manutenção desta lista, os critérios para adotar elementos novos na lista. O objetivo de padronizar as mensagens no formato FIPA ACL são: ajudar a assegurar a interoperabilidade fornecendo um conjunto padrão de estruturas para cada mensagem do ACL, e para fornecer um processo bem definido para manter este conjunto.

Para a troca de mensagens entre nodos a mensagem será representada em XML, seguindo o padrão proposto de "representação da mensagem FIPA XC00071 FIPA ACL" pela especificação de XML.

6.4 CODIFICAÇÃO XML PARA UMA TAREFA E PERFIS

Uma tarefa é representada em XML da seguinte forma:

```
<task>
        <service>$PERFORMATIVE$</service>
        <task-id>$ID$</task-id>
        <timestamp>$TIMESTAMP$</timestamp>
        [<parameter sequence="$COUNTER"
        type="$integer|string|list$">$VALUE$</parameter>]*
 </task>
```

Exemplo de uma tarefa usando XML:

```
<task>
        <service>MULTIPLY</service>
        <task-id>node-1-132131</task-id>
        <timestamp>32132112<timestamp>
        <parameter sequence="1" type="integer">128</parameter>
        <parameter sequence="2" type="integer">2</parameter>
</task>
```

6.4.1 SERVIÇO DE PERFIL (PROFILE)

O serviço de perfil é imperativo para cada execução do nodo. Este serviço é invocado recebendo/enviando tarefas como GET-PROFILE (para receber/enviar dados estáticos) e um GET-PROFILE-DYNAMIC (para receber/enviar dados dinâmicos). O nodo responderá com informações de perfil respectivamente estáticos ou dinâmicos. Um exemplo de uma requisição de perfil estático e dinâmico está descrito abaixo.

6.4.2 PERFIL ESTÁTICO DE DADOS

Os dados de um perfil estático são aqueles que não mudam freqüentemente por um determinado período, tais como: versão do sistema operacional, tipo de processador, quantidade memória entre outros. Estes dados são recuperados do nodo através de um GET-PROFILE.

Exemplo: Nodo-1 requisita o perfil do Nodo-2.

```
Nodo-1 requisita:
        (REQUEST
                :sender node-1
                :receiver node-2
                :content (GET-PROFILE
                                :task-id node-1-132131
                                :timestamp 321321312312
                        )
                :encoding: br.ufsc.lrg.grid.encoding.std
                :ontology br.ufsc.lrg.grid.ontology.std
                :reply-with node1-abc
        )

Nodo-2 responde:
        (REQUEST
                :sender node-2
                :receiver node-1
                :content (OK
                        :task-id node-1-132131
                        :timestamp 321321312888
                        :type(string) desktop-computer
                        :os(string) WindowsXP service pack 2
                        :running-environment(string) Java JRE 1.5
                        :cpu(string) 586 x86
                        :memory(integer) 512
                        :disk(integer) 0
                        :display(string) graphics
                        :display-width(integer) 170
                        :display-length(integer) 256
                        :display-colours(integer) 256
                        )
                :encoding: br.ufsc.lrg.grid.encoding.std
                :ontology br.ufsc.lrg.grid.ontology.std
                :reply-with node1-abc
        )
```

Dados estáticos recuperados Pela requisição do nodo 1

Descrição da tarefa

6.4.3 PERFIL DINÂMICO DE DADOS

Os dados em um perfil dinâmico têm por característica estarem em constante mudança, exemplo destes dados são: quantidade de memória livre, localização do nodo (dispositivo móvel) ente outros. Estes dados sao requisitados através de uma tarefa PROFILE-GET-DYNAMIC.

Exemplo: Nodo-1 requisita o perfil do Nodo-2.

Nodo-1 requisita:

```
(REQUEST
        :sender node-1
        :receiver node-2
        :content (PROFILE-GET-DYNAMIC
                :task-id node-1-132131
                :timestamp 321321312312
            )
        :encoding: br.ufsc.lrg.grid.encoding.std
        :ontology br.ufsc.lrg.grid.ontology.std
        :reply-with node1-abc
)
```

Nodo-2 responde:

```
(REQUEST
        :sender node-2
        :receiver node-1
        :content (OK
                :task-id node-1-132131
                :timestamp 321321312888
                :cpu-free(integer) 20
                :memory-free(integer) 127
                :disk-free(integer) 0
                :location(list): (350.0032,250.003232,12)
                )
        :encoding: br.ufsc.lrg.grid.encoding.std
        :ontology br.ufsc.lrg.grid.ontology.std
        :reply-with node1-abc
)
```

Dados dinâmicos recuperados Pela requisição do nodo 1

Descrição da tarefa

6.5 INTERFACE DE PROGRAMAÇÃO DE APLICAÇÕES

Nesta seção é apresentado um exemplo de uma programação básica de estruturas de tarefas (Task e ResultTask) baseadas na padronização de mensagens FIPA e originadas no módulo de Interface de programação de aplicações.

ESTRUTURA DE DADOS:

Task
TaskResult

INTERFACES:

MessageDispatcherInterface
    process(performative, Task)

ServiceInterface
    process(Task)

MÓDULOS:

Node(name, ProfilerService)
    register(URL)
    unRegister(URL)
    addService(id,ServiceInterface)
    removeService(id)
    addScheduledTask(id,startTime,deltaTime,Taks)
    removeScheduledTask(id)
    createTask(performative, ParameterList) ->Task
    submitTask(Task)->StatusCode
    submitTask(nodeName, Task)->StatusCode
    receiveTaskResult(id)->taskResult
    hasTaskResult(id)->Boolean

    # For asynchronous processors (based on TASK BROKER)
    hasTaskToProcess()->Integer
    receiveTasksToProcess(Integer)->List of Task
    processTask(Task)

    # For HTTP-server enabled nodes
    startListener(Port, MessageDispatcherInterface)
    stopListener(Port)

Exemplo de operação do Nodo-1

```
Service profiler = new ProfilerService();
Service collector = new CollectorService();

Node node = new Node("node-1", profiler);
node.addService("collect-data", collector);
node.startListener(50080, node.defaultDispatcher);
node.register(http://gridservice.lrg.ufsc.br);
```

Exemplo de operação do Nodo-2

```
Service profiler = new ProfilerService();

Node node = new Node("node-2", profiler);
node.register(http://gridservice.lrg.ufsc.br);

Task task = node.createTask("collect-data", null);
if(node. submitTask("node-1", task) == TASK_SUBMIT_OK){
        while(!node.hasTaskResult(task.id) && (now()-task.timestamp>TIMEOUT));

        if(now()-task.timestamp<TIMEOUT){
                TakResult result = node.receiveTaskResult(task.id);
                print(result.payload);
        }
}
```

6.6 CONSIDERAÇÕES E DISCUSSÃO DO CAPÍTULO

Neste capítulo foi realizado um aprimoramento da especificação do middleware, assim como um protótipo para troca de mensagens XML por meio de código C. Este aprimoramento tem como objetivo se tornar um documento de referência seja para futuras implementações seja para estudo para trabalhos futuros.

7. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

7.1 CONSIDERAÇÕES FINAIS

Nesse trabalho foi apresentada, segundo seu objetivo geral, uma proposta para um middleware de integração das tecnologias de Computação Móvel e Grades de Computadores para que este forneça uma solução para a área de Computação Distribuída para a criação de aplicações de serviços móveis. Este middleware integra as características oferecidas pela Computação em Grade, i.e., comunicação de dados, serviços de diretório e outros, a serviços comuns na área de computação móvel, tais como sensibilidade ao contexto e interfaces dos dispositivos móveis e oferece uma interface de programação unificada, que possibilite ao desenvolvedor da solução utilizar um único middleware para acessar aos recursos de ambas as tecnologias.

Definiram-se os requerimentos para o software proposto como sendo computacionalmente leve, visando dispositivos com processadores de 20Mhz e capacidade de memória de 512Kbytes ou maiores, interface de programação genérica e compatibilidade com as plataformas Java J2SE e Java J2ME.

Esse trabalho contribui com uma infra-estrutura para a implementação dos serviços visando o problema de implementação do serviço cujo desafio é integrar a infra-estrutura de computação móvel disponível e as soluções da computação em grade. Almeja-se contribuir com uma estrutura de software que facilite a criação de serviços móveis distribuídos e forneça ao desenvolvedor uma plataforma que integre as soluções na área da computação distribuída com os recursos e capacidades da computação móvel.

Ponderamos que grids também não resolve todos os tipos de problemas, mas sim contribui para uma nova abordagem para solução de problemas que antes necessitavam de um conjunto de tecnologias. A possibilidade de tornar esta grid móvel aumenta ainda mais a gama de solução de problemas que neste contexto envolvem mobilidade, informação em tempo real e homogeneização de recursos.

7.2 TRABALHOS FUTUROS

Como trabalhos futuros as seguintes atividades podem ser avaliadas:

- Especificar a arquitetura de grades de dispositivos móveis para um outro middleware, tal como, Globus ou Condor, isto é, utilizar do poder computacional e da infra-estrutura destes e mapear para dispositivos móveis.
- Desenvolvimento de um protótipo que concretize toda a especificação pesquisada durante essa dissertação.
- Desenvolver aplicações para outras áreas aumentando o poder do middleware grid móvel.
- Melhorar e acrescentar mais regras/métodos para gerenciamento dos nodos.
- Fazer medições de tempo e capacidade de transmissão das informações em uma determinada rede.
- Fazer testes com diferentes tipos de dispositivos móveis em diferentes tipos de redes.
- Realizar outros estudos de casos em outras áreas de atuação.

REFERÊNCIAS

[ACE 2005] Advanced Collaborative Environments Research Group site. Acessado em 26 de dezembro de 2005, disponível em: https://forge.gridforum.org/projects/ace-rg

[ASSUNÇÃO 2004] ASSUNÇÃO, Marcos (2004). "implementação e análise de uma arquitetura de grids de agentes para a gerência de redes e sistemas". Dissertação de Mestrado. PPGCC-UFSC, Florianópolis, 2004.

[BHATTI 2005] Bhatti, R.; Shafiq, B.; Shehab, M.; Ghafoor, A. Distributed access management in multimedia IDCs. IEEE Computer Volume 38, Issue 9, Sept. 2005 Page(s):60 - 69 Digital Object Identifier 10.1109/MC.2005.296

[BUYYA 2005] Buyya Rajkumar; Yu Jia. "A Taxonomy of Workflow Management Systems for Grid Computing" eprint arXiv:cs/0503025 Journal 03/2005 ARXIV 29 pages

[CAMPBELL 1999] Campbell Andrew T., Geoff Coulson, Michael E. Kounavis. "Managing Complexity: Middleware Explained," *IT Professional*, vol. 01, no. 5, pp. 22-28, September/October, 1999.

[CASTANO 2004] F. J.Gonzalez-Castano, J.Vales-Alonso, M.Livny, E.Costa-Montenegro, L.Anido-Rifon, "Condor grid Computing from Mobile Handheld Devices," Mobile Computing and Communications Review, 2004. Vol. 6, No. 2, pp18-27

[DAS 2003] S. Das and S. K. Das, editors. 5th International Workshop on Distributed Computing, volume 2918 of Lecture Notes in Computer Science, Kolkata, India, December 2003.

[FERREIRA 2004] FERREIRA L.; DIRKER J.; HERNANDEZ O.; QUEIROZ C.; ROHLEDER V.; HYATT M.; "he information grid, Part 1: The infrastructure" IBM site, 2004. Acessado em 06 de dezembro de 2005, disponível em: http://www-128.ibm.com/developerworks/grid/library/gr-info1/

[FIPA 2006] FIPA Communicative Act Library Specification. Foundation for Intelligent Physical Agents, 2000. Acessado em 15 de abril de 2006, disponível em: http://www.fipa.org/specs/fipa00037/.

[FOSTER 2005] FOSTER I. Globus toolkit version 4: Software for service-oriented systems. IFIP In ternational Conference on Network and Parallel computing, Springer-Verlag LNCS 3779:pp 2-13, 2005.

[FOSTER 1999] FOSTER, I.; KESSELMAN, C. The Grid: Blueprint for a New Computing Infrastructure. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1999.677p.

[FOSTER 2001] FOSTER, I; KESSELMAN, C.; TUECKE S. The Anatomy of the Grid:
Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, v.15 n.3, 2001.

[FOSTER 2002] What is the Grid? A Three Point Checklist. I. Foster, GRIDToday, July 20, 2002.

[FOSTER 2005] FOSTER, I; TUECKE S. The Different Faces of IT as Service Ian Foster and Steven Tuecke's article from the Enterprise Distributed Computing issue of ACM Queue (Vol. 3, No. 6 - July/August 2005)

[GAI 2005] Gai S. “Internetworking IPv6 with Cisco Routers”. Acessado em 28 de dezembro de 2005 , disponível em: http://www.ip6.com

[GAYNOR 2004] Gaynor, M. Moulton, S.L. Welsh, M. LaCombe, E. Rowan, A. Wynne, J. Integrating wireless sensor networks with the grid. Internet Computing, IEEE Volume 8, Issue 4, July-Aug. 2004 Page(s):32 – 39 Digital Object Identifier 10.1109/MIC.2004.18

[GUIGERE 2001] E. Guigere. Java 2 Micro edition: The ultimate guide on programming handheld and embedded devices. John Wiley and Sons, Inc., USA, 2001.

[GGF 2005] Global Grid Forum, site do Forum, acessado em 18 de dezembro de 2005, disponível em: http://www.ggf.org/

[GRIDLAB 2005] GridLab Project web-site, A Grid Application Toolkit and Testbed GridLab Mobile, Acessado em 07 de Novembro de 2005, disponível em: http://www.gridlab.org/WorkPackages/wp-12/

[GRIDBUS 2005] Gridbus Project web-site, Grid Computing and Distributed Systems Laboratory, Acessado em 06 de Novembro de 2005, disponível em: http://www.gridbus.org/

[GRIDFORUM 2005] GridForum Website “Understanding Grids” Acessado em 08 de dezembro de 2005, disponível em: http://www.gridforum.org/UnderstandingGrids/ggf_grid_understand.php

[HENRICKSEN 2001] Henricksen k; J. Indulska and A. Rakotonirainy. Infrastructure for pervasive computing: Challenges. In K. Bauknecht, W. Brauer, and T. A. MÄuck, editors, Informatik 2001: Wirtschaft und Wissenschaft in der Network Economy Visionen und Wirklichkeit, volume 1 of Tagungsband der GI/OCG-Jahrestagung, pages 214-222, Universitat Wien, September 2001.

[J2ME 2005] Java 2 Micro Edition (J2ME) web-site, sun corporation, acessado em 12 de Novembro de 2005, disponível em: http://java.sun.com/j2me.

[J2SE 2005] Java 2 Standart Edition (J2SE) web-site, sun corporation, acessado em 12 de Novembro de 2005, disponível em: http://java.sun.com/j2se.

[JEFFERY 2002] JEFFERY, K. G. Knowledge, Information and Data, A briefing to the Office ofScience and Technology. Disponível em:<www.itd.clrc.ac.uk/Publications/1433/KnowledgeInformationData20000124.htm>. Acesso em: 02 set. 2002.

[KOCH 2005] F. Koch, J.-J. C. Meyer, F. Dignum, and I. Rahwan. Programming deliberative agents for mobile services: the 3apl-m platform. In Proceedings of AAMAS'05 Workshop on Programming Multi-Agent Systems (ProMAS'2005)., 2005.

[LDAP 2005] LDAP, openLDAP, acessado em 15 de Novembro de 2005, disponível em: http://www.openldap.org/

[LEGION 2005] Legion, World Wide Virtual Computer, Acessado em 06 de Novembro de 2005, disponível em: http://legion.virginia.edu/

[LITKE 2004] A. Litke, D. Skoutas, and T. Varvarigou, “Mobile grid computing: Changes and challenges of resource management in a mobile grid environment,” in Proc. of Practical Aspects of Knowledge Management (PAKM 2004), Austria, December 2004.

[MCKNIGHT 2004] McKnight, L.W., Howison, J. and Bradner, S. (2004) “Wireless grids: Distributed resource sharing by mobile, nomadic, and fixed devices”, IEEE Internet Computing, vol. 8, nº. 4, pp. 24-31.

[NAVARRO 2005] NAVARRO F., SCHULTER A., KOCH F., ASSUNÇÃO M., C. B. WESTPHALL. “Grid Middleware for Mobile Decision Support Systems”. 5th International Conference on Networking (ICN'06).

[OBJWEB 2005] Object Web Open Source Middleware. Acessado em 28 de dezembro de 2005, disponível em: http://middleware.objectweb.org/

[OGSI 2005] University of Virginia Grid Computing Group site, acessado em 20 de dezembro de 2005, disponível em: http://www.cs.virginia.edu/~gsw2c/ogsi.net.html

[SATYANARAYANAN 2001] SATYANARAYANAN M; Pervasive computing: vision and challenges. IEEE Personal Communications, 8(4):10-17, 2001.

[SETI 2005] SETI@HOME. Search for Extraterrestrial Intelligence, 2005. Site do projeto, acessado em 15 de dezembro de 2005, disponível em: http://setiathome.ssl.berkeley.edu

[SPOONER 2004] Spooner D.P., J. Cao, S. A. Jarvis, L. He, and G. R. Nudd. “Performance-aware Workflow Management for Grid Computing”. The Computer Journal, Oxford University Press, London, UK, 2004.

[THAM 2005] THAM chen-khong; BUYYA Rajkumar (2005) “SensorGrid: Integrating Sensor Networks and Grid Computing”. CSI Communications, July 2005. pages 24-29, Vol.29, No.1, Computer Society of India (CSI) Publication.

[UNICORE 2005] Unicore, UNIform Interface to COmputer Resources, Acessado em 07 de Novembro de 2005, disponível em: http://www.unicore.org/

[USKELA 2003] S. Uskela. Key concepts for evolution toward beyond 3g networks. IEEE Wireless Communications, 10(1):43-48, 2003.

[WEISER 1991] WEISER Mark, "The Computer for the 21st Century". Acessado em 06 de dezembro de 2005. Disponível em: http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[XML 2005] XML, W3C Consortium, acessado em 13 de Novembro de 2005, disponível em: http://www.w3.org/XML/

[YANGUANG 2004] Yanguang Wang; Yong Xue; Jianqin Wang; Jiakui Tang; Guoyin Cai; YincuiHu; Shaobo Zhong; Ying Luo. "Analysis of remotely sensed images on the Grid platform from mobile handheld devices - a trial" Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 IEEE International Volume 5, 20-24 Sept. 2004 Page(s):2964 - 2966 vol.5

ANEXOS

Nesta seção estão descritos alguns códigos referentes à implementação da proposta.

Código para Login de um dispositivo e cadastro de tarefas:

----------Início----------

```
package testes;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;

public class Login extends MIDlet implements CommandListener, Runnable {
    private Display tela;

    private Form formLogin;

    private TextField login;

    private TextField senha;

    private Command ok, voltar, cadTask, lerTask;

    public int tread = 0; // Variavel para controlar as chamadas do run()

    public Login() {
        tela = Display.getDisplay(this);
        login = new TextField("Login:", "", 10, TextField.ANY);
        senha = new TextField("Senha:", "", 6, TextField.PASSWORD);

        ok = new Command("Ok", Command.SCREEN, 2);
        voltar = new Command("Voltar", Command.BACK, 1);
        cadTask = new Command("Cad.Task", Command.SCREEN, 2);
        formLogin = new Form("Digite Matrícula e Senha");
        formLogin.append(login);
        formLogin.append(senha);
        formLogin.addCommand(ok);
        formLogin.addCommand(voltar);
```

```
        tela.setCurrent(formLogin);
        formLogin.setCommandListener(this);
    }

    public void startApp() {
        tela.setCurrent(formLogin);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {
        if (c == voltar) {
                destroyApp(false);
                notifyDestroyed();
        } else if (c == ok) {
                try { // chama o validaLogin()
                        Thread minhathread = new Thread(this);
                        tread = 1;
                        minhathread.start();// vai chamar o metodo run()

                } catch (Exception e) {
                        System.out.println(e.toString());
                }
        }
        // Opçao que chama o procedimendo de cadastro de Tarefas
        else if (c == cadTask) {
                Thread minhathread2 = new Thread(this);
                tread = 2;
                minhathread2.start();// vai chamar o metodo run()
        }
    }

    // metodo que chama o validaLogin() utilizando uma thread
    public void run() {

        if (tread == 1)// Para chamadas do ValidaLogin
        {
                String loginSenha = login.getString() + "/" + senha.getString();
                try {
                        validaLogin(loginSenha);
                } catch (Exception e) {
                        System.out.println(e.toString());
                }
        }
    }

    private void validaLogin(String loginSenha) throws IOException {
        HttpConnection http = null;
        InputStream iStrm = null;

        // Pegar o Login e a Senha no Formato ( login/senha )
//      String loginSenha = login.getString() + "/" + senha.getString();
        System.out.println(loginSenha);
        String url = "http://127.0.0.1/arquivo.txt?";
```

```
        try {
                // Abre conexão com o servidor
                http = (HttpConnection) Connector.open(url);
                // Envia requisição GET
                http.setRequestMethod(HttpConnection.GET);
                // Agurada a resposta do servidor
                if (http.getResponseCode() == HttpConnection.HTTP_OK) {
                        iStrm = http.openInputStream();

                        // Recebe os dados enviados pelo servidor
                        int length = (int) http.getLength();
                        if (length > 0) {
                                byte servletData[] = new byte[length];
                                char data[] = new char[length];
                                int j = 0;
                                iStrm.read(servletData);
                                for (int i = 0; i < length; i++) {
                                        data[j] = (char) servletData[i];
                                        j++;
                                }

                                String dado_retorno = new String(data);
                                System.out.println(dado_retorno);

                                if (dado_retorno.equals(loginSenha)) {
                                        Alert alert = new Alert("OK", "Usuário cadastrado",
                                                        null, AlertType.CONFIRMATION);
                                        alert.setTimeout(6000);
                                        tela.setCurrent(alert);
                                        formLogin.removeCommand(ok);
                                        formLogin.addCommand(cadTask);
                                } else {
                                        Alert alert = new Alert("Erro",
                                                        "Usuário não cadastrado", null,
    AlertType.ERROR);
                                        alert.setTimeout(6000);
                                        tela.setCurrent(alert, formLogin);
                                }
                        }

                } else
                        formLogin.append("Impossível ler o dado");
        }

        catch (Exception e) {
                formLogin.append("Network error");
                System.out.println(e.toString());
        } finally {
                if (iStrm != null)
                        iStrm.close();
                if (http != null)
                        http.close();
        }

    }

}
```

----------Fim----------

Código que informa o resultado em um formulário, de um Login.

----------Início----------

```
package testes;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class ExemploForm1 extends MIDlet implements CommandListener {
    Display tela;

    Form login, resultado;

    TextField nome, senha;

    StringItem resultadoMsg;

    Command sair, proximo;

    public ExemploForm1() {
        // inicializar Commands
        this.sair = new Command("Sair", Command.EXIT, 0);
        this.proximo = new Command("Prox", Command.SCREEN, 1);
        // form de login
        this.login = new Form("Login");
        this.nome = new TextField("Nome:", "", 20, TextField.ANY);
        this.senha = new TextField("Senha:", "", 20, TextField.ANY
                            | TextField.PASSWORD);
        // adiciona-se os componentes ao Form Login
        this.login.append(this.nome);
        this.login.append(this.senha);
        this.login.addCommand(this.sair);
        this.login.addCommand(this.proximo);
        this.login.setCommandListener(this);
        // form de resultado
        this.resultado = new Form("Resultado");
        this.resultadoMsg = new StringItem("", "");
        // adiciona-se o componente ao Form Resultado
        this.resultado.append(this.resultadoMsg);
        this.resultado.addCommand(this.sair);
        this.resultado.setCommandListener(this);
    }

    public void startApp() {
        this.tela = Display.getDisplay(this);
        this.tela.setCurrent(this.login);
    }
```

```
    public void pauseApp() {
    }

    public void destroyApp(boolean condicional) {
    }

    public void commandAction(Command c, Displayable d) {
        if (c == this.sair) {
                this.destroyApp(true);
                this.notifyDestroyed();
        }
        if (c == this.proximo) {
                // O Label sempre aparecerá antes do Text não importando
                // a ordem que vc adicione ele ao componente
                // faça o teste trocando de ordem as chamdas.
                this.resultadoMsg.setLabel(this.nome.getString() + " ");
                this.resultadoMsg.setText(this.senha.getString());
                this.tela.setCurrent(this.resultado);
        }
    }
}
```

----------Fim----------

Código exemplo de criação de um menu de opções.

----------Início----------

```
Package testes;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExemploCommandsAndTicker extends MIDlet implements CommandListener {
    Display tela;

    TextBox texto;

    Ticker ticker;

    Command sair, opcao1, opcao2, opcao3, opcao4, opcao5;

    public ExemploCommandsAndTicker() {
        // instanciar TextBox
        this.texto = new TextBox("Commands", "Aplicação iniciada", 20,
                        TextField.ANY);
        this.ticker = new Ticker(
                        "Exemplo de ticker que deve rodar em cima da tela");
        // seta o ticker na tela
        this.texto.setTicker(this.ticker);
        // comandos do TextBox
        this.sair = new Command("Sair", Command.EXIT, 0);
        this.opcao1 = new Command("Opção 1", Command.SCREEN, 1);
        this.opcao2 = new Command("Opção 2", Command.SCREEN, 2);
```

```
        this.opcao3 = new Command("Opção 3", Command.SCREEN, 3);
        this.opcao4 = new Command("Opção 4", Command.SCREEN, 4);
        this.opcao5 = new Command("Opção 5", Command.SCREEN, 5);
        // relacionar Commands com TextBox
        this.texto.addCommand(sair);
        this.texto.addCommand(opcao1);
        this.texto.addCommand(opcao2);
        this.texto.addCommand(opcao3);
        this.texto.addCommand(opcao4);
        this.texto.addCommand(opcao5);
        // registrar TextBox com o CommandListener
        this.texto.setCommandListener(this);
    }

    public void startApp() {
        // obter tela do dispositivo
        this.tela = Display.getDisplay(this);
        // setar Displayable corrente para a tela
        this.tela.setCurrent(this.texto);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean i) {
    }

    // aqui precisamos de um command e um displayable
    // o displyable será o componente ativo na tela
    // porque ele extend de Display
    public void commandAction(Command c, Displayable d) {
        if (c == this.sair) {
                // sair da aplicação
                this.destroyApp(true);
                this.notifyDestroyed();
        }
        if (c == this.opcao1) {
                // alterar texto do TextBox
                this.texto.setString("Opção 1 selecionada.");
                this.ticker.setString("Opção 1 Selecionada");
        }

    }
}
```

----------Fim----------