Hans Alberto Franke

GRIDM: MIDDLEWARE PARA INTEGRAR DISPOSITIVOS MÓVEIS, SENSORES E COMPUTAÇÃO EM GRID

TRABALHO DE CONCLUSÃO DE CURSO

Florianópolis, SC – BRASIL
2007

GridM: Middleware para Integrar Dispositivos Móveis, Sensores e Computação em Grid

Por

Hans Alberto Franke

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Catarina, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Florianópolis, SC – BRASIL.
Janeiro de 2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO CIENTÍFICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

A COMISSÃO EXAMINADORA, ABAIXO ASSINADA, APROVA O TRABALHO DE CONCLUSÃO DE CURSO

GridM: Middleware para Integrar Dispositivos Móveis, Sensores e Computação em Grid

ELABORADO POR

HANS ALBERTO FRANKE

COMO REQUISITO PARCIAL PARA A OBTENÇÃO DO TÍTULO DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

COMISSÃO EXAMINADORA: _____
Carlos Becker Westphall, Dr. – Orientador

_____
Fernando Luiz Koch, MSc.

_____
Mario Dantas, Dr.

Florianópolis, ___ janeiro de 2007.

Agradecimentos

Acima de tudo aos meus pais Alberto e Sandra, que sempre me apoiaram e mostraram a importância do estudo na vida e nunca se importaram em bancar todas as despesas para que eu pudesse me dedicar apenas a estudar. A minha irmã Anna por todo o carinho e simpatia que uma irmãzinha deve ter.

Ao pessoal do LRG: Westphall por acreditar no meu trabalho oferecendo uma bolsa de pesquisa, possibilitando assim que produzisse um trabalho dentro da metodologia científica. Ao Koch e Oberdan, pelas dicas e idéias tanto na parte teórica quanto prática, na geração da idéia do projeto, na escrita, nos caminhos a seguir, na publicação de artigos.

Aos amigos excepcionais que fiz durante o curso e que estiveram presentes em todas as horas: Arthur, Douglas, Duda, Edson, Luiz Fernando, Marafon, Pedro, Garcia, Bazzo, Kleber, Giandrei, Grazi.

E um agradecimento especial a Déia que sempre esteve ao meu lado apoiando e incentivando durante estes 4 anos mostrando ser uma pessoa única e especial.

Sumário

Resumo

Autor: Hans Alberto Franke
Orientador: Dr. Carlos Becker Westphall

Neste trabalho propõe-se a criação de um middleware que ofereça suporte a integração entre rede de sensores (WSN), grids computacionais e dispositivos móveis. Discute-se como o middleware fornece serviços como gerência, comunicação de dados, diretório de serviços, descoberta de recursos e segurança, provendo ao desenvolvedor um conjunto de recursos reutilizáveis e homogêneos. Serão discutidos também problemas inerentes a este cenário e a forma de solucioná-los.

Palavras chaves: Aplicações em Grid; Computação Móvel; Middleware.

Abstract:

Author: Hans Alberto Franke
Advisor: Dr. Carlos Becker Westphall

In this work, we introduce a middleware for grid computing that supports network of sensors, like in a network management environment. We present its features, such as manageability, data communication, services directory, resources discovery and security; discuss about inherent problems, and; propose solutions.

Keywords: Grid Applications, Mobile Computing, Middleware.

Lista de tabelas

Lista de figuras

1 Introdução

A solução fornecida pela computação em Grid vai além de criar redes de processamento e armazenamento distribuídos. Argumenta-se que é também uma técnica poderosa para promover a homogeneização ou a virtualização criando redes de fornecedores de serviço onde cada nó compartilha serviços ao Grid. Particularmente nas Wireless Sensor Networks (WSN), os nós executam em diversos ambientes, com sistema operacional (SO) diferente (Windows, Linux, J2ME), em hardware (desktop, Personal Digital Assistent (PDA), dispositivos móveis), e em redes (TCP/IP, Wi-Fi, WAP). As aplicações devem lidar com as diversidades e entregar um serviço de qualidade. Conseqüentemente, a pergunta está em *Como integrar estes ambientes em uma infra-estrutura comum para fornecer serviços*? ou; *Qual a plataforma ideal para promover a virtualização em um Grid de WSN* ?

Diversos estudos estão sendo realizados no âmbito desta pergunta. Por exemplo, na comunicação de dados, simples, rápida e mais confiável em redes de transmissão de dados [USKELA 2003]; no hardware, novos dispositivos com mais recursos e em plataformas mais simples de desenvolvimento, e; na tecnologia de programação há a criação de metodologias do desenvolvimento capazes de lidar com as restrições de recursos existentes [GUIGERE 2001]. Entretanto, sugere-se que falta, na literatura, ainda uma solução completa e reutilizável para integrar WSN, provedores de serviços, e dispositivos estáticos (workstations) e móveis (PDA's, celulares, ...) de computação. Esta argumentação será abordada quando se fizer a revisão dos trabalhos relacionados, no capítulo 4. Sugere-se que esta abertura na literatura apresenta uma oportunidade de contribuir com um middleware que integre os dispositivos móveis, WSN e o Grid.

Neste trabalho, será apresentado o middleware: Grid-M. Ele é uma plataforma para construir aplicações de computação em Grid e aplicações em dispositivos embargados e móveis. Fornece uma Application Programming Interface (API) para conectar aplicações desenvolvidas em Java em um ambiente de computação em Grid. Seu perfil runtime é pequeno bastante para ser usado em aplicações de computação móvel. Acredita-se que as soluções do Grid-M respondam à pergunta sobre uma plataforma para promover a homogeneização neste ambiente de desenvolvimento.

1.1 Motivação e Análise

1.1.1 Motivação

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:

a) 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?
b) 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 estas perguntas, o presente trabalho apresenta a integração de computação em grid e computação móvel, através da criação de um grid middleware para o desenvolvimento de serviços móveis. O grid middleware descrito neste trabalho, recebe o nome de GRID-M. Ele oferece serviços como comunicação de dados, diretório de serviços, descoberta e segurança, desta forma provendo ao desenvolvedor um conjunto de recursos reutilizáveis e homogêneos, o que é garantido pela utilização do middleware como uma API de programação para facilidade do usuário.

1.1.2 Análise de Requisitos

Para que este sistema seja realizável, alguns requisitos são necessários, e seguindo o paradigma de Orientação a Objetos, serão propostos alguns métodos e classes para regulamentar este cenário.

Para que haja comunicação entre os dispositivos e os sensores a) alguns métodos de comunicação de dados; b) para que não haja interferência externa indevida aos dados faz-se necessário alguns métodos de segurança que façam o controle de acesso ao sistema; c) para

agregarmos recursos a grid, devemos ter métodos de registro de recursos que permitam, por exemplo, o controle de entrada e saída de novos sensores ou assistentes pessoais cadastrados no sistema; d) para que seja possível controlarmos as tarefas a serem processadas, os dispositivos pertencentes a grid e outros recursos, devemos ter métodos para gerenciamento dos dispositivos, que permitam gerenciar sensores, coletores e assistentes pessoais de forma centralizada, apesar da distribuição, dinamicidade do ambiente e heterogeneidade dos dispositivos; e) para que todo sistema possa ser mais facilmente operável, devemos ter interfaces amigáveis e padronizadas que possibilitem uma melhor interação; e, f) métodos para a coordenação de trabalhos entre os usuários móveis e que permita colaboração entre os assistentes pessoais nesse sistema.

O suporte para estes requisitos está além do escopo das linguagens de programação que devem ser executadas em vários dispositivos (e.g. Java na sua versão Java 2 Micro edition [J2ME]) 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.

Acredita-se que este suporte seja oferecido pela integração das tecnologias de Grid Computing e Computação Móvel [THAM 2005], o que é proposto através do Grid-M que será apresentado com maiores detalhes no capítulo 3.

1.2 Objetivos Gerais

Este trabalho visa contribuir para encontrar uma solução que possa integrar WSN, sensores, dispositivos móveis e computação em Grid; e, como esta solução possa transformar esse ambiente claramente heterogêneo em um ambiente homogêneo que possibilite facilmente fornecer serviços de qualidade aos usuários explorando todos os recursos providos pelo Grid e pelos seus dispositivos.

1.3 Objetivos Específicos

Como objetivos específicos pretende-se criar um middleware (GridM) que responda as perguntas levantadas pelos Objetivos Gerais. Este middleware deve prover ao usuário facilidade para implementação de aplicações para o Grid e funcione como uma API para auxilio de tarefas intrínsecas ao Grid.

1.4 Organização do Trabalho

Após a introdução o trabalho é organizado da seguinte forma: o capítulo 2 apresenta conceitos sobre os temas discutidos no trabalho, destacando-se as Grids, Redes de Sensores, Computação Móvel, Middleware, que são fundamentais para a compreensão da proposta a ser apresentada no capítulo 3. No capítulo 4, são listados os trabalhos relacionados e a sua comparação com o presente trabalho, mostrando os requerimentos satisfeitos por cada um. No Capítulo 5 são apresentadas algumas soluções utilizando o GridM, para exemplificação de uso e a validação da proposta. No capítulo 6 tem-se a conclusão do trabalho. Em seguida no capítulo 7 têm-se as referências bibliográficas. Finalmente um capítulo de Anexos, onde são inseridos o código fonte do GridM e alguns exemplos do mesmo.

2 Conceitos

2.1 Grids Computacionais

A tecnologia de Grades de computadores ou Grids, segundo [ASSUNÇÃO 2004], começou a ser difundida no final da década de 90 com a descrição de uma arquitetura que possibilitava 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 pervasivo, consistente, acessível e confiável acesso a capacidades computacionais de alto desempenho"*.

Em um trabalho mais recente [FOSTER 2001] define Grid como sendo *"compartilhamento coordenado de recursos e solucionador de problemas em instituições ou organizações virtuais dinâmicas"*.

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

a) 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;

b) 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, e;

c) entrega qualidade de serviço não trivial, 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].

Logo, grades de computadores possuem um conceito intrínseco que permeia este trabalho, pois além de possibilitar troca de informações e compartilhamento de recursos,

possuem 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) ou um sensor. A comunicação entre estes e o compartilhamento de recursos torna o uso da tecnologia de grades de computadores fundamental para solucionar estas questões de forma eficaz.

O termo Grid é o que compreende melhor esse novo paradigma de serviços Tecnologia de Informação (TI), 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 seriam 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 “on-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 “Utility Computing” que se refere a uma separação entre provedores de serviços e consumidores e uma capacidade para negociar níveis de qualidade de serviços. Esta superioridade pode ser visualizada na Figura 2.1.

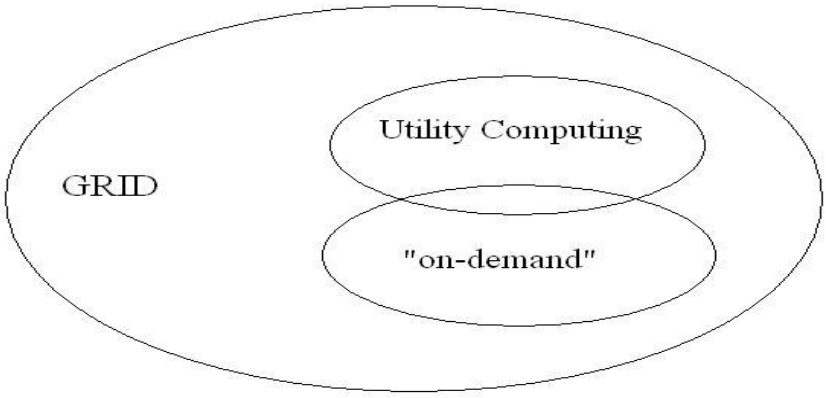

Figura 2.1 Visão da relação entre Grid, Utility e On-demand Computing. Adaptado de [FOSTER 2005]

2.2 Redes de Sensores

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

Redes de sensores, segundo 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 se deve 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, segundo uma determinada medição verificada pelo sensor, que pode transmitir informações continuas deste ambiente e, por meio, destas informações pode-se programar avisos sobre determinado cenário, ou seja, quando for verificada certa medição tomar alguma decisão.

Um exemplo na área da saúde 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. 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.

2.3 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 PDA's.

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.

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:

a) restrições como limite de armazenamento de dados;

b) baixo poder de processamento;

c) tempo de uso limitado pela capacidade das baterias devem ser consideradas ao desenvolver aplicações que tenham a característica de mobilidade.

2.4 Middleware

Segundo OBJWEB (2005), 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 vários outros 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 específicas 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: a) mascarar o ambiente distribuído, parecendo ao usuário ou ao desenvolvedor um sistema único; b) ocultar a heterogeneidade de dispositivos de hardware, de sistemas operacionais e de protocolos de comunicação; e, c) prover ao desenvolvedor interfaces padronizadas para que as aplicações desenvolvidas possam ser portáveis, reusáveis, interoperáveis e sejam facilmente construídas.

No entanto, o conceito de middleware neste trabalho e na comunidade atual utiliza o termo middleware como um software que faz a interconectividade entre dois sistemas. Particularmente nesta 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.

3 Grid-M

Apresentada a motivação e análise dos requisitos necessários para implementação de aplicações que suportem serviços móveis surgiu a questão: *Como integrar um grid computacional com rede de sensores e dispositivos móveis?*

Em [THAM 2005] são descrito duas formas de implementação de uma grid de sensores baseadas em dois focos. O primeiro chamado de (*centralized sensor-grid computing approach)* aponta para uma simples conexão e interface de todos os sensores e sub-redes de sensores na grid, deixando para o middleware a gerência de todos os dispositivos. Este tipo de implementação tem algumas desvantagens, pois necessita de comunicação excessiva com o sensor, o que acarreta um desgaste maior de sua bateria, além de possuir comunicação direta com o sensor e em caso de pane este ficaria inoperante.

Uma alternativa mais eficiente seria o (*distributed sensor-grid computing approach*), o qual possui uma arquitetura distribuída, poder de processamento e capacidade para tomar decisões por parte dos sensores, eliminando a comunicação excessiva presente no primeiro approach.

3.1 Arquitetura do Grid-M

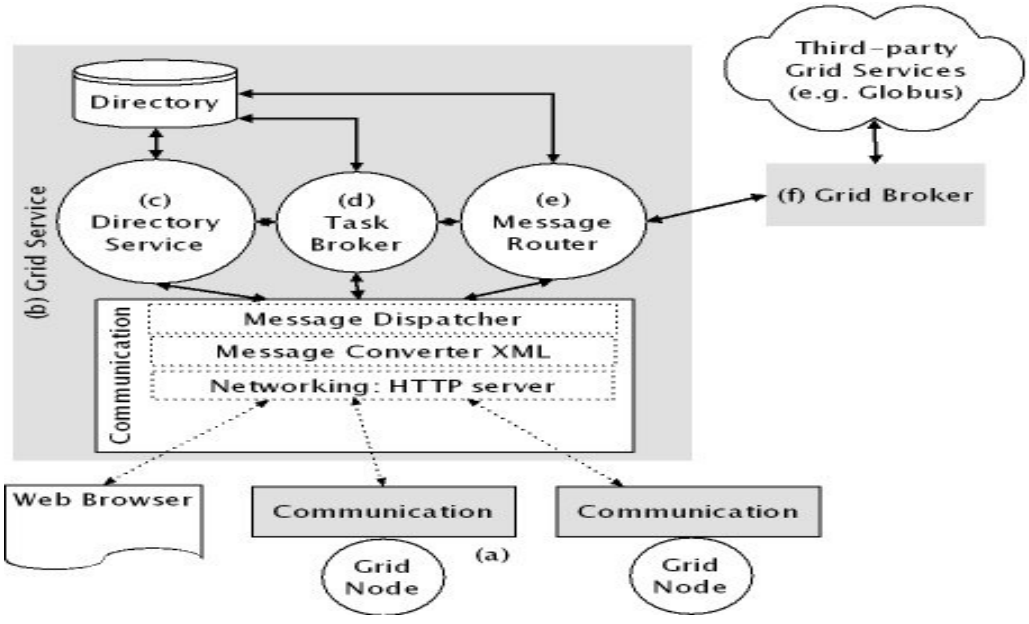

Figura 3.1 Arquitetura do Grid-M**,** GRIDM[2007]

A Figura 3.1 apresenta os componentes da arquitura proposta. Na Figura 3.2 é mostrado a arquitura de um nodo, que seria a unidade funcional do middleware, pois é o responsavel por mandar e receber tarefas, e disponibilizar os serviços através do provedor de serviços. Os modulos dessa arquitetura são descritos abaixo.

O sistema é composto por:

a) GRID NODES, como será descrito abaixo, ver Figura 3.2;

b) GRID SERVICES são as representações abstratas de provedor de serviços do Grid. Não é necessariamente um host (ou nodo) ou um agrupamento de nodos. Pode ser implementado como um serviço distribuído ao longo do sistema. È importante dizer que o Grid Service também é um provedor de serviços que deve responder as requisições de serviços;

c) DIRECTORY SERVICE é a estrutura onde serviços são registrados e correspondem a nodos capazes de implementá-los. Toda vez que um novo nodo registra-se no Grid Service ele deve se autenticar e publicar a lista de serviços que é capaz de executar;

d) TASK BROKER é a estrutura que controla pedidos de serviço, coordena distribuição de pedidos de tarefa (p.e. no caso de separação de tarefa) e no caso específico de grades de dispositivos móveis e embutidos onde o dispositivo não tem conectividade permanente, isto pode armazenar temporariamente o resultado da execução de tarefa para recuperação tardia;

e) módulo de MESSAGE ROUTER implementa o roteamento de mensagens de comunicação, por resolução de nome e acessando informação sobre endereço de rede físico do banco de dados de diretório, esta informação foi inserida durante processo de inscrição de nodo. Além disso, no caso de Grades de dispositivos Móveis e Embutidos, alguns destes dispositivos poderiam não estar transmitindo em rede diretamente (p.e. TCP/IP) ou disponível como ouvinte de porta (p.e. dispositivos de WAP não têm um real endereço de TCP/IP e não podem ser um ouvinte de porta de TCP/IP). Este módulo implementa a roteamento de mensagem a estes dispositivos, o que em alguns casos poderia requerer proxy, store-and-forward facility ou outras técnicas.

f) GRID BROKER é a estrutura para interconectar Grid Services externos. Promove tradução de mensagem e codifica em diferentes grid services (i.e. *gateway*).

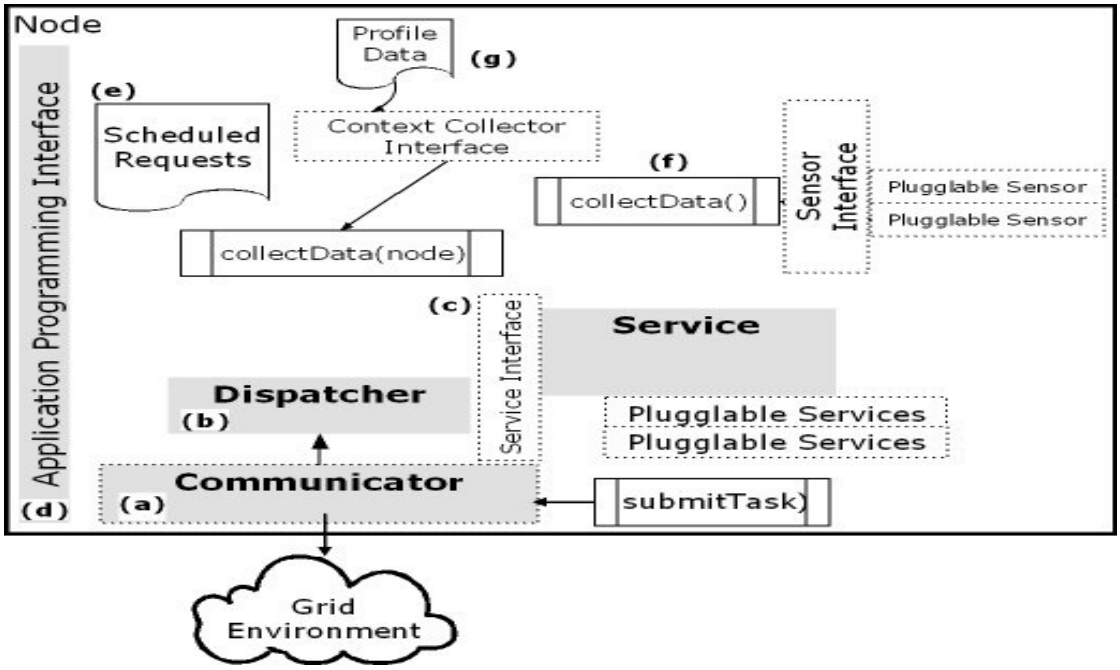

Figura 3.2 Arquitetura de um Nodo no GridM.GRID-M[2007]

Um nodo possui os seguintes módulos:

a) COMMUNICATION module: contém as sub-rotinas no escopo da comunicação inter-nodo. O protocolo escolhido foi HTTP sobre TCP/IP. É esperado que nodos sejam capazes de HTTP-cliente (ou por uma proxy para esses nodos não equipados com TCP/IP). Nodos podem receber pacotes tanto por HTTP-server port (ouvindo a porta padrão do Grid Middleware) ou os pacotes são recebidos via http-server que funciona como um *storage-relay*. O Communication module guarda as sub-rotinas para converter em representação XML para estruturas de dados internas e tem interface para (b) o message dispatcher module;

b) DISPATCHER module: contém as sub-rotinas para despachar mensagens recebidas (aqui já carregadas dentro das estruturas de dados internas) para os serviços plugados. Serviços de perfil em obrigatório para todo nodo. Serviços novos podem ser plugados pela interface de programação. O Dispatcher é baseado no PERFORMATIVE contido na mensagem recebida. São associados PERFORMATIVES a serviços novos durante o processo de plug-in;

c) SERVICE INTERFACE permite plugar serviços para serem implementados; ele seta *call-backs* para pontos de execução de serviços;

d) APPLICATION PROGRAMMING INTERFACE, permite a conexão de aplicações externas em cima do node middleware. São implementadas aplicações

de Java neste módulo. Eles podem usar os *Service Executors* para receber chamadas do nodo e responder a chamadas de comunicação existentes;

e) SCHEDULED REQUESTS e SCHEDULER MODULE fazem parte da implementação do middleware e permitem a programação de pedidos de comunicação internamente gerados em períodos estabelecidos de tempo. A programação de pedidos terminados está no item APPLICATION PROGRAMMING INTERFACE;

f) SENSOR INTERFACE permite plug-in de sensores que coletam dados de módulos externos e que eventualmente podem conectar em hardware externo;

g) CONTEXT COLLECTOR INTERFACE permite plug-in de novos coletores de dados de contexto através de Node.addProfileContextCollector(ProfileDataCollectorInterface collector) e addProfileStaticCollector(ProfileDataCollectorInterface collector) . Nodo implementa SensorInterface que de fato coleta seus próprios dados (i.e. Context). Auto coleta de dados e transmissão para o Service Provider podem ser agendadas através do Node.scheduleContextCollect(int intervalSec) .

3.2 Modelos de Comunicação do Grid-M

3.2.1 Modelo de Comunicação J2SE

A figura 3.3 apresenta o modelo de comunicação do Grid-M. Existem duas classes de comunicação no Grid-M: a) **IMMEDIATE**, quando o Service processa a tarefa imediatamente e retorna um *TaskResult* através da mesma conexão de comunicação (e.g. HTTP -- e.g. a *TaskResult* é retornado pelo *HTTP-response* pela *HTTP-request* para *Task processing*).

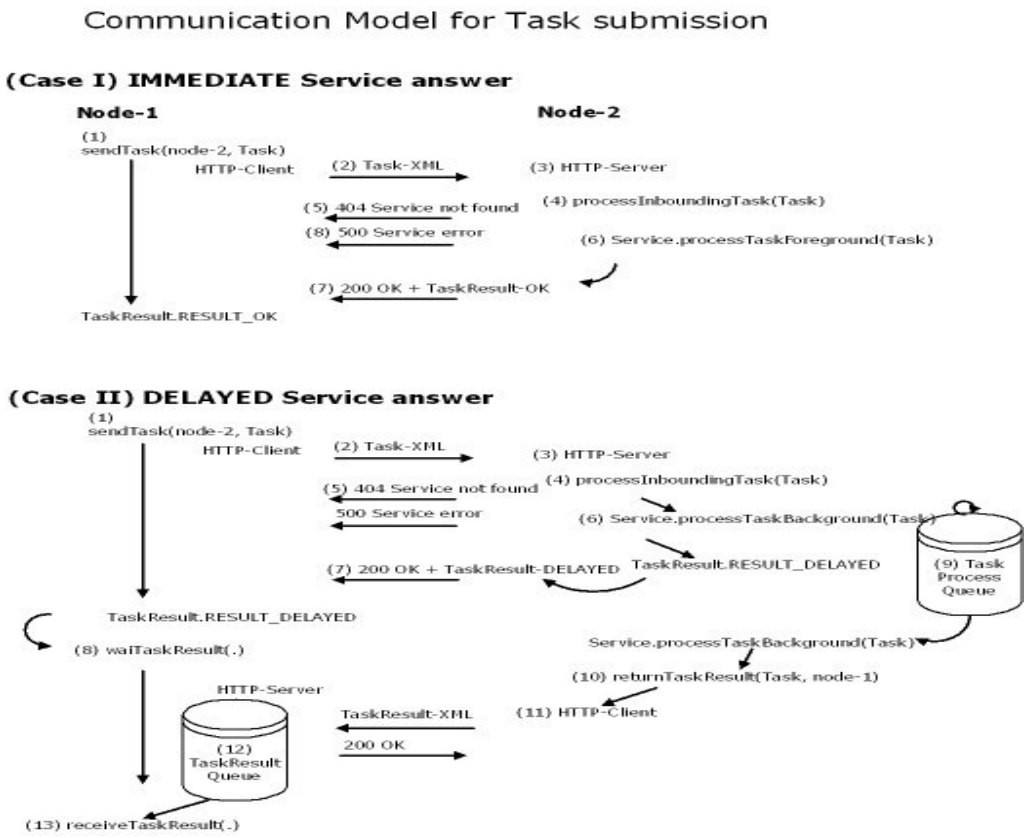

Figure 3.3 Modelo de Comunicação Grid-M. GRID-M[2007]

Os serviços **IMMEDIATE PROCESSING** funcionam assim:

1. node-1 envia a tarefa para ser processada no node-2: *sendTask(node-2, Task)*;
2. a tarefa é representada no formato XML e a *Task-XML* é transmitida para o node-2 por um *HTTP-request* através de um *httpClient* criando assim a conexão-1;
3. O node-2 recebe a *Task-XML* através de um *httpServer* decodifica a XML novamente para representação *Task* e envia o processo através *processInboundingTask(Task);*
4. *processInboundingTask(Task)* encontra o *ServiceInterface* para processar a *Task.*
5. Se o serviço não está registrado no node-2 é retornado *HTTP-response (404 NOT FOUND*) para a conexão-1;
6. *processInboundingTask(Task)* tenta enviar com "foreground process" utilizando *Service.processTaskForeground(Task);*
7. Se *Service.processTaskForeground(Task)* retorna *TaskResult.OK* então este resultado é codificado em XML e retornado através de *HTTP-response (200 OK / TaskResult-XML)* para a conexão-1;
8. Em caso de uma exceção durante o processamento é retornado *HTTP-response (500 SERVICE ERROR)* para a conexão-1.

Os serviços b) **DELAYED** trabalham da seguinte forma (Passos (1) ao (6) são os mesmos do IMMEDIATE PROCESSING):

7) Se *Service.processTaskForeground(Task)* retornar *DELAYED* (significando que o foreground process não pode ser realizado agora), então a *Task* é colocada para o *TASK PROCESS QUEUE* e *HTTP-response (200 OK, TaskResult.DELAYED)* é retornado através de *connection-1*. *connection-1*é fechada;
8) No node-1, *TaskResult.DELAYED* é recebido, o código deve esperar um *waitTaskResult(Task)* – mas alguns processos podem ser executados enquanto espera-se a resposta de *TaskResult*;
9) No node-2, *processInboundingTask(.)* inicia um background process thread (se ainda não estiver executando) que processa a *Task* que estava na fila do *TASK PROCESS QUEUE* utilizando *Service.processTaskBackground(Task)*;
10) Quando a *Task* é processada pelo *Service.processTaskBackground(Task)*, ela retorna *returnTaskResult(node-1, TaskResult)*;
11) O node-2 abre um *httpClient connection-2* para o node-1 e envia a *TaskResult* codificada em XML;
12) No node-1, *TaskResult* é enfileirado para *TASKRESULT QUEUE* e espera *waitTaskResult(Task)* sinalizando o recebimento, e;
13) Node-1 recebe *TaskResult* invocando *receiveTaskResult(Task.id)*.

3.2.2 Modelo de Comunicação para Dispositivos Móveis

A Figura 3.4 mostra as interações entre um Mobile Node (PDA, Celular, etc..) com o comunicator do Grid-M. Esta comunicaçao é diferente de uma comunicação j2se (figura 3.3) porque o dispositivo móvel não é socket-server e por isso não tem suporte ao TCP/IP, mas o WAP faz a tradução MSN<->TCP/IP. Ele apenas é um cliente http e por isso deve se comunicar com o grid, para checar através de polling se tem mensagem.

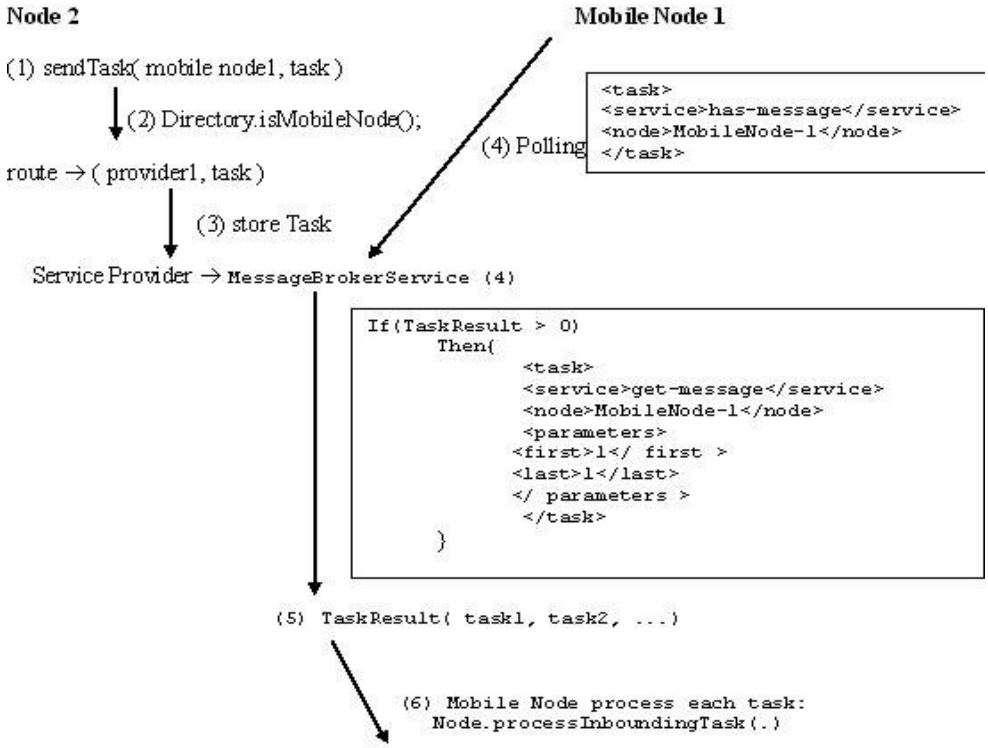

Figura 3.4 Modelo de Comunicação para Dispositivos Móveis.

As interaçoes são as seguintes:

a) Um nodo quer enviar uma tarefa para outro nodo;
b) O Diretorio verifica se o nodo é um mobile node, caso isto aconteça a rota entre eles passa a ser o Service Provider, no caso provider1;
c) A tarefa é armazenada dentro do Storage;
d) O mobile node continuamente realiza um polling, o http-client abre uma conexão para o service provider, e pergunta se tem algum pacote para ele;
e) Caso tenha mensagens elas são armazenadas num TaskResult e são enviadas para o mobile node, e;
f) Então o mobile node processa cada tarefa do TaskResult.

3.3 Como Programar com o Grid-M

A Grid-M API é composta de grupos de funções relativas à arquitetura do Nodo. A API completa pode ser vista em http://agentgrid.lrg.ufsc.br/javadoc/common/index.html .

Tabela 3.1 Alguns métodos do Grid-M.

Type	Method	Description
Node	**createTask**(java.lang.String destination, java.lang.String service, XMLTree parameters)	Create new Task for submission.
Node	**createTaskResult**(Task task, java.lang.String resultCode, XMLTree results)	Create new TaskResult.
Node	**sendTask**(Task task) Send Task to execute in other Grid Node (direct submission).	Send Task to execute in other Grid Node (direct submission).
Node	**fetchTaskResult**(java.lang.String taskId)	Fetch the result for a Task submission with DELAYED response.
Node	**addSensor**(java.lang.String sensorID, SensorInterface sensor)	Add Sensor to this Node.
Node	**addService**(ServiceInterface service)	Add new service to this Grid Node.
Node	**register**()	Register this Node to Service provider.
Node	**setProfiler**(Profiler profiler)	Set the PROFILER (i.e. profile collector object) to be used by this Node.
Node	**setServiceProvider**(java.lang.String url)	Set ServiceProvider to where this Node is connected.
Task	**createTaskID**()	Create new unique Task identificator

Fonte: GRID-M[2007]

3.4 Exemplos

3.4.1 Criando Sensor Nodes

::: Node.java :::

```
/**
 * Add Sensor to this Node.
 *
 * @param sensorID is the identification for this sensor
 * @param sensor is the SensorInterface implementation
 * @see SensorInterface
 * @see http://grid.lrg.ufsc.br/docs/GridMProgramming.pdf
 */
```

```
public final void addSensor(String sensorID, SensorInterface sensor)
```

Neste exemplo, a SENSOR NODE é configurado:

```
// initialize a Sensor Node
HTTPServerNode node1 = new HTTPServerNode("node-1", 8001);
node1.setServiceProvider("http://localhost:8000/");
node1.setLogOutput(System.err);
node1.setLogLevel("node", Logger.DETAIL);
node1.setLogLevel("net", Logger.DETAIL);

// initialize sensor
SensorInterface sensor = new ExampleSensor();
node1.addSensor("collector", sensor);
node1.scheduleSensorCollect("collector" , 5, "*", "store", Task.PRIORITY_NORMAL, null);
node1.scheduleContextCollect(5);

// start and register
node1.start();
node1.register();
```

Example Sensor.java pode ser criado assim:

```
public class ExampleSensor implements SensorInterface {

   public XMLTree collectData(){
      XMLTree xml = new XMLTree("vital-data");
      xml.put("heartbeat", Long.toString((System.currentTimeMillis()%30)+30));
      xml.put("breathrate", Long.toString((System.currentTimeMillis()%10)+10));
      return xml;
   }
}
```

3.4.2 Criando Service Provider Nodes

Code example:

```
// initialize and start Service Provider
```

```
ServiceProviderNode providerNode = new ServiceProviderNode("provider", "myGrid", 8000);

// set logger
providerNode.setLogOutput(System.err);
providerNode.setLogLevel("node", Logger.DETAIL);
providerNode.setLogLevel("provider", Logger.DETAIL);
providerNode.setHTTPSnoopMode(true);

// add STORE service
ExampleStoreServices storeService = new ExampleStoreServices();
providerNode.setHTTPServerLog(10, System.err);
providerNode.addService(storeService);
providerNode.addXSLTInterface("/store", null, "store.xsl", new XMLInterface[] {storeService});
// start provider
providerNode.start();
```

3.5 Cumprimento de Requisitos

Baseado na arquitetura acima apresentada e nas interações entre módulos, verifica-se então o cumprimento dos requisitos levantados no Capitulo 1, item 1.1.2:

i) Para satisfazer (a) métodos de comunicação de dados, é utilizado o protocolo HTTP, garantindo compatibilidade entre os dispositivos;

ii) Para satisfazer (b) segurança na transmissão de dados e controle de acesso aos recursos, utilizamos métodos de criptografia e autenticação no registro de novos nodos;

iii) Para satisfazer (c) registro de recursos, o componente DM (Directory Module) utiliza o protocolo LDAP [LDAP 2006] e mantém o registro dos recursos;

iv) Para satisfazer (d, f) gerenciamento e coordenação dos dispositivos, é utilizado o componente BM (Broker Module) o qual faz o gerenciamento dos dispositivos;

v) Para satisfazer (e) interface com os dispositivos móveis e sensores, é utilizada a tecnologia XML [XML 2006] para representação dos dados (protocolos) e visualização nos possíveis dispositivos do sistema.

O Grid computacional proporcionou a homogeneidade e virtualização de recursos e serviços no gerenciamento dos equipamentos conforme foi proposto inicialmente. Pelo fato de

estar usando apenas um pequeno conjunto de funcionalidades proporcionadas pelo Grid computacional, o middleware abstraiu a complexidade de implementação das funções que fazem uso da grade. Dessa forma conseguimos desenvolver o protótipo do ambiente para rodar em diferentes tipos de equipamentos sem deixar o código atrelado a um tipo de equipamento.

Também foi criado uma API para programação Java em Ambientes de Grids. Isto facilita o desenvolvimento de aplicações que precisem utilizar o grid. Pois basta a aplicação utilizar da arquitetura do Grid-M para utilizar os seus métodos. A aplicação precisa ser baseada em *Send and Receive Tasks*, padronizadas no formato XML, utilizando o protocolo TCP/IP.

4 Trabalhos Relacionados

BHATTI (2005) mostra uma aplicação no campo da saúde, descrevendo uma arquitetura na qual, por meio de Internet Data Center (IDC) 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 encontram-se dois aspectos afins ao Grid-M, primeiro, o uso de grids como tecnologia que homogeneíza e integra os dispositivos e, segundo, no aspecto de que esta arquitetura disponibiliza aos médicos acesso a informações, através de PDA, inclusive fora do ambiente do hospital (evidenciando a mobilidade).

Este trabalho complementa o Grid-M 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 prove colaboração (colaboration) outra propriedade da computação móvel e distribuída.

Em [GAINOR 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.

Neste artigo encontramos citação ao padrão [FOSTER 2002] Open Grid Services Architecture (OGSA) o qual define um conjunto de interfaces para desenvolvimento de sistemas grid, padronizando e viabilizando compatibilidade, isto se assemelha ao Grid-M, pois também propõe uma API para desenvolvimento em cima do Middleware. Em comum também há 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 o nosso middleware proposto.

4.1 Principais Middlewares

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 reduzidos. [USKELA 2003] propõe que essas limitações são inerentes ao ambiente e devem ser amenizadas, mas não suplantadas, por desenvolvimentos tecnológicos futuros. Baseado nos requisitos descritos anteriormente, avaliamos os middlewares existentes 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 [GLOBUS 2006] é um software de código aberto, desenvolvido pela Globus Alliance, que oferece um kit de ferramentas para desenvolver aplicações e sistemas de computação em grade.

Seu suporte para as características necessárias em aplicações para serviços móveis acima descritas é encontrado em (a) 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 (c) 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.

O GRIDBUS [GRIDBUS 2006] 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.

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

O Legion [LEGION 2006] 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, 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 (a) suporte a colaboração, onde por meio do sistema é possível colaboração através de tuplas como <LOID, LOA, timeout> que fornecem, endereçamento aos objetos (LOID) e gerenciamento provido por (LOA). Existe também (c) 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.

O UNIform Interface to COmputer REsources (UNICORE) [UNICORE 2006] 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 (a) 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 (c) distribuição de recursos, é feito pelo Abstract Job Object (AJO) que é uma classe/biblioteca que controla a comunicação, envio e recebimento dos jobs e faz a distribuição dos recursos.

Tabela 4.1 Comparação dos principais middlewares em relação às características para aplicações em serviços móveis.

Características dos middleware	Globus	GridBus	Legion	Unicore	Grid-M
Suporte a Colaboração	Sim	Sim	Sim	Sim	Sim
Suporte a Sensibilidade ao Contexto	Não	Não	Não	Não	Sim
Suporte a Alocação de Recursos	Sim (GRAM)	Sim	Sim (LOA	Sim (AJO)	SIM
Suporte a Ambientes Dinâmicos	Não	Não	Não	Não	Sim
Suporte a Execução em Dispositivos Móveis	Não	Sim	Não	Não	Sim

A Tabela 4.1 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 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 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 $S=\{\alpha,\beta,\gamma,\delta,\varepsilon\}$.

5 Ambiente Experimental

Neste capítulo serão apresentadas algumas simulações em diferentes áreas para validar a proposta do middleware e concluir se ele responde a questões fundamentais levantadas no decorrer do trabalho e comparadas com outros middlewares, ver 4.1, (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.

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.

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) tanto como facilidade para criação e utilização de serviços por dispositivos móveis (e.g., apresentação da informação na interface dos dispositivos e sensibilidade ao contexto, tal como localização do dispositivo), sensores (e.g, homogeneização de diversos sensores sem a necessidade de uma interface diferente para cada dispositivo).

5.1 Estudo de Caso 1: Telemedicina

Com a criação de novas aplicações médicas devido ao resultado do continuo avanço tecnológico a telemedicina hoje está sendo usada amplamente em diversos paises. Os principais objetivos da telemedicina são de garantir rapidez, segurança e confiabilidade em diagnósticos e monitoramentos do paciente à distância.

A aplicação da telemedicina que se utiliza nesse trabalho é o monitoramento remoto de pacientes em um leito hospitalar, o qual é feito utilizando sensores. Pelo fato de existirem diversos equipamentos espalhados pelo hospital, a homogeneização desses equipamentos se

torna uma questão importante. Ou seja, os dados dos diversos equipamentos têm que ser fornecidos ao médico de forma rápida, segura e proveniente de todos os equipamentos.

[FRANKE 2006] e [ROLIM 2006] propõem a utilização de grades computacionais para a solução deste problema, pois permite a visão de cada equipamento distinto como sendo um nodo pertencente à grade. Isso facilita muito as funções de gerenciamento, comunicação e integridade dos dados.

Procura-se a criação de uma arquitetura que pudesse garantir duas questões fundamentais:

a) garantir que os diversos coletores, atuadores e dispositivos móveis possam ser gerenciados de maneira homogênea, sem a necessidade da criação de uma interface para cada dispositivo, e;

b) como divulgar esses dados em tempo real.

5.1.1 Motivação e Análise

A telemedicina é a aplicação do trabalho e recursos médicos através de meios computacionais e de telecomunicação para tratamento e diagnostico médico, ou seja, é o uso das tecnologias de telecomunicação para proporcionar informações e serviços médicos não presenciais.

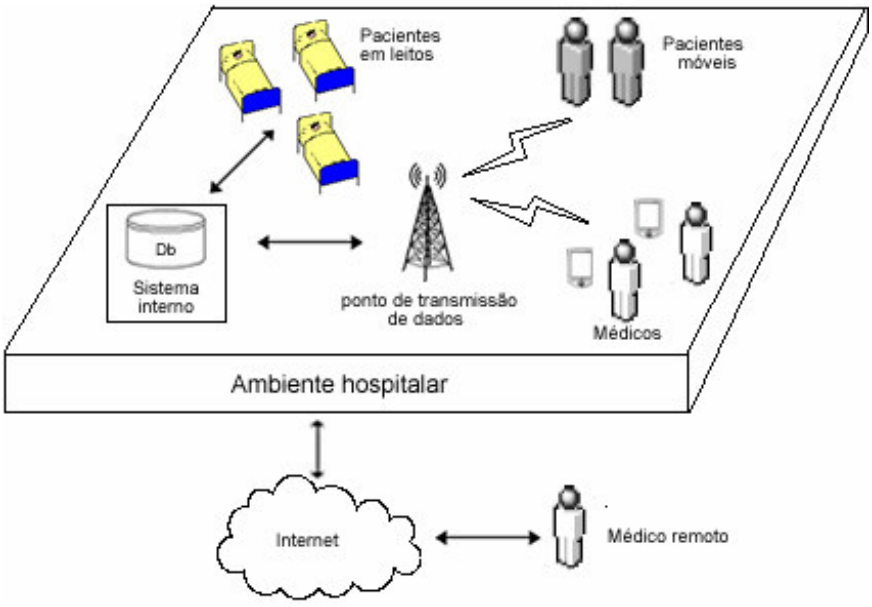

Figura 5.1 Coleta de dados vitais e disponibilização ao médico.

O cenário problema é um hospital onde o corpo médico necessita de apoio no monitoramento dos pacientes. Existem sensores nos quartos que coletam os dados vitais dos pacientes, como: freqüência cardíaca, temperatura, etc.. Estes dados são coletados através de sensores que são colocados no leito. Estes dados são enviados através de ondas de rádios para pontos de recepção de dados no interior do hospital.

Os dados podem ser acessados por qualquer pessoa que se autentique na grade. Esse acesso se da por PDA's carregado por cada membro do corpo médico. Com isso os médicos podem monitorar os pacientes em tempo real, contribuindo significativamente para o tratamento destes de pacientes. Além do monitoramento eles podem configurar o seu dispositivo de forma que receba alerta quando certas condições pré-estabelecidas forem atingidas.

Além do acesso interno, o médico pode querer acesso externo ao ambiente. Isso pode acontecer utilizando qualquer aparelho conectado à internet, por exemplo, um celular ou PC. Para isso foi desenvolvido um httpServer, que após a coleta dos dados da grade, fornece-os aos médicos a partir da internet.

Ao analisar o cenário problema acima, percebe-se que a utilização de grade é uma solução muito interessante, pois permite que os diversos equipamentos sejam vistos de maneira homogênea, ou seja, um nodo da rede. Em virtude disso não é necessário à criação de uma interface para cada dispositivo se comunicar com a grade.

5.1.2 Implementação

Para que este cenário seja possível alguns passos ou interações devem ser seguidos:

a) Registro de um PDA na grid;

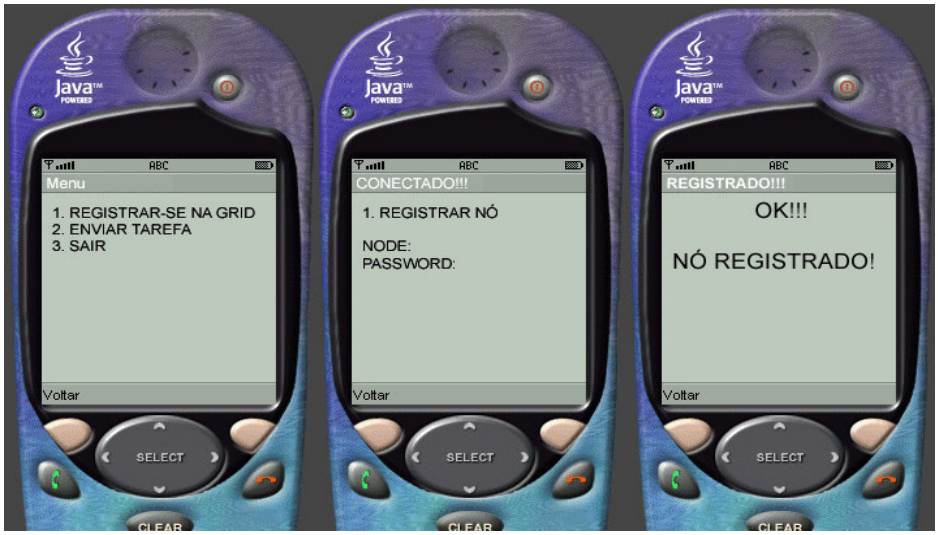

Figura 5.2. Interface de cadastro de um PDA na grid.

Neste passo devemos tratar o PDA como um nodo pertencente à grade e cadastrar o mesmo, e também se deve ter a possibilidade do médico enviar tarefas para algum nodo da grade. Para isto deve-se acessar o servidor através de um proxy (httpServer) e efetuar a execução do seguinte método:

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

b) Prévio registro do Sensor na grid;

Este passo é necessário, pois devemos cadastrar um sensor específico para realizar a coleta de dados e futuramente o médico poder requisitar os dados de um monitoramento que lhe interessa sem a necessidade de saber todos os sensores da grade.

```
public class RandomSensor extends Sensor {
  Random r = new Random(323141346);
  protected XMLTree _collect(Node node) throws Exception {
    int chance = Math.abs(r.nextInt()) % 10;
    XMLTree result = null;
    // this sensor fails 30% of time
    if (chance > 3) {
      result = new XMLTree("data", "" + (Math.abs(r.nextInt()) % 1000));
```

```
        // simulate long data collect
        try {
          int delay = Math.abs(r.nextInt()) % 1500;
          Thread.sleep(delay);
          Thread.yield();       }
        catch (InterruptedException ex) {       }
      }

      else {
        throw new Exception("chance=" + chance);     }
      return result;   }
    }
```

c) Criação de uma tarefa dentro do sensor

Este nodo sensor é criado e dentro dele cadastramos os serviços que ele é capaz de atender, neste caso o de monitorar os batimentos cardíacos.

```
    // Node2 é um sensor que marca os batimentos cardiacos
      Node node2 = new Node("node-2", "http://localhost:8002");
      node2.setLogLevel(3);
      node2.addService("HeartBeat", new HealthCareService());
      httpNetInterface = new HTTPServerInterface(node2, 8002);
      node2.setNetInterface(httpNetInterface);
      node2.scheduleLogHeartBeat(3);
      node2.start();
```

d) Requisição dos dados pelo médico;

Neste passo o médico deve utilizar o seu PDA e se logar na Grid, utilizando um proxy (httpServer). Uma vez logado ele tem acesso às estatísticas enviadas pelos nodos para o Servidor.

O código completo desse passo pode ser visto no capítulo Anexos, item 1.

5.1.3 Conclusões

Procurou-se resolver o problema de gerenciamento de diferentes tipos de equipamentos e a sua disponibilização em tempo real. Para esse propósito adotou-se uma

arquitetura em grades computacionais, permitindo assim uma homogeneização dos mesmos, garantindo uma facilidade na implementação e gerenciamento dos recursos provenientes de diferentes equipamentos, pois cada equipamento passa a ser visto como um nodo na rede igual aos demais.

A utilização da grade permitiu também critérios como: segurança, comunicação e compartilhamento de recursos, características naturais de uma grade, garantindo assim um reaproveitamento de tecnologia, pois não foi necessária a criação de uma interface para cada equipamento especifico.

A principal contribuição deste caso de estudo, foi a validação da idéia de homogeneização de recursos dentro da arquitetura proposta. O que prova que o uso de uma grade computacional para o problema de coleta de recursos e disponibilização em tempo real é uma ótima solução pela facilidade de implementação, uma vez que a grade fornece reaproveitamento de tecnologia e o grande poder computacional.

5.2 Estudo de Caso 2: Gerenciamento – NetManager

O Netmanager é um projeto do Laboratório de redes de Gerencia – LRG, da Universidade Federal de Santa Catarina, desenvolvido por Doublas Balen [FRANKE 2007] que tem como objetivo construir um gerenciador de redes em um ambiente de grades de computadores. Toda a parte de comunicação HTTP, transmissão de dados, representação, codificação e decodificação do XML, entre outras, é feito pelo Grid-M, assim tem-se uma grande reusabilidade de código e facilidade para implementação.

5.2.1 Arquitetura

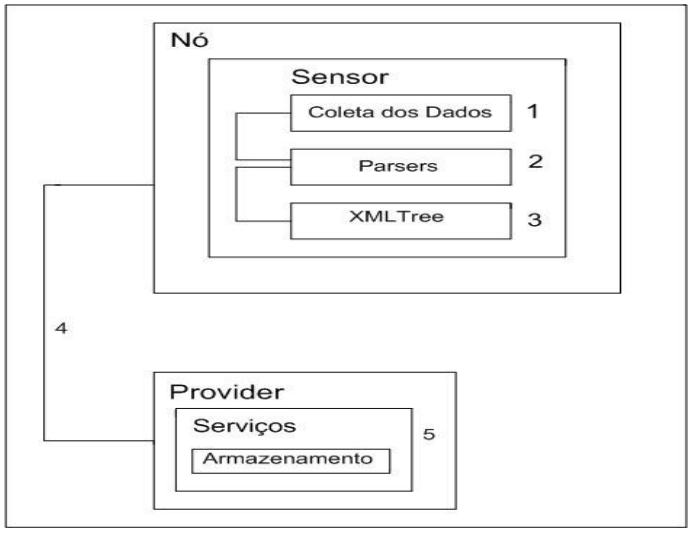

Figura 5.3 Arquitetura do Gerenciador

Para realizar o gerenciamento, cada nodo da grade deve possuir um sensor que será o responsável por realizar uma seqüência de passos. Estes passos são ilustrados pela Figura 5.3 que apresenta a arquitetura do gerenciador.

a) Coleta dos Dados: Neste passo os nós da grade irão executar funções internas, que revelarão o estado do seu contexto de execução. Após cada execução de método, os dados retornados serão repassados ao seu respectivo parser (cada comando tem o seu parser específico).

b) Parsers: Este passo tem a função de filtrar os dados coletados e utilizar só o que é realmente importante. Ao executar um comando, por exemplo, são retornados dados relevantes e irrelevantes. Cabe ao parser analisar e filtrar apenas os relevantes. Depois de realizada a filtragem os dados são repassados ao módulo responsável por criar as XMLTree.

c) XMLTree: Para que os dados tenham um formato padrão, os dados relevantes oriundos de todos os parsers de cada comando executado serão formatados em forma de uma árvore XML. Isso facilita tanto a compreensão dos dados como também a fácil manipulação no momento do seu armazenamento no Provider.

d) Envio ao Provider: Após a montagem da árvore XML, ocorre o envio dos dados do contexto de execução do nodo ao Provider. Isso é feito utilizando a parte de comunicação do Grid-M. Esta comunicação é feita pelo protocolo HTTP.

e) Armazenamento: As XMLTree de todos os nós da organização virtual serão armazenadas no serviço de armazenamento do Provider. Estes dados que chegarão

em intervalos periódicos serão a base para a construção do gerenciador, já que poderemos ter os dados de cada nó em vários períodos de tempo.

Dentro do coletor de Contexto de um nodo, são feitos coletores podendo ser implementações de uma interface Java que coletasse dados do OS, memória, processador, network e outros dados relevantes.

5.2.2 Métodos de Coleta dos Dados

Os sensores irão utilizar basicamente dois métodos de coleta, os scripts escritos em Virtual Basic Script para ambiente Windows, e a execução de linhas de comandos para todos os ambientes (Linux, Windows, MacOS).

5.2.3 Interface

Figura 5.4. Tela do Gerenciador

A Figura 5.4, apresenta a tela principal do gerenciador. São mostrados todos os nós da grade, dando a possibilidade ao gerenciador da rede obter mais dados sobre algum nó, apenas clicando sobre o a seta são apresentados os dados na forma de gráficos e relatórios, ver Figura 5.5, que podem ser exportados no formato pdf e csv.

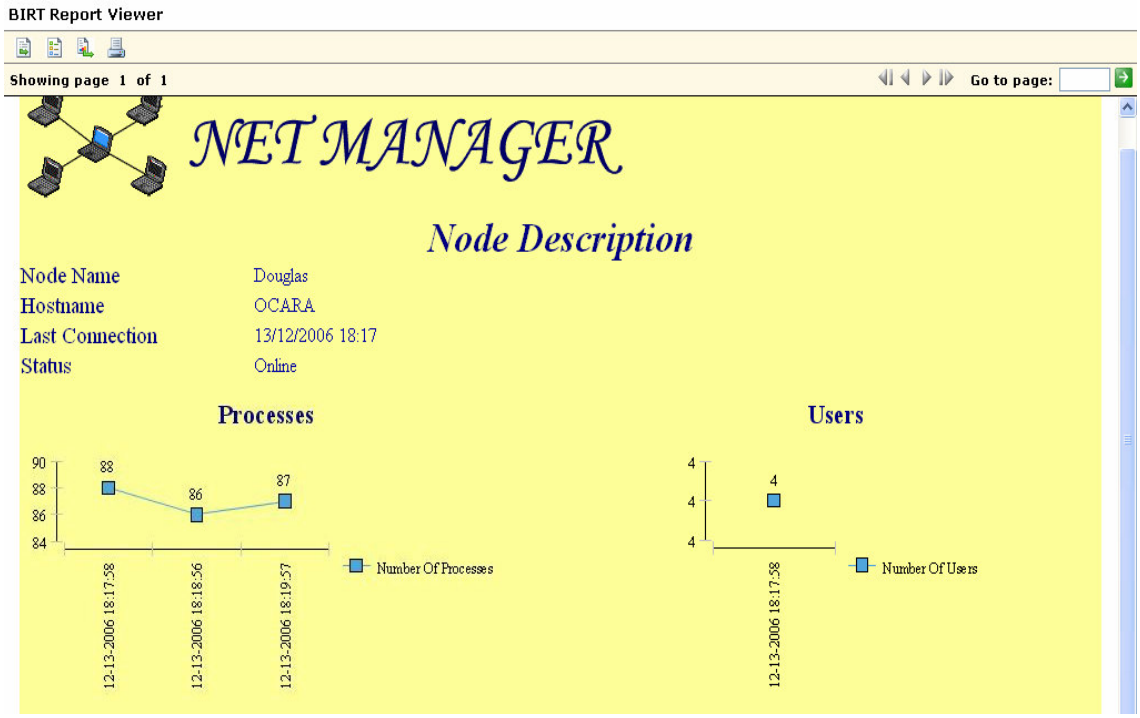

Figura 5.5. Gráfico de informações detalhadas de um nodo especifico.

5.2.4 Justificativa

A estrutura de Grid proporciona que cada nodo seja um provedor de serviços, portanto o NetManager passa a ser um serviço executado em um nodo. Esta estrutura de mapeamento onde todos os dispositivos da grade são tratados como nodos, provido pela virtualização, facilita o roteamento, comunicação e o armazenamento, pois estes são feitos facilmente com o uso da API do GridM.

O NetManager pode ser usado para suportar algumas aplicações:

a) Pode ser usado pelo gerenciador da rede para monitorar a performance de um nodo do Grid, podendo visualizar os dados coletados através de um Browser;

b) Os dados coletados podem ser usados por algum serviço que precise garantir Qualidade de Serviço (QoS);

c) Os dados de Contexto podem ser usados para detector intrusão em um nodo do Grid. Ele detecta intrusão por um uso anômalo dos recursos de um dispositivo. Assumindo que o uso padrão de CPU é 5%, se em algum momento o gerenciador coletar 80% de uso de CPU por um longo período, isto pode ser visto como uma intrusão.

6 Conclusão e Trabalhos Futuros

Neste trabalho foi proposto um middleware para integrar os dispositivos móveis, sensores e o Grid computacional como uma tentativa para responder à pergunta: *Qual a plataforma ideal para promover a homogeneização em um Grid de Wireless Sensor Networks?*

Apresentou-se o middleware Grid-M. Ele é uma plataforma para construir aplicações de computação em Grid e aplicações em dispositivos embargados e móveis. Fornece uma Application Programming Interface (API) para conectar aplicações desenvolvidas em Java em um ambiente de computação em Grid. Seu perfil runtime é pequeno bastante para ser usado em aplicações de computação móvel.

Comparado aos trabalhos relacionados, destaca-se que o Grid-M cobre uma necessidade de um middleware que forneça um meio de desenvolver serviços para aplicações baseadas em Grid que integrem computação móvel. Conclui-se que a Grid-M responde à pergunta sobre uma *plataforma que promova a homogeneização para o desenvolvimento de aplicações em Wireless Sensor Networks.*

Entretanto, o Grid-M é ainda um trabalho em andamento. Como trabalho futuro será adicionado melhorias na plataforma, por exemplo, implementar diversas novas funções, como as listadas abaixo:

a) segurança, melhorando mecanismos de autenticação, criptografia e introduzir mecanismos de detecção de intrusão;

b) gerência e QoS introduzindo ferramentas para monitorar recursos e serviços, em um nível do módulo e em mecanismos para garantir a qualidade do serviço;

c) melhoramento do código em diversos aspectos, tais como o sistema de plug-in onde o novo código pode ser introduzido para controlar o ciclo de deliberação, novo sistema de comunicação com introdução de interfaces de rede configuráveis, e assim por diante.

No futuro pretende-se desenvolver cenários maiores de teste para validar a sustentação da plataforma em diferentes ambientes.

Este trabalho procurou fornecer uma visão geral do middleware Grid-M. O material adicional, tal como a documentação, os exemplos e o código fonte podem ser alcançados com o Web-Site: http://grid.lrg.ufsc.br.

7 Referências bibliográficas

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, 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.

CAMPBELL, Andrew T; COULSON, Geof; KOUNAVIS, Michael. "Managing Complexity: Middleware Explained," *IT Professional*, vol. 01, no. 5, pp. 22-28, September/October, 1999.

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

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

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, I. “What is the Grid? A Three Point Checklist”. GRIDToday, July 20, 2002.

FOSTER, C. KESSELMAN, J. NICK, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002

FOSTER, I. & KESSELMAN, C. Foster, I. & Kesselman, C. (ed.) The Grid 2: Blueprint for a New Computing Infrastructure Elsevier, 2004.

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)

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

FRANKE, Hans Alberto; Westphall, Carlos Becker; Rolim, Carlos Oberdan. MOBILIDADE EM AMBIENTES DE GRADES COMPUTACIONAIS. 58ª Reunião Anual da SBPC. Florianópolis – SC, 16 a 21 de julho de 2006. (Resumo).

FRANKE, Hans Alberto; WESTPHALL, Carlos Becker; ROLIM, Carlos Oberdan; NAVARRO; Fabio. Mobilidade em Ambiente de Grades Computacionais. XXIV Simpósio Brasileiro de Redes de Computadores. IV Workshop on Computational Grids and Applications - WCGA. Curitiba - PR, de 29 de maio a 2 de junho de 2006 (Resumo).

FRANKE, Hans A.; ROLIM, Carlos O.; WESTPHALL, Carlos B.; KOCH, Fernando; BALEN, Douglas O. Grid-M: Middleware to Integrate Mobile Devices, Sensors and Grid Computing. In The Third International Conference on Wireless and Mobile Comunications – ICWMC 2007, Guadeloupe, French Caribbean, March 4-9, 2007.

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

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

GRID-M website, http://grids.lrg.ufsc.br/

GRIDBUS Project web-site, Grid Computing and Distributed Systems Laboratory, http://www.gridbus.org/

GRIDFORUM Website "Understanding Grids" Acessado em 08 de dezembro de 2005, disponível em: http://www.gridforum.org/UnderstandingGrids/ggf_grid_understand.php

J2ME. Java 2 Micro Edition web-site, Sun Corporation, http://java.sun.com/j2me.

KOCK, F; MEYER, J.-J. C; DIGNUM, F and RAHWAN. I. Programming deliberative agents for mobile services: the 3apl-m platform. In Proceedings of AAMAS'05 Workshop on Programming Multi-Agent Systems (ProMAS'2005)., 2005.

LEGION, World Wide Virtual Computer, http://legion.virginia.edu/

LDAP, openLDAP http://www.openldap.org/

LITKE, A.; SKOUTAS, D and VARVARIGOU, T. "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.

OBJECT WEB, Open Source Middleware. Acessado em 28 de dezembro de 2005, disponível em: http://middleware.objectweb.org/

ROLIM, Carlos Oberdan; WESTPHALL, Carlos Becker; KOCH, Fernando; ASSUNÇÃO, Marcos. Towards a Grid of Sensors for Telemedicine. CBMS 2006 - 19th IEEE International Symposium on Computer-Based Medical Systems. Salt Lake City, USA, 22-23 June, 2006.

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

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

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

UNICORE, UNIform Interface to COmputer Resources, http://www.unicore.org/

XML, W3C Consortium http://www.w3.org/XML/

Anexos

1. Cadastro e Consultas de tarefas através de um PDA

```
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();
}
}
}
```

Artigo

Grid-M : Middleware para Integrar Dispositivos Móveis, Sensores e Grids

Hans Alberto Franke[1], Carlos Oberdan Rolim[1], Fernando Koch[1], Douglas de Oliveira Balen[1], Carlos Westphall[1].
Laboratório de Redes e Gerência
Departamento de Informática e Estatística – INE
Universidade federal de Santa Catarina - UFSC
Florianópolis – SC - Brasil
{koiote, oberdan, koch, douglasb, westphall} @ inf.ufsc.br

Abstract. *Neste artigo propõe-se a criação de um middleware que ofereça suporte a integração entre rede de sensores (WSN), grids computacionais e dispositivos móveis. Será discutido como o middleware fornece serviços como gerência, comunicação de dados, diretório de serviços, descoberta de recursos e segurança, provendo ao desenvolvedor um conjunto de recursos reutilizáveis e homogêneos. Serão discutidos também problemas inerentes a este cenário e a forma como foi solucionado.*

Palavras chaves { Aplicações em Grid; Computação Móvel; Middleware}

1. Introdução

A solução fornecida pela computação em Grid vai além de criar redes de processamento e armazenamento distribuídos. Argumenta-se que é também uma técnica poderosa para promover a homogeneização ou a virtualização criando redes de fornecedores de serviço onde cada nó compartilha serviços ao Grid. Particularmente nas Wireless Sensor Networks (WSN), os nós executam em diversos ambientes, com sistema operacional (SO) diferente (Windows, Linux, J2ME), em hardware (desktop, PDA, dispositivos móveis), e em redes (TCP/IP, Wi-Wi-Fi, WAP). As aplicações devem lidar com as diversidades e entregar um serviço de qualidade. Conseqüentemente, a pergunta está em *Como integrar estes ambientes em uma infra-estrutura comum para fornecer serviços?* ou; *Qual a plataforma ideal para promover a homogeneização em um Grid de Wireless Sensor Networks ?*

Diversos estudos estão sendo realizados no âmbito desta pergunta. Por exemplo, na comunicação de dados, simples, rápida e mais confiável em redes de transmissão de dados [7]; no hardware, novos dispositivos com mais recursos e em plataformas mais simples de desenvolvimento, e; na tecnologia de programação há a criação de metodologias do desenvolvimento capazes de lidar com as restrições de recursos existentes [4]. Entretanto, sugere-se que falta na literatura ainda uma solução completa e reusável para integrar Wireless Sensor Networks, provedores de serviços, e dispositivos estáticos (workstations) e móveis de computação. Será detalhada esta argumentação em nossa revisão de trabalhos relacionados, na seção IV. Sugere-se que esta abertura na literatura apresenta uma oportunidade de contribuir com um middleware que integre os dispositivos móveis, wireless sensor networks e o Grid.

Neste trabalho, será apresentado o middleware: Grid-M. Ele é uma plataforma para construir aplicações de computação em Grid e aplicações em dispositivos embargados e móveis. Fornece uma Application Programming Interface (API) para conectar aplicações

desenvolvidas em Java em um ambiente de computação em Grid. Seu perfil runtime é pequeno bastante para ser usado em aplicações de computação móvel. Nós acreditamos que as soluções do Grid-M respondam à pergunta sobre uma plataforma para promover a homogeneização neste ambiente de desenvolvimento, as razões serão detalhadas a seguir.

Este artigo é organizado da seguinte forma: A seção seguinte apresenta o cenário e a motivação para este estudo. Então, a seção 3 introduz a proposta para uma plataforma leve de computação em Grid, o Grid-M. Na seção 4 apresenta-se uma simulação de uma ferramenta em cima do GridM, uma plataforma de gerenciamento de rede: NetManager. As características do GridM são comparadas a trabalhos relacionados apresentados na seção 5. A conclusão sobre a proposta e as suas soluções para este cenário é apresentado na seção 6.

2. Cenário e Análise de Requisitos

2.1. Motivação

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 computação em grid e computação móvel, através da criação de um grid middleware para o desenvolvimento de serviços móveis. O grid middleware descrito neste trabalho, recebe o nome de GRID-M. Ele oferece serviços como comunicação de dados, diretório de serviços, descoberta e segurança, desta forma provendo ao desenvolvedor um conjunto de recursos reutilizáveis e homogêneos, o que é garantido pela utilização do middleware como uma API de programação para facilidade do usuário.

2.2 Análise de Requisitos

Para que este sistema seja realizável, alguns requisitos são necessários, e seguindo o paradigma de Orientação a Objetos, será proposto alguns métodos e classes para regulamentar este cenário.

Para que haja comunicação entre os dispositivos e os sensores é necessário alguns (a) métodos de comunicação de dados; para que não haja interferência externa indevida aos dados faz-se necessário alguns (b) métodos de segurança que façam o controle de acesso ao sistema; Para agregarmos recursos a grid, devemos ter (c) métodos de registro de recursos que permita, por exemplo, o controle de entrada e saída de novos sensores ou assistente pessoais cadastrados no sistema; para que seja possível controlarmos as tarefas a serem processadas, os dispositivos pertencentes a grid e outros recursos, devemos ter (d) métodos para gerenciamento

dos dispositivos, que permitam gerenciar sensores, coletores e assistentes pessoais de forma centralizada, apesar da distribuição, dinamicidade do ambiente e heterogeneidade dos dispositivos; e para que todo sistema possa ser mais facilmente operável, devemos ter (e) interfaces amigáveis e padronizadas que possibilitem uma melhor interação, e (f) métodos para a coordenação de trabalhos entre os usuários móveis e que permita colaboração entre os assistentes pessoais nesse sistema.

O suporte para estes requisitos está além do escopo das linguagens de programação que devem ser executadas em vários dispositivos (e.g. Java na sua versão Java 2 Micro edition [1]) 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.

Acredita-se que este suporte seja oferecido pela integração das tecnologias de Grid Computing e Computação Móvel [15], o que é proposto através do Grid-M que será apresentado com maiores detalhes na seção seguinte.

3. GridM

Apresentada a motivação e análise dos requisitos necessários para implementação de aplicações que suportem serviços móveis surgiu à questão: *Como integrar um grid computacional com rede de sensores e dispositivos moveis?*

Em [15] é descrito duas formas de implementação de uma grid de sensores baseadas em dois focos. O primeiro chamado de (*centralized sensor-grid computing approach)* aponta para uma simples conexão e interface de todos os sensores e sub-redes de sensores na grid, deixando para o middleware a gerência de todos os dispositivos. Este tipo de implementação tem algumas desvantagens, pois necessita de comunicação excessiva com o sensor, o que acarreta um desgaste maior de sua bateria, além de possuir comunicação direta com o sensor e em caso de pane este ficaria inoperante.

Uma alternativa mais eficiente seria o (*distributed sensor-grid computing approach*), o qual possui uma arquitetura distribuída, poder de processamento e capacidade para tomar decisões por parte dos sensores, eliminando a comunicação excessiva presente no primeiro approach.

3.1 Arquitetura do Grid-M

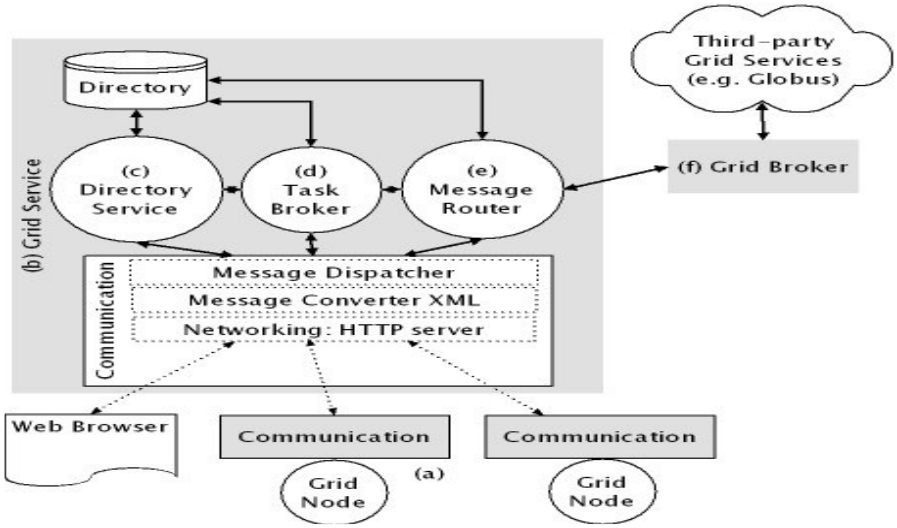

Figura 1 – Arquitetura do Grid-M

A figura 1 apresenta os componentes da arquitura proposta. Na figura 2 é mostrado a arquitura de um nodo, que seria a unidade funcional do middleware, pois é o responsavel por mandar e receber tarefas, e disponibilizar os serviços através do provedor de serviços. Os modulos dessa arquitetura são descritos abaixo.

Figura 1 apresenta a arquitetura do Grid-M. O sistema é composto por:

a. GRID NODES, como descrito abaixo.

b. GRID SERVICES são as representações abstratas de provedor de serviços de Grade. Necessariamente não é um host (ou nodo) ou um agrupamento de nodos. Pode ser implementado como um serviço distribuído ao longo do sistema. È importante dizer que o Grid Service também é um provedor de serviços que deve responder as requisiçoes de serviços.

c. DIRECTORY SERVICE é a estrutura onde serviços são registrados e correlatam a nodos capazes de implementá-los. Toda vez que um novo nodo registra-se no Grid Service ele deve se autenticar e publicar a lista de serviços que é capaz de executar.

d. TASK BROKER é a estrutura que controla pedidos de serviço, coordena distribuição de pedido de tarefa (por exemplo no caso de separação de tarefa) e no caso específico de grades de dispositivos móveis e embutidos onde o dispositivo não tem conectividade permanente isto pode armazenar temporariamente resultado de execução de tarefa, para recuperação tardia.

e. módulo de MESSAGE ROUTER implementa o roteamento de mensagens de comunicação, por resolução de nome e acessando informação sobre endereço de rede físico do banco de dados de diretório (esta informação foi alimentada lá durante processo de inscrição de nodo). Além disso, no caso de Grades de dispositivos Móveis e Embutidos, alguns destes dispositivos poderiam não ser transmitidos em rede diretamente (i.e. TCP/IP) ou disponível como ouvinte de porta (por exemplo dispositivos de WAP não têm um real endereço de TCP/IP e não podem ser um ouvinte de porta de TCP/IP). Este módulo implementa a roteamento de mensagem a estes dispositivos, o que em alguns casos, poderia requerer proxy, store-and-forward facility ou outras técnicas.

f. GRID BROKER é a estrutura para interconectar Grid Services externos. Promove tradução de mensagem e codifica em diferentes grid services (i.e. gateway).

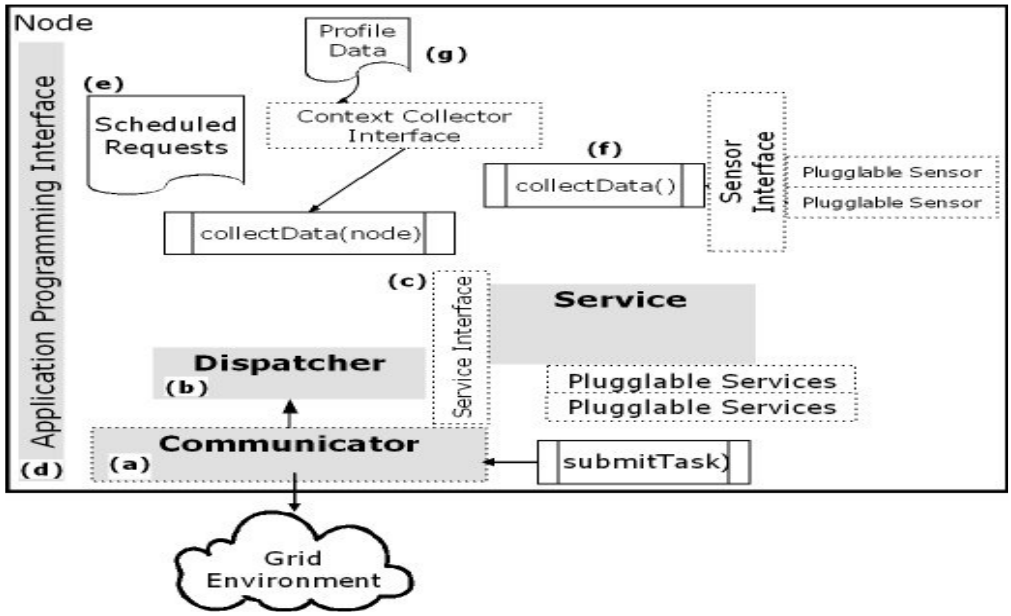

Figura 2 - Grid-M Node Architecture

Um nodo possui os seguintes modulos:

a. COMMUNICATION module: contem as sub rotinas no escopo da comunicação inter-nodo. O protocolo escolhido foi HTTP sobre TCP/IP. É esperado que nodos sejam capazes de HTTP-cliente (ou por uma proxy para esses nodos não equipados com TCP/IP). Nodos podem receber pacotes tanto por HTTP-server port (ouvindo a default Grid Middleware port) ou os pacotes são recebidos via http-server que funciona como storage-relay. O communication module guarda as sub-rotinas para converter representação XML para estruturas de dados internas, e tem interface para (b) o message dispatcher module.

b. DISPATCHER module: contem as sub-rotinas para dispachar mensagens recebidas (aqui ja carregadas dentro das estruturas de dados internas) para os serviços plugados. Serviços de perfil em obrigatório para todo nodo. Serviços novos podem ser plug-in pela interface de programação. O dispatcher é baseado no PERFORMATIVE contido na mensagem recebida. São associados PERFORMATIVES a serviços novos durante o processo de plug-in.

c. SERVICE INTERFACE permite plugar serviços para serem implementados; ele seta call-backs para pontos de execuçao de serviços.

d. APPLICATION PROGRAMMING INTERFACE, permite a conexão de aplicações externas em cima do node middleware. São implementadas aplicações de Java neste módulo. Eles podem usar os Service Executors para receber chamadas do node middleware e resposta de instrumento para pedidos de comunicação entrantes.

e. SCHEDULED REQUESTS e SCHEDULER MODULE fazem parte da implementação do node middleware e permitem a programação de pedidos de comunicação internamente gerados em períodos estabelecidos de tempo. A

programação de pedidos terminados está na (d) APPLICATION PROGRAMMING INTERFACE.

f. SENSOR INTERFACE permite plug-in de sensores que coletam dados de modulos externos e que eventualmente podem conectar em hardware externo

g. CONTEXT COLLECTOR INTERFACE: permite plug-in novos coletores de dados de contexto através de Node.addProfileContextCollector(ProfileDataCollectorInterface collector) e addProfileStaticCollector(ProfileDataCollectorInterface collector) . Node implementa SensorInterface que de fato coleta seus proprios dados (i.e. Context). Self-data collection e transmissao para o Service Provider pode ser agendafas atraves do Node.scheduleContextCollect(int intervalSec) .

3.2 Modelos de Comunicação do Grid-M

3.2.1. Modelo de Comunicação J2SE

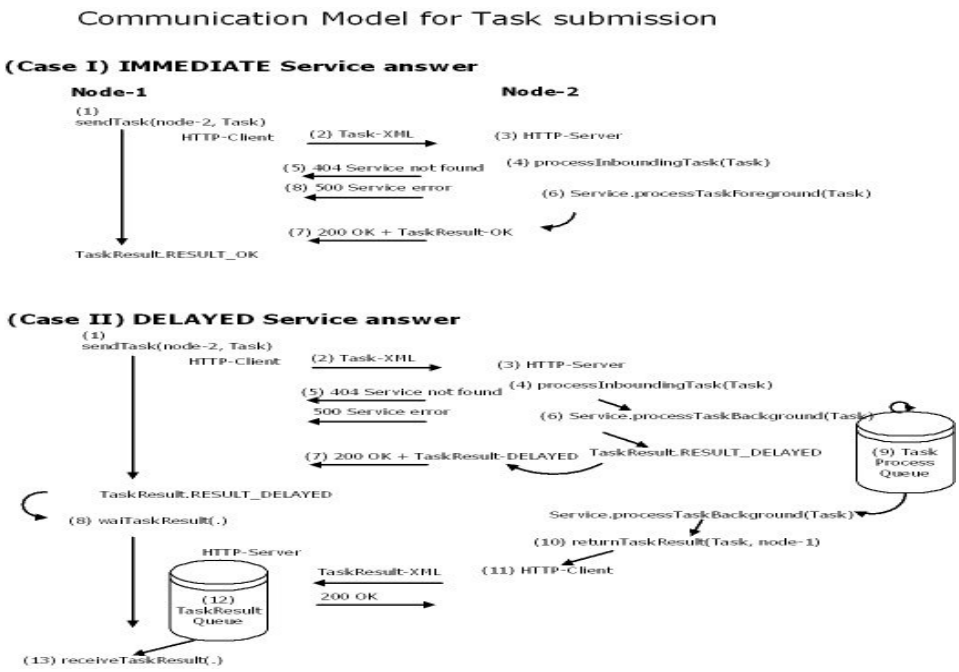

Figure 3. Grid-M Communication Model

A figura 3 apresenta o modelo de comunicação do Grid-M. Existem duas classes de comunicação no Grid-M: **IMMEDIATE**, quando o *service* processa a tarefa imediatamente e retorna um *TaskResult* através da mesma conexão de comunicação (e.g. HTTP -- e.g. a TaskResult é retornado pelo *HTTP-response* pela *HTTP-request* para *Task processing)*. O **DELAYED** Service por sua vez joga a *Task* para ser processada depois através de um *TaskResultQueue*, assim que o nodo puder processar a tarefa o seu *TaskResult* será enviado.

3.2.2. Comunnication Model For Mobile Devices

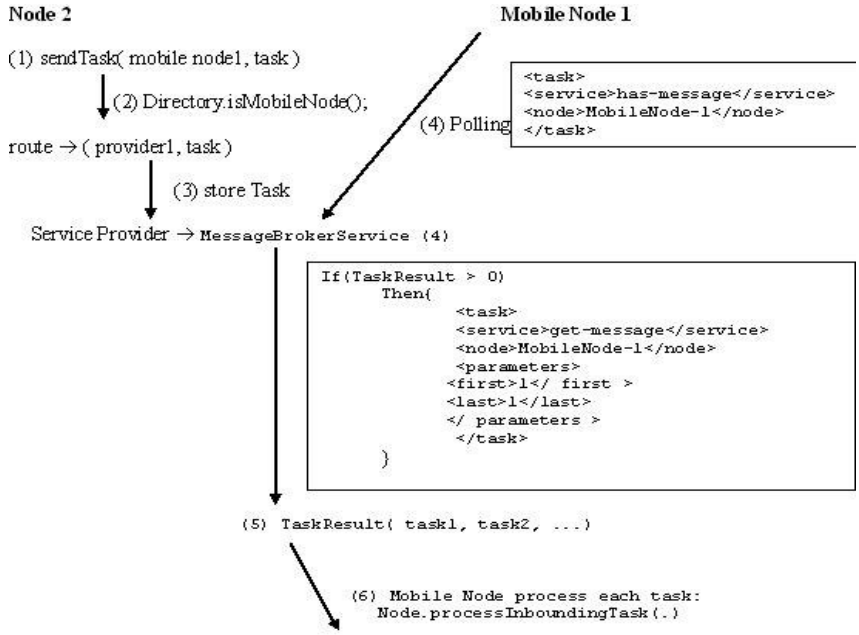

Figura 4. Communication Model para Dispositivos Móveis

A figura 4 mostra as interações entre um Mobile Node (PDA, Celular, etc..) com o comunicator do Grid-M. Esta comunicaçao é diferente de uma comunicação j2se (figura 3) porque o dispositivo móvel não é socket-server e por isso não tem suporte ao TCP/IP, mas o WAP faz a tradução MSN<->TCP/IP. Ele apenas é um cliente http e por isso deve se comunicar com o grid, para checar se tem mensagem para si, através de polling.

As interaçoes são as seguintes:

1. Um nodo quer enviar uma tarefa para outro nodo;
2. O Diretorio verifica se o nodo é um mobile node, caso isto aconteça a rota entre eles passa a ser o Service Provider, no caso provider1;
3. A tarefa é armazenada dentro do Storage;
4. O mobile node continuamente realiza um polling, o http-client abre uma conexão para o service provider, e pergunta se tem algum pacote para ele;
5. Caso tenha mensagens elas são armazenadas num TaskResult e são enviadas para o mobile node.
6. Então o mobile node processa cada tarefa do TaskResult.
7.

4. Simulação: NetManager

O Netmanager é um projeto do Laboratório de redes de Gerencia da Universidade Federal de Santa Catarina, que tem como objetivo construir um gerenciador de redes em um ambiente de grades de computadores. Toda a parte de comunicação HTTP, transmissão de dados, representação, codificação e decodificação do XML, entre outras, é feito pelo Grid-M, assim tem-se uma grande reusabilidade.

4.1. Arquitetura

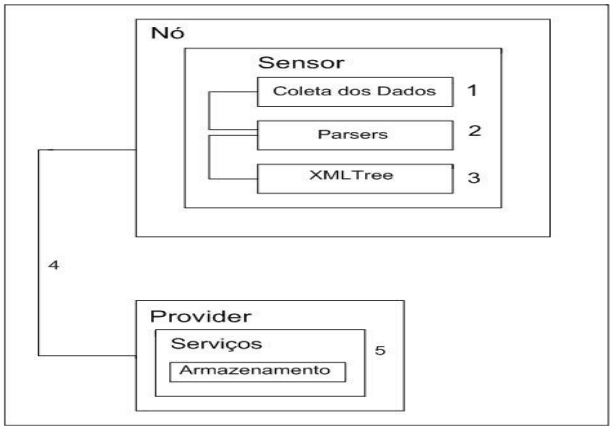

Figura 5. Arquitetura do Gerenciador

Para realizar o gerenciamento, cada nodo da grade deve possuir um sensor que será o responsável por realizar uma seqüencia de passos. Estes passos são ilustrados pela figura5 que apresenta a arquitetura do gerenciador.

- **Coleta dos Dados**: Neste passo os nós da grade irão executar funções internas, que revelarão o estado do seu contexto de execução. Após cada execução de método, os dados retornados serão repassados ao seu respectivo parser (cada comando tem o seu parser específico).
- **Parsers**: Este passo tem a função de filtrar dos dados coletados o que realmente é importante. Ao executar um comando, por exemplo, são retornados dados relevantes e irrelevantes. Cabe ao parser analisar isso. Após realizada a filtragem, os dados são repassados ao módulo responsável por criar as XMLTree.
- **XMLTree**: Para que os dados tenham um formato padrão, os dados relevantes oriundos de todos os parsers de cada comando executado serão formatados em forma de árvore XML. Isso facilita tanto a compreensão dos dados como também a fácil manipulação no momento do seu armazenamento no Provider.
- **Envio ao Provider**: Após a montagem da árvore XML, ocorre o envio dos dados do contexto de execução do nodo ao Provider. Isso é feito utilizando a parte de comunicação do Grid-M. Esta comunicação é feita pelo protocolo HTTP.
- **Armazenamento**: As XMLTree oriundas de todos os nós da organização virtual serão armazenadas no serviço de armazenamento do Provider. Estes dados que chegarão em intervalos periódicos serão a base para a construção do gerenciador, já que poderemos ter os dados de cada nó em vários períodos de tempo.

Dentro do coletor de Contexto de um nodo, são feitos coletores podendo ser implementações de uma interface Java, que coletassem dados do OS, memória, processador, network e outros dados relevantes.

4.2. Métodos de Coleta dos Dados

Os sensores irão utilizar basicamente dois métodos de coleta, os scripts escritos em *Virtual Basic Script* para ambientes Windows, e a execução de linhas de comandos para todos os ambientes (Linux, Windows, MacOS).

4.3. Tela do gerenciador

Figura 6. Tela do Gerenciador

A figura acima, apresenta a tela principal do gerenciador. São mostrados todos os nós da grade, dando a possibilidade ao gerenciador da rede obter mais dados sobre algum nó, apenas clicando sobre o a seta, são apresentados os dados na forma de gráficos e relatórios, que podem ser exportados no formato *pdf* e *csv*.

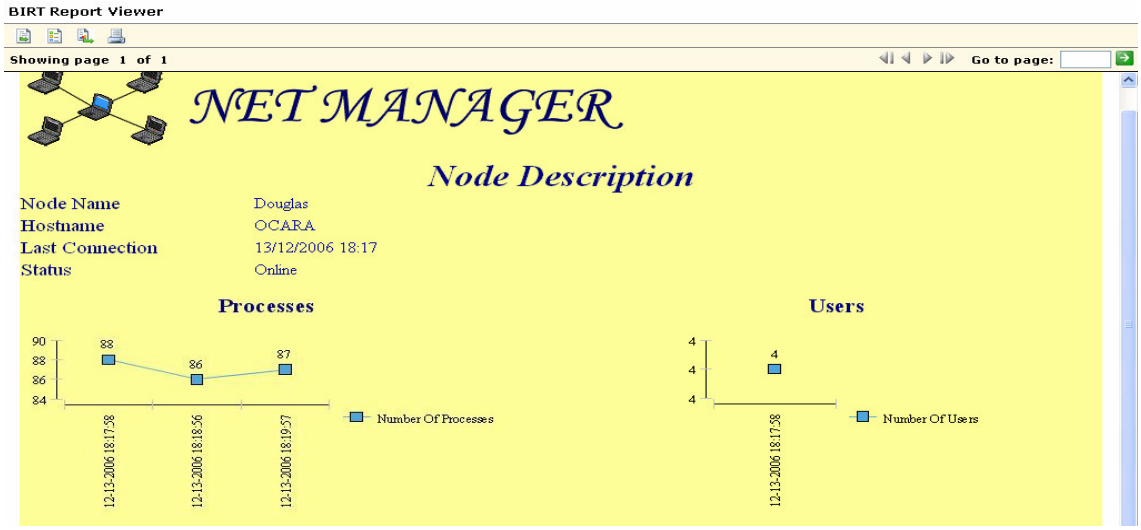

Figura7. Gráfico de informações detalhadas de um nodo especifico.

4.4. Justificativa

A estrutura de Grid proporciona que cada nodo seja um provedor de serviços, portanto o NetManager passa a ser um serviço executado em um nodo. Esta estrutura de mapeamento de todos os dispositivos da grade serem tratados como nodos através da virtualização facilita o roteamento, comunicação e o armazenamento, pois estes são feitos facilmente com o uso da API do GridM.

O NetManager pode ser usado para suporta algumas aplicações:

- Pode ser usado pelo gerenciador da rede para monitorar a performance de um nodo do Grid, podendo visualizar os dados coletados através de um Browser;

- Os dados coletados podem ser usados por algum service que precise garantir Qualidade de Serviço(QoS);
- Os dados de Contexto podem ser usados para detector intrusão em um nodo do Grid. Existe um trabalho em andamento no LRG, ele detecta intrusão por um uso anômalo dos recursos de um dispositivo. Assumindo que o uso padrão de CPU é 5%, se em algum momento o gerenciador coletar 80% de uso de CPU por um longo período, isto pode ser visto como uma intrusão.

5. Trabalhos Relacionados

[14] 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 encontram-se dois aspectos afins ao Grid-M, primeiro, o uso de grids como tecnologia que homogeneíza e integra os dispositivos e, segundo, no aspecto de que esta arquitetura disponibiliza aos médicos, acesso a informações, através de PDA, inclusive fora do ambiente do hospital (evidenciando a mobilidade).

Este trabalho complementa o Grid-M 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 prove colaboração (*colaboration*) outra propriedade da computação móvel e distribuída.

Em [16] 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.

Neste artigo encontramos citação ao padrão [17] OGSA (Open Grid Services Architecture) o qual define um conjunto de interfaces para desenvolvimento de sistemas grid, padronizando e viabilizando compatibilidade, isto se assemelha ao Grid-M pois também propõe uma API para desenvolvimento em cima do Middleware. Em comum também há 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

5.1. Principais Middlewares

Como descrito em [5], 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. [7] propõe que essas limitações são inerentes ao ambiente e devem ser amenizadas, mas não suplantadas, por desenvolvimentos tecnológicos futuros. Baseado nos requisitos descritos anteriormente, nós avaliamos os middlewares existentes 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 [3] é um software de código aberto, desenvolvido pela *Globus Alliance*, que oferece um kit de ferramentas para desenvolver aplicações e sistemas de computação em grade.

Seu suporte para as características necessárias em aplicações para serviços móveis acima descritas é encontrado em (a) 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 (c) 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.

O GRIDBUS [9] 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.

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

O Legion [10] 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, 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 (a) suporte a colaboração, onde por meio do sistema é possível colaboração através de tuplas como <LOID, LOA, timeout> que fornecem, endereçamento aos objetos (LOID) e gerenciamento provido por (LOA). Existe também (c) 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.

O UNICORE (UNIform Interface to COmputer REsources) [11] 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 (a) 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 (c) 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.

TABELA1. COMPARAÇÃO DOS PRINCIPAIS MIDDLEWARES EM RELAÇÃO AS CARACTERÍSTICAS PARA APLICAÇÕES EM SERVIÇOS MÓVEIS

Características dos middleware	Globus	GridBus	Legion	Unicore	Grid-M
Suporte a Colaboração	Sim	Sim	Sim	Sim	Sim
Suporte a Sensibilidade ao Contexto	Não	Não	Não	Não	Sim
Suporte a Alocação de Recursos	Sim (GRAM)	Sim	Sim (LOA	Sim (AJO)	SIM
Suporte a Ambientes Dinâmicos	Não	Não	Não	Não	Sim
Suporte a Execução em Dispositivos Móveis	Não	Sim	Não	Não	Sim

A Tabela 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* 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 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 $S=\{\alpha,\beta,\gamma,\delta,\varepsilon\}$.

6. Conclusão e Trabalhos Futuros

Neste trabalho foi proposto um middleware para integrar os dispositivos móveis, sensores e o Grid computacional como uma tentativa para responder à pergunta: *Qual a plataforma ideal para promover a homogeneização em um Grid de Wireless Sensor Networks* ?

Apresentou-se o middleware Grid-M. Ele é uma plataforma para construir aplicações de computação em Grid e aplicações em dispositivos embargados e móveis. Fornece uma Application Programming Interface (API) para conectar aplicações desenvolvidas em Java em um ambiente de computação em Grid. Seu perfil runtime é pequeno bastante para ser usado em aplicações de computação móvel.

Comparado aos trabalhos relacionados, destaca-se que o Grid-M cobre uma necessidade de um middleware que forneça um meio de desenvolver serviços para aplicações baseadas em Grid que integrem computação móvel. Conclui-se que a Grid-M responde à pergunta sobre uma plataforma que promova a homogeneização para o desenvolvimento de aplicações em Wireless Sensor Networks.

Entretanto, o Grid-M é ainda um trabalho em andamento. Como trabalho futuro será adicionado melhorias na plataforma, por exemplo, implementar diversas novas funções, como as listadas abaixo:

- Segurança, melhorando mecanismos de autenticação, criptografia e introduzir mecanismos de detecção de intrusão;
- Gerência e QoS introduzindo ferramentas para monitorar recursos e serviços, em um nível do módulo e em mecanismos para garantir a qualidade do serviço;
- Melhoramento do código em diversos aspectos, tais como o sistema de plug-in onde o novo código pode ser introduzido para controlar o ciclo de deliberação,

novo sistema de comunicação com introdução de interfaces de rede configuráveis, e assim por diante.

No futuro pretende-se desenvolver cenários maiores de teste para validar a sustentação da plataforma em diferentes ambientes.

Este artigo procurou fornecer uma visão geral do middleware Grid-M que é longe de completa. O material adicional, tal como a documentação, os exemplos, e o código de fonte, podem ser alcançados com o Web-Site: http://grid.lrg.ufsc.br.

Referências

2 Java 2 Micro Edition (J2ME) web-site, sun corporation, http://java.sun.com/j2me.

3 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.

4 I. Foster. 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.

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

6 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.

7 M. Satyanarayanan. Pervasive computing: vision and challenges. IEEE Personal Communications, 8(4):10-17, 2001.

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

9 Foster, I. & Kesselman, C. Foster, I. & Kesselman, C. (ed.) The Grid 2: Blueprint for a New Computing Infrastructure Elsevier, 2004

10 Gridbus Project web-site, Grid Computing and Distributed Systems Laboratory, http://www.gridbus.org/

11 Legion, World Wide Virtual Computer, http://legion.virginia.edu/

12 Unicore, UNIform Interface to COmputer Resources, http://www.unicore.org/

13 XML, W3C Consortium http://www.w3.org/XML/

14 LDAP, openLDAP http://www.openldap.org/

15 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

16 Chen-Khong Tham and Rajkumar Buyya. SensorGrid: Integrating Sensor Networks and Grid Computing. CSI Communications, pages 24-29, Vol.29, No.1, Computer Society of India (CSI) Publication, July 2005.

17 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

18 I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002

[18] Rolim, C. O, Koch, F. L., Assunção M., Westphall C. B, "Towards a Grid of Sensors for Telemedicine," cbms, pp. 485-490, 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06), 2006.

[19] Franke, Hans Alberto; Westphall, Carlos Becker; Rolim, Carlos Oberdan; NAVARRO; Fabio. Mobilidade em Ambiente de Grades Computacionais. XXIV Simpósio Brasileiro de Redes de Computadores. IV Workshop on Computational Grids and Applications. Curitiba - PR, de 29 de maio a 2 de junho de 2006 (Resumo).