UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Carlos Oberdan Rolim

Uma arquitetura para submissão de aplicações de dispositivos móveis e embarcados para uma configuração de grade computacional

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

Orientador: Prof. Dr. Carlos Becker Westphall

Florianópolis, Abril de 2007

Uma arquitetura para submissão de aplicações de dispositivos móveis e embarcados para uma configuração de grade computacional

Carlos Oberdan Rolim

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

Prof. Dr. Rogério Cid Bastos

Banca Examinadora

Orientador: Prof. Dr. Carlos Becker Westphall

Prof. Dr. Mário Antonio Ribeiro Dantas

Prof. Dr. João Bosco Mangueira Sobral

Prof. Dr. Bruno Richard Schulze

Nascer, morrer, renascer ainda,
e progredir sempre, tal é a Lei

Aos meus pais e minha irmã....

com amor e gratidão.

Agradecimentos

"Quando você quer alguma coisa, todo o Universo
conspira para que você realize seu desejo."
(Paulo Coelho, "O Alquimista")

Acima de tudo obrigado a Deus por todas as oportunidades. Que eu consiga cada vez mais deixar de ver meu próprio umbigo e prestar atenção nas dádivas que me são dadas a cada dia.

Obrigado aos meus pais Carlos e Cecília pela maior recompensa que um filho pode receber: ser amado! A forma honesta, simples e batalhadora de vocês transmitida através de conselhos e exemplos contribuíram e contribuem para meu crescimento. Obrigado pelo apoio dado em todos os momentos mas acima de tudo, obrigado por me receberem como filho. Amo vocês!

Agradeço também a minha irmã Cássia, por demonstrar desde cedo sua conduta firme e correta, que está sempre pronta a ajudar estendendo a mão aqueles que precisam. Os laços que nos unem são eternos e a ligação amorosa que temos faz com que caminhemos de mãos dadas. Te amo mana!

É inútil tentar descrever certos indivíduos. Só se pode mesmo seguilos... São pessoas que não se limitam em teorias, que demonstram através de ações toda a sua grandeza. Obrigado Fernando Koch pela paciência, pelos sábios conselhos, pelas conversas francas, pela dedicação comigo. Obrigado por me ajudar a descobrir o que fazer de melhor e, assim, fazê-lo cada vez melhor, por afastar o medo das coisas que não compreendia; levando-me, por fim, a compreendê-las de forma a despertar em mim a vontade de conhecer cada vez mais... Agradeço os momentos dos almoços na churrascaria, dos

cafés onde tu sempre me explicava o motivo e a forma de funcionamento das coisas me fazendo refletir, das caronas até a rodoviária em que quase perdia o ônibus, dos convites para fazer churrasco em plena hora de aula, do pouso oferecido, da mão amiga. Não existem palavras que demonstrem toda a minha gratidão por você ter sido meu co-orientador, me ajudando a chegar até aqui e também por me dar a honra de lhe chamar de amigo. Serei eternamente grato!

Obrigado ao meu orientador Prof. Dr. Carlos Becker Westphall pela confiança e oportunidade dada. Por me receber em sua sala em momentos em que eu não sabia que rumo tomar em minhas pesquisas e com toda paciência e prestatividade teres me demonstrado o melhor caminho a seguir esclarecendo minhas dúvidas.

Não poderia deixar de agradecer a Tati pela paciência e compreensão nos momentos em que deixamos de ficar juntos, pelos dias em que deixamos de passear ou ir ao cinema. Por ter me ouvido, acompanhado meu trabalho, me questionado, me motivado nos momentos que precisei e por estar comigo sempre. Querida, que o Bondoso Pai te abençõe em tua caminhada.

Agradeço aos amigos que fiz durante o mestrado como o Kleber Magno, Marcelo Cendron, Leonardo Kunrath, Hans Frank e Marcos Assunção do Laboratório de Redes e Gerencia. A todo o pessoal do Laboratório de Telemedicina, em especial ao Antonio da Luz, Levi Ferreira, Rodrigo Copetti, Daniel Duarte Abdala (Caju), Rafael Andrade e Rafael Simon Maia pelos momentos de labor, lazer e companherismo e ao Prof. Dr. Aldo von Wangenheim pela oportunidade de trabalho.

Agradeço aos meus familiares em especial a minha tia Lúcia pelo enorme auxilio que me deu durante a graduação em momentos de dificuldades financeiras, me empurrando e motivando para continuar caminhando que a recompensa viria a seu tempo. Por fim, sou grato a todos meus amigos, colegas de trabalho, professores da graduação e também da pós-graduação. Não irei colocar nomes pois cada um sabe o quanto é importante para mim e o quanto contribuiu para eu chegar até aqui. Muito obrigado a todos!

Sumário

Lista de Figuras

Lista de Tabelas

Lista de siglas

ECG eletrocardiograma

FIPA Foundation for Intelligent Physical Agents

GRAM Globus Resource Allocation Manager

GPRS General Packet Radio Service

GRIP GRid Interoperability Project

GPRS General Packet Radio Service

GSM Global System for Mobile Communications

HTTP HyperText Transfer Protocol

IDC Internet Data Center

OGSA Open Grid Services Architecture

OGSI Open Grid Service Infrastructure

P2P Peer-to-Peer

PCO2 pressão de gás carbônico

PDA Personal Digital Assistant

RIP Routing Information Protocol

SpO2 saturação de oxigênio

SNMP Simple Network Management Protocol

SOAP Simple Object Access Protocol

XSL Extensible Style Language

XSLT Extensible Style Language Transformation

WSN Wireless Sensor Network

WSRF Web Services Resource Framework

Resumo

O uso de aplicações móveis no dia-a-dia está se tornando cada vez mais comum. As facilidades do uso de redes sem fios podem ser percebidas na possibilidade de comunicação em qualquer lugar e a qualquer momento, permitindo integrar diversos aplicativos e plataformas. Surge então, a necessidade de prover uma solução integrada e homogênea para um ambiente, de forma que ocorra a comunicação, integração e gerenciamento de dispositivos móveis e embarcados juntamente com as aplicações que são executadas nesses dispositivos. Este trabalho visa demonstrar que uma grade computacional pode ser utilizada como software de base para efetuar o gerenciamento e a integração de dispositivos móveis e embarcados de forma que aconteça a homogeinização ou virtualização dos recursos existentes no ambiente. É efetuada a análise de um cenário-problema e então levantados os elementos e interações existentes de forma que são definidos os requisitos necessários a uma plataforma de grade capaz de ser aplicada a este cenário. Esta plataforma é implementada a partir dos conceitos apresentados no trabalho e seus resultados são demonstrados através de testes quantitativos e de um estudo de caso onde ela é aplicada para resolver um problema na área de telemedicina.

Abstract

The usage of mobile applications in day-by-day is becoming more common. The wireless networks facilities can be noticed in communication possibilities at any place and any moment, allowing to integrate several devices and plataforms. This calls for a homogeneous and integrated solution that promotes easy integration and management of mobile and embedded devices with their applications.

The aim of this study was to demostrate that computacional grid can be used as base software to manage and integrate that devices, so perfoming the virtualization of existent resources. We analyzed a problem-scenario to demonstrate existent elements and interactions, so we could define necessary requisites to a plataform that can be applied in that case.

A plataform's prototype was coded using the concepts defined in this research and its pratical results was demonstrated by quantitative tests and by a case study in telemedicine area.

Capítulo 1

Introdução

O uso de aplicações móveis no dia-a-dia está se tornando cada vez mais comum. As facilidades do uso de redes sem fios podem ser percebidas na possibilidade de comunicação em qualquer lugar e a qualquer momento, permitindo integrar diversos aplicativos e plataformas.

O constante aperfeiçoamento tecnológico proporciona que equipamentos como celulares, smartphones, Personal Digital Assistants (PDAs) e sensores sejam utilizados nas mais diversas áreas formando cenários mistos, onde dispositivos móveis, sensores e também dispositivos fixos sejam utilizados em conjunto para solução de algum tipo de problema. Um exemplo disso é o cenário apresentado na figura 1.1 onde diferentes tipos de equipamentos são utilizados para proporcionar o monitoramento de um ambiente. Para coletar os dados do ambiente uma rede de sensores (Wireless Sensor Network (WSN)) é utilizada de forma que os dados sejam coletados automáticamente. Os dados são então transmitidos via redes sem fios para um sistema central onde são processados e armazenados em uma base de dados. Usuários localizados no interior do ambiente podem consultar os dados coletados e também interagir com os sensores através de computadores de mesa ou PDAs. Existe ainda a possibilidade de usuários localizados fora do ambiente interno consultem os dados coletados via Internet através de um navegador, celular, smartphone ou outro tipo de dispositivo móvel.

Nesse cenário, existem problemas comuns em redes de sensores (como

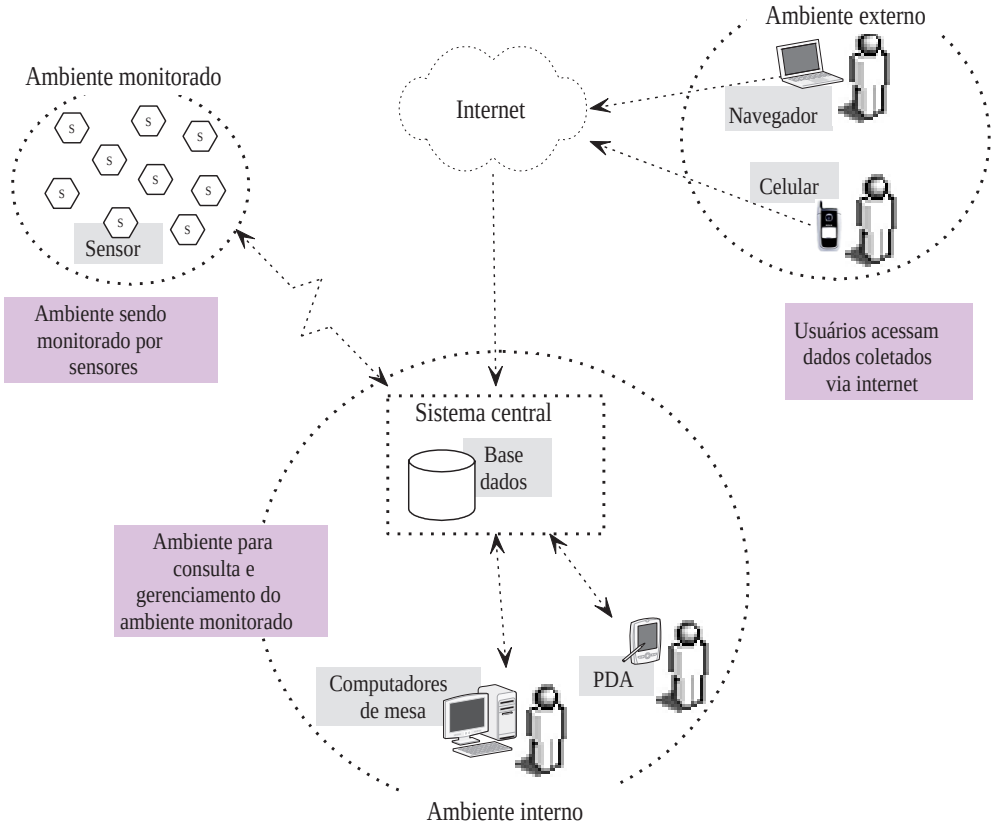


Figura 1.1: Exemplo de um cenário onde diferentes tipos de equipamentos são utilizados para proporcionar o monitoramento de um ambiente

gerenciamento de energia, roteamento eficiente, auto-organização e calibragem dos sensores) e também problemas associados à computação móvel (como heterogeneidade do ambiente, latência na comunicação, instabilidades da rede, necessidade de aplicações sensíveis ao contexto). Além deles, existe um outro problema que é comum aos dois mundos (móvel e embarcado) e que apesar dos avanços nas pesquisas científicas ainda não foi completamente solucionado: *como prover uma solução integrada e homogênea para um ambiente formado por dispositivos móveis e embarcados que permita a comunicação, integração e gerenciamento desses dispositivos juntamente com as aplicações que são executadas nesses dispositivos?* Outra carência refere-se a como garantir a re-usabilidade desta solução para que não seja necessário re-implementar ou alterar a infra-estrutura existente, sempre que for necessário o desenvolvimento de um serviço móvel que faça

uso dos dados coletados a partir de sensores ?

Ao analisar o cenário-problema, percebe-se que ele é propicio ao uso de grade computacional devido à diversidade de equipamentos e tarefas envolvidas na obtenção e disponibilização dos dados coletados. A aplicação de grade de computacional, nesse caso, é justificado pois cada equipamento utilizado é visto como um nodo da grade e não como um tipo específico. Dessa forma, não se torna necessária a implementação de uma interface de comunicação para cada um dos equipamentos existentes no ambiente, sendo que os dispositivos passam a ser gerenciados de uma forma homogênea. Aliado a isto, existe o reaproveitamento tecnológico dos serviços de transmissão de dados e segurança proporcionados de forma intrínseca pela grade.

Entretanto, para ser aplicada no cenário, uma plataforma de grade deve ser capaz de interagir com os sensores e também com os dispositivos móveis e fixos existentes, ou seja, a grade para esse tipo de aplicação deve possuir uma combinação das características de grades de sensores juntamente com as de grades móveis. Na literatura existente pode-se concluir que existem propostas de grades de sensores como descrito em [GHA 04] ou a arquitetura chamada SPRING [LIM 05] as quais poderiam ser utilizadas em um ambiente formado exclusivamente por sensores. Por outro lado as plataformas de grades móveis como o AKOGRIMO [BEC 06], GRIDKit [COU 04] ou MAG [dSeS 06] atendem alguns requisitos de mobilidade e poderiam ser utilizadas em um ambiente móvel. Porém, no caso do cenário acima que é composto tanto por dispositivos embarcados quanto por dispositivos móveis as plataformas apresentadas não podem ser usadas para solucionar o problema na totalidade, ou seja, ainda falta uma solução completa e re-usável que una os dois tipos de grades e seja voltada para a integração e o gerenciamento homogêneo dos equipamentos.

Sendo assim, esta situação apresenta uma oportunidade de contribuição com a construção de um plataforma que permita o gerenciamento e a integração de dispositivos móveis e embarcados via grade computacional de forma que aconteça a homogeinização ou virtualização dos recursos existentes no ambiente. Uma plataforma de grade formada pela combinação das características de grades de sensores com grades móveis é um aspecto inovador com diversas questões que a comunidade científica

internacional ainda não apresentou respostas.

Desde já define-se que o escopo desse trabalho é fazer uma pesquisa sobre quais os elementos, interações, requisitos, componentes e métodos são necessários para a integração de dispositivos móveis e embarcados via grade computacional de forma que o conjunto dessas informações torne possível a construção de uma plataforma que seja capaz de efetuar a homogeinização no gerenciamento de um cenário semelhante ao apresentado acima. Não são considerados em profundidade aspectos como segurança, sensibilidade ao contexto, descobrimento, auto-organização, roteamento, transmissão e qualidade de serviço que existem de forma intrínseca em grades móveis, nem mesmo aspectos relacionados a redes de sensores sem fios como gerenciamento de energia, segurança, protocolos usados, calibragem de sensores, tolerância a falhas e auto-organização.

1.1 Definição do problema e proposta

Como exposto anteriormente uma grade computacional genérica formada pela união das características de grades de sensores com grades móveis permite que ambientes móveis e embarcados sejam integrados e gerenciados de forma homogênea.

Dessa forma, no presente trabalho pretende-se responder a seguinte questão:

> *Qual é a estrutura de software necessária para desenvolver grades computacionais capazes de integrar e gerenciar ambientes formados por dispositivos móveis e embarcados?*

Para responder a questão, esse trabalho aborda quais são as pesquisas existentes na área de computação em grade e também aquelas que visam levar esse paradigma de computação distribuída para aplicações em ambientes móveis e embarcados. Além de abordar pesquisas na área, é efetuado um estudo para definir quais os elementos e interações existem em um ambiente de forma que possam ser definidos requisitos necessários a uma arquitetura capaz de integra-lo e gerencia-lo. Visando demonstrar o

comportamento e usabilidade da plataforma implementada - e conseqüentemente a funcionalidade da arquitetura proposta - é efetuado um conjunto de testes quantitativos onde são avaliados alguns experimentos que simulam algumas situações de uso da plataforma; também é apresentado um estudo de caso na área de telemedicina onde a plataforma é aplicada em um ambiente que possui uma rede de sensores coletando dados que podem ser então visualizados em tempo real através de dispositivos móveis.

No decorrer do trabalho as seguintes sub questões relacionadas serão respondidas:

- Qual a finalidade de utilização de uma grade computacional que combine as características de redes de sensores com computação móvel ?

  Essa pergunta visa demonstrar que uma plataforma de grade que possua a união das características de grades de sensores com grades móveis pode ser genérica ao ponto de ser aplicada em ambientes formados tanto por dispositivos móveis quanto por dispositivos embarcados. Com seu uso, o gerenciamento e a integração dos dispositivos existentes no ambiente acontece de forma homogênea, além de possibilitar que as aplicações que fazem uso desses dispositivos sejam desenvolvidas com maior facilidade.

- Quais as características e os módulos de software necessários em uma plataforma para suportar o elementos e interações de um ambiente formado por dispositivos móveis e embarcados ?

  Uma vez demonstrado que a aplicação de uma grade que combine características móveis e embarcadas em um ambiente proporciona uma série de facilidades, é feita uma análise de um ambiente de forma que sejam levantadas as informações necessárias para a definição de uma arquitetura que quando implementada forneça suporte aos diferentes elementos, interações e requisitos existentes nesse ambiente.

- Como se comporta uma plataforma de grade computacional aplicada na integração de um ambiente formado por dispositivos móveis e embarcados ?

Essa pergunta visa demonstrar a implementação da plataforma obtida a partir da definição da arquitetura proposta e a sua forma de comportamento quando aplicada em um ambiente formado por dispositivos móveis e embarcados. A funcionalidade é demonstrada através de testes qualitativos e da sua aplicação em um estudo de caso na área de telemedicina, mais especificamente no monitoramento remoto de pacientes.

Como contribuições do trabalho, pode-se destacar:

- definição de um modelo conceitual contendo elementos, componentes e interações necessárias para a construção de uma plataforma de grade que é capaz de ser aplicada no cenário-problema.

- apresentação de uma plataforma de grade computacional que possui características necessárias para ser aplicada em ambientes móveis e embarcados cobrindo uma lacuna existente na literatura.

- abordagem de como o paradigma de grade computacional pode ser utilizado para efetuar a integração e o gerenciamento de ambientes móveis e embarcados.

1.2 Organização da dissertação

Para apresentação dos resultados, o trabalho foi dividido nos seguintes capítulos:

- *Capitulo 1 - Introdução:* Visa introduzir e contextualizar o problema, além de definir o escopo que o trabalho abordará e quais são as suas contribuições.

- *Capitulo 2 - Motivação e Trabalhos Relacionados:* Apresenta o cenário-problema que é usado como motivação para desenvolvimento desse trabalho e quais os benefícios de aplicar grade computacional nesse cenário. Em trabalhos relacionados

são apresentadas informações referentes à área de computação móvel, redes de sensores, telemedicina e também são analisadas algumas plataformas de grade computacional de forma a demonstrar a inexistência de uma plataforma capaz de atender a todos os requisitos de um ambiente formado por dispositivos móveis e embarcados.

- *Capitulo 3 - Proposta:* Analisa um cenário-problema de forma a demonstrar quais os elementos e interações existentes. São então definidos alguns requerimentos necessários para que seja aplicado o paradigma de computação em grade nesse ambiente e posteriormente é proposta uma arquitetura que contemple todas as necessidades apresentadas no cenário inicial.

- *Capitulo 4 - Resultados:* Aborda a forma de implementação da proposta em uma plataforma de grade. Para demonstrar o seu comportamento, são apresentados resultados de testes quantitativos efetuados em alguns experimentos. Como forma de exemplificar a utilização e comportamento na prática é apresentado um estudo de caso na área de telemedicina onde sensores e dispositivos móveis são usados para coleta e visualização de dados de pacientes em tempo real.

- *Capitulo 5 - Conclusão e trabalhos futuros:* Sumariza as conclusões obtidas pelo progresso do trabalho e também apresenta quais são os rumos que serão tomados em trabalhos futuros.

Capítulo 2

Motivação e trabalhos relacionados

Este capítulo procura abordar quais as vantagens obtidas pelo uso de grade computacional para a solução dos problemas inerentes a ambientes formados por dispositivos móveis e embarcados. Além de apresentar o cenário-problema onde uma rede de sensores é usada juntamente com dispositivos móveis na área da telemedicina, são abordados estudos que serviram como base teórica para o trabalho.

O capítulo está organizado da seguinte forma:

- *Seção 2.1 - Apresentação do cenário problema:* Descreve o cenário, abordando quais são os seus problemas e os requisitos necessários para que um software seja capaz de resolver esses problemas. Com base nesses requisitos é demonstrado como o paradigma de grade computacional pode ser utilizado como software de base para a interligaçao e gerenciamento dos dispositivos existentes no cenário.

- *Seção 2.2 - Trabalhos relacionados:* Visa demonstrar quais os trabalhos existentes na área que foram utilizados como base de pesquisa. É efetuada uma análise de diferentes propostas existentes e então demonstrado como essas propostas não conseguem atender as necessidades do cenário-problema.

2.1 Apresentação do cenário problema

Redes de sensores são formadas a partir da distribuição espacial de dispositivos autônomos usando sensores para cooperativamente monitorar condições físicas ou ambientais como temperatura, som, vibração, pressão, movimento ou poluição em diferentes localidades. O contínuo avanço tecnológico está permitindo que as redes de sensores possam ser integradas com dispositivos móveis para aplicações de diversos fins. Um dos exemplos de uso dessas aplicações é na área de telemedicina.

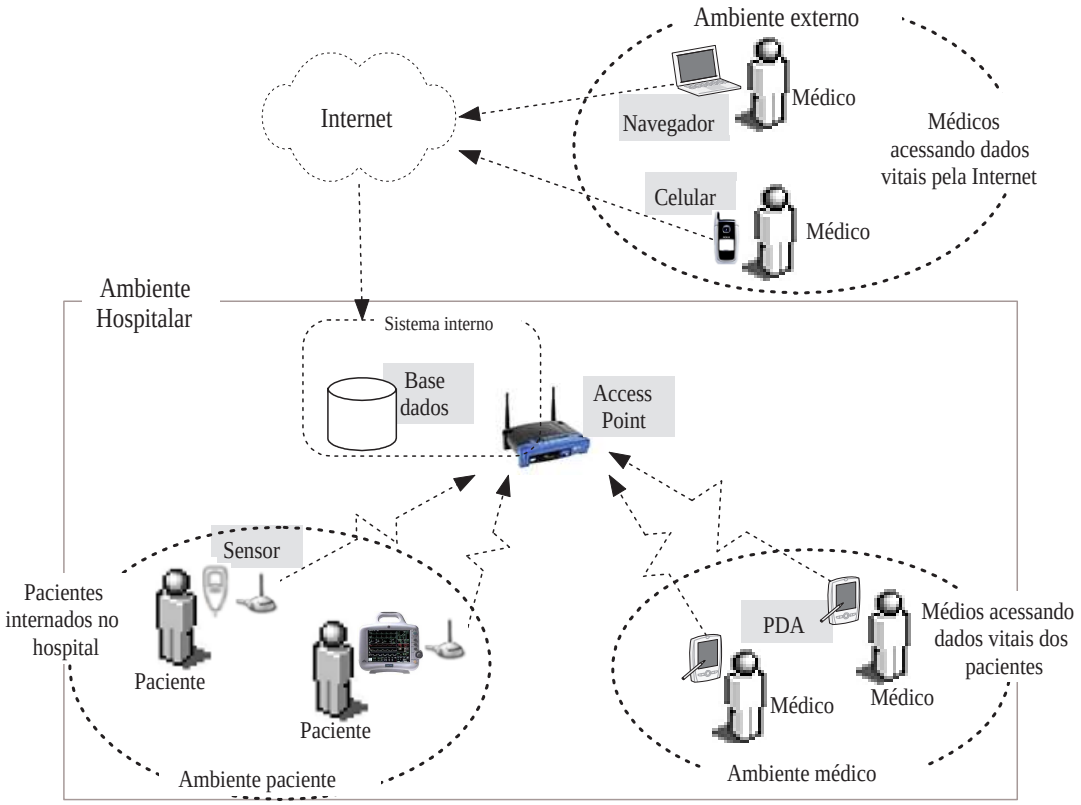


Figura 2.1: Cenário-problema demonstrando o uso de redes de sensores em conjunto com dispositivos móveis na área de telemedicina

A figura 2.1 apresenta um cenário-problema formado pela união de uma rede de sensores juntamente com dispositivos móveis na área de telemedicina. Nesse cenário um hospital quer disponibilizar um sistema de apoio à decisão para o seu corpo médico, sendo que os dados vitais dos pacientes internados ficam disponíveis aos médicos

em tempo real e podem ser acessados de qualquer lugar e a qualquer hora.

A coleta dos dados vitais dos pacientes é feita através de sensores conectados a coletores dispostos de forma fixa ao lado de cada leito ou então através de sensores móveis que transmitem os sinais através de ondas de rádio para pontos de recepção de dados dispostos no interior do hospital. Esses dispositivos estão conectados ao sistema interno do hospital o qual é responsável pela catalogação e armazenamento dos dados de cada paciente.

Nesse ambiente é necessário um padrão de segurança rígido entre os dispositivos responsáveis pela coleta de dados e o sistema de gerenciamento para evitar perdas de dados ou acessos indevidos. O acesso e análise aos dados são feitos por assistentes pessoais móveis (PDAs) carregados pelos médicos da instituição. Os médicos podem visualizar os dados de um ou vários pacientes em tempo real a medida que são coletados, tendo um acesso sem precendentes às condições vitais de seus pacientes. Além de monitorar, eles podem também programar seus assistentes pessoais para receber alarmes quando certas condições vitais de saúde dos pacientes são atingidas.

Outra possibilidade existente é que médicos localizados em pontos distantes, externos ao hospital, precisam de acesso aos dados dos pacientes internados. Usando qualquer dispositivo conectado a Internet (como um navegador internet, celular ou smartphone) o médico pode fazer acesso aos dados da mesma forma como se estivesse usando o seu PDA no interior do hospital.

Para que os dados dos pacientes possam ser acessados a partir de qualquer lugar e a qualquer hora é necessário o uso de equipamentos que coletem os dados dos pacientes de forma contínua. Em um ambiente hospitalar são usados diversos tipos de sensores que coletam os dados vitais dos pacientes como pressão sangüínea, pulsação, temperatura corporal, respiração abdominal, fluxo nasal, saturação de oxigênio (SpO2), pressão de gás carbônico (PCO2), nível de glicose e eletrocardiograma (ECG). Esses dados são adquiridos de maneira eficiente, sem a necessidade de uma pessoa para execução da tarefa e podem ser disponibilizados de forma digital pelo equipamento.

Os maiores problemas encontrados nesse cenário são:

1. Como fazer para que os diferentes tipos de sensores responsáveis pela coleta de dados e dispositivos de consulta móveis possam ser gerenciados no ambiente de forma homogênea sem a necessidade de implementação de uma interface de comunicação para cada uns dos equipamentos usados no ambiente;
2. Como disponibilizar os dados do paciente para os médicos em tempo real acessível de qualquer local e a qualquer hora; e
3. Como lidar com o dinamismo e mobilidade existente no cenário.

Para que esses problemas possam ser solucionados são necessários que os seguintes requisitos sejam atendidos:

i. método de comunicação para coleta e transmissão dos dados dos sensores sem fios e também dos coletores fixos existentes no hospital;
ii. método para garantir que os dados coletados sejam entregues com segurança ao destinatário;
iii. método de controle de recursos e registro de novos dispositivos no ambiente, como por exemplo, a entrada de um novo dispositivo de coleta ou outro dispositivo móvel utilizado por um médico na rede de comunicação;
iv. uma interface de acesso e visualização dos dados para os assistentes pessoais dos médicos;
v. uma interface de acesso e visualização dos dados para acesso via Internet;
vi. mecanismos para interação com os diferentes tipos de sensores usados na coleta de dados;
vii. mecanismos capazes de efetuar a reconfiguração automática dos dispositivos devido ao dinâmismo existente no ambiente.
viii. mecanismos que proporcionem a construção de aplicações que trabalhem de forma colaborativa e sejam sensíveis ao contexto.

ix. mecanismos que proporcionem a suporte a qualidade de serviço.

Optando por uma solução baseada no uso de grade computacional o desenvolvedor de aplicações re-aproveita os recursos disponíveis pela grade, re-usando assim as características inerentes ao ambiente da grade computacional como software de base capaz de fornecer os requisitos de (i) a (iii) descritos acima. Dessa forma ele passa a concentra-se em aspectos particulares da implementação do sistema, como os requisitos (iv) e (v).

Os requisitos de (vi) a (ix) não são atendidos pelo uso de grades computacionais "convencionais" as quais possuem como finalidade proporcionar a disponibilização de recursos em larga escala geralmente necessários por aplicações científicas. Uma plataforma de grade capaz de lidar com esses requisitos, fornecendo assim recursos para a integração e o gerenciamento de ambientes formados por dispositivos móveis e embarcados, é um aspecto inovador e relevante com diversas questões que a comunidade científica internacional ainda não apresentou respostas.

A próxima seção aborda quais são os trabalhos relacionados que foram utilizados para embasamento teórico do trabalho.

2.2 Trabalhos relacionados

Essa seção apresenta informações obtidas a partir de alguns trabalhos relacionados que foram usadas para embasamento teórico. As pesquisas foram efetuadas no sentido de buscar trabalhos relacionados às áreas de desenvolvimento de aplicações móveis, redes de sensores, grades computacionais e telemedicina.

2.2.1 Aplicações móveis, redes de sensores e telemedicina

Com o crescente uso de dispositivos móveis e embarcados - como notebooks, PDAs, celulares e sensores - na vida cotidiana surge um novo tipo de ambiente onde dados são disponibilizados o tempo todo em qualquer lugar. Esses ambientes são conhecidos como ambientes móveis. Segundo [PHA 02] ambientes móveis são formados

por dispositivos com baixo poder de processamento, capacidade de comunicação intermitente e não confiável, grandes períodos de desconexão, limitações no fornecimento de energia bem como o uso de interface de comunicação com o usuário de tamanho reduzido.

Devido a essa mudança na abrangência do ambiente e conseqüentemente na forma de uso dos dispositivos surge a necessidade de aplicações que sejam capazes de trabalhar com esse novo contexto no qual estão inseridas. Estas aplicações possuem algumas diferenças em relação às aplicações desenvolvidas para serem executadas em dispositivos fixos, como computadores de mesa e servidores. Conforme descrito em [KOC 05], o desenvolvimento de aplicação móveis precisam dar suporte aos seguintes aspectos: cooperação, interface com usuário, interface com os elementos do ambiente (sensibilidade ao contexto) e processos de inferência que possibilitem o desenvolvimento de sistemas que não sejam puramente reativos, ou seja, que possuam uma certa capacidade de autonomia. Outro fator extremamente importante são que as aplicações devem ser capazes de lidar com as limitações existentes no ambiente. [USK 03] propõe que essas limitações são inerentes ao ambiente e devem ser contornadas, porém não esquecidas, pelos desenvolvedores.

Diversos estudos estão sendo feitos para levar o uso de aplicações móveis as diferentes áreas do conhecimento. Uma das possibilidades emergentes que vêm sendo explorada pela comunidade científica é o uso de sensores para a aquisição de dados de um ambiente em tempo real. Esses dados são processados e posteriormente podem ser acessados através de dispositivos móveis.

Como sensores define-se pequenos dispositivos que possuem baixa capacidade computacional, fornecimento de energia limitado (através de baterias) e capacidade de comunicação sem fios utilizados para finalidades genéricas de aquisição de dados [STA 07] ou então dispositivos especializados, com maior capacidade de processamento, maior tempo de vida (alimentado por baterias ou corrente contínua) e usados para tarefas específicas de monitoramento.

Uma rede de sensores sem fios é formada por um conjunto de sensores que são capazes de detectar e transmitir através de ondas de rádio as características físicas do fenômeno ou ambiente no qual estão imersos. Os dados coletados são então ar-

mazenados em uma base de dados para posterior consulta [MAL 02]. Segundo [HEI 01] uma rede de sensores é um tipo particular de sistema distribuído, onde a comunicação de baixo nível não depende da estruturação topológica da rede. Devido a natureza dos dispositivos que formam este tipo de rede, ela possui algumas características particulares como por exemplo a extrema necessidade de economia de energia, constantes alterações de topologia, eleição de nodos que serão os líderes e coordenarão a aquisição de dados do ambiente, roteamento eficiente, calibragem para assegurar que os dados que estão sendo obtidos estão corretos, tolerância a falhas e auto-organização. Estas características fazem com que sejam necessários protocolos e estudos específicos para redes de sensores que não podem ser aplicados em outras áreas de sistemas distribuídos.

Em [AKY 02] é abordado que redes de sensores são utilizadas em diversos campos como militar, saúde ou doméstico. No campo militar as características de auto-organização e tolerância a falhas fazem delas um recurso para auxiliar no comando, controle, organização e definição de alvos. No campo da saúde sensores são usados para monitorar e assistir pacientes. Esse tipo de aplicação onde ocorre a união da computação com a medicina é conhecido como teleledicina. Além de proporcionar o monitoramento de pacientes a distância os objetivos da telemedicina são de garantir rapidez, segurança e confiabilidade em diagnósticos. [OBR 02] traz um exemplo de trabalho na área apresentando um framework baseado em agentes para dispositivos médicos virtuais. Esse framework faz o uso de um ambiente de agentes móveis para monitoramento, mineração de dados, e alertas em caso de anormalidades com pacientes.

Já arquitetura apresentada em [BHA 05] descreve a forma como dados multimídia de pacientes podem ser gerenciados e compartilhados por diferentes hospitais através de Internet Data Center (IDC). Esses dados podem ser acessados pelos médicos de fora do ambiente hospitalar. A arquitetura têm como foco a sensibilidade ao contexto, proporcionando ao paciente a localização do centro de tratamento baseado no seu plano de saúde e provendo ao médico acesso ao registro do paciente que podem estar localizados de forma distribuída em vários hospitais.

Em [JAS 05] é proposta a construção de uma rede sem fios para implementação de um sistema de telemedicina em tempo real para propósitos gerais. Essa rede seria cons-

truída usando o protocolo Bluetooth, redes Global System for Mobile Communications (GSM) ou redes General Packet Radio Service (GPRS). O trabalho explora os fatores que devem ser considerados ou avaliados em aplicações de telemedicina. Além disso é demonstrado o projeto e a implementação da plataforma para comunicação usando o protocolo Bluetooth. Um trabalho como esse, com a existência de uma plataforma que visa integrar uma diversidade de dispositivos, teria sua implementação simplificada através do re-uso de componentes proporcionados por uma grade computacional.

Uma plataforma chamada UbiMon utilizada para monitoramento de pacientes, usando sensores corporais implantáveis é apresentada em [NG 04]. Tal plataforma faz o uso de nodos para desempenhar a tarefa de aquisição, processamento e armazenamento dos dados de maneira semelhante à proposta apresentada nesse trabalho. Porém, a abordagem possui limitações e necessita da implementação de mecanismos para gerenciar diferentes tipos de sensores implantáveis que possam existir no ambiente.

Outro trabalho que faz o uso de sensores para monitoramento de dados vitais é apresentado por [CHE 03]. Nessa proposta, sensores são implantados no corpo do paciente e passam a formar uma rede responsável pela coleta dos dados vitais. Segundo o autor, os dados necessitam ser armazenados de uma forma confiável, sendo que a forma como os dados são protegidos requer alto poder computacional. Essa necessidade de grande poder de processamento é típica de aplicações na área de telemedicina. Uma forma de atender aos requisitos de infra-estrutura necessários por aplicações que demandam elevados recursos computacionais foi apresentado por [FOS 98]. Nesse trabalho é proposto o uso de grade computacional para fornecer a idéia de um metacomputador capaz de fornecer os mecanismos de base necessários para proporcionar o uso da telemedicina em larga escala. O uso de grade computacional na medicina por aplicações que necessitam de grande poder de processamento também é apresentado por [BER 03b]. Esse trabalho é baseado em uma arquitetura voltada à serviços construída sobre os padrões de grade e tecnologias web existentes que visam ocultar a complexidade de transformar as aplicações existentes em serviços para grade. A primeira versão já está operacional disponibilizando protótipos para simulações maxilo-facial, simulações cardio-vasculares, reconstrução avançada de imagens e também para simulação e análise de drogas.

Como percebe-se a telemedicina é uma área bastante explorada em pesquisas. Porém grande parte dos trabalhos existentes concentram-se em usar a telemedicina para atendimento, exames e diagnóstico remotos, simulações de comportamento do corpo e desenvolvimento de drogas. São poucos aqueles que propõem o uso da telemedicina para monitoramento remoto de pacientes. Quando procurado sobre trabalhos que usam grades computacionais na telemedicina todos os encontrados usam os serviços fornecidos pelo paradigma somente para explorar o alto poder de processamento oferecido pelas grades. O uso de grade computacional para gerenciamento de dispositivos médicos (como sensores corporais) e também para disponibilização de dados vitais de pacientes em tempo real para dispositivos fixos e móveis é algo inovador e não foi encontrado nenhum trabalho que possuísse essas características.

2.2.2 Grades de computadores

A área de grades computacionais [FOS 04] [BER 03a] [FOS 02a] [FOS 02b] é uma área emergente que usa o paradigma de construir aplicações em larga escala de forma distribuída, colaborativa e segura através da virtualização da infraestrutura [KRA 04].

Uma dos benefícios fornecidos pelas grades computacionais é a possibilidade de gerenciamento homogêneo de ambientes. A arquitetura de gerência de redes baseada em grades de agentes apresentada por [ASS 04] demonstra como uma grade pode ser utiliza em conjunto com agentes para coletar e analisar grande quantidade de dados de diferentes equipamentos de forma a ser uma ferramenta de auxilio para o gerenciamento de um ambiente.

O potencial de gerenciamento de dispositivos fornecido pelas grades muitas vezes não é aproveitado. Em [TIL 06] é apresentado um cenário para coleta de dados de um lago através de sensores. O autor propõe uma arquitetura orientada a serviços que visa lidar com a heterogeneidade dos sensores em uso e também formas de usar dispositivos móveis como PDAs ou smartphones para manipular os sensores. Uma grade computacional poderia ser emprega para o gerenciamento dos dispositivos de forma homogênea, porém haveria a necessidade de algumas mudanças na grade para ela ser capaz

de trabalhar com os dispositivos móveis.

Tomando como base o trabalho acima, percebe-se que existe a possibilidade de unir grades computacionais com sensores. A união de grade computacional com rede de sensores forma um tipo de grade que permite o acompanhamento e análise de fenômenos ou eventos que acontecem em um ambiente. Esse tipo de grade é conhecido como grade de sensores e possui métodos capazes de lidar com dispositivos de grande capacidade de processamento (como computadores de mesa ou servidores) e também de interagir com equipamentos de recursos limitados como sensores.

Na área de grades de sensores o trabalho apresentado por [GHA 04] faz uso do projeto DiscoveryNet [CUR 02] para a construção de uma infraestrutura usada na análise de poluição do ar e identificação de situações de risco em tempo real. Outro trabalho existente é a arquitetura chamada SPRING [LIM 05] que faz o uso de proxy para conectar uma rede de sensores a grade. Um proxy também é usado pelo MPAS [YUJ 05] para coordenar a comunicação e a integração de diversas redes de sensores a uma estrutura de grade. Esse framework usa uma estrutura de software baseada em módulos e serviços web para analisar, gerenciar e armazenar os dados de sensores.

Em [THA 05] é apresentado que redes de sensores podem ser integradas com grades computacionais em uma única plataforma combinando assim a aquisição de dados em tempo real do ambiente com a vasta disponibilização de recursos computacionais. Segundo o objetivo do presente trabalho, a grade computacional não é usada com a finalidade de disponibilização de recursos em larga escala mas como forma de gerenciamento e integração dos sensores existentes no ambiente.

Indo em direção a aplicação de grade computacional em ambiente móveis constata-se que é um tema inovador com diversas questões ainda não respondidas pela comunidade acadêmica. Dessa forma alguns conceitos são usados por alguns autores porém ainda não padronizadas pela comunidade. Um exemplo é a definição dada por [MCK 04] e [KLE 96] apud [dSeS 06] segundo o qual a mobilidade proporcionada pela infraestrutura de uma grade pervasiva pode ser dividida em dois diferentes mecanismos, de acordo com o tipo de mobilidade que devera ser suportada pela infraestrutura da grade: (1) usuários nômades e (2) usuários moveis. A mobilidade nômade é caracte-

rizada pelo deslocamento de usuários e/ou dispositivos através de limites institucionais, permanecendo desconectados enquanto se locomovem, e.g. um usuário que esteja utilizando um acesso dial-up em uma localidade deve desconectar-se antes de mudar para outra localidade,reconectando-se novamente no destino. Ao contrário do que ocorre com usuários nômades, usuários moveis não devem ter suas conexões interrompidas durante sua locomoção, sendo estas conexões mantidas através de tecnologias de comunicação sem fio, e.g. um usuário acessando a Internet através de um Smart-Phone utilizando tecnologia GPRS, representa um usuário móvel. No contexto desse trabalho tanto o suporte a usuários móveis quanto nômades será tratado como suporte a mobilidade.

No âmbito de plataformas de grade em ambiente móveis em [NAV 06] é abordado que um middleware para esse tipo de ambiente deve possuir os seguintes requisitos: suporte a colaboração; suporte a sensibilidade ao contexto; suporte a alocação de recursos; suporte a ambientes dinâmicos e suporte a execução em dispositivos móveis. De forma resumida o autor efetuou a seguinte análise de middlewares de grade computacional baseando-se nos requisitos:

- O GLOBUS [FOS 97] 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. É muito utilizado para o desenvolvimento de aplicações científicas que necessitam de alto poder computacional. O Globus possui mecanismos como o Globus Resource Allocation Manager (GRAM), GRid Interoperability Project (GRIP), GridFTP Também existe o (c) suporte a alocação de recursos, provido pelo gerenciador de recursos GRAM que fornece uma interface para envio e monitoramento de tarefas.

- O GRIDBUS [BUY 07] 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.

- O Legion [GRI 99] [LEG 97] 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 UNICORE (UNIform Interface to COmputer REsources) [ALM 99] 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.

Baseado nessa análise o autor montou a tabela 2.1 que esquematiza o suporte dos middlewares aos requisitos definidos.

Conforme detalhado no inicio da seção, ambientes móveis possuem restrições inerentes devido a natureza limitada dos dispositivos que formam esses ambientes. Algumas destas restrições também existem em ambientes que fazem uso de redes de sensores. Dessa forma, usando os requisitos de aplicações móveis apresentados anteriormente e assumindo que esses requisitos de forma genérica podem ser usados em redes de sensores devido a similaridade dos dispositivos envolvidos, uma plataforma de grade computacional para ambientes formados por dispositivos móveis e embarcados deve, basicamente, fornecer suporte a dinamicidade (suporte a ambientes dinâmicos), mobilidade (suporte a usuários e dispositivos nômades e móveis), sensibilidade ao contexto (significa que o dispositivo e a aplicação que está rodando no dispositivo são capazes de obter informações a respeito do ambiente onde estão operando e então são capazes de adap-

	Suporte a colaboração	Suporte a sensibilidade ao contexto	Suporte a alocação de recursos	Suporte a ambiente dinâmico	Suporte a execução em dispositivos móveis
Globus	✔	✖	✔	✖	✖
Gridbus	✔	✖	✔	✖	✔
Legion	✔	✖	✔	✖	✖
UNICORE	✔	✖	✔	✖	✖

Tabela 2.1: Comparação entre os principais middlewares segundo os critérios análisados por [NAV 06]

tar seu comportamento para tirar o máximo proveito desse ambiente), reconfiguração ou auto-adaptação dos dispositivos existentes (adaptação do dispositivo de forma autônoma - sem a necessidade de intervenção administrativa - visando adequar-se a alguma situação ou facilitar a agregação de novos recursos à grade), mecanismos para qualidade de serviço (proporcionar que as tarefas sejam executadas de forma eficiente) e também capacidade de comunicação de forma homogênea com sensores. Porém na análise dos middlewares efetuada acima não foram levados em consideração todos esses aspectos, de forma que acredita-se que a proposta não seria capaz de fornecer o devido suporte as necessidades apresentadas para o cenário-problema apresentado inicialmente. Como o foco do presente trabalho é o uso de grade computacional em ambientes móveis e também embarcados, abaixo é feita uma análise que aborda os requisitos deixados de lado mais a necessidade de suporte a dispositivos embarcados que não existia no estudo anterior. A análise têm o objetivo de expandir o estudo realizado anteriormente suprindo as carências do mesmo e também de abranger as novas necessidades do trabalho corrente.

➥ MoGRID [dSL 05] é um middleware de grade baseado no MoCa [SAC 04] e no In-

tegrade [GOL 04]. Seu foco é o suporte ao desenvolvimento de aplicações móveis sensíveis ao contexto. O trabalho propõe o uso de tecnologias ponto a ponto em grades móveis, tendo como elemento central do middleware o P2P Discovery Protocol (P2PDP). O P2PDP é o protocolo responsável pela coordenação de distribuição de tarefas na grade. Um fator importante, é que ele trabalha de forma a atenuar a sobrecarga de mensagens de controle que são trocadas entre os nodos otimizando assim o tráfego da rede. Segundo os resultados descritos no trabalho o MoGrid está em fases iniciais e seu trabalho futuro envolverá a implementação de proxies para permitir que dispositivos conectados a grade móvel possam trocar informações com os conectados a grades convencionais.

➥ AKOGRIMO [BEC 06] é um projeto Europeu proposto a ser a próxima geração de grade computacional baseada na Open Grid Services Architecture (OGSA). Seu foco é o desenvolvimento de aplicações para dispositivos móveis que fazem o uso do protocolo IP versão 6 (IPv6). A arquitetura é formada por quatro camadas: (1) Mobile Internet: camada formada pelos equipamentos que fazem o uso de IPv6. Têm por finalidade permitir que usuários de dispositivos móveis façam acesso as suas aplicações de uma forma simplificada. (2) Network Middleware: introduz os mecanismos para manipulação da camada de rede e também aspectos de autenticação, autorização e bilhetagem. (3) Grid Infrastructure Layer: é o núcleo dos serviços de descoberta e alocação, serviço de informação, gerênciamento de energia e notificação assíncrona. Para o desenvolvimento dessa camada será usado Open Grid Service Infrastructure (OGSI) e posteriormente Web Services Resource Framework (WSRF). (4) Generic Application Support Services Layer: é a camada mais externa da arquitetura. Têm por finalidade abstrair ao usuário o acesso aos recursos fornecidos pelas camadas mais baixas. O projeto Akogrimo têm concentração nas áreas de ensino a distância, telemedicina e manipulação de desastres. Ainda não foram apresentados resultados do projeto.

➥ SOGOS [BEC 06] é uma arquitetura especificamente voltada para dar suporte a auto-organização em ambientes de grade. A arquitetura proposta pelo trabalho é

capaz de trabalhar com ambientes dinâmicos através de informações semânticas (metadados) que descrevem as organizações envolvidas, papéis, direitos e capacidades dos agentes participantes e também da forma como eles interagem para solucionar o problema. São essas informações que coordenam todas as ações do sistema.

- Grid Mobile Service [PIO 07] faz parte do projeto GridLab [DAV 02]. É considerado como um pacote do GridLab, desta forma possui todas as suas características. O pacote têm por finalidade permitir que as aplicações rodando em dispositivos móveis possam acessar os recursos de uma grade fixa através de gateways. Uma das restrições desse projeto é que os dispositivos móveis são usados única e exclusivamente para consulta de dados e não para compartilhamento de recursos com outros dispositivos da grade.

- GRIDKit [COU 04] Possui uma arquitetura baseada em componentes sendo que seu funcionamento é orientado a serviços através de serviços web (web services). Para dar suporte as necessidades da grade são abordados quatro dominios: (i) Prestação de serviços: proporciona os mecanismos de comunicação, integração e gerenciamento de qualidade de serviços usando Simple Object Access Protocol (SOAP). (ii) Descoberta de recursos: fornece flexibilidade para o uso de diferentes tecnologias para descoberta de serviços como GRAM para descoberta de CPU e Peer-to-Peer (P2P) para descoberta de recursos em geral. (iii) Gerenciamento de recursos: fornece mecanismos necessários para ajustes dos recursos visando suporte a qualidade de serviço (iv) Segurança da grade: proporciona os mecanismos para que a comunicação entre os nodos ocorra de forma segura. O projeto merece um estudo mais aprofundado pois possui quase os mesmos requisitos do trabalho corrente, porém o seu foco não é na integração e gerenciamento de dispositivos através de grades e sim na integração da plataforma com outras redes que usam diferentes formas de interação como as redes overlay - rede overlay (overlay network) é uma rede de computadores construída sobre outra rede como por exemplo as redes que usam protocolos peer-to-peer como Gnutella, Freenet e I2P.

- Condor [GC 02] utiliza uma arquitetura baseada em camadas de forma que sua modularidade proporciona que vários componentes sejam reutilizados. O seu foco é a incorporação de dispositivos móveis como clientes para submissão de tarefas para grades de grande porte. Para submissão de tarefas e visualização dos dados é utilizado um navegador internet.

- MAG (Mobile Agents for Grid Computing Environments) [dSeS 06] explora a tecnologia de agentes móveis como uma forma de superar os desafios de construção de grades de computadores. O MAG executa aplicações carregando dinamicamente seus códigos nos agentes móveis. O agente do MAG pode ser realocado dinamicamente entre nós da grade atravéss de um mecanismo de migração transparente chamado MAG/Brakes, como uma forma de prover suporte a nós não dedicados. O MAG inclui mecanismos de tolerância a falhas de aplicações, de grade pervasiva e de grade de dados. O paradigma de agentes foi extensivamente utilizado para projetar e implementar os componentes do MAG formando uma infraestrutura multiagente para grades computacionais. A sua relação ao trabalho é mecanismos chamado PervMAG que trabalha como um proxy provendo suporte a mobilidade de forma a permitir que usuários nomâdes e usuários móveis consigam interagir com o MAG para solicitar serviços.

- Mobile OGSI.NET [CHU 04] é uma extensão do projeto OGSI.NET para dispositivos móveis. O trabalho aborda questões de limitações computacionais de processamento, força, armazenamento e intermitência na comunicação dos dispositivos móveis. O seu foco principal é a colaboração através dos dispositivos móveis. Para isso é proposta uma arquitetura composta de três camadas: um servidor web móvel utilizado para manipulação de requisições usando SOAP; um módulo que manipula as requisições e as encaminha para o serviço correto de serviços da grade; e os serviços da grade que são os responsáveis pelo processamento. A plataforma proposta ainda é restrita a dispositivos que usam o PocketPC como sistema operacional em conjunto com o .NET Compact Framework.

A tabela 2.2 sumariza a análise efetuada sobre middlewares móveis:

	Dinamicidade	Mobilidade	Sensibilidade ao contexto	Execução em disp. móveis	Reconfiguração	Qualidade de serviço	Dispositivos embarcados
MoGRID	✔	✔	✔	✔	✔	✖	✖
AKOGRIMO	✔	✔	✔	✔	✔	✔	✖
SOGOS	✔	✔	✖	✖	✔	✖	✖
Grid Mobile Service	✔	✔	✖	✔	✔	✔	✖
GRIDKit	✔	✔	✔	✔	✔	✔	✖
Condor	✔	✔	✖	✖	✖	✖	✖
MAG	✔	✔	✖	✖	✖	✖	✖
Mobile OGSI.NET	✔	✔	✖	✔	✔	✖	✖

Tabela 2.2: Análise de middlewares móveis

Com a análise dos trabalhos acima conclui-se que eles não atendem a todos os requisitos necessários para o uso de grade computacional em ambientes formados por dispositivos móveis e embarcados. Sendo assim, verifica-se a possibilidade de contribuição na área de grade computacional com uma plataforma que seja capaz de atender aos requisitos.

No próximo capítulo é efetuada a análise de um cenário e então a partir dela são apresentadas informações que acabam por definir a proposta da plataforma capaz de suprir a lacuna verificada acima.

Capítulo 3

Proposta

Uma plataforma de grade computacional capaz de lidar com ambientes móveis e embarcados deve, basicamente, fornecer suporte a dinamicidade, mobilidade, sensibilidade ao contexto, reconfiguração ou auto-adaptação dos dispositivos existentes, suporte a qualidade de serviço e também uma interface que possibilite a comunicação de forma homogênea com sensores. No capítulo anterior destacou-se a inexistência de uma plataforma que consiga atender a esses requisitos. Essa falta leva os desenvolvedores a aderir soluções "ad hoc", as custas de pouca (ou nenhuma) re-usabilidade.

Este capítulo responde a segunda sub pergunta levantada na seção 1.1: *Quais as características e os módulos de software necessários em uma plataforma para suportar o elementos e interações de um ambiente formado por dispositivos móveis e embarcados?*.

Para isso, o capítulo faz um levantamento de um conjunto de informações que definem uma arquitetura a ser usada, a qual será utilizada para a implementação da plataforma que será capaz de suprir a falta identificada.

A estrutura do capítulo é a seguinte: a seção 3.1, apresenta um cenário-problema, fazendo um análise de quais os elementos e interações existentes no cenário. A seção 3.2 aborda quais são os requisitos necessários em uma arquitetura que estenda o uso de grades computacionais para esse tipo de ambiente. Esses requisitos levam a definição dos componentes necessários à arquitetura, abordados na seção 3.3. A seção 3.4 descreve

as camadas da arquitetura definidas para estruturar a disposição dos componentes. A seção 3.5 aborda quais são as estruturas de dados que precisam ser criadas para atender as necessidades da arquitetura. Os métodos existentes são descritos na seção 3.6 e servem para definir o comportamento da arquitetura. Por fim, o capítulo é finalizado na seção 3.7 que conclui como a plataforma implementada a partir das definições do capítulo pode ser aplicada ao cenário problema, suprindo assim a falta identificada na seção 2.2 de uma plataforma de grade computacional que consiga lidar com as características existentes em ambientes móveis e embarcados.

3.1 Análise do cenário problema

A presente seção efetua uma análise empírica de um cenário-problema que faz o uso da telemedicina para o monitoramento remoto de pacientes. Esse cenário é um exemplo típico de uso de redes de sensores em conjunto com dispositivos móveis e serve como base de análise para a identificação de quais são os elementos existentes no ambiente e também de quais são as interações que acontecem entre esses elementos. A partir dessa análise podem ser identificadas as características do mesmo. Uma plataforma de grade computacional para ambientes móveis e embarcados deve ser capaz de lidar com essas características.

Para analise inicial, a figura 3.1 apresenta os seguintes elementos existentes no cenário:

(i) Pacientes são aqueles que estão internados dentro do ambiente hospitalar e possuem conectados ao seu corpo (ii) sensores responsáveis pela coleta de dados vitais. Os sensores podem ser equipamentos fixos ao lado do leito do Paciente ou então móveis. Tanto os sensores fixos quanto os móveis transmitem os dados usando sinal de rádio através de (iii) Pontos de acesso espalhados no interior do hospital. O (iv) Ambiente do paciente é formado pelo conjunto de (i) e (ii) e fornece condições para que a coleta e transmissão de dados ocorram com sucesso para o (v) Sistema Interno do hospital. O Sistema Interno é responsável pelo gerenciamento dos dispositivos do ambiente, pela coleta, análise, armazenamento e disponibilização dos dados coletados para o (vi) dispositivo

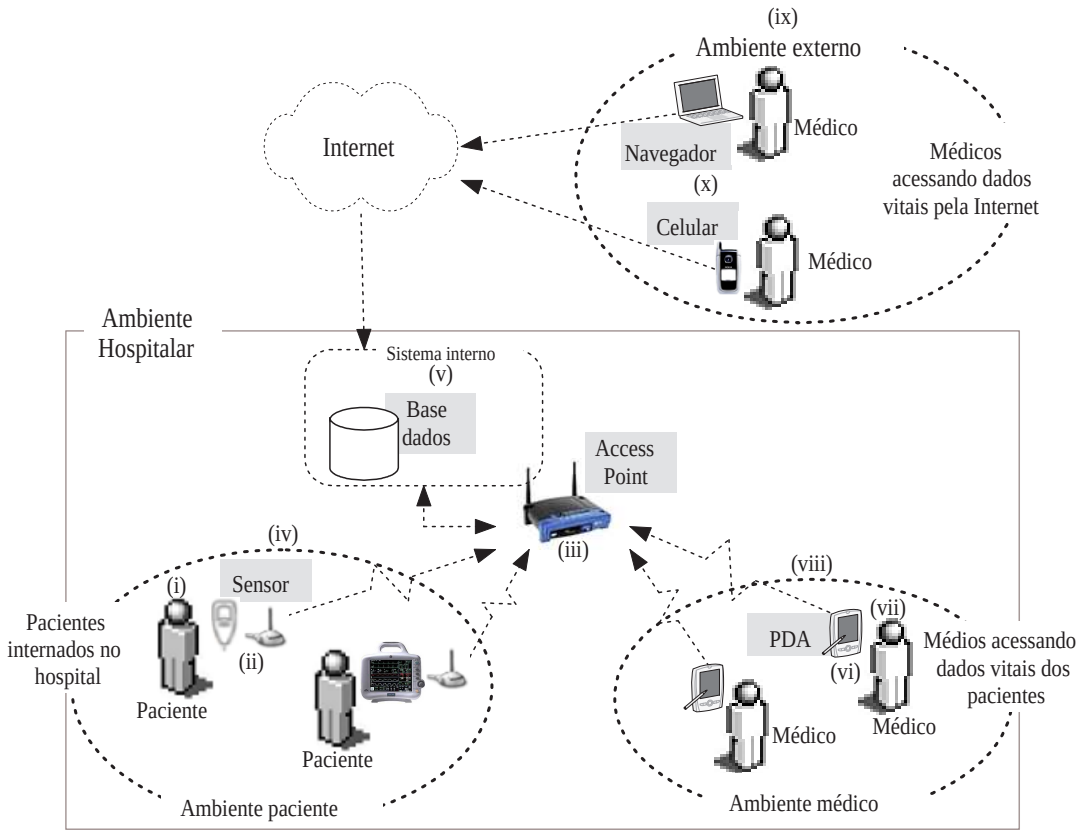


Figura 3.1: Elementos existentes no cenário-problema

móvel do (vii) Médico. O Médico através de seu PDA pode consultar os dados vitais do paciente em tempo real. O (viii) Ambiente do médico é formado pelo conjunto de (vi) e (vii) e fornece condições para que o médico consulte os dados vitais e interaja com o paciente. Além de consultar os dados no interior do hospital, médicos localizados em um (ix) Ambiente Externo também são capazes de acessar os dados dos pacientes usando (x) celulares, navegadores ou algum outro dispositivo que seja capaz de efetuar conexão à Internet.

Os elementos fornecem a visão estática do cenário. Para que cada elemento possa executar o seu papel, dando dinamismo ao cenário, são necessárias que aconteçam as interações representadas na figura 3.2. As interações precisam ser abordadas pois são elas que demonstram como os elementos se relacionam entre si, determinando a forma como o ambiente se comporta e conseqüentemente a forma como a

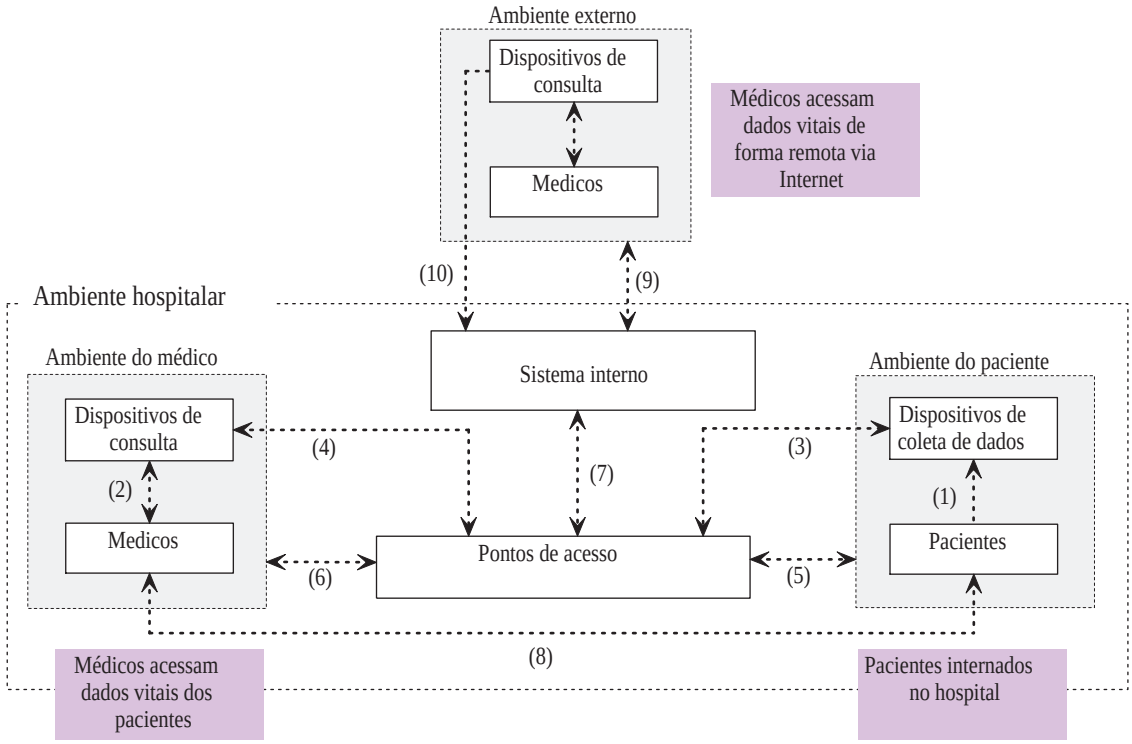


Figura 3.2: Interações existentes no cenário-problema

plataforma deverá se comportar.

1. Interação entre o sensor e o coletor de dados: é a interação que existe no momento em que o sensor troca dados com o dispositivo responsável por transmitir os sinais coletados de uma forma inteligível para a plataforma. Essa interação torna necessária a existência de uma padronização dos dados coletados para posterior processamento e também algum mecanismo que permita a comunicação de forma homogênea com os diferentes tipos de sensores existentes.

2. Interação entre o médico e o dispositivo móvel: aborda a entrada de dados do médico no equipamento móvel (como por exemplo, a solicitação de sinais vitais de um determinado paciente) e a resposta da plataforma para o médico (resposta da solicitação efetuada). Essa interação existe para que o usuário consiga fazer requisições a plataforma e receber as respostas geradas por ela de uma forma transparente não tendo conhecimento da forma interna de seu funcionamento.

3. Interação entre o coletor e a plataforma: no momento em que o coletor obtém os dados do equipamento de monitoramento deve fazer com que esses dados sejam enviados para a plataforma de uma forma confiável e segura. Essa interação também torna necessária a padronização no gerenciamento e na troca de informações entre os dispositivos existentes no ambiente uma vez que podem existir diferentes tipos, marcas e modelos e cada um deles pode usar um padrão diferente na saída dos dados coletados.

4. Interação entre o dispositivo móvel e plataforma: a interação que existe quando o dispositivo móvel do médico acessa as informações dos pacientes. Como trata-se de um dispositivo móvel essa interação pode acontecer em diferentes pontos do ambiente. A plataforma deve reconhecer de onde a requisição partiu e então garantir que a informação solicitada chegará ao seu destino de forma correta.

5. Interação entre o ambiente do paciente e o ponto de acesso: é a interação que efetua o controle dos dispositivos ligados ao local onde o paciente se encontra e que proporciona a comunicação com os demais dispositivos existentes no ambiente. Essa interação envolve mecanismos capazes de lidar com a entrada de um sensor no ambiente, conexão com a plataforma, autenticação, verificação do funcionamento, ajustes e desconexão quando necessário. Esses mecanismos permitem o gerenciamento do ambiente de forma homogênea.

6. Interação entre o ambiente do médico e o ponto de acesso: é a interação existente para que o médico e seu dispositivo móvel possuam as condições necessárias para efetuar a consulta dos dados fornecidos pela plataforma. Assim como a interação anterior, também deve prover mecanismos capazes de lidar com a entrada de um novo médico e seu equipamento móvel no ambiente, autenticação, verificação do funcionamento, ajustes e desconexão quando necessário. Os mecanismos da interação anterior que permitem o gerenciamento de forma homogênea do ambiente podem ser reaproveitados.

7. Interação entre o ponto de acesso e o sistema interno: são as interações existen-

tes para que os pontos de acesso espalhados no interior do hospital sejam capazes de manipular as mensagens trocadas entre os médicos, paciente e o sistema interno. Essa interação necessita de condições para localização de onde estão vindo as requisições e para onde devem ser enviadas as respostas.

8. Interação entre o paciente e o médico: é a interação que ocorre indiretamente no momento em que o médico solicita a visualização de dados vitais de um paciente ou quando o paciente necessita de atenção de um médico devido a uma anormalidade dos seus sinais vitais.

9. Interação entre o ambiente externo e o Sistema Interno: é a interação que acontece quando médicos localizados em ambiente externo ao hospital acessam os dados dos pacientes. Essa interação necessita que os dados coletados sejam entregues de forma correta e segura ao ambiente externo, sendo necessário para isso mecanismos de segurança como autenticação e criptografia.

10. Interação entre os dispositivos externos e a plataforma via internet: é a interação que acontece entre o dispositivo que o médico está usando e a plataforma. A fim de garantir a homogeneidade no gerenciamento são necessários mecanismos capazes de lidar com os dispositivos externos como se eles estivessem localizados no interior do hospital.

A análise dos elementos e suas interações proporcionam uma melhor compreensão do cenário. Pode-se perceber que existe grande diversidade de dispositivos envolvidos na coleta, análise, armazenamento e posterior disponibilização dos dados coletados para usuários dentro e fora do ambiente hospitalar. As trocas de informações entre os dispositivos podem acontecer em diferentes locais devido a mobilidade e dinamismo existente no ambiente. Os sensores utilizados podem ser fixos ou móveis e serem de diferentes modelos ou com diferentes finalidades. Dessa forma, as aplicações devem ser capazes de rodar em dispositivos móveis, devem ser capazes de se adaptarem ao local onde estão sendo executadas, devem possuir algum mecanismo que garanta qualidade de

serviço nas tarefas executadas e também proporcionarem mecanismos capazes de efetuar a comunicação com os sensores.

Uma plataforma de grade computacional proporciona de forma intrínseca os mecanismos necessários para o gerenciamento de forma homogênea do ambiente e também para auxiliar no desenvolvimento de aplicações para esse ambiente. Porém, para ser capaz de lidar com as características descritas acima não basta que sejam proporcionados os mecanismos de segurança (autenticação, segurança de dados), de controle de recursos (registro e busca de serviços), de escalonamento de tarefas (distribuição) e de comunicação de dados (envio/recebimento de pacotes, endereçamento) [GRI 03] presentes em plataformas de grade computacional "convencionais". São necessários que sejam atendidos mais alguns requisitos que estendam as funcionalidades da grade de forma a possibilitar que ela saiba lidar com as características de ambientes móveis e também de redes de sensores. Esses requisitos serão abordados na próxima seção.

3.2 Requisitos da arquitetura

Como visto anteriormente as plataformas de grade "convencionais" não conseguem lidar com todas as características existentes em um ambiente formado por dispositivo móveis e embarcados. Usando como base o cenário-problema apresentado acima, foram identificadas alguns requisitos necessários para a arquitetura ser capaz de lidar com os elementos e interações existentes. Esses requisitos estão listados abaixo:

a) uso de protocolos e padrões abertos: protocolos e padrões abertos e de propósito geral, possibilitam maior facilidade na integração dos dispositivos móveis e embarcados juntamente com o desenvolvimento de aplicações. Dessa forma os dados trocados dentro da arquitetura seguem o mesmo padrão atendendo assim as necessidades para que ocorra a interação 3.

b) coordenação de recursos existentes no ambiente: no contexto desse trabalho são os dispositivos móveis e embarcados que serão integrados à grade computacional. Para atender as interações 5 e 6, o middleware deve proporcionar mecanismos de

comunicação de dados, autenticação, verificação de conexão e desconexão de dispositivos de forma que o gerenciamento dos recursos existentes no ambiente seja feito de forma homogênea.

c) flexibilidade para comunicação com diferentes dispositivos: devido a diversidade de dispositivos que podem existir em um ambiente, deve haver mecanismos que garantam a padronização dos dados coletados e tornem a interface de comunicação com esses dispositivos facilmente adaptável. Essa flexibilidade e padronização atende as necessidades da interação 1.

d) mecanismos para proporcionar qualidade de serviço: o middleware deve ser capaz de considerar aspectos como latência, disponibilidade e carga de processamento para assegurar que os serviços sejam executados de forma eficiente. Esse é um requisito necessário pelas interações 3 e 4.

e) mecanismos para gerenciar ambientes dinâmicos: um ambiente formado por dispositivos móveis e embarcados possui problemas intrínsecos de intermitência de comunicação, latência e perda de dados. A topologia da rede pode sofrer constantes alterações devido a mobilidade dos dispositivos o que ocasiona constantes desconexões (por perda de comunicação ou falta de força) e conexões (de novos dipositivos ou daqueles que foram desconectados previamente). Devem haver mecanismos que sejam capazes de gerenciar a mobilidade e o dinamismo existente no ambiente, atendendo assim as necessidades da interação 4.

f) mecanismos de descoberta e reconfiguração: devido ao dinamismo existente no ambiente, deve haver mecanismos que permitam descobrir qual dispositivo que presta um serviço solicitado por outro em determinado momento. Esses mecanismos devem também ser capazes de lidar com as constantes mudanças de topologia, permitindo a reconfiguração ou auto-organização dos dispositivos de forma autônoma. Esses mecanismos atendem as necessidades da interação 7.

g) transparência no acesso aos resultados das tarefas: devem haver mecanismos que permitam que os resultados da execução das tarefas sejam visualizados de forma correta

e transparante em qualquer dispositivo. Ou seja, independente do dispositivo em uso, seja um Personal Digital Assistant (PDA), celular ou navegador internet o resultado apresentado será formatado automaticamente de acordo com ele. Isso atende a necessidade da interação 2.

Os requisitos apresentados foram definidos tomando como base as interações do ambiente, de forma que cada um deles atenda a uma interação identificada na seção anterior. Uma vez que as interações representam a forma de comportamento do ambiente, uma arquitetura de grade computacional para ambientes móveis e embarcados precisa implementar esses requisitos.

Portanto, se a arquitetura for capaz de implementa-los, conseqüentemente, ela será capaz de lidar com as interações e assim estaria estendendo-se a aplicação das grades computacionais para ambientes móveis e embarcados, ou seja, conseguiria-se aplicar o paradigma de grade computacional no cenário-problema inicial.

3.3 Componentes da arquitetura

Os requisitos apresentaram quais as funcionalidades que a arquitetura deve possuir.A seguir são definidos quais os componentes (módulos de software) que devem estar presentes na arquitetura. Essa definição visa diminuir o nível de abstração e tornar possível a implementação dos requisitos.

De acordo com a figura 3.3 têm-se os seguintes componentes e suas respectivas funções:

- Interface sensor: responsável pela troca de informações entre a arquitetura e o sensor. Esse componente possui a finalidade de assegurar a flexibilidade na implementação da comunicação com os diferentes tipos de sensores que podem existir no ambiente. Esse componente implementa o requisito (c) flexibilidade para comunicação com diferentes dispositivos

- Interface usuário: disponibiliza mecanismos para que o usuário consiga interagir com a arquitetura. Esse componente possui a finalidade de garantir que qualquer

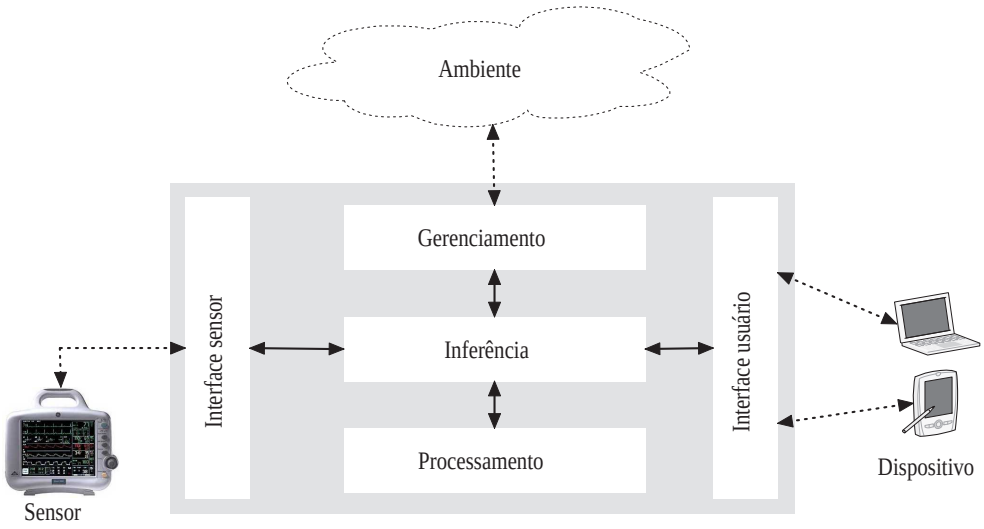


Figura 3.3: Componentes da arquitetura

dispositivo suportado pela arquitetura será capaz de enviar tarefas para a arquitetura e conseguirá receber os resultados obtidos pela execução desta. Implementa o requisito (g) transparência no acesso aos resultados das tarefas.

- Gerenciamento: responsável pela coordenação e descoberta de recursos, comunicação entre os dispositivos existentes, assegurar qualidade de serviço no processamento de tarefas e gerenciamento do dinamismo existente no ambiente além de efetuar os controles internos necessários para garantir o correto funcionamento da arquitetura. Esse componente implementa os requisitos (b) coordenação de recursos existentes no ambiente; (d) mecanismos para proporcionar qualidade de serviço; (e) mecanismos para gerenciar ambiente dinâmicos; e (f) mecanismos de descoberta e reconfiguração

- Inferência: executa a inferência e a conversão dos dados para o padrão utilizado pela arquitetura. Para permitir a troca de informações de forma estruturada, atendendo assim ao requisito de padronização dos dados que são transmitidos e recebidos na arquitetura, esse componente faz a codificação e decodificação dos dados usando padrões abertos e reconhecidos por todas as partes da arquitetura. Esse componente atende ao requisito (a) uso de protocolos e padrões abertos

- Processamento: As tarefas dentro da arquitetura são definidas como estruturas de dados que possuem instruções que quando executadas geram um resultado. Esse componente é o responsável pela execução das instruções que formam as tarefas. Pelo fato da arquitetura usar tarefas para a execução de ações, esse módulo contempla de forma indireta todos os requisitos.

A definição dos componentes apresenta como a arquitetura deve ser organizada e demonstra como os requisitos podem ser implementados. Além disso, também auxilia no processo de implementação, uma vez que o desenvolvedor concentra-se na codificação das funcionalidades que cada componente em específico deve possuir. A idéia é deixar cada um dos componentes com responsabilidades bem definidas tornando a arquitetura modular, com baixo acoplamento (acoplamento mede o grau de dependência entre componentes, quais são os impactos causados quando acontece a mudança na implementação de um componente em relação aos demais) e alta coesão (coesão é o nível de integridade interna de uma classe, classes com alta coesão têm responsabilidades bem definidas e são difíceis de dividir em duas ou mais classes;).

3.4 Camadas da arquitetura

Em uma arquitetura camadas são separações lógicas que visam tornar mais simples a execução de tarefas complexas, dividindo-as em pequenas sub-tarefas. No contexto desse trabalho, além de usar camadas para delimitação de sub-tarefas, a arquitetura as usa para auxiliar na separação de responsabilidades, servindo como mecanismo para alcançar o baixo acoplamento e alta coesão dos componentes.

A figura 3.4 demonstra a visão em camadas da arquitetura sendo que a descrição de cada uma é apresentada a seguir:

- **Sensor:** é a camada mais baixa, interagindo diretamente com os dispositivos embarcados (sensores) existentes no ambiente. A sua existência visa a flexibilidade para incorporação de novos tipos de dispositivos. No caso de um ambiente hospitalar existe equipamentos de diferentes marcas, modelos, fabricantes e com diferentes

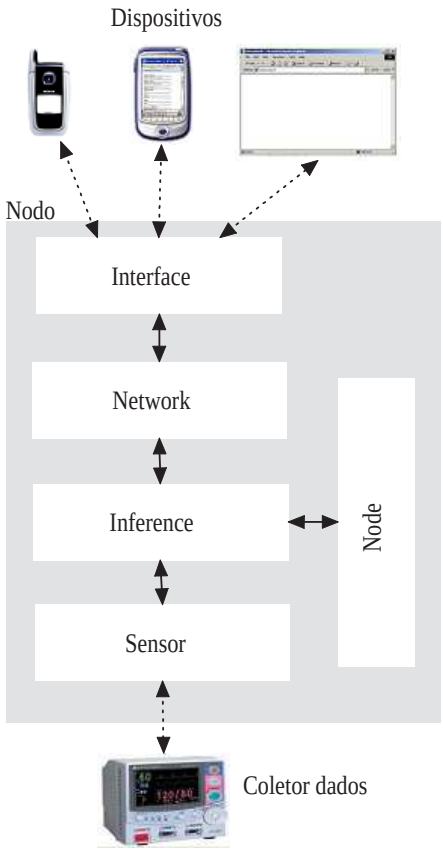


Figura 3.4: Camadas que formam a arquitetura

finalidades os quais não seguem um padrão comum de exteriorização dos dados coletados. Nesse caso, basta adaptar essa camada de forma que ela consiga obter os dados do equipamento em questão e enviá-los no padrão requerido pela camada superior. Essa camada possui métodos necessários para coleta automatizada dos dados capturados pelos dispositivos.

- **Inference:** é a camada que possui os métodos responsáveis pela inferência e também pela conversão dos dados no formato utilizado pela arquitetura.

- **Node:** uma vez que em um ambiente de grade computacional todos os dispositivos são vistos como nodos essa camada é considerada o "coração" da arquitetura. Possui métodos para gerenciamento das funções internas do nodo, métodos de descoberta e autoconfiguração e também aqueles necessários para assegurar qualidade

de serviço. Além disso, possui métodos relacionados a criação, escalonamento, coordenação e execução de tarefas, bem como os necessários para a obtenção de resultados da execução destas tarefas.

- **Network:** é a camada responsável pela comunicação entre os nodos. Ela implementa os métodos relacionados ao nível de rede da arquitetura, como por exemplo, roteamento e manipulação do protocolo de comunicação. Para garantir transparência e organização dos dados que são transportados entre os nodos da arquitetura, toda a comunicação ocorre através de um protocolo pervasivo. O protocolo escolhido é o HTTP devido a ser considerado padrão de fato, tendo sua implementação possível em diversos tipos de dispositivos.

- **Interface:** é a camada acessada diretamente pelo usuário. Possui métodos que proporcionam a interação do usuário com a arquitetura, recebendo tarefas para serem executadas pelas camadas mais baixas e também formatando os resultados que serão apresentados de acordo com o dispositivo utilizado.

A visão em camadas possibilita ao desenvolvedor uma melhor compreensão dos métodos que necessitam serem implementados para que cada camada desempenhe a sua função. Na arquitetura, as funções desempenhadas por uma camada servem como base para a camada imediatamente superior ou inferior. Isso garante maior flexibilidade, simplicidade e rapidez na implementação pois estando as funções bem definidas quaisquer alterações feitas em uma camada não afetarão o funcionamento de outra.

3.5 Estruturas de dados da arquitetura

Para implementar a arquitetura as estruturas de dados nativas proporcionadas pelas linguagens de programação não são suficientes. Os métodos, que serão apresentados logo a seguir, necessitam de estruturas de dados específicas para a arquitetura. Essa seção demonstra quais são as estruturas de dados que precisam ser definidas:

- XMLTree: encapsula os dados em uma estrutura XML simples. É a estrutura utilizada como base para a codificação de todas as informações que trafegam na arquitetura. A figura 3.5 representa de forma gráfica a estrutura de uma XMLTree.

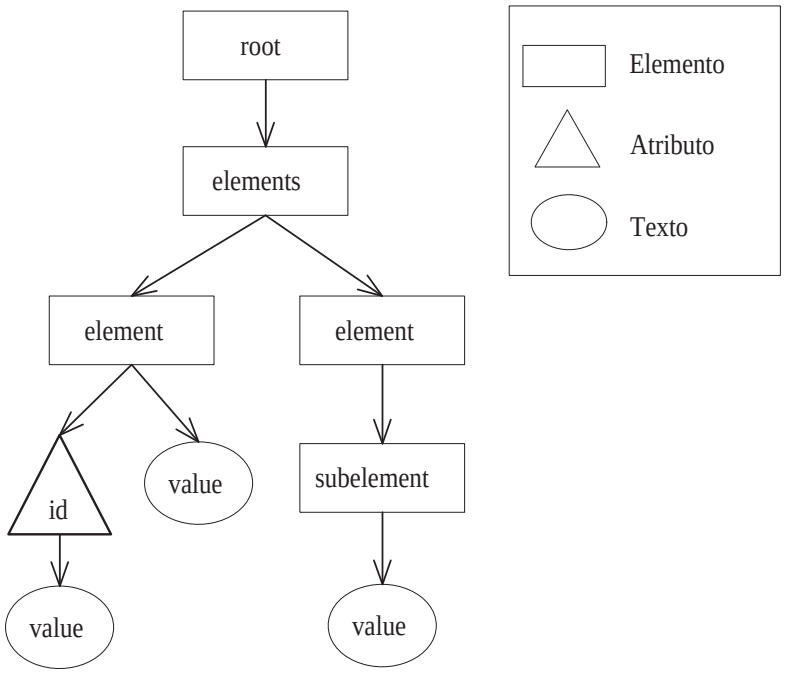


Figura 3.5: Representação da estrutura XMLTree

Os campos existentes nessa estrutura e representados na figura seguem o mesmo padrão de estruturação do XML. A finalidade de criação da estrutura XMLTree é de tornar mais simples a manipulação do XML com métodos específicos para a arquitetura.

- Tarefa: define a tarefa a ser executada. Contém o nome do serviço a ser invocado, o remetente e o destinatário, identificação da tarefa, tempo em que ela foi criada, sua prioridade e também os parâmetros necessários para a sua execução. A Tarefa é codificada usando XMLTree.

- ResultadoTarefa: contém as informações relativas ao resultado da execução de uma tarefa, como por exemplo identificação da tarefa original, código de resultado, quem originou o resultado, tempo em que o resultado foi gerado, destinatário, prioridade e

também o resultado da execução. Assim como a Tarefa o ResultadoTarefa também é codificado usando XMLTree.

- Node: é a abstração de um nodo da grade.
- Sensor: abstração de um sensor que pode ser conectado a um node.
- Gancho: abstração dos mecanismos de gancho proporcionados pela arquitetura. Os ganchos são estruturas usadas para proporcionar a flexibilidade da arquitetura para implementação de novas funcionalidades.

Como a arquitetura trabalha com dados e dispositivos diversificados, tornou-se necessário criar estruturas de dados capazes de representa-los. Essa representação não seria possível unicamente através de tipos nativos como inteiros, strings e floats e outros disponíveis pelas linguagens de programação. Os tipos definidos acima são usados pelos métodos da arquitetura os quais serão apresentados na próxima seção.

3.6 Métodos da arquitetura

Para ser capaz de desempenhar suas funções a arquitetura necessita que métodos sejam definidos. A fim de melhor organiza-los, eles estão agrupados em grupos, segundo as suas finalidades. A visão conceitual da organizacão dos grupos de métodos é representada na figura 3.6.

- **Scheduler:** grupo de métodos responsável pelo agendamento de tarefas que são executadas em intervalos de tempo.
- **Application Programing Interface:** contém os métodos disponíveis para o desenvolvimento de aplicações.
- **Internal Management Methods:** são os métodos responsáveis pelo gerenciamento do nodo como roteamento de mensagens, gerenciamento de logs de eventos, gerenciamento de traps, além de mecanismos de “heart beat” (usados para aviso que o nodo “está vivo”) e de monitoramento de dados de perfil do dispositivo.

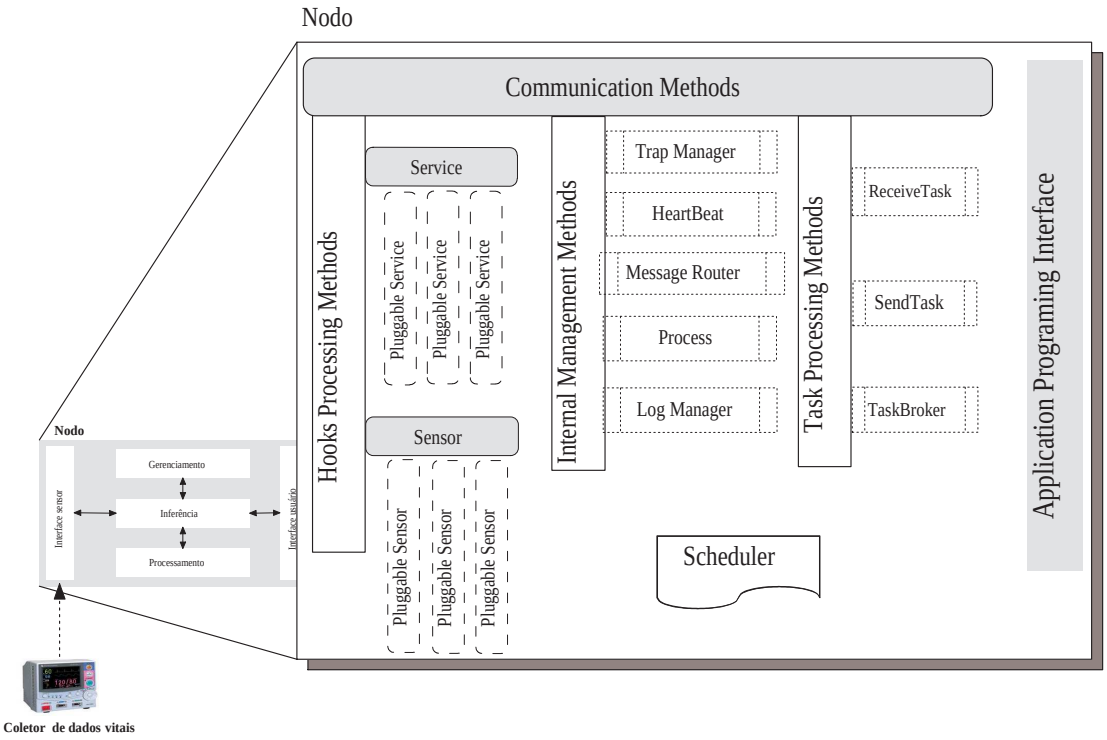


Figura 3.6: Visão conceitual da organização dos grupos de métodos de um nodo

- **Task Processing Methods:** contém os métodos usados para criação, envio, recepção e processamento de tarefas.

- **Communication Methods:** formado pelos métodos usados para comunicação e troca de informações entre os nodos.

- **Hooks Processing Methods:** composto de métodos utilizado para o processamento dos "ganchos" existentes. Existe a necessidade de dois tipos de ganchos: Service (gancho responsável pela criação e execução de novos serviços) e Sensor (responsável pela coleta de dados dos sensores e também para permitir a flexibilidade na implementação da comunicação com os diferentes tipos de sensores).

A visão conceitual em grupo de métodos auxilia na melhor compreensão daquilo que cada grupo deverá fazer. A descrição abstrata de como os métodos podem ser implementados está descrita em detalhes no anexo A. Os métodos possuem a

finalidade de descrever como a arquitetura deve se comportar para atender aos requisitos levantados pela análise do cenário-problema.

O conjunto das informações apresentadas até o momento fornece os subsídios necessários para a construção de uma plataforma de grade computacional que é capaz de desempenhar funções de grades de sensores e também de grades móveis, ou seja, dessa forma ela é capaz de lidar com ambientes formados por dispositivos móveis e embarcados.

3.7 Conclusão do capítulo

Esse capítulo respondeu a segunda sub pergunta levantada na seção 1.1: *Quais as características e os módulos de software necessários em uma plataforma para suportar o elementos e interações de um ambiente formado por dispositivos móveis e embarcados?*.

Para responder a pergunta, partindo do cenário-problema foram analisados os componentes e as interações existentes. Pela análise, constatou-se que devido ao dinamismo do ambiente, mobilidade dos dispositivos e emprego de sensores com diferentes finalidades as aplicações devem ser capazes de rodar em dispositivos móveis, de se adaptarem ao local onde estão sendo executadas, possuírem algum mecanismo capaz de garantir qualidade de serviço e também mecanismos que permitam a comunicação com sensores de uma forma homogênea. Como uma plataforma de grade computacional "convencional" com o Globus ou o Legion não consegue lidar com essas características, verificou-se a oportunidade de proposta de uma plataforma capaz de suportalas, formando um misto de características presentes em grades de sensores com grades móveis. Dessa forma conseguiria-se estender o uso de grades computacionais para ambientes móveis e embarcados.

A fim de definir a arquitetura a ser usada como base para a implementação da plataforma, foi apresentado um conjunto de requisitos que precisam ser atendidos para a arquitetura ser capaz de lidar com os elementos e interações. Para proporcionar a implementação dos requisitos foram definidos componentes (módulos de software) que

devem estar presentes na arquitetura. Cada componente foi concebido com funções bem definidas tornando a arquitetura completamente modular. Conceitualmente a arquitetura foi organizada em camadas. O uso de camadas visa auxiliar na separação de responsabilidades, servindo como mecanismo para alcançar o baixo acoplamento e alta coesão dos componentes.

Para ser capaz de desempenhar suas funções foram apresentados abstrações de métodos que definem a forma como a arquitetura deverá comporta-se. Porém, para que os métodos pudessem ser definidos houve a necessidade de criação de algumas estruturas de dados específicas para arquitetura, uma vez que os tipos de dados nativos das linguagens de programação não conseguem representar os dados complexos usados pelos métodos.

O conjunto das informações da arquitetura apresentadas no decorrer do capítulo fornece os subsídios necessários para a implementação da plataforma. Essa plataforma será capaz de atender a todos os requisitos necessários para lidar com as interações e elementos do ambiente estendendo assim o uso de grades computacionais para ambiente móveis e embarcados. De forma resumida, os componentes, camadas e métodos definidos pela arquitetura são capazes de atender aos requisitos, ou seja, com o uso da proposta consegue-se aplicar o paradigma de grade computacional no cenário-problema inicial, solucionando dessa forma o problema. A figura 3.7 ilustra o caminho percorrido até chegar as definições que solucionam o problema.

O desenvolvimento do capítulo apresentado contribui de duas formas: primeiro, ele adiciona ao estado-da-arte, completando uma deficiência na literatura, apresentada na seção 2.2, de uma plataforma de grade computacional para ambientes formados por dispositivos móveis e embarcados. A segunda contribuição é a enumeração de forma clara do ciclo analítico necessário no desenvolvimento de um sistema de computação em grade capaz de fornecer o suporte as características existentes no cenário-problema.

Para demonstrar a funcionalidade da proposta o próximo capítulo apresenta a aplicação dos conceitos usados na implementação da plataforma. São apresentados alguns resultados quantitativos obtidos em simulações que buscam demonstrar o seu comportamento em diferentes situações de uso. Por fim é apresentado o desenvolvimento

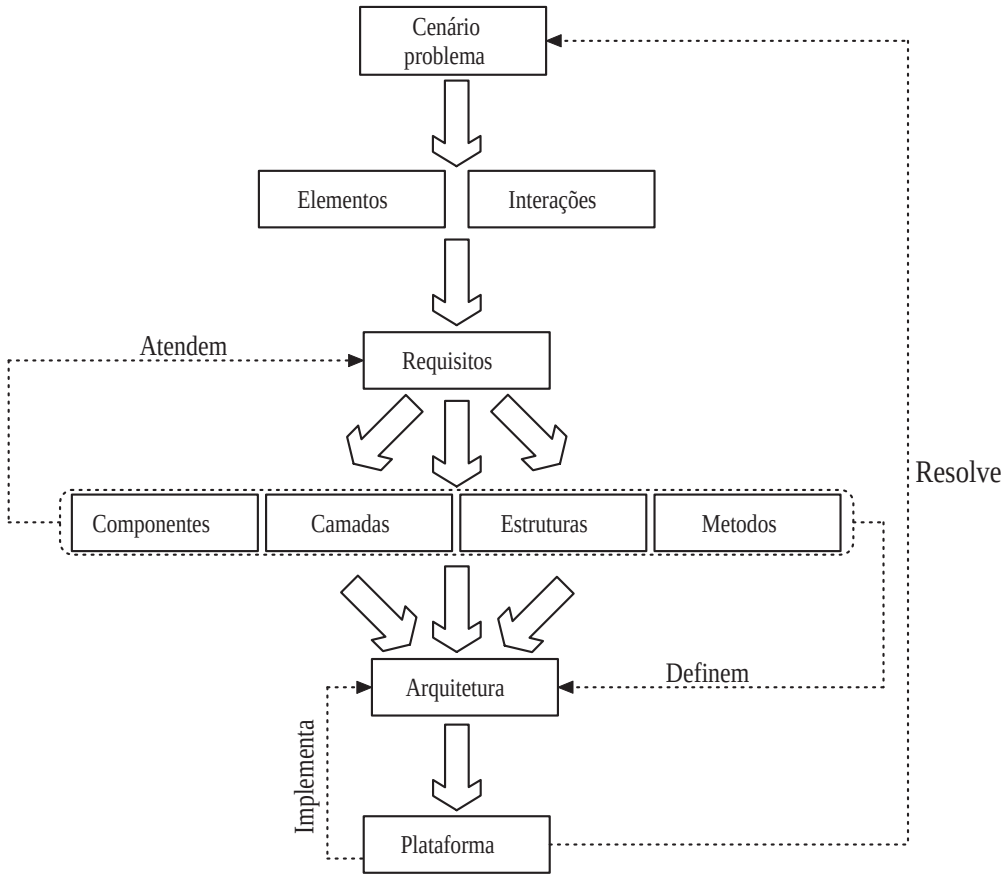


Figura 3.7: Caminho analítico percorrido para definir a plataforma

de um estudo de caso completo na área de telemedicina onde sensores e dispositivos móveis são usados para monitoramento remoto de pacientes. Esse estudo de caso servirá para validação da proposta e estudo de suas limitações.

Capítulo 4

Resultados

O capítulo anterior efetuou a análise de um cenário-problema e então propôs uma arquitetura capaz de atender aos requisitos necessários para estender o uso de grade computacional a esse cenário. Este capítulo, além de demonstrar como a arquitetura proposta é implementada, responde a terceira sub pergunta efetuada na seção 1.1: *Como se comporta uma plataforma de grade computacional aplicada na integração de um ambiente formado por dispositivos móveis e embarcados ?*.

Para responder a pergunta, o capítulo está estruturado da seguinte forma: a seção 4.1 demonstra como a plataforma foi implementada a partir das definições da arquitetura efetuadas no capítulo anterior. A interface de programação de aplicativos da plataforma é apresentada na seção 4.2. Na seção 4.3 são apresentados os resultados de testes quantitativos realizados com a plataforma. Para demonstrar de forma prática o uso de plataforma a seção 4.4 apresenta um estudo de caso na área da telemedicina. Finalmente o capítulo é finalizado na seção 4.5

4.1 Apresentação da plataforma

Uma carência levantada anteriormente em ambientes de grade "convencionais" é que eles não são capazes de lidar com todas as características presentes em ambientes móveis e embarcados. A plataforma apresentada nesta seção foi obtida a partir

da implementação da arquitetura proposta. Ela visa tornar mais simples a integração e construção de aplicações para dispositivos móveis e embarcados usando grade computacional. No contexto do trabalho, a computação em grade é o conceito utilizado para a integração e gerenciamento dos dispositivos no ambiente.

Implementando a arquitetura a plataforma é capaz de fornecer suporte a dinamicidade, mobilidade, reconfiguração ou auto-adaptação dos dispositivos existentes no ambiente, suporte a qualidade de serviço e também comunicação de forma homogênea com dispositivos embarcados, que são os requisitos apresentados na seção 3.2.

Para lidar com a dinamicidade, mobilidade e também com a reconfiguração ou auto-organização dos dispositivos existem mecanismos internos da plataforma que fazem com que os demais nodos da grade consigam assimilar a mudança de topologia ocasionada pela movimentação, conexão ou desconexão de um dispositivo. Os mecanismos asseguram que quando um desses eventos acontecer sejam disparados métodos que ajustam as tabelas de roteamento e de disponibilização de serviços prestados pela grade para refletir a nova localização do nodo. Isso assegura que no momento em que uma aplicação fizer a requisição por determinado serviço ela não necessita saber qual o nodo que presta o serviço nem mesmo onde ele se encontra. Basta requisitar o serviço e a plataforma assegura que o mesmo será encontrado e executado caso estiver disponível. Além disso, esses mecanismos proporcionam facilidades para a agregação de novos recursos à grade.

O suporte a qualidade de serviços é proporcionado com o uso de atributos dinâmicos e de perfil os quais são índices atualizados em tempo de execução que retratam o comportamento de cada nodo da rede. O desenvolvedor, com base nesses dados, pode então criar diferentes mecanismos para garantir a qualidade esperada na execução das tarefas.

A comunicação de forma homogênea com dispositivos embarcados é fornecida através do uso de métodos que fornecessem flexibilidade para a implementação de interfaces específicas para cada tipo de dispositivo sem necessitar a alteração na estrutura interna da plataforma. Isso é de grande valia para os desenvolvedores uma vez que basta implementar a interface e “pluga-la” a plataforma facilitando o processo de desenvolvimento.

Na concepção da plataforma, tentou-se não sacrificar a flexibilidade, escalabilidade, confiabilidade e extensabilidade. Outro ponto considerado, foi a independência de plataforma, isso significa que ela é capaz de ser executada tanto em pequenos dispositivos portáteis quanto em servidores de grande porte.

As próximas seções descrevem com maior profundidade como a plataforma implementa a arquitetura e também quais são as suas funcionalidades.

4.1.1 Orientação a serviço

A arquitetura definida anteriormente segue o conceito de que grades computacionais são prestadoras de serviços. Cada nodo é um prestador de serviços na organização virtual a que pertence. Os serviços prestados pela grade não são pré-definidos, isso quer dizer que a plataforma possui mecanismos que permitem que os desenvolvedores implementem os serviços necessários conforme a aplicação.

Para que os serviços prestados por um nodo sejam invocados por outro nodo, de alguma forma, ele deve ser capaz de ser encontrado. A primeira idéia que surge é de um repositório central ou diretório de serviços, onde cada nodo registra seus serviços e então sempre que um nodo precisar invocar determinado serviço ele faz uma consulta a esse repositório para saber a qual nodo deve fazer a requisição. O problema de usar um repositório central é que ele passa a ser um centralizador de operações. Se ele falhar por algum motivo a grade deixa de funcionar pois os nodos não terão como encontrar os serviços.

Por esse motivo, a plataforma não faz uso de um diretório de serviços. Cada nodo possui os seus próprios serviços e através de mecanismos de descoberta internos da plataforma os demais nodos conseguem encontrar esses serviços quando for necessário. Esse método de funcionamento elimina a possibilidade de interrupção de prestação de serviços pela grade e segue o conceito de que a grade não deve possuir nodos que concentrem tarefas que possam fazer com que toda a estrutura de prestação de serviços seja interrompida em caso de falha deles.

4.1.2 Estrutura das mensagens

Segundo a definição de prestação de serviços apresentado anteriormente a grade proporciona serviços através dos nodos que a formam. A intercomunicação entre os nodos sempre refere-se a invocação desses serviços. Esse axioma conduz a idéia da teoria de diálogo de atores (speech-act) que ocorre em ambientes multi-agentes. Dessa forma, a estrutura das mensagens que são trocadas entre os nodos bem como a forma com que estas mensagens são trocadas seguem padrões utilizados em ambientes multi-agentes. Esses padrões são definidos pela Foundation for Intelligent Physical Agents (FIPA).

A estrutura das mensagens segue o padrão definido pelo documento "FIPA XC00061 - FIPA ACL Message Structure Specification" [fIPA 07b]. Esse documento, contém as especificações para os parâmetros das mensagens, incluindo: a lista dos elementos da mensagem corrente, procedimentos para manter esta lista e os critérios para inclusão de novos elementos na lista. A padronização das mensagens visa auxiliar na garantia de inter-operabilidade através de um conjunto comum de estruturas e também processos para garantir a manutenção desse conjunto.

As mensagens que são trocadas entre os nodos são representadas em XML, seguindo o padrão definido pelo documento "FIPA XC00071 FIPA ACL Message Representation in XML" [fIPA 07a]. Uma mensagem contém um conjunto de um ou mais elementos. Precisamente quais elementos são necessários em cada mensagem variam conforme a situação. Na plataforma, os elementos mandatórios são a ação ou serviço a ser executado e o remetente da mensagem, embora seja esperado que as mensagens também contenham o destinatário e mais algumas informações para execução da ação solicitada.

O motivo de uso desse tipo de estrutura de mensagens deve-se ao fato do padrão FIPA ACL ser um padrão consolidado e que atende perfeitamente as necessidades da arquitetura. Ressalta-se que os padrões definidos pelos documentos acima são usados como referência para a troca de mensagens, ou seja, não são implementados em sua totalidade.

4.1.3 Definição de tarefas

Segundo a documentação do Alchemi [NAD 07] [LUT 07]: As implementações "convencionais" de grades computacionais fornecem um nível alto de abstração de "máquina virtual", onde a menor unidade de execução paralela é um processo (tipicamente referenciado como um trabalho ou job, com vários trabalhos constituindo uma tarefa).

Em um ambiente formado por dispositivos móveis e embarcados a execução de tarefas de forma assíncrona pode ser a única forma de fazer as coisas funcionarem. Nesse tipo de ambiente, os nodos submetem tarefas (que são estruturas de dados) para serem executados em outros nodos e recebem uma resposta indicando se a tarefa foi executada com sucesso ou se aconteceu algum problema com a execução da mesma.

A plataforma trabalha exatamente dessa forma: para que um nodo da grade consiga invocar um serviço que está rodando em outro nodo é criada uma tarefa (Task). Assim que o nodo destino recebe a requisição de uma tarefa e a processa, ele sempre retorna um resultado (TaskResult) a qual indica a situação da tarefa recebida.

A figura 4.1 demonstra conceitualmente a troca de tarefas entre dois nodos.

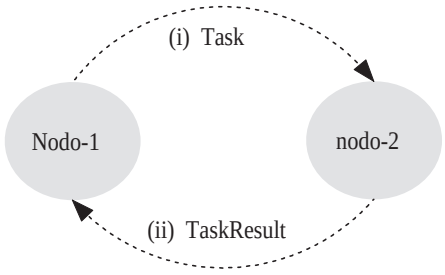


Figura 4.1: Visão conceitual de tarefas executadas entre nodos da grade

A estrutura de uma tarefa possui o formato demonstrado na tabela 4.1. Cada um dos campos utilizados possui a seguinte função:

➢ **Service:** define qual serviço será invocado no destino.

➢ **Originator:** define qual nodo originou a tarefa.

➢ **Destination:** define qual nodo será o destinatário da tarefa. Se esse

```
<task>
  <service> ...  </service>
  <originator> ... </originator>
  <destination> ... </destination>
  <taskid> .... </taskid>
  <timestamp> .... </timestamp>
  <priority> ... </priority>
  <parameters>
      ....
  </parameters>
</task>
```

Tabela 4.1: Definição da estrutura de uma tarefa

campo conter um * (asterisco) a grade será responsável por descobrir quem é o destinatário da tarefa (maiores detalhes são apresentados na seção 4.1.9).

➢ **Taskid:** define a identificação da tarefa. Sempre que uma nova tarefa é criada por um nodo é definido um identificador para ela.

➢ **Timestamp:** indica o tempo de criação da tarefa. Esse campo é utilizado para cálculos de qualidade de serviço pela plataforma.

➢ **Priority:** define qual a prioridade que deve ser dada à tarefa. A plataforma possui os seguintes níveis de prioridade:

→ Nível 1 = Prioridade baixa

→ Nível 2 = Prioridade normal. É a prioridade padrão na criação de tarefas

→ Nível 3 = Prioridade alta

→ Nível 4 = Prioridade muito alta

→ Nível 5 = Prioridade tempo-real

➢ **Parameters:** esse campo contém os parâmetros necessários para execução do serviço. Os parâmetros são codificados usando XML.

A TaskResult possui a estrutura definida na tabela 4.2 e de maneira análoga a estrutura da tarefa, cada um dos campos possui a função descrita a seguir:

```
<task-result>
  <result-code> ... </result-code>
  <originator> ... </originator>
  <destination> ... </destination>
  <taskid> .... </taskid>
  <timestamp> .... </timestamp>
  <priority> ... </priority>
  <parameters>
      ....
  </parameters>
</task-result>
```

Tabela 4.2: Definição da estrutura do resultado de uma tarefa

➢ **Result-code:** contém o resultado da execução da tarefa. Podendo conter:

→ OK = tarefa foi executada com sucesso.

→ DELAYED = tarefa foi recebida, aceita, porém ainda não foi executada.

→ TOO-BUSY = O nodo destino estava muito ocupado para executar a tarefa.

→ FAILURE = A execução da tarefa apresentou alguma falha.

→ NOT_UNDERSTOOD = O nodo destino não entendeu a tarefa.

→ NOT_FOUND = O serviço solicitado pela tarefa não foi encontrado.

→ REFUSE = O nodo destino recusou a tarefa.

➢ **Originator:** define qual nodo originou a resposta.

➢ **Destination:** define a qual nodo será enviado o resultado.

➢ **Taskid:** indica a identificação da tarefa executada.

➢ **Timestamp:** indica o tempo em que o resultado foi criado. Esse campo é utilizado para cálculos de qualidade de serviço pela plataforma.

➢ **Priority:** indica qual a prioridade que foi dada à tarefa. Possui os mesmos níveis de prioridade definidos pela tarefa.

➢ **Parameters:** contém os parâmetros retornados pelo nodo que rece-

beu a tarefa.

4.1.4 Codificação e decodificação dos dados

Conforme descrito na seção 3.5 a estrutura de dados XMLTree serve de base para representação de algumas estruturas e dos dados utilizados na plataforma. Para a manipulação da XMLTree a plataforma faz uso dos seguintes métodos:

➢ **add:** acresce um elemento ou sub-árvore a uma árvore existente

➢ **addText:** acresce texto a um elemento

➢ **get:** obtém uma sub-árvore na árvore

➢ **getSubTrees:** obtém as sub-árvores da árvore

➢ **getTagCount:** obtém a quantidade de elementos da árvore

➢ **getTagNames:** obtém os nomes dos elementos da árvore

➢ **getText:** obtém um texto da árvore

➢ **setAttribute:** define um atributo de um elemento

➢ **setText:** define um texto a um elemento ou atributo

A XMLTree é uma estrutura de dados criada com a finalidade de tornar mais simples a representação de dados usando XML. Os métodos apresentados acima são utilizados para manipulação da XMLTree e possibilitam ao desenvolvedor realizar todas as operações que são necessárias para a representação e trocas de dados que ocorrem no ambiente no qual a plataforma está sendo utilizada.

4.1.5 Suporte a qualidade de serviços

Para proporcionar a implementação de mecanismos que garantam qualidade de serviço na execução de tarefas, a plataforma possui uma relação de atributos que são atualizados em tempo de execução. Por questões de facilidade, cada nodo que compõe a grade controla os seus próprios atributos e cada módulo da plataforma possui o seu conjunto de atributos.

Essa separação de atributos por módulos permite grande flexibilidade para o desenvolvedor trabalhar com qualidade de serviço na plataforma. Ressalta-se, que

a plataforma controla esses atributos e os disponibiliza ao desenvolvedor, porém ela não implementa os mecanismos que usam esses atributos para garantir qualidade de serviço pois o assunto está fora do escopo do trabalho que foi definido inicialmente.

A descrição dos atributos que são controlados pela plataforma e disponibilizados para o desenvolvedor é apresentada no anexo B.

4.1.6 Agendamento de tarefas

Com a finalidade de automatizar tarefas que são executadas rotineiramente, é usado um conjunto de métodos para agendar a execução de tarefas em intervalos definidos de tempo. Os métodos são:

➢ **scheduleTimerTask:** agenda alguma tarefa para ser executada em intervalos definidos de tempo.

➢ **scheduleGarbaggeCollect:** executa o coletor de lixo de maneira automática no nodo.

➢ **scheduleLogHeartBeat:** efetua a coleta e log dos dados estatisticos do nodo.

➢ **scheduleHeartBeat:** transmite de tempo em tempo os dados estatisticos do nodo para os nodos de monitoramento.

➢ **scheduleSensorCollect:** efetua a coleta dos dados de um sensor em intervalos de tempo definido.

Os métodos usados para agendamento de tarefas além de efetuarem as tarefas de rotina interna dos nodos, também proporcionam ao desenvolvedor uma forma de expandir a capacidade da plataforma para a execução de ações que precisam ser executadas em intervalos definidos de tempo.

4.1.7 Atributos dinâmicos e perfis dos dispositivos

Para auxiliar nos mecanismos de qualidade de serviço, de monitoramento e localização de nodos que melhor se enquadrem para execução de determinada tarefa a plataforma implementa o conceito de perfil de dispositivos. Ao perfil do dis-

positivo pertencem as informações que caracterizam o nodo e também aquelas que são atualizadas em tempo de execução e demonstram a sua situação de funcionamento.

As informações que formam o perfil de um nodo são as seguintes:

➢ **memoryTotal:** total de memória disponível

➢ **memoryFree:** total de memória livre

➢ **memoryUsed:** total de memória usado

➢ **runningTime:** tempo de execução

➢ **cpuUsage:** valor indicando o percentual de uso da cpu

➢ **java_version:** versão do java

➢ **java_vendor:** distribuidor do java

➢ **os_name:** sistema operacional que está em execução

➢ **os_arch:** qual a arquitetura usada

➢ **os_version:** versão do sistema operacional

Se houver a necessidade de monitoramento do perfil de determinado nodo por outro nodo da grade pode ser usado o método addMonitorDestination(String destination). Esse método faz com que os dados de perfil sejam enviados a outro nodo em intervalos de tempo.

A definição de perfil de dispositivo têm por finalidade fornecer ao desenvolvedor uma forma simples de monitorar o funcionamento de cada nodo e também de proporcionar facilidade para implementação de aplicações que precisem executar determinadas ações de acordo com os recursos disponíveis em cada dispositivo que está sendo gerenciado pela grade.

4.1.8 Suporte a traps

Usando a idéia do Simple Network Management Protocol (SNMP) [CAS 07], a plataforma possui o conceito de notificação (trap) para reportar notificações a um nodo alocado (através do método addMonitorDestination) para ser o responsável pelo monitoramento da grade. Enquanto as informações de perfil de dispositivo são enviadas de tempo em tempo ao nodo monitor um trap somente é enviado quando determinados even-

tos acontecerem.

Os traps gerados podem ser de cinco tipos:

➢ **Note:** indica uma simples notificação.

➢ **Warn:** indica algum aviso, nada de grave.

➢ **Err:** indica algum erro considerado crítico.

➢ **Urgt:** indica algum problema que merece atenção urgente.

➢ **Fatl:** indica algum problema fatal que impede o funcionamento.

Os traps possibilitam ao desenvolvedor criar aplicações mais robustas para o gerenciamento dos dispositivos existentes em um ambiente. Outro ponto importante é que o seu uso ocasiona menor sobrecarga de tráfego na rede pois um nodo somente irá reportar ao monitor quando realmente estiver acontecendo alguma situação problemática que requer atenção.

4.1.9 Descobrimento de serviço e auto-organização

Para atender ao requisito de dinâmicidade e reconfiguração a plataforma possui a capacidade de efetuar o descobrimento de serviço e auto-organização de topologia. Segundo [AG 06] a auto-organização substitui a necessidade de intervenções administrativas, facilitando a agregação e o uso de recursos pela grade e também simplificando o desenvolvimento de aplicações.

A figura 4.2 ilustra o processo de descoberta de serviços e reconfiguração em um cenário formado por cinco nodos.

O nodo-0 deseja invocar um serviço que se encontra no nodo-4, porém o nodo-0 não possui conhecimento disso, nem sabe da existência do nodo-4 na grade. O nodo-0 solicita o serviço ao nodo-1. Como o nodo-1 não possui o serviço, ele cria uma tarefa de descoberta do serviço tendo como destino o nodo-2. Como o nodo-2 também não possui o serviço ele repassa a tarefa de descoberta ao nodo-3. O nodo-3 também não possui o serviço, então a tarefa de descoberta é encaminhada ao nodo-4. Como o nodo-4 possui o serviço solicitado pela tarefa de descoberta, ele responde ao nodo-3 que ele possui o serviço. O nodo-3 repassa essa informação ao nodo-2, que por sua vez repassa

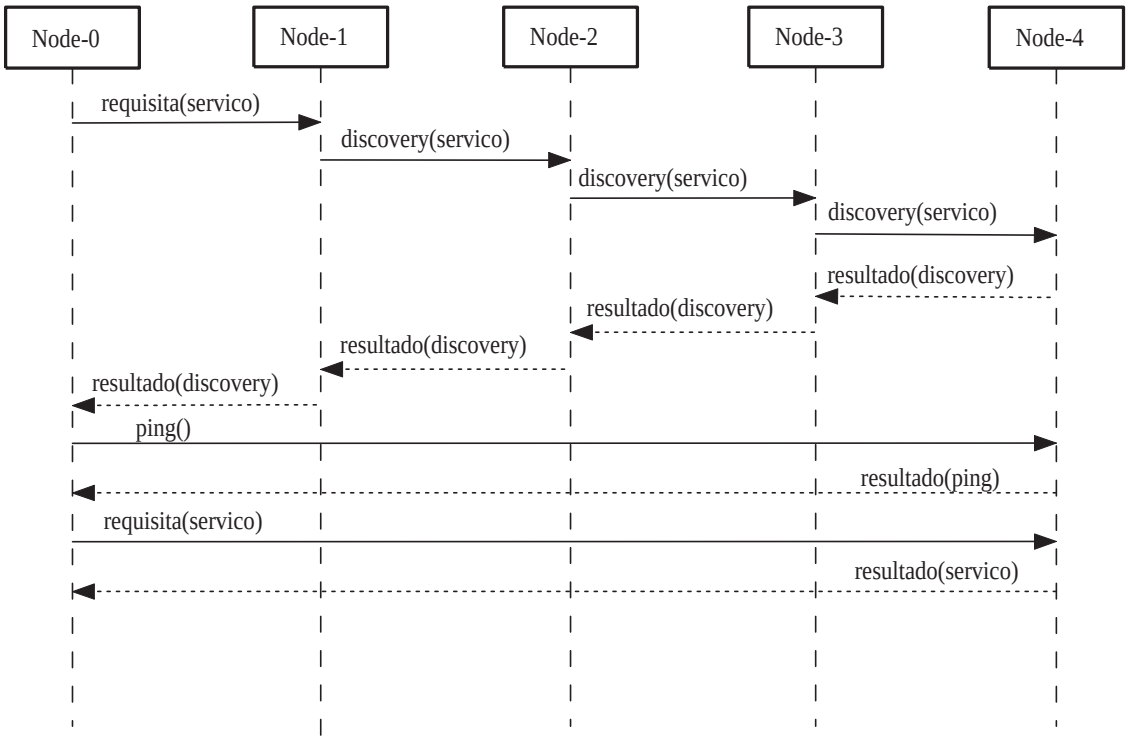


Figura 4.2: Processo de auto-descoberta de um serviço e reconfiguração

a informação ao nodo-1. O nodo-1, por sua vez, acaba informando ao nodo-0 onde está localizado o serviço. O nodo-0 então tenta efetuar contato direto com o nodo-4 enviando um ping. Quando o nodo-4 responde o ping, o nodo-0 faz uma alteração em sua tabela de roteamento acrescendo uma rota direta (métrica = 1) ao nodo-4. Como o nodo-0 agora "aprendeu" onde está rodando o serviço que necessita, a requisição do serviço é enviada diretamente ao nodo-4. Por fim o nodo-4 retorna a resposta do serviço invocado.

Se por algum motivo o nodo-0 não conseguir resposta ao ping enviado ao nodo-4 a rota direta não é criada. Isso pode acontecer se por exemplo o nodo-4 está atrás de algum servidor nat ou firewall. Nesse caso, a requisição do serviço vai sendo repassada de nodo em nodo até chegar ao nodo-4. A resposta proveniente da execução do serviço pelo nodo-4 também vai sendo repassada de nodo em nodo até chegar ao nodo requisitante do serviço, no caso o nodo-0.

O suporte a descobrimento e auto-organização possibilita o desenvolvimento de aplicações para ambientes onde existe a constante mudança de topologia de

rede devido a constante entrada e saída de dispositivos em diferentes pontos do ambiente. O uso de reconfiguração ou auto-organização permite que um dispositivo seja capaz de mover-se mesmo estando conectado a grade. A medida que ocorre a movimentação os mecanismos internos da grade asseguram que os demais nodos assimilem as mudanças de localização e sejam capaz de efetuar a comunicação quando necessário. Para uma aplicação invocar um serviço prestado por um nodo da rede ela não necessita saber qual nodo possui o serviço ou mesmo onde encontra-se tal nodo. Basta efetuar a requisição e a grade fica responsável por localizar o serviço e fazer os devidos ajustes de interconexão entre os nodos para assegurar que o serviço será invocado.

4.1.10 Roteamento entre nodos

No ambiente dinâmico formado por dispositivos móveis e embarcados as rotas entre os nodos podem sofrer constantes alterações. Visando manter a tabela de roteamento consistente entre os nodos uma adaptação do Routing Information Protocol (RIP) [HED 07] é utilizada.

Cada nodo da plataforma possui a sua própria tabela de roteamento contendo o nome do nodo destino, endereço de como alcança-lo (next hop) e métrica (indica a distância para alcançar o nodo destino). O método scheduleSendRoute é o responsável pelo envio da tabela de roteamento de um nodo para todos os nodos que possuem ligação direta com ele (métrica = 1). Assim como no RIP, cada nodo armazena somente as rotas com menores métricas para atingir o nodo destino sendo que quando um nodo recebe a tabela de roteamento do nodo vizinho e percebe que aconteceu alterações em alguma das entradas que ele possui em sua tabela local, ele atualiza-a para refletir as alterações que aconteceram no ambiente. No momento em que não acontecerem mais alterações nas tabelas de roteamento por propagação de rotas assumi-se que aconteceu a convergência das tabelas e todos os nodos do ambiente sabem como alcançarem uns aos outros. Esse tempo de convergência varia conforme a quantidade de nodos que existem no ambiente.

A outra forma de atualização da tabela de roteamento acontece quando um nodo tenta fazer uma descoberta de serviço conforme descreve a seção 4.1.9.

4.1.11 Ganchos (hooks) para novas aplicações

Os ganchos (hooks) utilizados na plataforma são definidos em interfaces ou classes abstratas e devem necessariamente ser implementados por uma classe que contenha o código real da aplicação.

Dois ganchos são proporcionados:

- No módulo Sensor existe o gancho _collect que é usado para implementação de novos métodos para coleta de dados dos sensores. Esse gancho possibilita que não seja necessária a re-implementação de um mecanismo de coleta de dados para cada sensor. Basta implementar os métodos para um sensor específico e então "pendura-los" no gancho do módulo Sensor.

- No módulo Service, existe o gancho _execute que é o responsável pela execução de novos serviços que venham a ser implementados. De forma análoga ao gancho anterior, basta implementar o método responsável pelo serviço e "pendurar" no módulo Service.

O ganchos são de grande valia para desenvolvedores pois proporcionam facilidades na implementação de novas funcionalidades e extensão daquelas já existentes, garantindo assim maior flexibilidade e reaproveitamento de código no desenvolvimento das aplicações para uso em grade.

4.1.12 Acesso aos resultados das tarefas

O acesso aos resultados de tarefas executadas pelas nodos, como por exemplo dados coletados a partir de sensores, é efetuado pelo módulo Network.

No caso do dispositivo possuir a plataforma em execução, o acesso a esses dados pode ser efetuado através de métodos internos da própria plataforma. Caso o dispositivo não a possua em execução também é possível ter acesso aos dados através de requisições ao servidor de requisições que fazem uso do HyperText Transfer Protocol (HTTP) (servlet) que é executado de forma acoplada a cada nodo da grade. A idéia é usar um servlet com suporte Extensible Style Language Transformation (XSLT) para retornar

os dados a interface do cliente conforme descrito em [GRA 03]. Nesse caso, o retorno de resultados acontece da seguinte forma:

I - Cliente faz uma requisição ao endereço do nodo através de um navegador internet.

II - A plataforma através de seus mecanismos internos verifica qual é o tipo de cliente que está fazendo a requisição.

III - É verificada a existência de um "template" contendo código Extensible Style Language (XSL) de acordo com o cliente detectado. Se não existir o template para o cliente, um template padrão é selecionado.

IV - É executada uma transformação usando o XML retornado pela execução da tarefa e o XSL contido dentro do arquivo de template de forma que o resultado é formatado de acordo com as configurações do cliente XSL.

V - O resultado da formatação é retornado ao cliente.

Usando essa forma de acesso é possível que diferentes tipos de dispositivos como PDAs, navegadores de internet ou até mesmo telefones celulares acessem os resultados de tarefas sem a necessidade de possuirem a plataforma instalada, garantindo assim maior capilaridade da grade.

4.1.13 Processos internos Node

A plataforma possui um conjunto interno de métodos padrão que podem ser invocados por outros nodos para a execução de tarefas consideradas básicas.

Os métodos internos disponíveis são:

➢ **Stop:** solicita que o nodo seja desconectado do ambiente.

➢ **Register:** é utilizado quando um nodo deseja entrar na grade. Como não existe um papel de centralizador de registros de novos nodos, cada nodo da grade possui o método register que se encarrega de registrar o novo nodo e informar ao demais membros da grade sobre a existência dele.

➢ **Ping:** envia uma requisição perguntando se um nodo está “operante” (up). Caso positivo, como resposta é retornado um “pong”. Se dentro de um tempo limite não houver resposta assume-se que o nodo destino está “inoperante” (down).

➢ **Get-delayed:** obtém o resultado de uma tarefa marcada como “delayed”

➢ **Route-table:** método responsável por receber a tabela de roteamento de outro nodo e efetuar os devidos ajustes na tabela local (utilizado pelos mecanismos definidos na seção 4.1.10).

➢ **Add-route:** acresce uma rota na tabela de roteamento local.

➢ **Set-default-route:** define uma rota padrão.

➢ **Discovery:** solicita a criação de uma tarefa de descoberta de serviço (o processo de descoberta de serviço é descrito na seção 4.1.9).

➢ **Check-attached-services:** retorna quais são os serviços que estão disponíveis no nodo.

➢ **Check-attached-sensors:** retorna quais são os sensores que estão disponíveis no nodo.

➢ **Get-statistics:** retorna dados estatísticos de cada um dos módulos do middleware.

Como dito anteriormente esses métodos visam manter o correto funcionamento do nodo. Se for necessário que novas funcionalidades sejam acrescidas às rotinas internas de núcleo da platasforma, elas podem ser acrescidas nesse conjunto de métodos.

4.2 Interface de Programação de Aplicativos

Interface de Programação de Aplicativos, ou API (Application Programming Interface), é um conjunto de rotinas e padrões estabelecidos por um software para utilização de suas funcionalidades por programas aplicativos – isto é: programas que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

A API da plataforma foi escrita na linguagem Java devido a portabilidade da linguagem sendo que a versão binária é distribuída em Java 2 Micro Edition [SM 07a] e Java 2 Standard Edition [SM 07b]. Seu pequeno tamanho permite o uso no desenvolvimento de aplicações de computação em grade para dispositivos móveis e embarcados.

Na próxima seção são demonstrados os passos necessários para a construção de uma aplicação que faz uso da plataforma.

4.2.1 Passos para uso da API

Para a criação de um ambiente de grade computacional usando a API são necessários que sejam executados os seguintes passos para cada nodo que compõe a grade:

I - Criar um nodo: para criar o nodo é necessário definir o nome que ele irá possuir, em qual endereço e em qual porta que ocorrerá a comunicação com os demais nodos do ambiente.

Exemplo: Node node1 = new Node(”node-1”, ”http://localhost:8001”);

II - Definir a interface de comunicação: é o processo de criação da interface (servlet) que irá receber/enviar requisições e associar essa ao nodo.

Exemplo: NetInterface httpNetInterface = new HTTPServerInterface(node1, 8001);

III - Disponibilizar serviços: é opcional e define se o nodo vai possuir algum serviço que será disponibilizado para ser invocado por outros nodos.

Exemplo: node1.addService(”multiply”, new MathService());

IV - Associar sensores: é opcional e visa definir se o nodo possuirá associado algum sensor (ou sensores). Caso seja associado algum, torna-se necessário definir o intervalo de tempo em que dados do sensor serão obtidos, para qual nodo os dados obtidos serão enviados e qual será o serviço que será o responsável por manipular os dados nesse nodo.

Exemplo: node1.addSensor(”sensor-1”, new RandomSensor());
node1.scheduleSensorCollect(”sensor-1”, ”monitor_node”, ”store”, 5);

V - Definir tabela de roteamento: efetua a ligação do nodo com outro nodo da grade.

Exemplo: node1.addRoute(”node-2”, ”http://localhost:8002”);

VI - Definir configurações especificas: é opcional e depende da função do nodo na grade. Podem ser utilizados métodos para definição do nível de log, de “snoop” da interface de comunicação ou tarefas de agendamento de monitoramento do nodo.

Exemplo: node1.setLogLevel(5);
node1.addMonitorDestination(”monitor_node”);

VII - Iniciar o nodo: é o simples processo de colocar o nodo em funcionamento.

Exemplo: node1.start();

A próxima seção visa demonstrar de forma prática como utilizar os passos abordados acima para a construção de uma aplicação.

4.2.2 Exemplo de uso

Para demonstrar o uso da API foi codificada a aplicação esquematizada na figura 4.3. Nessa aplicação são criados dois nodos, sendo que um deles (node-2) possui um serviço chamado multiply que recebe dois números, multiplica-os e retorna o resultado. O node-1 então invoca o serviço de multiplicação do node-2 solicitando para serem multiplicados os números 3 e 6 e recebe como resposta o resultado enviado pelo node-2 que é 6.

O exemplo apresentado é uma simples aplicação que visa demonstrar a simplicidade e flexibilidade de desenvolvimento de aplicações que fazem uso de grade proporcionada pela plataforma. Aplicações mais complexas podem ser construídas facilmente através das funções existentes na API.

O código fonte da aplicação é apresentado no anexo C e a descrição de todos os métodos existentes na API pode ser encontrada no site do projeto em http://grid.lrg.ufsc.br.

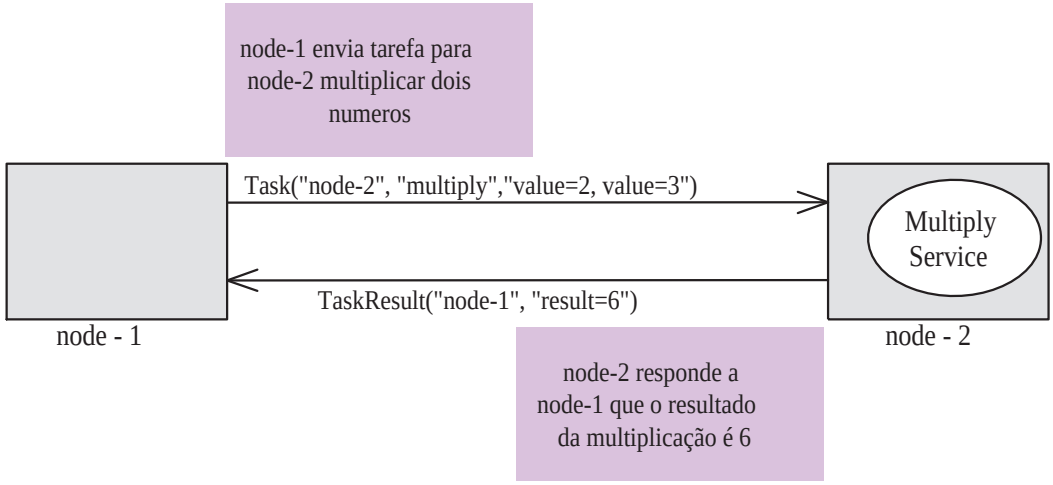


Figura 4.3: Esquema da aplicação desenvolvida para demonstrar o uso da API

4.3 Testes quantitativos

Esta seção apresenta resultados de alguns resultados quantitativos obtidos em simulações que buscam demonstrar o comportamento da plataforma em diferentes situações de uso.

Para realização dos testes foi montado o ambiente representado pela figura 4.4

No ambiente existem um total de trinta computadores (1) conectados a sensores. A quantidade de sensores existentes em cada um vária conforme o teste realizado. Os dados coletados pelos sensores são então armazenados em dois computadores que possuem o serviço de armazenamento (2). Em intervalos definidos de tempo cada computador do ambiente envia seus dados estatísticos para os Monitores (3) os quais armazenam os dados para posterior análise. Todos os computadores estão interligados entre si através de switch.

Nesse ambiente as seguintes situações foram simuladas:

- **Experimento 1:** Foram usados trinta nodos com cinco sensores cada de forma que cada nodo possuia seu próprio serviço para armazenamento dos dados coletados a partir de seus sensores. Foram dispostos quatro nodos monitores responsáveis pelo

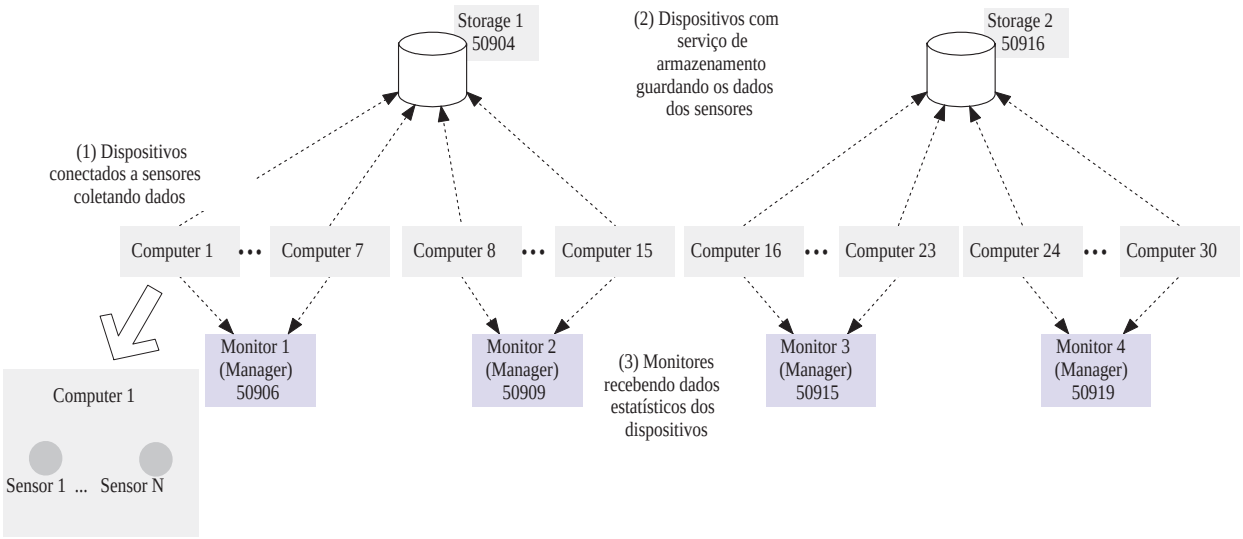


Figura 4.4: Ambiente montado para teste quantitativo

recebimento dos dados estatísticos dos nodos coletores. O objetivo desse experimento foi verificar o comportamento dos nodos monitores.

- **Experimento 2:** Também foram usados trinta nodos com cinco sensores cada. Porém nesse experimento foram disponibilizados dois serviços de armazenamento para os nodos guardarem os dados coletados a partir do sensores.

- **Experimento 3:** Foram usados trinta nodos com vinte e cinco sensores cada. Foram disponibilizados dois serviços de armazenamento para os nodos guardarem os dados coletados a partir do sensores.

- **Experimento 4:** Foram usados trinta nodos com cem sensores cada. Foram disponibilizados dois serviços de armazenamento para os nodos guardarem os dados coletados a partir do sensores.

4.3.1 Análise dos resultados

Para análise dos resultados foram utilizados os dados estatísticos obtidos em cada experimento e então observado o comportamento de três tipos diferentes de

nodos: Um nodo monitor (Nodo 50906) que é o responsável pelo monitoramento dos nodos do nodos do ambiente, um nodo coletor (Nodo 50910) que é o responsável pela coleta de dados e um nodo de armazenamento (Nodo 50916) que é o responsável pelo armazenamento dos dados coletados.

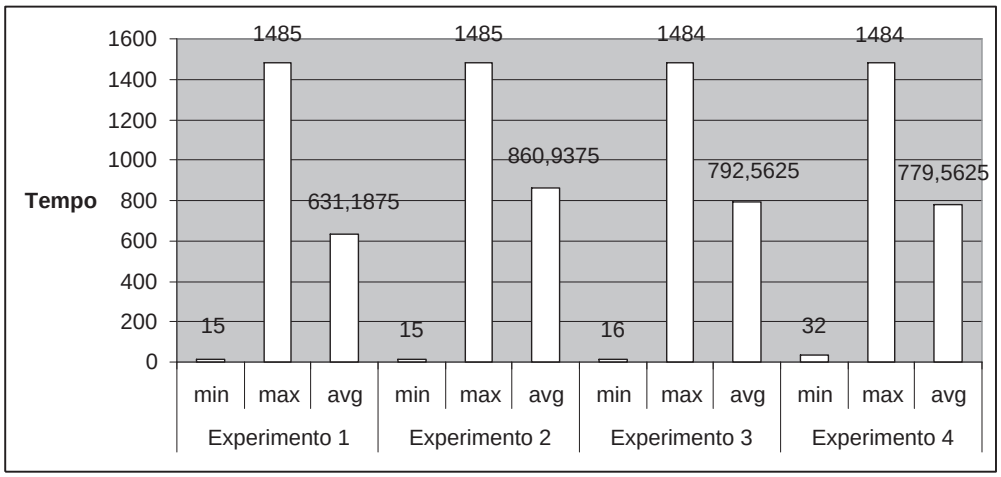


Figura 4.5: Gráfico demonstrando os tempos mínimos, médio e máximo do nodo monitor (50906) em cada experimento

Segundo a figura 4.5 os tempos máximo, mínimo e médio para execução da tarefa de recepção dos dados estatísticos de um monitor não sofrem grande interferência com o aumento da quantidade de sensores.

O mesmo não acontece com relação ao consumo de memória. A figura 4.6 demonstra o total de memória consumida em cada um dos experimentos. Pode-se perceber um aumento significativo do uso de memória no experimento 4.

A figura 4.7 representa o total de bytes enviados e recebidos pelo nodo coletor(50910), no caso um nodo de coleta de dados. Pode-se perceber que quando maior o número de sensores maior é quantidade de bytes que trafegam. No caso do experimento 1 o tráfego de bytes recebidos é maior que o experimento 2 pois os dados estão sendo enviados para o próprio nodo armazenar. A partir do experimento 2 começa a ocorrer um crescimento regular pois os dados estão sendo enviados para um nodo de armazenamento por isso que a quantidade de bytes enviados é maior que recebido.

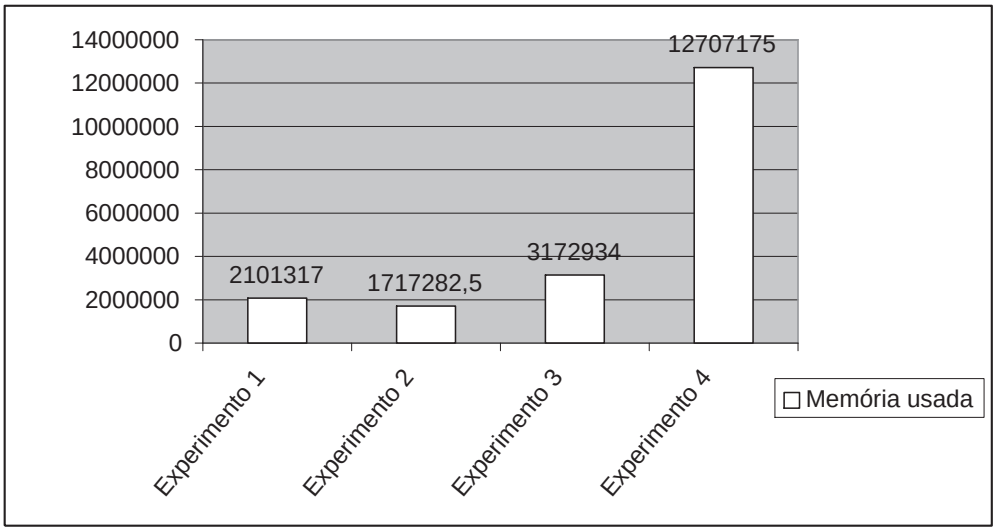


Figura 4.6: Gráfico demonstrando o consumo de memória do nodo monitor (50906) em cada experimento

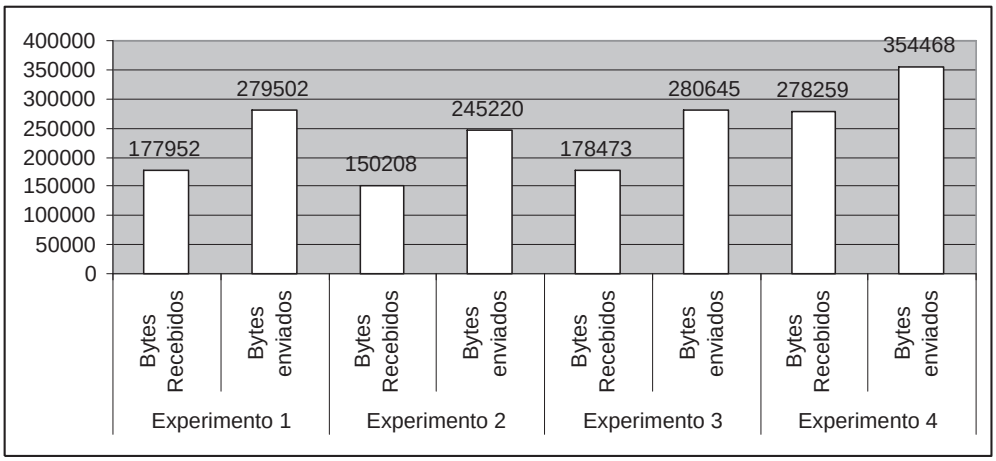


Figura 4.7: Gráfico demonstrando total de bytes enviados e recebidos pelo nodo coletor (50910) em cada experimento

A figura 4.8 também demonstra que quanto mais sensores estiverem anexados em um nodo coletor maior será o consumo de memória. No caso do experimento 1 percebe-se um consumo maior que o experimento 2, isso deve-se ao fato do nodo além de coletar dados também estar disponibilizando o serviço de armazenamento.

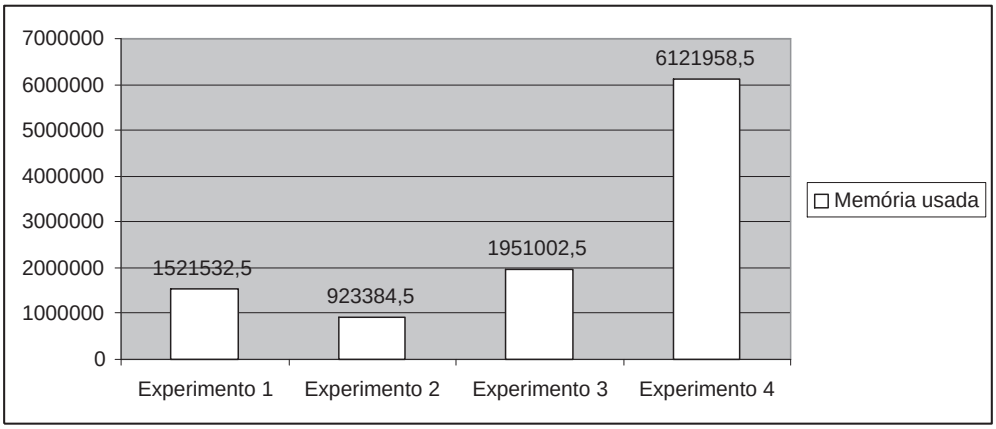


Figura 4.8: Gráfico demonstrando o consumo de memória do nodo 50910 em cada experimento

No caso do nodo de armazenamento 50916 o consumo de memória sofreu diretamente o impacto da quantidade de dados que estavam sendo armazenados. A figura 4.9 representa gráficamente o consumo.

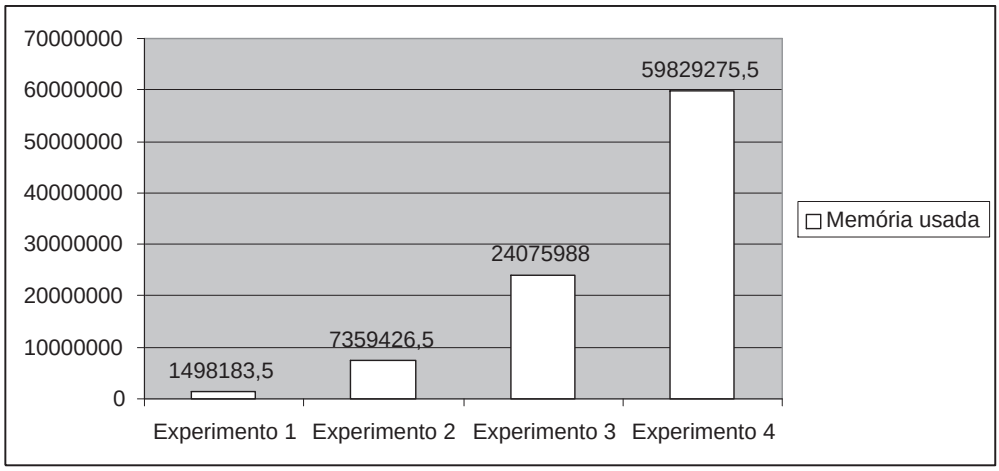


Figura 4.9: Gráfico demonstrando o consumo de memória do nodo 50916 em cada experimento

O mesmo ocorre com a quantidade de bytes recebidos por esse nodo (figura 4.10). Esse comportamento já era esperado uma vez que todos os nodos estão enviando os seus dados para serem armazenados pelos nodos que possuem os serviços de armazenamento.

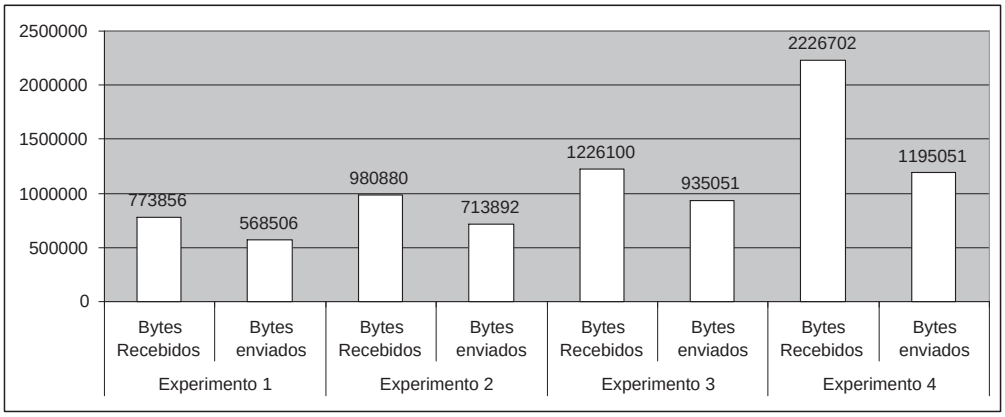


Figura 4.10: Gráfico demonstrando total de bytes enviados e recebidos pelo nodo de armazenamento (50916) em cada experimento

A figura 4.11 demonstra claramente que a quantidade de requisições atendidas interfere no tempo de resposta. Com o nodo de armazenamento recebendo requisições dos nodos coletores com cem sensores (experimento 4) o tempo máximo cresceu muito.

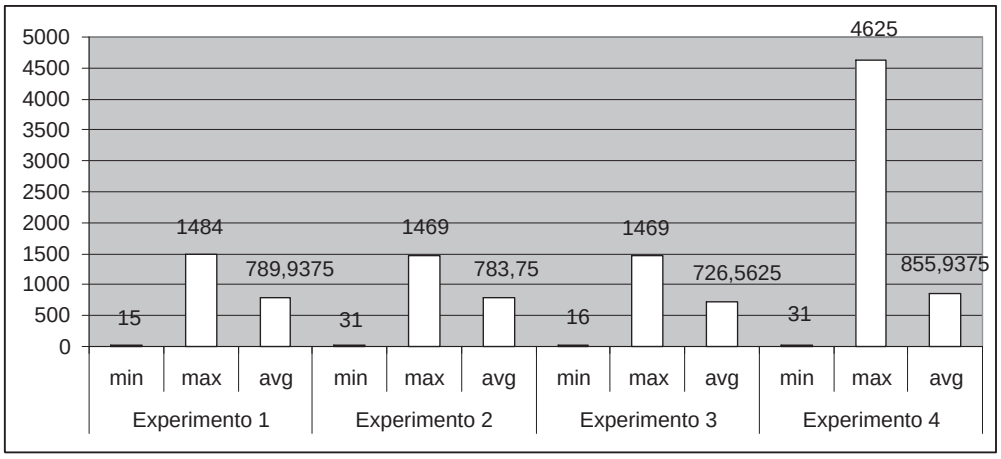


Figura 4.11: Gráfico demonstrando os tempos mínimos, médio e máximo do nodo de armazenamento (50916) em cada experimento

Pelos resultados desse conjunto de experimentos pode-se concluir que:

- A quantidade de sensores conectados em um nodo interfere diretamente no con-

sumo de memória dos dispositivos

- Os nodos monitores não sofrem interferência no atendimento de requisições com o aumento de nodos ou sensores

- Como esperado a quantidade de bytes que entra e sai de cada nodo é proporcional a quantidade de nodos e sensores

- Apesar do consumo de memória aumentar a medida que aumenta-se a quantidade de nodos a plataforma necessita de alguma forma de otimização de memória. Isso é algo referente específicamente a implementação.

- O uso de XML possibilita maior versatilidade porém, aumenta o consumo de banda e a quantidade de processamento para interpretação. O ideal seria fazer testes com a plataforma usando uma forma compactada de XML como o WBXML [MAR 07] e comparar qual é o impacto ocasionado pela mudança.

- O uso da plataforma em rede cabeada não deve sofrer diferenças de consumo de banda ou no consumo de memória dos nodos. Porém o ideal serial que esses testes fossem efetuados usando redes sem fios.

Nos experimentos a plataforma comportou-se sem problemas até cem sensores por nodo. Na tentativa de aumentar mais a quantidade de sensores ocorreram problemas relativos ao consumo de memória e sobrecarga de processamento de cada nodo. Com trinta nodos o ambiente não apresentou problemas. Pelas avaliações estatísticas a quantidade máxima de nodos que podem existir em um ambiente está limitada a disponibilidade de recursos físicos de cada dispositivo e a largura de banda disponível para tráfego de dados e não relacionada a alguma restrição da arquitetura.

Os experimentos demonstrados acima visaram fornecer uma visão inicial do comportamento da plataforma em algumas situações de uso, porém diferentes cenários e objetivos podem ser definidos e simulados para tornar possível uma análise mais aprofundada dos dados.

4.4 Estudo de caso

Para apresentar a funcionalidade da plataforma e sua forma de comportamento em uma situação prática esta seção demonstra um estudo de caso na área da telemedicina. O estudo de caso consiste na demonstração de como a plataforma pode ser aplicada para resolver o cenário-problema apresentado no capítulo 1.

O objetivo é automatizar o processo de coleta de dados vitais dos pacientes a partir dos equipamentos de monitoramento existentes em um hospital. Os dados coletados são então disponibilizados para consulta pelo corpo médico através de dispositivos móveis que podem estar localizados dentro ou fora do hospital.

Seguindo o conceito da plataforma todos os dispositivos usados são vistos como nodos e nodos formam o ambiente que será responsável pela disponibilização dos serviços que serão prestados pela grade. Dessa forma, os dados são coletados por nodos sensores, usando protocolos ubíquos trafegam via ondas de rádio até os nodos que possuem o serviço de armazenamento e então podem ser visualizados pelos dispositivos de consulta através da invocação do serviço de visualização pelas aplicações móveis. A figura 4.12 demonstra a arquitetura usada no estudo de caso.

Para implementar os (i) nodos sensores é usado um equipamento comercial de rádio juntamente com uma instância da plataforma. O equipamento usado é da marca Linksys modelo WRT54G rodando uma versão minimalista do sistema operacional linux. Algumas alterações no equipamento de rádio foram efetuadas para possibilitar que a comunicação entre ele e o monitor vital seja efetuada via porta serial.

A instância da plataforma é executada no equipamento de forma que o (ii) módulo que implementa a interface com sensores são os responsáveis pela coleta dos dados que são enviados pelo monitor de sinais vitais ao equipamento. Após os dados serem coletados é feita a (iii) inferência e a conversão dos dados segundo o padrão usado pela plataforma pelo módulo responsável. O (iv) módulo Network então os envia através da interface de rádio do equipamento.

Os (v) nodos prestadores de serviço possuem os serviços que serão invocados pelos demais nodos do ambiente. Internamente esses nodos executam uma instância

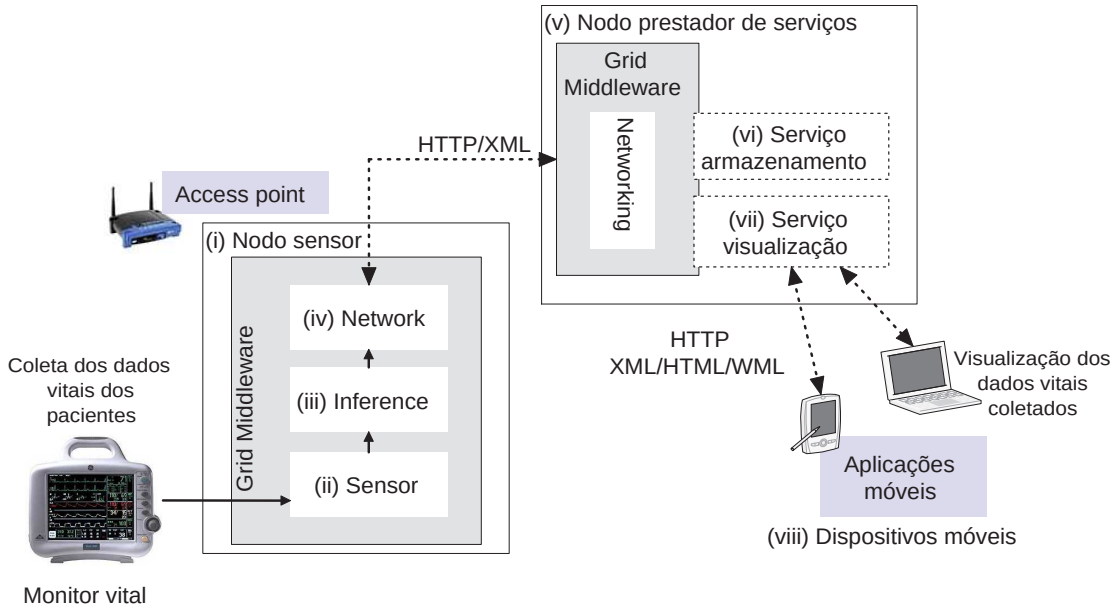


Figura 4.12: Arquitetura usada no estudo de caso para monitoramento remoto de pacientes

da plataforma. Usando os ganchos proporcionados por ela, é implementado o (vi) serviço de armazenamento que será invocado pelos nodos sensores para armazenar os dados coletados. O serviço de armazenamento faz uso da biblioteca bdb [ORA 07] para gerenciar a base de dados obtida pelas requisições de armazenamento.

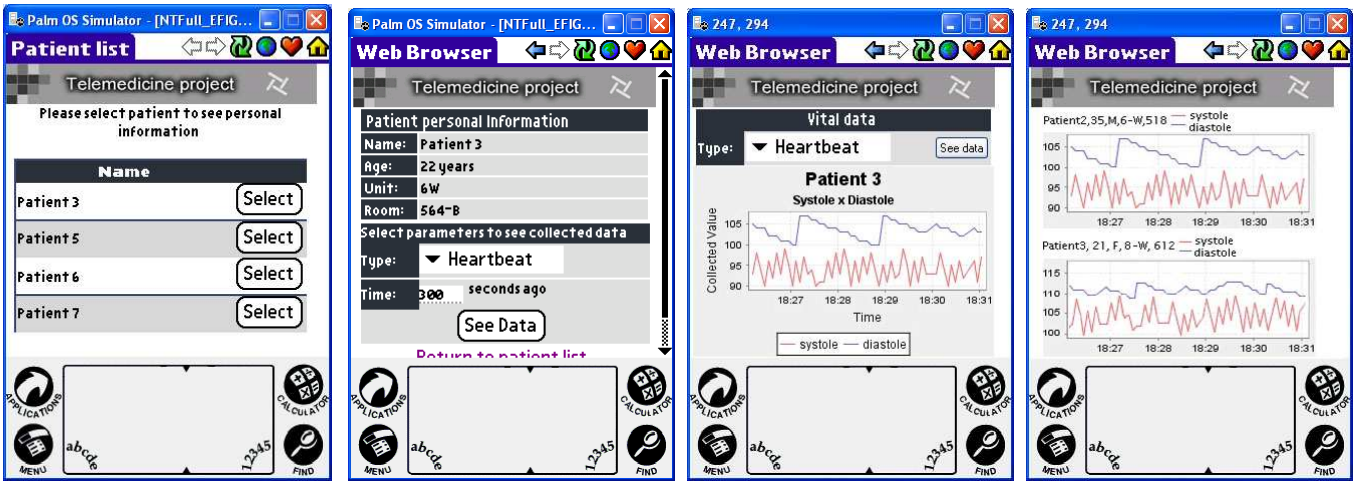
Os (viii) dispositivos de consulta fazem uso das aplicações móveis para visualização dos dados armazenados na base de dados. A consulta pode ser feita no interior do hospital através de PDAs, que usam a mesma infra-estrutura de coleta, ou então externamente através da Internet por celulares ou navegadores. Para efetuar a visualização dos dados vitais dos pacientes, os dispositivos de consulta invocam o (vii) serviço de visualização do ambiente. Então, através dos mecanismos internos da plataforma de sensibilidade ao contexto o resultado é formatado e exibido de acordo com o dispositivo usado.

A figura 4.13 demonstram os dados vitais dos pacientes visualizados a partir de um celular. Em (a) é apresentada a tela para seleção do paciente (b) apresenta as opções para visualização (c) apresenta os dados para visualização.

A figura 4.14 apresenta a saída caso fosse utilizado um PDA para acesso

(a) seleção do paciente (b) opções de visualização (c) visualização dos dados

Figura 4.13: Visualização dos dados através de um celular

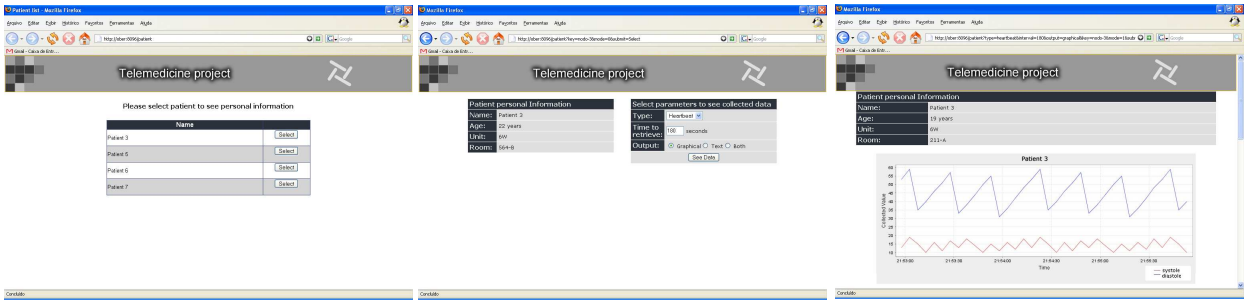


(a) seleção do paciente (b) opções de visualização (c) visualização dos dados (d) visualização dos dados

Figura 4.14: Visualização dos dados através de um PDA

aos dados vitais. (a) apresenta a relação de pacientes ao qual o dispositivo possui acesso. (b) apresenta a tela onde podem ser selecionados os parametros para visualização dos dados vitais. (c) apresenta a visualização do gráfico representando o batimento cardíaco

de um paciente. (d) apresenta a tela onde são visualizados os batimentos cardíacos de mais de um paciente ao mesmo tempo.



(a) seleção do paciente (b) opções de visualização (c) visualização dos dados

Figura 4.15: Visualização dos dados através de um navegador internet

A figura 4.15 apresenta os dados visualizados através de um navegador Internet. (a) é a tela para seleção do paciente a visualizar. (b) apresenta a tela para seleção de quais dados vitais deseja-se visualizar e qual o intervalo de tempo a ser usado. (c) apresenta um gráfico com os dados vitais do paciente selecionado.

Os resultados obtidos no estudo de caso demonstraram como a plataforma pode ser aplicada no cenário-problema apresentado inicialmente. O uso da plataforma possibilitou atender os requisitos apresentados na seção 2.1 de: (1) mecanismos para gerenciamento do ambiente de forma homogênea; (2) mecanismos para disponibilização dos dados vitais dos pacientes para os médicos em tempo real; (3) mecanismos capazes de lidar com o dinamismo e mobilidade existentes no cenário; e (4) mecanismos para interação com os diferentes tipos de sensores usados na coleta de dados.

4.5 Conclusão do capítulo

Esse capítulo demonstrou como a plataforma foi implementada a partir das definições da arquitetura proposta. A sua principal contribuição foi na apresentação dos conceitos e métodos usados para tornar a plataforma funcional.

Os resultados quantitativos auxiliaram na demonstração de comportamento da plataforma. Algo que ficou faltando e que necessita ser feito são mais testes,

agora utilizando dispositivos móveis para chegar-se a uma conclusão até onde o tipo de rede e de dispositivos usados interferem diretamente na performance da plataforma.

O estudo de caso serviu para demonstrar o comportamento da plataforma na prática e também como ela pode ser utilizada para solucionar um importante problema do mundo real. Uma necessidade encontrada no estudo de caso foi o uso de algum mecanismo de inteligência artificial (talvez uma rede neural) para construção do sistema de alerta ao médico responsável em caso de problemas com um paciente.

Para finalizar, esse capítulo pode ser concluído com o seguinte raciocínio lógico:

- Análise do cenário-problema define componentes e métodos existentes

- Requisitos da arquitetura são obtidos com base nos componentes e métodos

- Arquitetura é definida com base nos requisitos, logo arquitetura atende requisitos

- Plataforma implementa arquitetura, logo plataforma atende cenário-problema.

Ou seja, os conceitos definidos pelo trabalho solucionam o problema inicial de forma que consegue-se usar grade computacional para integração e gerenciamento de um ambiente formado por dispositivos móveis e embarcados.

O próximo capítulo encerra a pesquisa apresentando as conclusões finais e trabalhos futuros.

Capítulo 5

Conclusão

Esse trabalho apresentou qual é a estrutura de software necessária para desenvolver grades computacionais capazes de integrar e gerenciar ambientes formados por dispositivos móveis e embarcados. Na literatura utilizada constatou-se que grades computacionais são usadas como uma forma de compartilhamento de recursos em larga escala. Algo desconsiderado pela maioria dos trabalhos analisados foi que grades computacionais possuem recursos que possibilitam que elas sejam usadas como ferramenta para interligação e gerenciamento homogêneo dos dispositivos existentes em um ambiente. A proposta do trabalho foi justamente de explorar essa possibilidade, aplicando uma grade para facilitar a integração e o gerenciamento de um ambiente formado por dispositivos móveis e embarcados. Porém ficou claro que uma grade que possua as características presentes em grandes de sensores e também as presentes em grades móveis é algo inovador com requisitos que as plataformas analisadas não foram capazes de atender.

Como passo inicial para o desenvolvimento da plataforma, foi necessária a definição de maneira conceitual de como ela deveria comportar-se. Para essa definição no capítulo 3 foi efetuada uma análise de um cenário formado por dispositivos móveis e embarcados que apontou quais os elementos e interações existentes. Essas informações acabaram por definir que uma plataforma de grade computacional capaz de lidar com o cenário deve ter como requisitos o uso de protocolos e padrões abertos, coordenação de recursos existentes no ambiente, flexibilidade para comunicação com diferentes dis-

positivos, mecanismos para proporcionar qualidade de serviço, mecanismos para gerenciar ambientes dinâmicos, mecanismos de descoberta e reconfiguração, transparência no acesso aos resultados das tarefas e também interfaces capazes de efetuarem comunicação com sensores. Os requisitos serviram como base para a definição de uma arquitetura que quando implementada possui as características necessárias para ser aplicada na solução do cenário-problema.

A implementação da plataforma foi apresentada no capítulo 4 que demonstrou como se comporta um ambiente formado por dispositivos móveis e embarcados integrados por grades de computadores. Como a plataforma implementou a arquitetura, ela atendeu aos requisitos definidos anteriormente exatamente conforme esperado. Os resultados dos testes quantitativos obtidos na simulações de uso da plataforma demonstraram que ela é funcional. Analisando os gráficos constatou-se que os tempos de atendimento de requisições e disponibilização de dados estiveram dentro do aceitável. Porém pode-se perceber que ela consome muitos recursos de processamento e memória dos dispositivos o que demonstra que a implementação deve ser melhorada. Isso é aceitável uma vez que a implementação teve como finalidade verificar se os conceitos da arquitetura apresentados iriam ser capazes de resolver o problema de integração e gerenciamento do cenário e isso aconteceu. O que realmente demonstrou que os conceitos funcionam foi o estudo de caso que provou que com o uso da plataforma os dados foram coletados dos sensores de maneira automatizada, armazenados em servidores que a própria plataforma controlava e então puderam ser disponibilizados por ela para visualização por diferentes tipos de dispositivos móveis e fixos de forma transparente para o usuário e possibilitando ao desenvolvedor re-usar os componentes construídos.

Após a conclusão do trabalho ficou comprovado que o paradigma de grade computacional proporciona que os dispositivos móveis e embarcados existentes em um ambiente sejam integrados e gerenciados de forma homogênea (virtualizada). A plataforma implementada a partir das definições do trabalho foi capaz de ser aplicada para solucionar o cenário-problema apresentado como motivação do trabalho. Dessa forma conclui-se que os conceitos apresentados conseguiram estender o uso de grades computacionais para ambientes móveis e embarcados completando uma deficiência na literatura e

atendendo a proposta que foi definida inicialmente.

Algumas carências observadas na plataforma proposta é a necessidade de trabalhar com diferentes organizações virtuais, melhorar o mecanismo de descoberta de recursos, melhorar alguns aspectos básicos de segurança dos dados e também tornar mais compacta e otimizada as estruturas de mensagens que são trocadas entre os nodos visando menor consumo de recursos.

Como contribuições do trabalho pode-se destacar a definição de um modelo conceitual de grade computacional que quando implementada proporciona uma plataforma que possui as características necessárias para ser aplicada em ambientes móveis e embarcados. Esta plataforma cobre uma lacuna na literatura de uma plataforma que possua um misto das características presentes em grades de sensores e também em grades móveis. Outro ponto importante foi a abordagem de como o paradigma de grade computacional pode ser utilizado para efetuar a integração e o gerenciamento de ambientes móveis e embarcados. Grades computacionais na maioria dos casos são aplicadas como uma forma de compartilhamento de recursos em larga escala exigidos por aplicações científicas de forma que seu uso com a finalidade de integração e gerenciamento são deixados de lado. Como exposto no decorrer do trabalho usando uma grade para esse fim proporciona que o desenvolvimento de aplicações para ambientes móveis e embarcados seja simplificado pois são reaproveitados os recursos proporcionados de forma intrínseca pela grade.

5.1 Trabalhos futuros

O trabalho teve como escopo demonstrar como grades computacionais podem ser utilizadas para integração e gerenciamento de ambientes formados por dispositivos móveis e embarcados. Com a conclusão do mesmo verificou-se a abertura de um leque de áreas que futuramente podem ser exploradas por trabalhos científicos:

- desenvolvimento de interfaces que auxiliem na criação de aplicações para a avaliação dos diferentes aspectos envolvidos no ambiente de aplicação da grade. É necessário

um estudo mais aprofundado visando a definição de quais dados são relevantes para a execução de simulações que avaliem os aspectos relacionados a mobilidade na grade.

- execução de um estudo mais aprofundado sobre qualidade de serviço para a grade. Podem ser avaliados quais os aspectos necessitam ser controlados e como diferentes algoritmos podem ser empregados para trabalhar com QoS em uma grade para dispositivos móveis e embarcados.

- estudos relacionados a como garantir a segurança da infraestrutura da grade. Podem ser abordados quais os parâmetros, políticas de segurança e mecanismos que devem ser usados para proporcionar que diferentes níveis de acesso, autenticação e autorização sejam fornecidos.

- estudos para proporcionar o auto-gerenciamento da grade. É necessária a definição de interfaces e estruturas que possibilitem a aplicação de inteligência artificial para que a grade seja capaz de adaptar sua infraestrutura de forma autônoma de acordo com uma situação.

- estender os estudos de auto-gerenciamento para que a grade adapte sua infraestrutura de forma autônoma, proporcionando assim, os mecanismos de segurança e também os parâmetros de qualidade de serviço em resposta a determinadas situações.

- estudos sobre aplicação de diferentes protocolos de roteamento que podem ser empregados em uma grade para ambientes móveis e embarcados.

Apêndice A

Métodos da arquitetura

De forma abstrata os métodos podem ser implementado da seguinte forma:

I - **Métodos para agendamento de tarefas:** sãos os métodos relacionados com a execução de tarefas em intervalos definidos de tempo

→ void agendaTarefa(Tarefa tarefa, int intervalo) - agenda tarefas genéricas que são executadas em intervalos definidos de tempo.

→ void agendaColetaLixo(int intervalo) - agenda coleta de lixo do ambiente.

→ void agendaColetaSensor(Sensor sensor, int intervalo) - agenda coleta de dados de um sensor.

→ void agendaHeartBeat(int intervalo) - agenda envio de situação do nodo.

II - **Métodos para manipulação de log:** métodos responsáveis pela criação e gerenciamento de log de cada nodo

→ void defineNivelLog(int nivel) - define o nível de detalhamento das mensagens de log do nodo.

→ boolean verificaNivelLog(int nivel) - verifica se o nodo possui definido um determinado nivel de log.

→ void log(int nível, int tipo, int evento, string elemento, string mensagem) - cria uma entrada de log

III - **Métodos para comunicação entre nodos:** são métodos que necessários para que um nodo consiga comunicar-se com outro

→ void acresceRota(string nodo, string endereco) - acresce rota para determinado nodo

→ string encontraRota(string nodo) - retorna o endereço para alcançar determinado nodo ou nulo caso não encontrado.

→ boolean enderecoLocal(string endereco) - verifica se o endereço pertence ao nodo local

→ boolean ping(string nodo) - verifica se determinado nodo está respondendo

→ void defineRotaPadrao(string endereco) - define determinado endereço como rota padrão

IV - **Métodos para manipulação de tarefas:** responsáveis pela criação, codificação, envio, processamento e recebimento de tarefas e seus resultados

→ResultadoTarefa enviaTarefa(Tarefa tarefa) - envia uma tarefa através do canal de comunicação

→ ResultadoTarefa recebeTarefa(Tarefa tarefa) - recebe uma tarefa para execução retornando o seu resultado.

→ void registraTarefaRecebida(Tarefa tarefa) - registra os dados de uma tarefa recebida

→ void armazenaResultadoTarefa(ResultadoTarefa resultado) - armazena o resultado da execução de uma tarefa que foi executada com atraso (delayed)

→ void obtemResultadoTarefa(string tarefaIdentificacao) - obtém o resultado armazenado de uma tarefa que foi executada com atraso (delayed)

→ void defineSituaçãoResultadoTarefa(Tarefa tarefa, string situacao) - define a situação de uma tarefa. Situação pode ser definida como tarefa executada, postergada, nodo ocupado, falha na execução, não entendida, não encontrada, recusada

→ string obtemSituaçãoResultadoTarefa(Tarefa tarefa) - retorna a situação do resultado de uma tarefa

V - **Métodos para gerenciamento interno do nodo:** métodos responsáveis pelo funcionamento interno de cada nodo

→ XMLTree verificaServicosConectados(Node nodo) - verifica quais são os serviços que estão disponíveis no nodo

→ XMLTree verificaSensoresConectados(Node nodo) - verifica quais são os sensores que estão disponíveis no nodo

→ XMLTree obtemEstatisticas(Node nodo) - obtém os dados estatísticos de cada componente da arquitetura

VI - **Métodos para qualidade de serviço (QoS):** permitem quem as tarefas sejam executadas com qualidade de serviço

→ void acresceAtributoHistorico(string atributo, string valor) - acresce o valor histórico de um atributo

→ void defineAtributoEstatico(string atributo, string valor) - define o valor de um atributo estático

→ void incrementaAtributo(string atributo, int valor) - incrementa o valor de um atributo

→ XMLTree obtemAtributosEstatisticos(Node nodo) - obtém os atributos estatísticos que são atualizados durante o tempo de execução do nodo

→ void defineTempoMaximoProcessamento(Tarefa tarefa, int tempo) - define o tempo máximo de execução da tarefa

→ int obtemTempoMaximoProcessamento(Tarefa tarefa) - retorna o tempo máximo permitido para execução da tarefa

→ void definePrioridadeTarefa(Tarefa tarefa, int prioridade) - define a prioridade para execução da tarefa

→ int obtemPrioridadeTarefa(Tarefa tarefa) - retorna a prioridade definida para a execução da tarefa

→ long obtemTempoTransmissao(Tarefa tarefa) - obtém o tempo gasto para transmissão da tarefa

VII - **Métodos para descoberta e reconfiguração:** são métodos necessários para permitir que a arquitetura seja capaz de trabalhar com o dinâmismo existente no ambiente
→ void descobreServico(string servico) - solicita a criação de uma tarefa de descoberta de serviço

→ void recebeRespostaDescoberta(XMLTree resposta) - processa a resposta obtida pela execução da tarefa de descoberta

→ void enviaTabelaRoteamento(Node nodo) - envia a tabela de roteamento local para um nodo

→ void recebeTabelaRoteamento(XMLTree tabela) - recebe a tabela de roteamento de um nodo remoto efetuando os ajustes necessários na tabela de roteamento local

VIII - **Métodos para manipulação de sensores:** métodos para interação com sensores
→ void acresceSensor(Node nodo, Sensor sensor) - acresce sensor a determinado nodo

→ void removeSensor(Node nodo, Sensor sensor) - remove sensor de determinado nodo

→ XMLTree coletaDadosSensor(Node nodo) - efetua a coleta dos dados dos sensores associados a um nodo

→ boolean existeSensor(sring sensorNome) - verifica se existe determinado sensor em um nodo

IX - **Métodos para manipulação de serviços:** métodos para definição e execução de serviços
→ void acresceServico(Node nodo, Servico servico) - acresce um servico a um nodo

→ void removeServico(Node nodo, Servico servico) - remove um servico de um nodo

→ ResultadoTarefa executaServico(Node nodo, Tarefa tarefa) - executa um serviço definido em uma tarefa em determinado nodo

→ boolean possuiServico)(string servicoName) - verifica se um nodo possui determinado servico.

X - **Métodos para manipulação de ganchos:** permitem a flexibilidade para criação de novas aplicações que podem então serem "penduradas" nos ganchos
→ void acresceGancho(string contexto, Gancho gancho) - acresce um gancho a determinado contexto

→ void removeGancho(string contexto, Gancho gancho) - remove gancho de determinado contexto

→ void executaGancho(string contexto) - executa o gancho associado a determinado contexto

→ Gancho possuiGancho(string contexto) - verifica se existe um gancho para determinado contexto

XI - **Métodos para apresentação de resultados:** permitem flexibilidade na apresentação dos resultados obtidos pela execução de tarefas
→ String transformaResultado(string fileTemplate, ResultadoTarefa resultado) - efetua a transformação do resultado da tarefa usando como base um arquivo de template

Apêndice B

Atributos de qualidade de serviço

.

Os atributos de qualidade de serviço que são controlados pela plataforma e disponibilizados para uso são os seguintes:

- Módulo Networker

  ➢ **totalTaskSent:** total de tarefas enviadas

  ➢ **totalSendingBytesSent:** total de bytes enviados pelas tarefas

  ➢ **totalSendingBytesReceived:** total de bytes recebidos pelo envio de tarefas

  ➢ **totalSendingTimeTransmission:** tempo total gasto com a transmissão de tarefas.

  ➢ **historicSendingTimeTransmission:** histórico contendo os dez últimos tempos gastos com envio de tarefa.

  ➢ **numberTaskSendNoRoute:** contém o total de tarefas enviadas para endereços que não possuem rota definida

  ➢ **numberTaskSendException:** número de excessões geradas no envio de tarefas

  ➢ **historicTaskSendException:** histórico contendo os dez últimos tempos em que cada excessão originadas por uma tarefa enviada aconteceu.

➢ **historicTaskSendProcessingTime:** histórico contendo os últimos dez tempos gastos em processamento de tarefas enviadas.

➢ **totalTaskSendProcessingTime:** tempo total gasto no processamento de tarefas enviadas.

➢ **numberTaskReceivedException:** número de excessões que aconteceram devido a tarefas recebidas.

➢ **historicTaskReceivedException:** histórico contendo os dez últimos tempos em que cada excessão originadas por uma tarefa recebida aconteceu.

➢ **historicTaskReceivedProcessingTime:** histórico contendo os últimos dez tempos gastos em processamento de tarefas recebidas.

➢ **totalTaskReceivedProcessingTime:** tempo total gasto em processamento de tarefas recebidas.

➢ **totalTaskReceived:** número total de tarefas recebidas.

➢ **totalReceivingBytesSent:** total de bytes enviados devido ao envio de tarefas

➢ **totalReceivingBytesReceived:** total de bytes enviados devido ao recebimento de tarefas

➢ **totalReceivingTimeTransmission:** tempo total gasto com o recebimento de tarefas

➢ **historicReceivingTimeTransmission:** gasto no recebimento de resultados de tarefas.

➢ **numberTaskResultReceived:** número total de resultados de tarefas recebidos.

➢ **totalReceivingBytesReceived:** número de bytes recebidos pelo resultado de tarefas.

➢ **totalReceivingTimeTransmission:** tempo total gasto no recebimento de resultados de tarefas.

➢ **historicReceivingTimeTransmission:** histórico contendo os últimos dez tempos gastos no recebimento de resultados de tarefas.

- Módulo Node

  ➢ **historicServiceProcessingTime:** histórico contendo os últimos dez tempos gastos em processamento de serviços.

  ➢ **totalServiceProcessingTime:** total de tempo gasto em processamento de serviço.

  ➢ **totalServiceNotUnderstood:** total de serviços que não foram compreendidos pelo nodo.

  ➢ **totalServiceSuccess:** total de serviços executados com sucesso.

- Módulo Sensor

  ➢ **numberExecutionSuccess:** número de vezes que o sensor foi executado com sucesso.

  ➢ **numberExecutionException:** número de vezes que o sensor gerou uma excessão.

  ➢ **historicExecutionException:** histórico contendo os últimos dez tempos em que o sensor gerou uma excessão durante execução.

  ➢ **historicProcessingTime:** histórico contendo os últimos dez tempos gastos em processamento pelo sensor.

  ➢ **totalProcessingTime:** tempo total de processamento do sensor.

- Módulo Service

  ➢ **numberExecutionSuccess:** número de vezes que o serviço foi executado com sucesso.

  ➢ **numberExecutionException:** número de vezes que o serviço gerou uma excessão.

  ➢ **historicExecutionException:** histórico contendo os últimos dez tempos em que o serviço gerou uma excessão durante execução.

  ➢ **historicProcessingTime:** histórico contendo os últimos dez tempos gastos em processamento pelo serviço.

➢ **totalProcessingTime:** tempo total de processamento do serviço.

- Módulo de Ganchos (Hook)

  ➢ **numberExecutionSuccess:** número de vezes que o gancho (hook) foi executado com sucesso.

  ➢ **numberExecutionException:** número de vezes que o gancho (hook) gerou uma excessão.

  ➢ **historicExecutionException:** histórico contendo os últimos dez tempos em que o gancho (hook) gerou uma excessão durante execução.

  ➢ **historicProcessingTime:** histórico contendo os últimos dez tempos gastos em processamento pelo gancho (hook).

  ➢ **totalProcessingTime:** tempo total de processamento do gancho (hook).

Apêndice C

Exemplo de uso da API

O código apresentado a seguir têm por finalidade ilustrar o uso da API. Nele, são criados dois nodos, sendo que um deles possui um serviço chamado multiply que recebe dois números, multiplica-os e retorna o resultado:

```
1     // Set node-1
2     Node node1 = new Node("node-1", "http://localhost:8001");
3     node1.setLogLevel(5);
4     NetInterface httpNetInterface = new HTTPServerInterface(node1, 8001);
5     httpNetInterface.setSnoopMode(true);
6     node1.setNetInterface(httpNetInterface);
7     node1.start();
8
9     // Set node-2
10    Node node2 = new Node("node-2", "http://localhost:8002");
11    node2.addService("multiply", new MathService());
12    node2.setLogLevel(5);
13    httpNetInterface = new HTTPServerInterface(node2, 8002);
14    httpNetInterface.setSnoopMode(true);
15    node2.setNetInterface(httpNetInterface);
16    node2.start();
17
18    // Create route table
19    node1.addRoute("node-2", "http://localhost:8002");
20    node2.addRoute("node-1", "http://localhost:8001");
21
22
23    // Create Task
24    Task task = new Task("node-2", "multiply", null);
25
26    // Set parameters
27    task.addParameters(new XMLTree("value", "" + 2));
28    task.addParameters(new XMLTree("value", "" + 3));
29
```

```
30      // Send task
31      TaskResult result = node1.sendServiceRequest(task);
32
33      // Get result
34      System.out.println("Result=> " +
35                          result.getParameters().getText("result") + " <==");
36
```

A linha 2 cria um nodo chamado node-1 tendo como endereço http://localhost porta 8001. Na linha 3 o nível de log do nodo é setado para 5 (nível máximo de log). A linha 4 cria a interface de comunicação que será usada pelo nodo. Na linha 5 é setado o modo "snoop" da inteface. Esse modo faz com que todas as requisições que chegarem e saírem pela interface sejam mostradas na tela sendo útil para debug da aplicação. Na linha 6 essa interface é anexada do nodo. Finalmente a linha 7 faz com que o nodo seja inicializado.

O segundo nodo é criado na linha 10 com o nome de node-2 e tendo como endereço http://localhost port 8002. Na linha 11 é instânciada a classe MathService e associado o serviço Multiply ao nodo. As demais linhas seguem a mesma lógica usada no node-1.

As linhas 19 e 20 são usadas para criar o roteamento entre os dois nodos. A linha 24 cria uma tarefa invocando o serviço multiply tendo como destino o node-2 (que possui o serviço de multiply). Nas linhas 28 e 29 são setados os parâmetros dessa tarefa. Na linha 31 o node-1 submete a tarefa e seu resultado é obtido na linha 34.

O resultado da execução desse código é mostrado abaixo (a saída é um tanto extensa devido ao nível de log estar setado para o máximo).

```
1  [00000300] 5 node-1 Note HTTPServer http-server addServlet(1) address=/
2  [00000300] 5 node-1 Note HTTPServer http-server addServlet(1) address=/request
3  [00000300] 5 node-1 Note HTTPServer http-server addServlet(1) address=/response
4  [00000040] 5 node-2 Note HTTPServer http-server addServlet(1) address=/
5  [00000040] 5 node-2 Note HTTPServer http-server addServlet(1) address=/request
6  [00000040] 5 node-2 Note HTTPServer http-server addServlet(1) address=/response
7  [00000370] 5 node-1 Note RouteFound networker findRoute(1) host=node-2:url=http://loca
8  [00000901] 5 node-1 Note HTTPSending http-server httpClient(1) url=http://localhost:80
9
10 node-1<== HTTP CLIENT SENDING REQUEST (to http://localhost:8002/request)
11 POST /request http
12
13 <task>
```

```
14 <service>multiply</service>
15 <originator>node-1</originator>
16 <destination>node-2</destination>
17 <taskid>task-1000</taskid>
18 <timestamp>1171835000631</timestamp>
19 <priority>2</priority>
20 <parameters>
21 <value>2</value>
22 <value>3</value>
23 </parameters>
24 </task>
25 [00002013] 5 node-2 Note http-servlet HTTPServletReceived doPost(2) task=Task(task-100
26 [00003425] 5 node-2 Note http-servlet HTTPServletResponding doPost(4) task=Task(task-1
27 [00004386] 5 node-1 Note HTTPReceiving http-server httpClient(2) result=200:message=OK
28 [00004406] 5 node-1 Note HTTPSending http-server sendTask(4) done:taskResult for taskt
29
30 ==> HTTP CLIENT node-1 RECEIVED RESPONSE (for http://localhost:8002/request)
31 200 OK
32
33 <task-result>
34 <result-code>OK</result-code>
35 <originator>node-2</originator>
36 <destination>node-1</destination>
37 <taskid>task-1000</taskid>
38 <timestamp>1171835003996</timestamp>
39 <parameters>
40 <result>6</result>
41 </parameters>
42 </task-result>
43
44 Result=> 6 <==
```

Nele pode-se perceber que o node-1 envia a tarefa ao node-2 (linha 10). O node-2 então executa a tarefa (linha 25) e retorna a resposta ao node-1 (linha 26). O número 6 que é o resultado da multiplicação de 2 e 3 é retornado ao node-1 (linha 30). Esse resultado é então apresentado na tela (linha 44).

Referências Bibliográficas

[AG 06] ABU-GHAZALEH, N.; LEWIS, M. J. Short paper: Toward self organizing grids. **HPDC-15: The 15th IEEE International Symposium on High Performance Distributed Computing (Hot Topics Session)**, [S.l.], v.Paris, France, June, 2006.

[AKY 02] AKYILDIZ, I. et al. **A survey on sensor networks**. IEEE Commun. Mag. 40 (8) (2002) 102–114.

[ALM 99] ALMOND, J.; SNELLING, D. Unicore: uniform access to supercomputing as an element of electronic commerce. **Future Gener. Comput. Syst.**, Amsterdam, The Netherlands, The Netherlands, v.15, n.5-6, p.539–548, 1999.

[ASS 04] ASSUNçãO, M. D.; KOCH, F. L.; WESTPHALL, C. B. Grids of agents for computer and telecommunication network management: Research articles. **Concurr. Comput. : Pract. Exper.**, Chichester, UK, UK, v.16, n.5, p.413–424, 2004.

[BEC 06] BECKSTEIN, C.; DITTRICH, P.; ERFURTH, C. Sogos - a distributed meta level architecture for the self-organizing grid of services. In: MDM'06: PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON MOBILE DATA MANAGEMENT, 2006. **Proceedings...** Washington, DC, USA: EEE Computer Society, 2006.

[BER 03a] BERMAN, F.; FOX, G.; HEY, A. J. G. **Grid Computing: Making the Global Infrastructure a Reality**. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[BER 03b] BERTI, G. et al. Medical simulation services via the grid. In: IPDPS 2002:MEDICAL SIMULATION SERVICES VIA THE GRID. IN 17TH INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM, 2003. **Proceedings...** Nice, France, April 2003. IEEE Computer Society: [s.n.], 2003.

[BHA 05] BHATTI, R. et al. **Distributed Access Management in Multimedia IDCs**. IEEE Computer Magazine, vol. 38, no. 9, pp. 60-69, Sept., 2005.

[BUY 07] BUYYA, R. **GridBus (GRID computing and BUSiness technologies)**. Disponível em <http://www.gridbus.org/>. Acesso em: Mar 2007.

[CAS 07] CASE, J. et al. **A Simple Network Management Protocol (SNMP) - RFC 1157**. Disponível em <http://www.ietf.org/rfc/rfc1157.txt>. Acesso em: Mar 2007.

[CHE 03] CHERUKURI, S.; VENKATASUBRAMANIAN, K. K.; GUPTA, S. K. S. Biosec: A biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. In: ICPPW'03: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING WORKSHOPS, 2003. **Proceedings...** : [s.n.], 2003.

[CHU 04] CHU, D. C.; HUMPHREY, M. Mobile ogsi.net: Grid computing on mobile devices. In: Buyya, R., editor, FIFTH IEEE/ACM INTERNATIONAL WORKSHOP ON GRID COMPUTING, 2004. **Proceedings...** : IEEE Computer Society, 2004. p.182–191.

[COU 04] COULSON, G. et al. Towards a component-based middleware framework for configurable and reconfigurable grid computing. In: WETICE '04: PROCEEDINGS OF THE 13TH IEEE INTERNATIONAL WORKSHOPS ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES (WETICE'04), 2004. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2004. p.291–296.

[CUR 02] CURCIN, V. et al. Discovery net: Towards a grid of knowledge discovery. In: KDD 2002: THE EIGHTH ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2002. **Proceedings...** Canada, Edmonton, Canada: [s.n.], 2002.

[DAV 02] DAVE, G. A. Gridlab: Enabling applications on the grid. In: PROCEEDINGS OF GRID COMPUTING. SPRINGER: BERLIN, 2002; 39–45., 2002. **Proceedings...** : [s.n.], 2002.

[dSeS 06] DA SILVA E SILVA, F. J.; RAFAEL FERNANDES LOPES, B. B. D. S.; ANTONIO EDUARDO BERNARDES VIANA, S. A. D. S. Mag, um middleware de grade baseado em agentes: estado atual e perspectivas futuras. In: WCGA 2006: ANAIS DO IV WORKSHOP DE COMPUTAçãO EM GRID E APLICAçõES. SIMPóSIO BRASILEIRO DE REDES DE COMPUTADORES (SBRC 2006), 2006. **Proceedings...** Curitiba: [s.n.], 2006.

[dSL 05] DOS S. LIMA, L. et al. Peer-to-peer resource discovery in mobile grids. In: MGC '05: PROCEEDINGS OF THE 3RD INTERNATIONAL WORKSHOP ON MIDDLEWARE FOR GRID COMPUTING, 2005. **Proceedings...** : [s.n.], 2005.

[fIPA 07a] FOR INTELLIGENT PHYSICAL AGENTS, F. **FIPA ACL Message Representation in XML Specification**. Disponível em <http://www.fipa.org/specs/fipa00071/>. Acesso em: Mar 2007.

[fIPA 07b] FOR INTELLIGENT PHYSICAL AGENTS, F. **FIPA ACL Message Structure Specification**. Disponível em <http://www.fipa.org/specs/fipa00061/>. Acesso em: Mar 2007.

[FOS 97] FOSTER, I.; KESSELMAN, C. Globus: A metacomputing infrastructure toolkit. **Intl J. Supercomputer Applications**, [S.l.], v.11, n.2, p.115–128, 1997.

[FOS 98] FOSTER, I. et al. A computational framework for telemedicine. **Future Generation Computer Systems**, [S.l.], v.14, n.1–2, p.109–123, 1998.

[FOS 02a] FOSTER, I. et al. **The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration**.

[FOS 02b] FOSTER, I. **What Is The Grid? A Three Point Checklist**. GRIDtoday, v. 1, n. 6, July 22, 2002.

[FOS 04] FOSTER, I.; KESSELMAN, C. **The Grid: Blueprint for a New Computing Infrastructure**. 2nd Edition. ed. Elsevier, 2004.

[GC 02] GONZALEZ-CASTANO, F. J. et al. Condor grid computing from mobile handheld devices. **SIGMOBILE Mob. Comput. Commun. Rev.**, [S.l.], v.6, n.2, p.18–27, 2002.

[GHA 04] GHANEM, M. et al. Sensor grids for air pollution monitoring. In: 3RD UK E-SCIENCE ALL HANDS MEETING, 2004. **Proceedings...** Nottingham, UK: [s.n.], 2004.

[GOL 04] GOLDCHLEGER, A. et al. **InteGrade: object-oriented Grid middleware leveraging idle computing power of desktop machines**.

[GRA 03] GRABOWSKI, P.; LEWANDOWSKI, B. **Mobile-enabled grid middleware and/or grid gateways**. GridLab - www.gridlab.org/Resources/Deliverables/D12.2.pdf.

[GRI 99] GRIMSHAW, A. et al. Legion: An operating system for wide-area computing. **IEEE Computer**, [S.l.], v.32, n.(5), 1999.

[GRI 03] GRIMSHAW, A. S. et al. **Grid Computing: Making the Global Infrastructure a Reality**, chapter10, From Legion to Avaki: The Persistence of Vision, p.265–298. John Wiley & Sons, Mar, 2003.

[HED 07] HEDRICK, C. **Routing Information Protocol - RFC 1058**. Disponível em <http://www.ietf.org/rfc/rfc1058.txt>. Acesso em: Mar 2007.

[HEI 01] HEIDEMANN, J. et al. Building efficient wireless sensor networks with low-level naming. In: SOSP '01: PROCEEDINGS OF THE EIGHTEENTH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 2001. **Proceedings...** New York, NY, USA: ACM Press, 2001. p.146–159.

[JAS 05] JASEMIAN, Y.; NIELSEN, L. A. Design and implementation of a telemedicine system using bluetooth protocol and gsm/gprs network, for real time remote patient monitoring. **Technol. Health Care**, Amsterdam, The Netherlands, The Netherlands, v.13, n.3, p.199–219, 2005.

[KLE 96] KLEINROCK, L. Nomadicity: anytime, anywhere in a disconnected world. **Mob. Netw. Appl.**, Hingham, MA, USA, v.1, n.4, p.351–357, 1996.

[KOC 05] KOCH, F. et al. Programming deliberative agents for mobile services: the 3apl-m platform. In: PROMAS'2005: IN PROCEEDINGS OF AAMAS'05 WORKSHOP ON PROGRAMMING MULTI-AGENT SYSTEMS, 2005. **Proceedings...** : [s.n.], 2005.

[KRA 04] KRA, D. Six strategies for grid application enablement, part 1. IBM, April, 2004. Relatório técnico.

[LEG 97] LEGION. **A Worldwide Virtual Computer**. Disponível em <http://legion.virginia.edu>. Acesso em: Mar 2007.

[LIM 05] LIM, H. B. et al. Sensor grid: Integration of wireless sensor networks and the grid. In: LCN '05: PROCEEDINGS OF THE THE IEEE CONFERENCE ON LOCAL COMPUTER NETWORKS 30TH ANNIVERSARY, 2005. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2005. p.91–99.

[LUT 07] LUTHER, A. et al. **Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids**. citeseer.ist.psu.edu/731972.html.

[MAL 02] MALLADI, R.; AGRAWAL, D. P. Current and future applications of mobile and wireless networks. **Commun. ACM**, New York, NY, USA, v.45, n.10, p.144–146, 2002.

[MAR 07] MARTIN, B.; JANO, B. **WAP Binary XML Content Format**. Disponível em <http://www.w3.org/TR/wbxml/>. Acesso em: Mar 2007.

[MCK 04] MCKNIGHT, L. W.; HOWISON, J.; BRADNER, S. Guest editors' introduction: Wireless grids–distributed resource sharing by mobile, nomadic, and fixed devices. **IEEE Internet Computing**, Piscataway, NJ, USA, v.8, n.4, p.24–31, 2004.

[NAD 07] NADIMINTI, K.; BUYYA, R. **Enterprise Grid computing: State-of-the-Art**. citeseer.ist.psu.edu/nadiminti05enterprise.html.

[NAV 06] NAVARRO, F. P. **Um middleware para grades de dispositivos móveis**. UFSC - Universidade Federal de Santa Catarina, 2006. Dissertação de Mestrado.

[NG 04] NG, J. W. et al. Ubiquitous monitoring environment for wearable and implantable sensors (ubimon). In: UBICOMP'2004: INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, 2004. **Proceedings...** : [s.n.], 2004.

[OBR 02] OBRENOVIC, Z. et al. An agent based framework for virtual medical devices. In: AAMAS '02: PROCEEDINGS OF THE FIRST INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 2002. **Proceedings...** New York, NY, USA: ACM Press, 2002. p.659–660.

[ORA 07] ORACLE. **Oracle Berkeley DB Java Edition**. Disponível em <http://www.oracle.com/database/berkeley-db/je/index.html>. Acesso em: Mar 2007.

[PHA 02] PHAN, T.; HUANG, L.; DULAN, C. ”challenge: Integrating mobile wireless devices into the computational grid. In: MOBICOM’2002: PROC. OF THE 8TH ACM INTL. CONF. ON MOBILE COMPUTING AND NETWORKING, 2002. **Proceedings...** : [s.n.], 2002.

[PIO 07] PIOTRGRABOWSKI; BARTEKLEWANDOWSKI. **GridLab - A Grid Application Toolkit and Testbed - Access for Mobile Users**. Disponível em <http://www.gridlab.org/WorkPackages/wp-12/>. Acesso em: Mar 2007.

[SAC 04] SACRAMENTO, V. et al. Moca: A middleware for developing collaborative applications for mobile users. **IEEE Distributed Systems Online**, Piscataway, NJ, USA, v.5, n.10, p.2, 2004.

[SM 07a] SUN MICROSYSTEMS, I. **Java 2 Micro Edition**. Disponível em <http://java.sun.com/javame/index.jsp>. Acesso em: Mar 2007.

[SM 07b] SUN MICROSYSTEMS, I. **Java 2 Standard Edition**. Disponível em <http://java.sun.com/javase/>. Acesso em: Mar 2007.

[STA 07] STANKOVIC, J. et al. **A Network Virtual Machine for Real-Time Coordination Services**. Disponível em <http://www.cs.virginia.edu/ control/>. Acesso em: Mar 2007.

[THA 05] THAM, C.-K.; BUYYA, R. Sensorgrid:integrating sensor networks and grid computing. **Special Issue on Grid Computing**, [S.l.], v., July, 2005.

[TIL 06] TILAK, S. et al. The case for mobile devices in environmental observing systems. **World-Sensor-Web (WSW’2006)**, [S.l.], v.Boulder, Colorado,, October 31 - November 3, 2006.

[USK 03] USKELA, S. et al. **Key technologies and concepts for beyond-3G networks**. IEEE Wireless Communication Magazine, Feb. 2003, Volume:10, Issue:1, On page(s): 43- 48.

[YUJ 05] YUJIE, Y.; SHU, W.; HAO, Z. Mpas: a connecting platform for integrating wireless sensor network with grid. In: PROCEEDINGS OF ASIA-PACIFIC CONFERENCE ON COMMUNICATIONS, 2005. **Proceedings...** Perth, Australia: [s.n.], 2005.