UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Diego Souza Gomes

Acesso Móvel aos Serviços do Middleware InteGrade

São Luís
2009

Diego Souza Gomes

Acesso Móvel aos Serviços do Middleware InteGrade

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão, como requisito parcial para a obtenção do grau de MESTRE em Engenharia de Eletricidade.

Orientador: Francisco José da Silva e Silva

Doutor em Ciência da Computação – UFMA

São Luís

2009

Gomes, Diego Souza

Acesso Móvel aos Serviços do Middleware InteGrade / Diego Souza Gomes. – São Luís, 2009.

107 f.

Orientador: Prof. Dr. Francisco José da Silva e Silva.

Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia de Eletricidade, Universidade Federal do Maranhão, 2009.

1. Grades de computadores. 2. Sistemas distribuídos. 3. Computação móvel. 4. Middleware InteGrade. I. Título.

CDU 004.75

ACESSO MÓVEL AOS SERVIÇOS DO MIDDLEWARE INTEGRADE

Diego Souza Gomes

Dissertação aprovada em 07 de dezembro de 2009.

Prof. Francisco José da Silva e Silva, Dr.
(Orientador)

Prof. Markus Endler, Dr.
(Membro da Banca Examinadora)

Prof. Mário Antonio Meireles Teixeira, Dr.
(Membro da Banca Examinadora)

Aos meus pais, irmãos e familiares.

Agradecimentos

Aos meus pais Josué e Maria do Carmo, pelo apoio e carinho, me desculpem pelos muitos momentos de ausência durante o tempo em que me dediquei a este projeto.

Aos meus familiares, que sempre torceram por mim.

Ao professor Francisco pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos amigos do Laboratório de Sistemas Distribuídos: Gilberto, Vandecia, Eduardo Devidson, Domingos, Jaciara, Jesseildo, Igor, Paulo, Marcondes, pela ajuda nos momentos de trabalho e distrações no dia-a-dia do laboratório.

Um agradecimento especial a Eduardo Viana, Pablo Durans, Vinicius Araújo, Danilo Lauande e Regilaine Souza que se envolveram e muito contribuíram para o prosseguimento deste projeto.

Aos companheiros de trabalho da DGTI-IFMA, Leonardo, Ronaldson, José Maria, Ronaldo, Jackson, Jefferson, Fábio, Daniel, Eliberto, Renato, Euclides, Dona Rosa, e em particular ao professor Cláudio Fernandes pelo incentivo dado para prosseguir no desenvolvimento deste Mestrado.

Aos professores Markus Endler e Mário Meireles que gentilmente aceitaram compor esta banca.

A toda equipe de desenvolvimento do InteGrade, MoCA e SNU, utilizados no desenvolvimento deste trabalho.

A todos que de alguma forma contribuíram para a realização deste trabalho.

"Existe uma teoria que diz que, se um dia alguém descobrir exatamente para que serve o Universo e por que ele está aqui, ele desaparecerá instantaneamente e será substituído por algo ainda mais estranho e inexplicável."

"Existe uma segunda teoria que diz que isso já aconteceu."

Douglas Adams

Resumo

Tecnologias de computação móvel e de redes sem fio têm evoluído muito rapidamente, de forma que muitos dos dispositivos portáteis possuem atualmente considerável capacidade de processamento, armazenamento e comunicação. Paralelo a isso, a tecnologia de grades computacionais se consolidou como um ferramental importante para o trabalho colaborativo entre usuários e organizações, através do compartilhamento de recursos e serviços computacionais entre múltiplos domínios administrativos. Devido a esta popularização da computação móvel, usuários de dispositivos portáteis formam um importante e novo segmento da computação em grade, assumindo tanto o papel de usuários como também de provedores de recursos e serviços. A integração dessas duas categorias de sistemas distribuídos, objetiva estender as capacidades dos dispositivos computacionais móveis através do acesso a uma infra-estrutura de recursos compartilhados, além de fornecer aos usuários de serviços de grade meios mais rápidos e fáceis de acesso as informações produzidas por estes sistemas em qualquer hora e lugar.

Este trabalho descreve o MInteGrade (*Mobile InteGrade*), uma infra-estrutura de *software* para acesso aos serviços do *middleware* de grade InteGrade a partir de dispositivos móveis conectados através de redes sem fio IEEE 802.11 em modo infra-estruturado e redes *bluetooth* em modo Ad hoc. Através deste mecanismo para acesso a uma grade de computadores, clientes móveis podem solicitar a execução de aplicações na grade, realizar o acompanhamento da execução das aplicações e visualizar o resultado de computações já concluídas. O MInteGrade foi projetado para levar em consideração o dinamismo das redes sem fio, provendo suporte a períodos de desconexão e variações na topologia das redes Ad hoc, bem como a adaptação de conteúdo dos resultados das computações realizadas pela grade.

Palavras-chaves: Grades de computadores, Grades sem fio, Computação móvel, Redes Ad hoc.

Abstract

Mobile computing technologies and wireless networks have evolved very quickly. Nowadays, many of the portable devices have significant processing power, storage and communication capacity. At the same time, grid computing technology became an important tool for collaborative work among users and organizations enabling the sharing of computing resources and services through multiple administrative domains. Due to the popularity of mobile computing, users of handheld devices form an important and new segment of the computing grid, assuming both the role of consumers and providers of resources and services. The integration of these two categories of distributed systems aims to extend the capabilities of portable devices, enabling access to a infrastructure of shared resources, as well as provide faster and easier means for grid services users to access information produced by these systems at any time and place.

This work describes the MInteGrade (Mobile InteGrade), a software infrastructure that provides access to the InteGrade grid middleware from mobile devices connected through IEEE 802.11 wireless networks in infrastructure mode, and Bluetooth networks in Ad hoc mode. Through this mechanism for access to a computational grid, mobile clients can request the execution of applications on the grid, monitor the execution of applications and view the computation results already completed. The MInteGrade was designed so as to consider the dynamics of wireless networks. Specifically, it supports disconnections of devices and dynamic changes of the Ad hoc network topology, as well as content adaptation of the output of the grid computations.

Key-words: Grid computing, Wireless Grids, Mobile computing, Ad hoc networks.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Dispositivos computacionais móveis estão hoje disponíveis nas mais variadas formas como, por exemplo, *laptops*, *tablet PCs*, PDAs (*Personal Digital Assistants*) e *smartphones*, entre outros. Tecnologias de computação móvel e de redes sem fio têm evoluído muito rapidamente, de forma que muitos destes dispositivos possuem hoje considerável capacidade de processamento, armazenamento e comunicação. As diversas tecnologias de rede sem fio (*wireless networks*) que existem atualmente permitem que seus usuários possam ter acesso a dados corporativos, pessoais e conteúdos da Internet de modo conveniente em qualquer lugar e a qualquer hora. Todas estas funcionalidades têm tornado estes dispositivos cada vez mais populares e utilizados por diversos grupos de pessoas para os mais variados fins.

Devido a esta popularização da computação móvel, usuários de dispositivos portáteis formam um importante e novo segmento da computação em grade. A computação em grade envolve a agregação de computadores conectados em rede para formar um sistema distribuído de larga escala que pode ser utilizado para realizar as mais variadas computações. Através da divisão da carga de trabalho sobre uma grande quantidade de computadores, uma grade pode disponibilizar um enorme poder computacional, de armazenamento e de transferência de dados, além de outros recursos compartilháveis. Um ambiente de grades de computadores pode ainda constituir um ferramental tecnológico importante para o trabalho colaborativo entre usuários e organizações, dado que ambientes de grades computacionais habilitam o compartilhamento de recursos e serviços computacionais entre múltiplos domínios administrativos.

Existem três abordagens principais para a integração de dispositivos computacionais móveis à ambientes de grades de computadores. Na primeira abordagem, dispositivos móveis participam como clientes da grade [6, 23]. Neste caso, a grade serve como uma extensão dos recursos computacionais dos dispositivos móveis, permitindo que seus usuários requisitem serviços mesmo enquanto se deslocam. A infraestrutura de *software* que disponibiliza o acesso aos serviços da grade por usuários móveis deve resolver questões como a extensão do modelo que trata a heterogeneidade

dos recursos que compõem a infra-estrutura de grade. Este modelo passa a ter que levar em consideração características típicas dos dispositivos portáteis, como UCPs lentas, telas com diferentes capacidades em termos de resolução e definição de cores, variados mecanismos de interação com o usuário e limitações de memória volátil e não-volátil. O *middleware* de grade também deve levar em consideração que tecnologias de comunicação sem fio apresentam menor largura de banda, maior taxa de erro, altas variações na qualidade da comunicação a medida que o usuário se desloca, além de eventualmente ocorrerem períodos de desconexões ou conectividade intermitente [37]. Finalmente, um esquema de mapeamento deve decidir como diferentes componentes de uma aplicação e da infra-estrutura da grade devem ser mapeados: no dispositivo móvel (*front-end*) ou em nós fixos da grade (*back-end*). Este mapeamento deve levar em consideração aspectos como capacidade de processamento, consumo de energia (devido a limitações das baterias que equipam dispositivos móveis), requisitos de recursos, entre outros [24].

Em uma segunda abordagem, dispositivos móveis funcionam como provedores (nós de processamento) no contexto de uma grade fixa pré-existente [33]. Uma vez que os dispositivos móveis possuem menos recursos que estações de trabalho conectadas à redes fixas, geralmente é necessário uma maior quantidade desses dispositivos para realizar uma dada computação da grade. Tendo isso em vista, um dos desafios do *middleware* de grade nesta abordagem é o desenvolvimento de mecanismos que permitam a paralelização automática das aplicações em tarefas a serem escalonadas em dispositivos móveis heterogêneos. Um outro importante aspecto a ser levado em consideração é que dispositivos computacionais móveis podem repentinamente desaparecer da rede devido a desconexões ou falta de energia. Portanto, mecanismos de tolerância a falhas devem ser implementados de modo a garantir que tarefas já realizadas nesses dispositivos não sejam perdidas e comprometam a execução das aplicações.

Na terceira abordagem, dispositivos móveis formam, de maneira espontânea, grades puramente sem fio: as chamadas "grades móveis Ad hoc" [29, 26]. As redes sem fio Ad hoc (*Mobile Ad hoc NETworks* - MANETs), são redes caracterizadas pela total ausência de infra-estrutura, onde todos os dispositivos participantes são móveis e a topologia da rede é formada temporariamente, de forma dinâmica e independente, com os dispositivos podendo entrar ou sair da rede a qualquer momento.

Esta dinamicidade das redes Ad hoc implica no fato de que não se pode contar com nenhum tipo de controle centralizado nestas redes, devendo-se prover assim, maneiras mais sofisticadas para a descoberta e seleção de seus recursos e serviços [27]. O fato dos nós de uma rede Ad hoc moverem-se arbitrariamente leva a mudanças frequentes e imprevisíveis na topologia da rede, desta forma as grade móveis Ad hoc também devem estar preparadas para lidar com as altas variações tanto na disponibilidade de recursos quanto na demanda pelos mesmos.

1.1 Objetivos

Este trabalho tem por objetivo geral investigar a integração de dispositivos computacionais móveis a ambientes de grades de computadores no contexto do *middleware* InteGrade [17], uma infra-estrutura de *software* para o estabelecimento de grades computacionais, visando o desenvolvimento de mecanismos que permitam o acesso por usuários móveis aos serviços disponibilizados pelo mesmo.

Para este trabalho, foram definidos os seguintes objetivos específicos:

- Avaliar o estado da arte em grades computacionais e do acesso aos seus serviços por dispositivos móveis;
- Definição dos requisitos de projeto e arquitetura da infra-estrutura para acesso ao *middleware* InteGrade através de dispositivos móveis conectados à redes sem fio em modo infra-estruturado e em modo Ad hoc;
- Implementação, testes e avaliação desta arquitetura.

1.2 Estrutura da Dissertação

Esta dissertação está organizada da seguinte forma: o Capítulo 2 apresenta os principais conceitos relacionados à tecnologia de grades de computadores, uma taxonomia deste tipo de sistema e as classes de aplicações usualmente utilizadas. O capítulo também fornece uma introdução ao *middleware* InteGrade, descrevendo seus principais serviços.

O Capítulo 3 descreve os resultados obtidos em uma primeira etapa deste trabalho, na qual foi desenvolvida uma infra-estrutura de *software* para acesso aos serviços do *middleware* InteGrade por dispositivos móveis conectados através de uma rede infra-estruturada IEEE 802.11, denominada Mobile InteGrade (MInteGrade) [18] Através desta infra-estrutura, clientes móveis podem solicitar a execução de aplicações na grade, realizar o acompanhamento da execução das aplicações e visualizar o resultado de computações já concluídas. São apresentados os requisitos de seu projeto, sua arquitetura e detalhes de implementação, que levaram em consideração aspectos relacionados a ciência de contexto, suporte a períodos de desconexão e baixa conectividade bem como adaptação de conteúdo retornado pela grade relativo aos resultados das computações realizadas.

O Capítulo 4 apresenta os avanços decorrentes da segunda etapa de desenvolvimento deste trabalho, na qual foram desenvolvidos mecanismos que permitem o compartilhamento por usuários portadores de dispositivos móveis conectados em uma rede Ad hoc do acesso aos serviços da grade provido pelo MInteGrade. São apresentados os requisitos de seu projeto, sua arquitetura e detalhes de implementação, que levaram em consideração a topologia variável das redes Ad hoc, nas quais dispositivos podem entrar e sair da rede a qualquer momento e sem prévio aviso, passar por períodos de desconexão ou de baixa conectividade ou até mesmo falharem.

O Capítulo 5 descreve os testes funcionais utilizados na avaliação dos componentes de *software* desenvolvidos, bem como as simulações realizadas para avaliação da arquitetura proposta que levaram em consideração cenários de larga escala e padrões de mobilidade dos usuários. Finalmente, o Capítulo 6 apresenta as conclusões obtidas a partir deste trabalho e apresenta possíveis trabalhos futuros que podem ser desenvolvidos a partir deste esforço inicial.

2 Introdução a Grades de Computadores e ao *Middleware* InteGrade

Este capítulo apresenta conceitos gerais relacionados à tecnologia de grades computacionais e suas aplicações. É descrita uma taxonomia para sistemas de grades computacionais e suas principais classes de aplicações. Além disso, ainda neste capítulo são definidos o conceito de grade oportunista e apresentados os componentes, serviços e classes de aplicações do *middleware* InteGrade, o *middleware* de grade oportunista utilizado no desenvolvimento deste trabalho.

2.1 Grades Computacionais

Atualmente, ramos de atividades científicas, comerciais e industriais como biologia, processamento de imagens para diagnóstico médico, previsão do tempo, física de alta energia, previsão de terremotos, simulações mercadológicas, prospecção de petróleo e computação gráfica, têm demandado por grande capacidade de processamento, compartilhamento de dados e colaboração entre usuários e organizações. Porém, alternativas tradicionais para a realização de computação de alto desempenho, como o uso de supercomputadores ou de aglomerados de computadores como *clusters Beowulf*, requerem altos investimentos em *hardware* e *software*, podendo chegar a custar centenas de milhares de dólares.

Por outro lado, redes de computadores existentes em instituições tanto públicas quanto privadas formam hoje um enorme parque computacional interconectado principalmente por tecnologias ligadas à Internet. No entanto, apesar de permitir comunicação e troca de informações entre computadores, as tecnologias que compõem a Internet atual não disponibilizam abordagens integradas que permitam o uso coordenado de recursos pertencentes a várias instituições na realização de computações. Uma nova abordagem, denominada computação em grade (*grid computing*) [14, 7], tem sido desenvolvida com o objetivo de superar esta limitação.

Grade, em um nível conceitual, é um tipo de sistema paralelo e distribuído que possibilita o compartilhamento, seleção, e agregação de recursos autônomos geograficamente distribuídos em tempo de execução, dependendo de sua disponibilidade, capacidade, desempenho, custo e requisitos de qualidade de serviço de usuários [7].

A computação em grade permite a integração e o compartilhamento de computadores e recursos computacionais, como *software*, dados e periféricos, em redes corporativas e entre estas redes, estimulando a cooperação entre usuários e organizações, criando ambientes dinâmicos e multi-institucionais, fornecendo e utilizando os recursos de maneira a atingir objetivos comuns e individuais [2, 15]. A origem do termo *Grid Computing* deriva de uma analogia com a rede elétrica (*Power Grid*) e reflete o objetivo de tornar o uso de recursos computacionais distribuídos tão simples quanto ligar um aparelho na rede elétrica.

O compartilhamento multi-institucional e dinâmico proposto pela computação em grade deve ser, necessariamente, altamente controlado, com provedores de recursos e consumidores definindo claramente e cuidadosamente o que é compartilhado, a quem é permitido compartilhar, e as condições sob as quais ocorre o compartilhamento. Ian Foster et. al [15], conceituam os grupos formados por indivíduos e/ou organizações que compõem esses ambientes dinâmicos e multi-institucionais como Organizações Virtuais (OV).

O *middleware*[1] de grade é o componente de *software* central de um sistema de grade, sendo responsável pela integração dos recursos distribuídos, de modo a criar um ambiente unificado para o compartilhamento de dados, recursos computacionais e execução de aplicações.

2.1.1 Aplicações e Taxonomia de Grades de Computadores

Existem atualmente muitas aplicações para a computação em grade. Ian Foster e Carl Kesselman [14] identificaram as cinco principais classes de aplicações para grades de computadores: supercomputação distribuída, alto rendimento, compu-

[1] middleware é uma camada de software que reside entre o sistema operacional e a aplicação a fim de facilitar o desenvolvimento de aplicações, escondendo do programador diferenças entre plataformas de hardware, sistemas operacionais, bibliotecas de comunicação, protocolos de comunicação, formatação de dados, linguagens de programação e modelos de programação.

tação sob demanda, computação intensiva de dados e computação colaborativa. Estas classes de aplicações são apresentadas na Tabela 2.1.

Categoria	**Características**	**Exemplos**
Supercomputação distribuída	Grandes problemas com intensiva necessidade de CPU, memória, etc.	Simulações interativas distribuídas, cosmologia, modelagem climática
Alto rendimento	Agregar recursos ociosos para aumentar a capacidade de processamento. A grade é utilizada para executar uma grande quantidade de tarefas independentes ou fracamente acopladas.	Projeto de chips, problemas de criptografia, simulações moleculares
Computação sob demanda	Recursos remotos integrados em computações locais, muitas vezes por um período de tempo limitado	Instrumentação médica, processamento de imagens microscópicas
Computação intensiva de dados	Síntese de novas informações de muitas ou grandes fontes de dados	Experimentos de alta energia, modernos sistemas meteorológicos
Computação colaborativa	Suporte à comunicação ou a trabalhos colaborativos entre vários participantes	Educação, projetos colaborativos

Tabela 2.1: Cinco maiores classes de aplicações de grades

Krauter et al. [25] propõem uma taxonomia para sistemas de grades de computadores ilustrada na Figura 2.1, cujos componentes são descritos a seguir.

Figura 2.1: Taxonomia dos sistemas de grade

- Grade Computacional (*Computing Grid*): é um sistema de computação em grade que objetiva prover maior capacidade computacional através da agregação de recursos computacionais distribuídos. Como exemplo de classes de aplicações temos:

  - supercomputação distribuída, cujas aplicações usam grades computacionais para agregar recursos computacionais substanciais para resolver problemas que não poderiam ser resolvidos por um único sistema como, por exemplo, aplicações para planejamento e treinamento militar através de simulações interativas distribuídas e simulação precisa de processos físicos complexos;
  - modelagem climática;
  - aplicações de alto rendimento, onde a grade é usada para escalonar um grande número de tarefas independentes ou fracamente acopladas com o objetivo de utilizar ciclos de processadores ociosos como, por exemplo, a resolução de problemas criptográficos.

- Grade de Serviços (*Service Grid*): disponibiliza serviços viabilizados pela integração de diversos recursos computacionais. Exemplos de classes de aplicações são:
  - Computação sob demanda: aplicações usam as capacidades da grade por períodos curtos de acordo com solicitações por recursos como, por exemplo, aplicações de instrumentação médica e requisições de utilização de *software* especializado por usuários remotos;
  - Computação colaborativa: aplicações que utilizam a grade para dar apoio a trabalhos cooperativos envolvendo diversos participantes, como, por exemplo, em projetos colaborativos e educação;
  - Multimídia: a grade provê a infra-estrutura para aplicações multimídia em tempo real.

- Grade de Dados (*Data Grid*): é um sistema de computação em grade que objetiva prover mecanismos especializados para publicação, descoberta, acesso e classificação de grandes volumes de dados distribuídos. Como exemplo, temos a computação intensiva de dados: cujo foco está na síntese de novas informações de dados que são mantidos em repositórios, bibliotecas digitais e bancos de dados geograficamente distribuídos.

2.2 Grades Oportunistas

Atualmente, as instituições privadas e públicas têm um grande número de recursos de computação, tais como computadores pessoais e estações de trabalho, com

grande capacidade de processamento e armazenamento de dados. Os computadores estão ociosos na maior parte do tempo e, mesmo quando estão em uso, normalmente apenas uma pequena percentagem de sua capacidade de computação é efetivamente utilizada [30, 11, 31]. Grades oportunistas são sistemas computacionais que provêem meios para usar uma base instalada de computadores pessoais para execução de aplicações computacionais de alto desempenho, aproveitando o poder de computação ocioso disponível [17].

O foco de um *middleware* de grade oportunista não é a integração de aglomerados de computadores dedicados (ex: *clusters Beowulf*) ou recursos de supercomputação, mas é aproveitar ciclos de computação ociosa de computadores regulares e estações de trabalho que podem estar espalhados através de diversos domínios administrativos.

Desenvolvedores de um *middleware* de grade oportunista devem tratar diversos desafios tais como:

a) Grande instabilidade, uma vez que nós são geralmente não dedicados e aplicações não executam sobre um ambiente controlado;

b) Alta heterogeneidade de recursos computacionais e de enlaces de rede, uma vez que computadores estão geralmente espalhados através de diferentes domínios administrativos;

c) O *middleware* não deve interferir no uso regular dos recursos computacionais ou, pelo menos, deve prover o mínimo de impacto sobre o desempenho percebido pelos usuários que consentem o uso do recurso;

d) É desejável o fornecimento de mecanismos para prever o período de tempo, em que uma máquina estará ociosa, minimizando a necessidade de migração de código.

2.3 O *Middleware* InteGrade

O projeto InteGrade [17] é o resultado de um esforço multi-universitário com o objetivo de construir uma infra-estrutura de *software* robusta e flexível para computação em grade oportunista.

O InteGrade é estruturado em aglomerados, ou seja, unidades autônomas dentro da grade que contém todos os componentes necessários para funcionar independentemente. Estes aglomerados podem estar organizados de uma forma hierárquica ou conectados através de uma rede ponto-a-ponto (*Peer-to-Peer* ou P2P) [16]. A arquitetura de um aglomerado InteGrade pode ser visto na Figura 2.2.

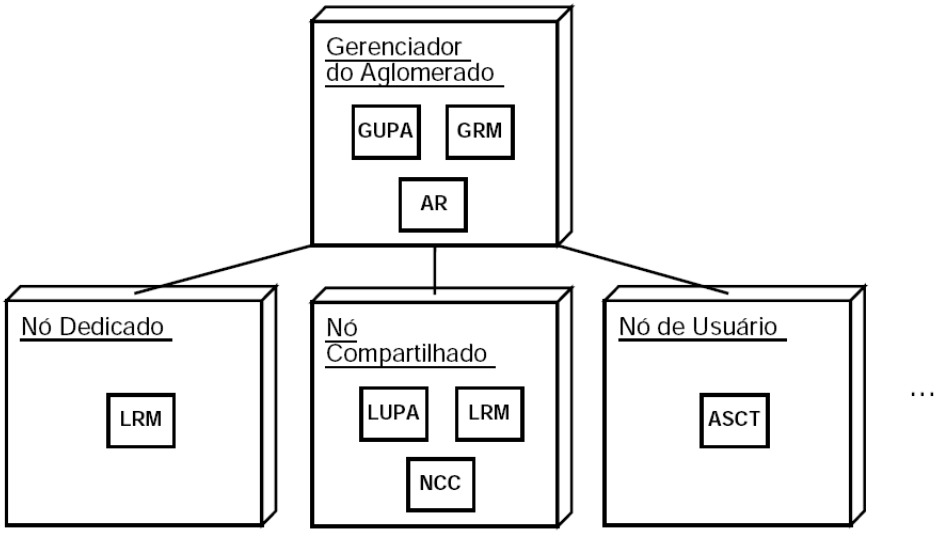

Figura 2.2: Componentes de um aglomerado InteGrade

Cada aglomerado possui um nó "Gerenciador do Aglomerado" que executa componentes do InteGrade responsáveis pela coleta de informações dos recursos computacionais do aglomerado e escalonamento das aplicações nos recursos locais com base nessas informações. No entanto, como os aglomerados do InteGrade são dispostos de forma hierárquica, caso não haja recursos locais suficientes no aglomerado para realizar o escalonamento de uma tarefa, o escalonador pode encaminhar a tarefa para que seja executada em outro aglomerado da hierarquia.

Um "Nó de Usuário" é um nó pertencente a um usuário que submete aplicações para execução na grade. Um "Nó Compartilhado" é tipicamente um PC ou estação de trabalho em um laboratório compartilhado que exporta parte de seus recursos, tornando-os disponíveis aos usuários da grade. Um "Nó Dedicado" é aquele reservado à execução de computações da grade. Estas categorias podem sobrepor-se: um nó pode ser ao mesmo tempo um "Nó de Usuário" e um "Nó Compartilhado". Como também pode ser visto na Figura 2.2, várias entidades de *software* compõem a arquitetura mostrada. Elas executam nos nós componentes do aglomerado de forma a prover as funcionalidades necessárias a cada categoria de nó. Estas entidades serão descritas a seguir:

- *GRM (Global Resource Manager):* gerenciador de recursos do aglomerado. Mantém informações a respeito do estado de disponibilidade dos recursos presentes nas máquinas que compõem o aglomerado, efetuando o escalonamento de tarefas nos nós de acordo com estas informações e efetuando negociação e reserva de recursos. Executa no nó gerenciador do aglomerado;

- *LRM (Local Resource Manager):* é executado em cada nó que fornece recursos ao aglomerado. Coleta informações sobre o estado do nó como memória, CPU, disco e utilização da rede. Os `LRMs` enviam periodicamente informações ao `GRM` de seus aglomerados, através de um "protocolo de atualização de informações";

- *LUPA (Local Usage Pattern Analyser):* executa junto ao `LRM` e coleta localmente informações de padrões de uso dos usuários do nó. Baseia-se em longas séries de dados derivadas de padrões semanais de uso do nó. Periodicamente transmite as informações locais ao `GUPA`;

- *GUPA (Global Usage Pattern Analyser):* gerencia as informações sobre os padrões de uso de recursos dos nós componentes do aglomerado. Fornece estas informações ao `GRM` de forma a colaborar para que ele tome melhores decisões de escalonamento;

- *AR (Application Repository):* armazena as aplicações que podem executar na grade. Toda aplicação submetida à execução deve ser primeiramente registrada neste repositório;

- *NCC (Node Control Center):* permite ao proprietário definir parâmetros sobre as condições de compartilhamento dos recursos de seu nó. É possível determinar horários em que a máquina não irá compartilhar recursos ou qual a percentagem de UCP e memória que ela deseja ceder à grade.

- *ASCT (Application Submission and Control Tool):* ferramenta para submissão de aplicações para execução na grade. Os usuários podem especificar pré-requisitos de execução, como a plataforma de *hardware* e *software* nas quais a aplicação deve ser executada, requisitos de memória necessários à execução da aplicação, entre outros. Também é possível monitorar o progresso da aplicação em execução.

O InteGrade atualmente permite a execução de três classes de aplicações:

1. **Aplicações regulares**, onde o código executável é designado para um único nó da grade;

2. **Aplicações paralelas fracamente acopladas** denominadas paramétricas ou BoT (*Bag-of-tasks*), onde várias cópias do código executável são designadas para nós diferentes da grade. Cada nó recebe um subconjunto dos dados de entrada da aplicação e computa o resultado em paralelo com os demais. Neste caso, não existe comunicação entre os nós;

3. **Aplicações paralelas fortemente acopladas** que seguem o modelo BSP [40] (*Bulk Synchronous Parallelism*) ou MPI, onde vários nós são designados para a execução da aplicação e as computações são realizadas em cada um dos nós participantes mas, neste caso, os processos ocasionalmente trocam dados entre si.

2.4 Serviços Disponibilizados pelo InteGrade

Os dois principais serviços disponibilizados pelo *middleware* InteGrade são a execução de aplicações e o armazenamento distribuído de dados. Nesta seção o protocolo de execução de aplicações do InteGrade e os componentes do serviço distribuído de dados são descritos em mais detalhes.

2.4.1 Execução de Aplicações Distribuídas

O principal serviço disponibilizado pelo InteGrade é o serviço de execução de aplicações. Através desse serviço, um usuário da grade pode submeter aplicações que necessitam de alto poder computacional e/ou trabalham com grande quantidade de dados para execução sobre recursos compartilhados que estão ociosos. A Figura 2.3 ilustra os passos necessários para a execução de uma aplicação no InteGrade. Uma vez que a aplicação tenha sido registrada no Repositório de Aplicações através do `ASCT`, o usuário solicita a execução da aplicação (1). O usuário pode, opcionalmente, especificar requisitos para a execução de sua aplicação, como por exemplo a arquitetura para a qual a aplicação foi compilada, ou a quantidade mínima de memória necessária para a execução. Também é possível especificar preferências, recursos desejáveis para execução da aplicação, como executar em máquinas com a maior quantidade de memória livre.

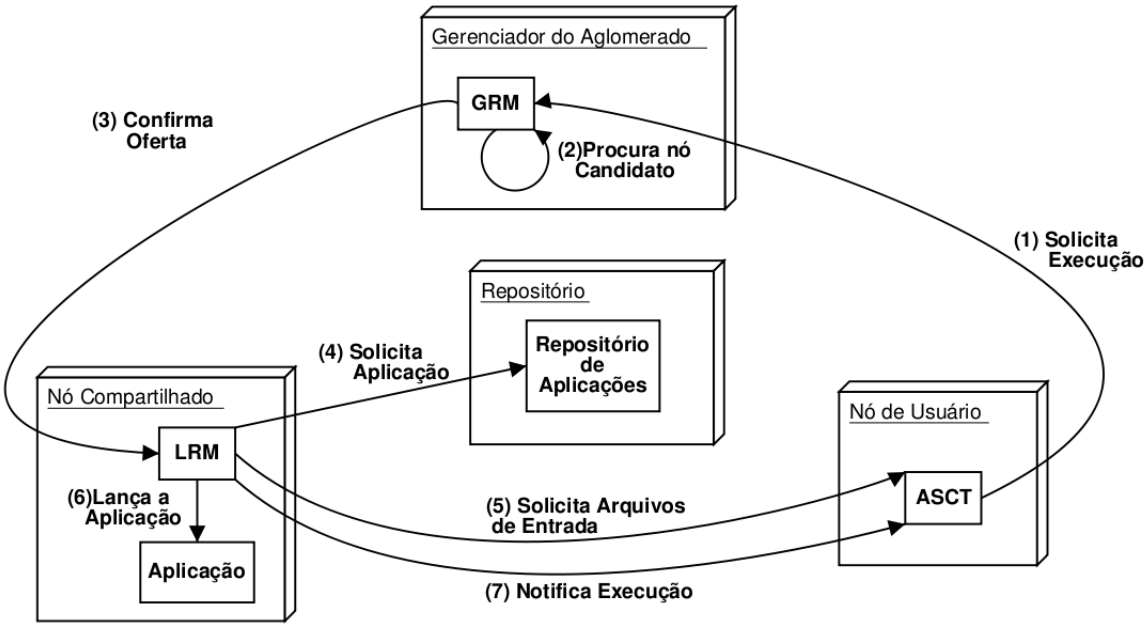

Figura 2.3: Protocolo de execução de aplicações do InteGrade

Assim que a requisição de execução é enviada ao GRM, este procura nós candidatos para executar a aplicação (2), baseado nos requisitos informados pelo usuário no processo de submissão. Aplicações regulares requerem apenas um nó para sua execução, enquanto aplicações paralelas fortemente ou fracamente acopladas necessitarão de uma quantidade de nós correspondente à quantidade de tarefas a serem instanciadas. Caso não haja recursos suficientes que permitam a execução da aplicação, o GRM notifica o ASCT utilizado no processo de submissão que a execução foi rejeitada por falta de recursos. Caso contrário, o GRM envia a solicitação para o LRM de cada máquina escolhida para executar a aplicação (3). O LRM verifica se de fato possui recursos disponíveis para realizar a execução solicitada. Este passo é necessário dado que o GRM mantem somente um visão aproximada da disponibilidade de recursos nos nós do aglomerado, atualizada periodicamente e que pode não corresponder à realidade corrente da máquina. Caso o nó não possua os recursos necessários para a execução da aplicação, o LRM notifica o GRM de tal fato e o GRM retorna ao passo (2), procurando por outro nó candidato. Caso contrário, o LRM baixa do Repositório de Aplicações o código da aplicação a ser executada (4), solicita os eventuais arquivos de entrada da mesma ao ASCT requisitante (5) e lança a aplicação (6) para execução, notificando ao ASCT que sua requisição foi atendida (7). Desta forma, o ASCT descobre em

qual(is) `LRM(s)` sua aplicação está executando, podendo assim, verificar o estado de sua execução, recuperar seus resultados, ou mesmo finalizar a execução da aplicação.

2.4.2 Armazenamento Distribuído de Dados

Além de executar aplicações, o *middleware* de grade necessita gerenciar e armazenar grandes quantidades de dados que podem ser gerados e utilizados por estas aplicações. Tendo em vista esta necessidade, o InteGrade utiliza o *middleware* OppStore [9, 10] para disponibilizar uma infra-estrutura de armazenamento de dados que utiliza o espaço livre em disco de máquinas compartilhadas pertencentes à grade.

No *middleware* OppStore, as máquinas são organizadas em uma federação de aglomerados, onde cada aglomerado é constituído por máquinas fisicamente próximas como, por exemplo, em um mesmo laboratório ou departamento. Cada aglomerado deve conter uma máquina que instancia um módulo de gerenciamento dos repositórios de dados daquele aglomerado denominado `CDRM` (*Cluster Data Repository Manager*). As demais máquinas funcionam como repositórios de dados e instanciam o módulo `ADR` (*Autonomous Data Repository*).

Para realizar o armazenamento de dados, o OppStore codifica arquivos em fragmentos redundantes e disponibiliza para a aplicação cliente dois métodos de armazenamento para os arquivos: (1) armazenamento perene e (2) armazenamento efêmero.

No modo de armazenamento perene, os fragmentos codificados são distribuídos em diversos aglomerados da grade. Esta distribuição melhora a disponibilidade de dados armazenados, uma vez que estes podem ser recuperados mesmo quando ocorrem falhas em um ou mais aglomerados. Além disso, durante a recuperação de arquivos, aplicações podem obter fragmentos localizados nos aglomerados mais próximos, melhorando o desempenho e diminuindo o tráfego de dados na rede.

Já no modo de armazenamento efêmero, os dados são armazenados apenas em máquinas do aglomerado de onde a requisição foi realizada. A vantagem deste modo é que os dados trafegam somente na rede local, gerando uma latência menor para o armazenamento e recuperação dos dados.

A figura 2.4 mostra a distribuição de componentes do InteGrade, como o

GRM, AR, EM e LRM e dos componentes do OppStore implantados sobre o InteGrade.

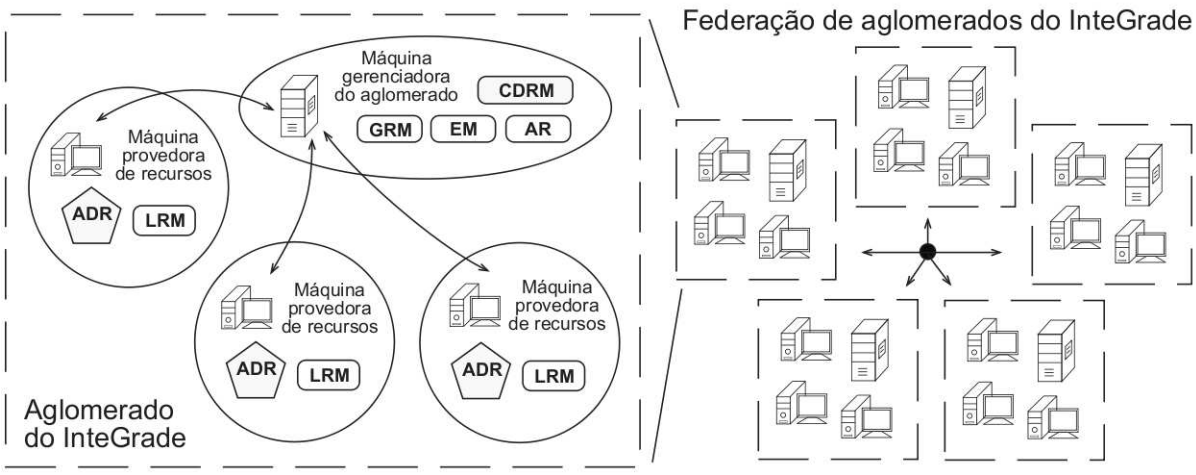

Figura 2.4: Implantação do OppStore sobre o InteGrade

Como pode ser notado na figura 2.4, ADRs são instanciados nas máquinas provedoras de recursos, de modo que as mesmas máquinas que disponibilizam seus ciclos computacionais passam a disponibilizar também o espaço livre em disco. Além disso, CDRMs são instanciados nas máquinas gerenciadoras do aglomerado.

Antes da adoção do OppStore pelo InteGrade, usuários utilizavam a interface gráfica ASCT (*Application Submission and Control Tool*) para armazenar arquivos executáveis de aplicações no módulo AR (*Application Repository*) e submeter tarefas para execução na grade. As tarefas eram repassadas para execução nos LRMs (*Local Resource Managers*), que obtinham os binários da aplicação junto ao AR e os arquivos de entrada a partir do ASCT. Quando a aplicação finalizava, os arquivos de saída das aplicações eram enviados para o ASCT de onde a execução foi requisitada. Este esquema trazia diversas limitações à execução de aplicações. A primeira é que o ASCT precisava ficar conectado ao sistema por todo o processo de submissão, escalonamento, execução e obtenção de resultados, uma vez que o ASCT interage diretamente com o LRM. Além disso, não era permitido ao usuário da grade visualizar os resultados da execução em máquinas diferentes da utilizada para a submissão da execução. Finalmente, não era possível compartilhar dados entre diferentes aplicações em uma grade computacional, uma vez que para cada execução os arquivos de entrada e saída eram obtidos e enviados diretamente ao ASCT. Após a integração do OppStore ao InteGrade, os arquivos de entrada e saída passaram a ser armazenados em repositórios distribuídos. No momento de submeter uma aplicação para execução, os arquivos de entrada

são enviados para armazenamento remoto. Quando o LRM recebe a requisição para execução, ele obtêm os arquivos de entrada da aplicação no OppStore e providencia a execução da mesma. Após seu término, os arquivos de saída também são armazenados no OppStore. Usuários da grade podem, então, obter estes dados armazenados de qualquer máquina utilizando a ferramenta gráfica ASCT. Finalmente, como dados de entrada e saída de uma aplicação ficam armazenados no OppStore, outras aplicações podem facilmente utilizar estes arquivos.

2.5 Conclusões

A tecnologia de grades de computadores recebe hoje grande atenção por parte tanto da academia quanto da indústria por ter se firmado como uma alternativa atraente para a execução de uma grande variedade de aplicações nas mais variadas áreas como biologia computacional, processamento e armazenamento de imagens, previsão do tempo e simulações de mercado.

Este capítulo apresentou um introdução à tecnologia de grades de computadores. Foram descritas as classes de aplicações que fazem uso deste tipo de tecnologia e uma taxonomia que classifica as grades de computadores em grades computacionais, de dados e de serviços.

Uma ênfase especial foi dada as grade oportunistas e ao *middleware* InteGrade, uma infra-estrutura de *software* livre que permite o aproveitamento de recursos ociosos para o estabelecimento de grades de computadores. O *middleware* InteGrade foi o *middleware* de grade escolhido para fornecer o serviço de execução de aplicações distribuídas a dispositivos computacionais móveis no desenvolvimento do trabalho apresentado nesta dissertação. Foram descritos ainda, os dois principais serviços do *middleware* de grade InteGrade: a execução de aplicações distribuídas e o armazenamento distribuído de dados.

3 Acesso aos Serviços do InteGrade por Dispositivos Móveis em Redes sem Fio Infra-estruturadas

O objetivo deste capítulo é apresentar uma infra-estrutura de *software* para acesso aos serviços do *middleware* de grade oportunista InteGrade por parte de usuários de dispositivos móveis que possuem conexão com redes sem fio IEEE 802.11 em modo infra-estruturado. Esta arquitetura foi denominada Mobile InteGrade (MInteGrade) [18]. Através deste mecanismo, clientes móveis podem solicitar a execução de aplicações na grade, realizar o acompanhamento da execução das aplicações e visualizar o resultado de computações já concluídas. Para alcançar estes objetivos, a arquitetura proposta possui mecanismos de ciência de contexto, provendo suporte a períodos de desconexão e baixa conectividade bem como à adaptação de conteúdo dos resultados das computações realizadas pela grade.

Nas seções deste capítulo serão descritos os requisitos de projeto que nortearam o desenvolvimento do *software*, o *middleware* MoCA, utilizado para provisão e processamento de informações de contexto em redes móveis sem fio, a arquitetura do MInteGrade e detalhes de sua implementação, além de trabalhos relacionados ao mesmo.

3.1 Requisitos do Projeto

O projeto do MInteGrade levou em consideração características da computação móvel, das redes sem fio e do *middleware* de grades oportunistas InteGrade, estabelecendo-se, assim, os seguintes requisitos para seu desenvolvimento:

1. A arquitetura deve ser elaborada de modo a causar o mínimo impacto aos componentes já implementados do InteGrade;

2. Do ponto de vista da grade não deve haver diferenciação entre requisições feitas por nós fixos e nós móveis;

3. Deve-se prever a possibilidade dos dispositivos móveis passarem por períodos

de desconexão ou de baixa conectividade;

4. A infra-estrutura de *software* deve dar suporte a grande heterogeneidade tecnológica de dispositivos decorrentes das diversas famílias de equipamentos existentes, fornecendo adaptações do conteúdo das respostas retornadas pela grade de acordo com características do dispositivo móvel cliente.

5. A aquisição e interpretação do contexto dos clientes móveis, assim como o processamento das adaptações correspondentes a cada contexto, deve ser realizada por componentes da rede fixa, de forma a permitir o desenvolvimento de "clientes leves" para execução nos aparelhos móveis.

3.2 Introdução a MoCA

A arquitetura para acesso aos serviços do InteGrade através de dispositivos móveis foi implementada com o auxílio do *middleware* MoCA. A arquitetura MoCA[1] (*Mobile Collaboration Architecture*) [36] foi desenvolvida com a finalidade de oferecer suporte ao desenvolvimento e execução de aplicações distribuídas sensíveis ao contexto, particularmente aquelas que envolvem dispositivos móveis interconectados através de redes *Wireless LAN* infra-estruturadas (IEEE 802.11b/g). Os serviços disponibilizados pela MoCA provêem meios para coletar, armazenar, processar e difundir informações de contexto obtidas a partir dos dispositivos móveis que interagem entre si em uma rede sem fio e com computadores da rede fixa. Além disso, é fornecido um conjunto de API's para o desenvolvimento de aplicações que interagem com esses serviços como consumidores de informações de contexto. Os serviços da MoCA, ilustrados na figura 3.1, são brevemente descritos a seguir.

O `Monitor` é um *daemon* executado em cada dispositivo móvel que tem o objetivo de coletar dados brutos de contexto do dispositivo (tais como percentual de uso da UCP, memória disponível, nível de energia da bateria) e da rede sem fio (AP corrente, sinal RF) e enviá-las periodicamente para o `CIS` (*Context Information Service*). Exitem implementações do `Monitor` para Windows XP, Windows CE, Linux e Symbian OS.

O Context Information Service (`CIS`) é um serviço distribuído no qual cada

[1] Página Inicial: `http://www.lac.inf.puc-rio.br/moca/`

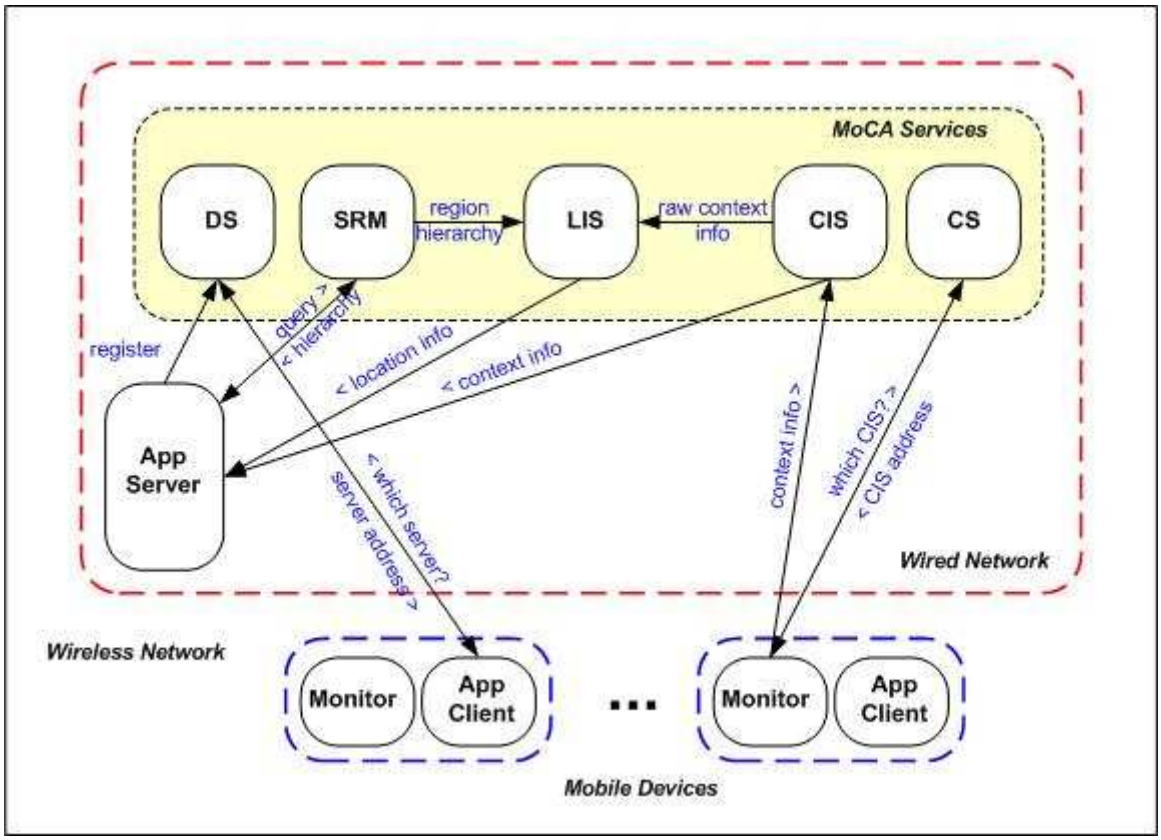

Figura 3.1: A arquitetura MoCA

servidor `CIS` recebe e processa dados brutos de contexto enviados por monitores de cada dispositivo móvel. O `CIS` utiliza as especificações CC/PP[2] e UAProf [1] como fontes de informação das características estáticas dos dispositivos. O CC/PP especifica uma estrutura geral de um documento XML para descrever as características dos dispositivos e dos usuários. O *Open Mobile Aliance*[3] criou a especificação UAProf com o objetivo de padronizar os vocabulários CC/PP. Para cada dispositivo desenvolvido pelos fabricantes, um documento UAProf é criado e armazenado no repositório *Web* no site do fabricante. Esse documento contém informações como características do dispositivo, informações sobre sistema operacional e *software* disponíveis.

A aquisição e manutenção de informações de contexto dos clientes móveis no `CIS` é ilustrada na figura 3.2. Como já mencionado, o `Monitor` coleta as informações de contexto dinâmicas sobre os dispositivos e os envia ao `CIS`. No `CIS`, estas informações são convertidas para o Modelo de Contexto da MoCA[4] e armazenadas em um banco de dados local. É o Modelo de Contexto da MoCA que define como

[2] *W3C composite capability/preference profiles* (CC/PP): `http://www.w3.org/Mobile/CCPP/`
[3] *Open mobile alliance*: `http://www.openmobilealliance.org/`
[4] Modelo de contexto da moca: `http://www.lac.inf.puc-rio.br/moca/cis/context-model.html`

as informações de contexto são descritas e como partes específicas das informações de contexto podem ser acessadas. O `Monitor` também envia ao `CIS` as informações sobre o fabricante e o modelo do dispositivo móvel. O `Controlador` repassa essas informações (modelo e fabricante) ao `UAProfHandler`, que utiliza o `UAProfCache` para verificar se o documento UAProf que descreve as propriedades estáticas dos dispositivos já se encontra no *cache*. Caso não seja encontrado, o *download* do documento é realizado junto aos fabricantes. O uso de *cache* é necessário para evitar *downloads* repetidos dos documentos UAProf que podem causar atrasos no processo de obtenção do contexto e sua utilização. O `UAProfHandler`, de posse do documento UAProf, o interpreta e repassa as informações estáticas ao gerenciador do Modelo de Contexto MoCA, que retira do documento as informações de contexto que estão definidas no Modelo de Contexto da MoCA e as armazena no banco local.

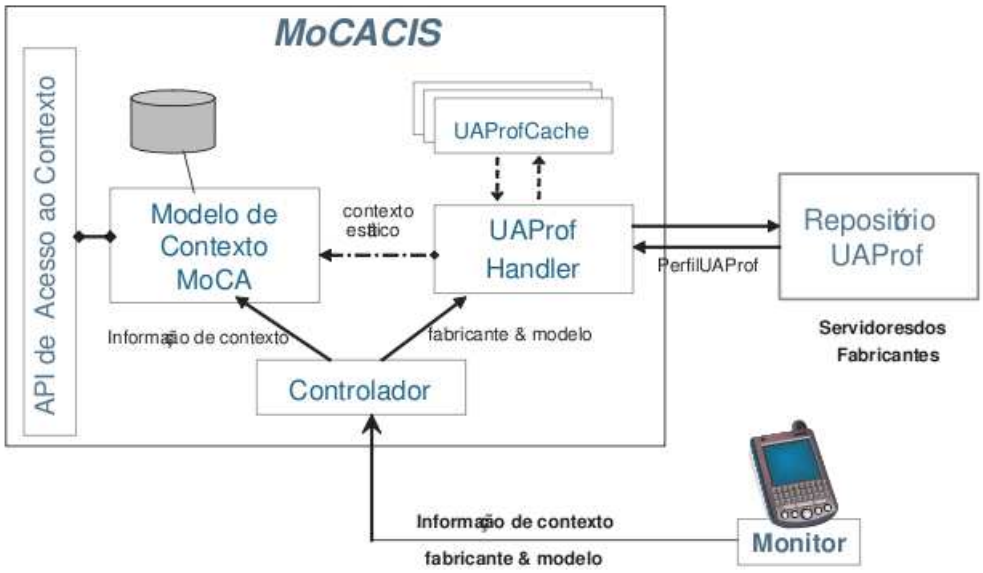

Figura 3.2: Obtenção de contexto no CIS

As aplicações podem consultar informações de contexto de forma síncrona ou assíncrona através da API de Acesso ao Contexto do `CIS`. Por meio de consultas síncronas, aplicações podem solicitar informações atualizadas sobre o contexto de um determinado dispositivo. Através de consultas assíncronas, aplicações podem registrar interesse em situações de contexto específicas descritas por expressões lógicas (de sintaxe semelhante a SQL), envolvendo diversos atributos de contexto de um dispositivo. Por exemplo, para uma aplicação registrar o interesse em ser notifi-

cada quando o nível de energia da bateria de um PDA ficar abaixo de 30%, pode-se usar a seguinte expressão de interesse: `Device.LocalResources.EnergyLevel > 30 AND DeviceProfile.HardwarePlatform.DeviceType = PDA`. A tabela 3.1 enumera algumas das informações de contexto providas pelo `CIS`.

Informação de contexto	**Descrição**
Device.LocalResources.CpuUsage	Uso da CPU (0 a 100%)
Device.LocalResources.EnergyLevel	Nível de energia disponível (0 a 100%)
Device.LocalResources.FreeMemory	Total de memória (kB)
Device.Connectivity.CurrentApMacAddr	Endereço MAC do ponto de acesso
Device.Connectivity.OnLine	True se o dispositivo está conectado
Device.Connectivity.ApChange	Verdadeiro se o dispositivo muda o AP
DeviceProfile.HardwarePlatform.DeviceType	Tipo do dispositivo, ex: PDA
DeviceProfile.HardwarePlatform.ScrenSizeWidth	Largura da tela do dispositivo
DeviceProfile.HardwarePlatform.ScrenSizeHeight	Altura da tela do dispositivo

Tabela 3.1: Informações de contexto fornecidas pelo CIS

O *Location Inference Service* (`LIS`) é responsável por inferir a localização aproximada de um dispositivo móvel comparando o padrão corrente de sinais de rádio-frequência observados pelo dispositivo (obtidos de pontos de acesso 802.11 dentro do raio de cobertura) com o padrão de sinais medidos em pontos de referência pré-definidos. O serviço permite ao usuário definir regiões simbólicas, ou seja, associar nomes a regiões físicas bem definidas (por exemplo, salas, prédios, corredores) que são de interesse para aplicações sensíveis à localização. Uma aplicação cliente pode consultar quais são as regiões simbólicas mapeadas no serviço, em que regiões simbólica se localiza um dado dispositivo ou quais dispositivos se encontram em uma determinada região simbólica. Ele pode ainda receber notificações sobre a mudança de área de um dispositivo específico ou a entrada/saída de um dispositivo em uma dada região simbólica.

Outros serviços complementares incluem o `SRM` (*Symbolic Region Manager*), que permite estabelecer uma relação entre as regiões atômicas definidas pelo `LIS`, descrevendo uma hierarquia em que regiões podem ser contidas em outras; O `DS` (*Discovery Service*), um serviço de descoberta, que armazena informações como nome, propriedades e endereço sobre os serviços e *proxies* registrados no *middleware* MoCA; e

o CS (*Configuration Service*), um serviço de configuração que armazena endereços dos serviços básicos da MoCA.

Um serviço de privacidade, chamado CoPS (*Context Privacy Service*), foi implementado para controlar como, quando e a quem devem ser fornecidas informações de contexto. O CoPS é um serviço opcional que permite ao usuário de aplicações sensíveis ao contexto ou baseadas em localização definir e gerenciar suas políticas de privacidade em relação às suas informações de contexto. Antes de atender a qualquer solicitação de informações de contexto, os serviços MoCA consultam o CoPS para decidir se o acesso àquelas informações deve ser concedido ou negado.

3.2.1 ProxyFramework

Para facilitar o desenvolvimento de *proxies* capazes de adaptações de conteúdo sensíveis aos contextos dos clientes móveis, um *framework* (ProxyFramework) [35] foi implementado como parte da arquitetura MoCA. O ProxyFramework intermedeia a comunicação entre o servidor da aplicação e seus clientes móveis, sendo responsável por interagir com os serviços da MoCA, escondendo detalhes desta interação. O *framework* tem a função de inscrever-se como interessado por notificações sobre os contextos de interesse para cada cliente cadastrado pela aplicação e por acionar as adaptações apropriadas, quando um determinado contexto é identificado. Além disso, este *framework* visa aumentar o reuso de código, permitindo sua extensão e personalização para criar instâncias de *proxies* de acordo com as necessidades específicas da aplicação. O ProxyFramework, também provê mecanismos para o armazenamento temporário (*buffering*) de mensagens em momentos de desconexão ou fraca conectividade do cliente.

Para criação de um *proxy* de aplicação através da extensão do ProxyFramework, alguns passos devem ser seguidos: primeiro, implementar as regras de adaptação de acordo com as necessidades específicas da aplicação; segundo, deve-se definir as regras de adaptação que determinam em que condições de contexto as adaptações devem ser aplicadas; por último determinar a estratégia de armazenamento das mensagens destinadas aos clientes móveis.

Além de fornecer uma biblioteca com um conjunto de adaptadores de imagens que podem ser reutilizados pelos usuários (desenvolvedores) do *framework*, o

`ProxyFramework` permite que o desenvolvedor da aplicação *proxy* implemente suas próprias classes de adaptação de conteúdo para atender necessidades específicas da aplicação, através da extensão da classe abstrata `Adapter`, implementando-se o método `execute`. Esse método recebe informações sobre o destinatário da mensagem e a própria mensagem a ser adaptada, e retorna a mensagem com o conteúdo adaptado. Além disso existe a possibilidade de criação de adaptadores parametrizados por contexto, para que seja possível adequar as adaptações às características do dispositivo e seu ambiente de execução. Adaptadores parametrizados por contexto utilizam o valor corrente dos atributos de contexto dos clientes para ajustar a adaptação. O contexto corrente de cada dispositivo pode ser obtido através da método `getCurrentContext` da classe `ClientInfo`. O método `getUsedContextAttribute` da classe `Adapter` deve ser utilizado pelo adaptador parametrizado para indicar quais atributos de contexto foram utilizados para realizar a adaptação. A figura 3.3 apresenta a classe abstrata `Adapter`.

```
public abstract class Adapter extends Action {

   public abstract Message execute(
                        ArrayList<ClientInfo> clientInfo,
                        Message msg)
      throws AdaptationException;

   public ArrayList<String> getUsedContextAttributes();

}
```

Figura 3.3: Classe abstrata Adapter do ProxyFramework

O `ProxyFramework` utiliza o paradigma de políticas de adaptação baseada em regras para determinar quais ações (adaptações) são necessárias de acordo com o contexto do cliente, o que inclui o estado do dispositivo e a qualidade do enlace sem fio. A configuração das regras de adaptação é feita através de um arquivo XML. Neste arquivo, cada estado é definido por uma expressão de contexto. A cada estado pode estar associado uma ação e um número não determinado de regras para adaptação de mensagens. Cada regra é composta por um filtro e ações de adaptação. O filtro serve pra validar as mensagens que podem ser adaptadas por determinado

adaptador. As ações dentro da regra são aplicadas na ordem em que aparecerem no arquivo XML. Além disso, cada regra possui uma prioridade, que se não for definida será considerada como de menor prioridade. Se diferentes regras tiverem a mesma prioridade, serão aplicadas na mesma ordem em que foram definas.

```
<ProxyConf>
<State>
        <Expression>
                <![CDATA[ OnLine=true AND DeltaT > 2000 ]]>
        </Expression>
        <Action class="moca.core.proxy.actions.listeners.DefaultCacheListener">
                <Parameter name="cacheClassName"> moca.core.proxy.cache.FIFOCacher </Parameter>
        </Action>
</State>
<State>
        <Expression>
                <![CDATA[ CPU > 80 AND FreeMemory < 10000 ]]>
        </Expression>
        <Rule priority="1">
                <Filter>
                        <StartWith>
                                <FieldValue> <Literal>datatype</Literal> </FieldValue>
                                <Literal>image/</Literal>
                        </StartWith>
                </Filter>
        <Action class="moca.core.proxy.actions.adapters.ScaleImageAdapter">
                <Parameter name="factor">0.5</Parameter>
        </Action>
        </Rule>
</State>
<State>
        <Expression>
                <![CDATA[ DeviceProfile.HardwarePlatform.DeviceType LIKE 'PDA' ]]>
        </Expression>
        <Rule priority="1">
                <Filter>
                        <StartWith>
                                <FieldValue> <Literal>datatype</Literal> </FieldValue>
                                <Literal>image/</Literal>
                        </StartWith>
                </Filter>
                <Action class="moca.core.proxy.actions.adapters.CropCenterParamAdapter">
                </Action>
        </Rule>
</State>
</ProxyConf>
```

Figura 3.4: Exemplo do arquivo de configuração do ProxyFramework

A Figura 3.4 apresenta um exemplo de arquivo de configuração de regras do `ProxyFramework`. Neste exemplo, são definidas três expressões de contexto: uma para desconexão e duas para adaptação de conteúdo. As expressões de contexto são expressões lógicas que combinam atributos de contexto, tanto estáticos (como tamanho da tela do dispositivo), quanto dinâmicos (como energia disponível). Neste exemplo, define-se que para uma desconexão superior a dois segundos, as mensagens serão armazenadas em uma fila temporária, seguindo a estratégia FIFO. O arquivo também indica que mensagens, cujo conteúdo seja uma imagem, deverão sofrer adaptações nas seguintes situações: i) caso o dispositivo móvel esteja com pouca memória e CPU disponíveis (contexto dinâmico), a imagem deve ser reduzida em 50%; ii) caso seja um

PDA (contexto estático), utiliza-se um adaptador parametrizado que recorta a imagem para o tamanho das dimensões da tela do PDA.

No que tange ao armazenamento de mensagens em *cache* em favor dos clientes móveis, o `ProxyFramework` provê duas políticas padrão. Na primeira, as mensagens são simplesmente descartadas e na outra implementa-se uma fila FIFO, com descarte de mensagens mais antigas após um certo número de mensagens. O `ProxyFramework` também fornece ao desenvolvedor a possibilidade de definir suas próprias estratégias de armazenamento de mensagens. Esta flexibilidade é fornecida pela interface `Cacher`, que estabelece os métodos que devem ser desenvolvidos para todas as classes que implementam políticas de armazenamento a serem utilizadas pelo `ProxyFramework`. A figura 3.5 apresenta a interface `Cacher`.

```
public interface Cacher {

   // Adiciona um pacote (mensagem) durante a desconexão
   void addPackage(Package pack);

   // Retorna todos os pacotes armazenados (na reconexão)
   Package[] getPackages();
}
```

Figura 3.5: Interface Cacher do ProxyFramework

3.3 Arquitetura do Serviço

A arquitetura proposta para a integração de dispositivos móveis ao InteGrade foi elaborada tentando tornar o mais transparente possível a este *middleware* de grade os desafios provenientes da computação móvel descritos no Capítulo 1. A arquitetura proposta é baseada no modelo cliente/*proxy*/servidor [34]. Nesta arquitetura, como mostrado na Figura 3.6, um agente ou *proxy* representa o cliente móvel na rede fixa. Toda a comunicação entre o cliente e o servidor passa pelo agente, que pode assumir responsabilidades em favor do cliente, como realizar requisições complexas ao servidor, retornando apenas os resultados; realizar *cache* de mensagens destinadas ao cliente; comprimir e formatar dados ou realizar tradução entre protocolos diferentes utilizados do lado cliente e servidor.

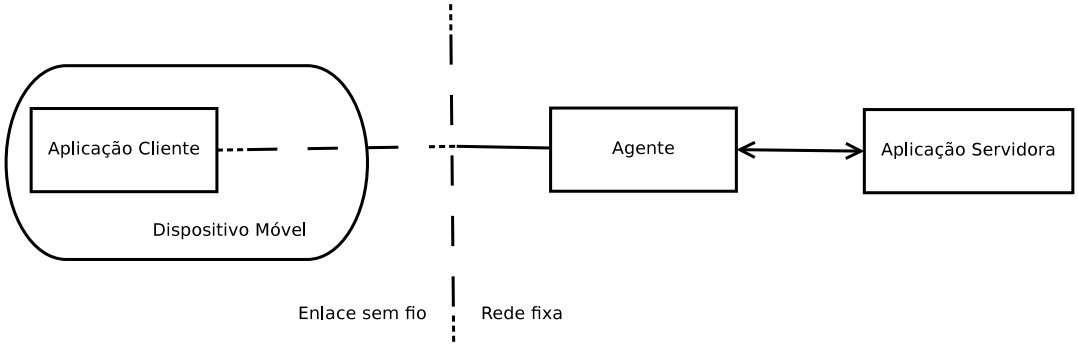

Figura 3.6: Arquitetura cliente/proxy/servidor

Na arquitetura do MInteGrade, o dispositivo móvel assume o papel de cliente e o *middleware* de grade de servidor. O cliente móvel executa dois componentes de *software*: o `MASCT` (*Mobile Application Submission and Control Tool*) e um `Monitor` de contexto. O papel do *proxy* cabe aos componentes `ProxyAdapter` e `GridProxy`, enquanto o *middleware* InteGrade atua como servidor. Esta arquitetura é ilustrada na figura 3.7 e seus componentes são descritos a seguir.

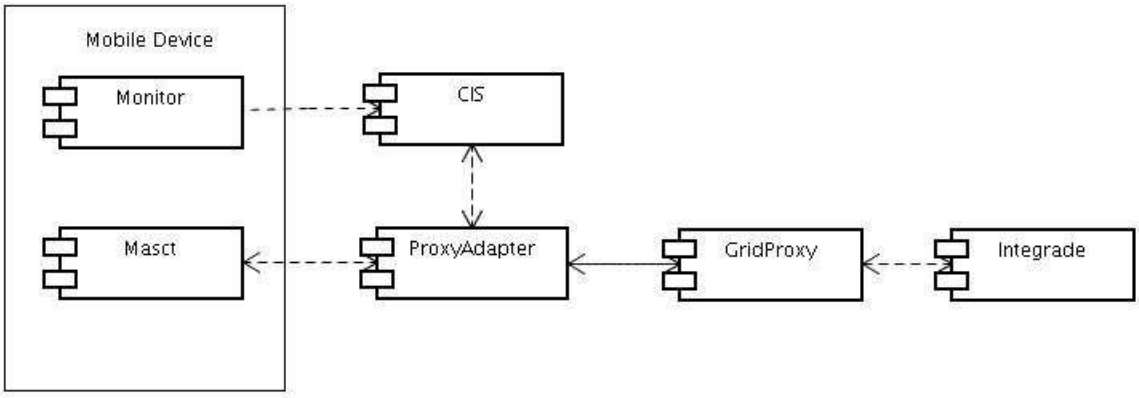

Figura 3.7: Componentes da arquitetura de acesso a grade através de dispositivos móveis

- **Mobile Application Submission and Control Tool (MASCT)**: interface gráfica que permite aos usuários de dispositivos móveis requisitar os seguintes serviços da grade: submeter aplicações, acompanhar o andamento da execução das aplicações submetidas e visualizar os resultados das computações finalizadas;

- **Monitor**: *daemon* integrante da arquitetura MoCA que executa em cada dispositivo móvel e é responsável por coletar informações de contexto do dispositivo, que incluem a qualidade da conexão, energia disponível, uso de CPU, quantidade de memória livre, ponto de acesso à rede atual, lista de pontos de acesso dentro do raio de cobertura e suas intensidades de sinal;

- **Context Information System (CIS)**: componente da arquitetura MoCA que processa as informações de contexto enviadas pelo `Monitor` e gera notificações de eventos relacionados ao contexto dos dispositivos móveis ao `ProxyAdapter`. Notificações sobre eventos de contexto são geradas sempre que forem detectadas mudanças no estado dos dispositivos móveis para as quais o `ProxyAdapter` tenha registrado interesse;

- **ProxyAdapter**: componente implementado através da extensão do *framework* `ProxyFramework` da arquitetura MoCA. O `ProxyAdapter` intermedeia toda a comunicação entre clientes móveis e o `GridProxy`, tratando eventuais desconexões através do armazenamento em uma *cache* das mensagens direcionadas ao dispositivo até que a conectividade seja restabelecida. É também responsável pela adaptação do conteúdo dos resultados das computações submetidas à grade, de modo a permitir sua exibição nos dispositivos móveis. O *ProxyAdapter* é capaz de redimensionar imagens retornadas pela grade de acordo com o tamanho da tela do dispositivo a quem elas se destinam. Como exemplo de aplicações que se beneficiariam deste tipo de adaptação podemos citar as aplicações que utilizam a técnica CBIR (*Content-Based Image Retrieval*) [21], para busca de imagens médicas armazenadas em sistemas PACS (*Picture Archiving and Communication System*) [22], utilizando características extraídas diretamente da própria imagem como, por exemplo, forma, tamanho, contorno e textura. Neste caso, a grande capacidade de processamento das grades computacionais pode ser utilizada para extração de características das imagens médicas e consulta, a partir dessas características, por similaridades entre imagens que podem estar dispersas por múltiplas instituições hospitalares. Médicos podem então, através da comparação dessas imagens identificar doenças nos pacientes em estudo. As buscas por imagens de raio-X, tomografia computadorizada, ressonância magnética e ultrasonografia armazenadas em sistemas PACS que utilizam a infra-estrutura de grades computacionais podem ser realizadas a partir de dispositivos móveis com a utilização da arquitetura do MInteGrade, com a vantagem de que as imagens recuperadas podem ser adaptadas ao tamanho da tela desses aparelhos.

- **GridProxy**: é o componente responsável por tornar os dispositivos móveis transparentes aos demais componentes da grade. O `GridProxy` recebe as requisições de cada `MASCT` executando nos dispositivos móveis que são repassadas pelo

`ProxyAdapter`, formata estas requisições em favor do dispositivo, utilizando o protocolo definido pelo InteGrade e as encaminha para execução na grade. O `GridProxy` também é responsável por manter um banco de dados que relaciona cada solicitação a um serviço da grade ao dispositivo móvel responsável por sua emissão e por atualizar o estado da execução nesse banco de dados, quando notificações de aplicação aceita para execução, recusada, executando ou finalizada são encaminhadas pela grade.

- **InteGrade**: é o *middleware* de grade que irá fornecer serviços aos clientes móveis. Através das chamadas às APIs fornecidas pelo InteGrade, componentes podem ter acesso aos serviços da grade, como a solicitação para execução de aplicações regulares, paramétricas ou paralelas, acompanhamento do estado das execuções submetidas e obtenção de seus resultados.

3.3.1 Interação entre Componentes

Como ilustração das interações entre os componentes da arquitetura apresentada, o diagrama de sequência ilustrado na Figura 3.8 mostra os passos de uma requisição por parte de um usuário de dispositivo móvel para a execução de uma dada aplicação na grade.

A requisição para execução é realizada no dispositivo móvel através da interface do `MASCT` e encaminhada ao componente `ProxyAdapter` (1). A comunicação entre `MASCT` e `ProxyAdapter` é realizada através de *sockets* TCP. O `ProxyAdapter` repassa a mensagem com a solicitação de execução da aplicação ao `GridProxy`(2). Este componente mantêm um banco de dados que relaciona cada solicitação a um serviço da grade ao dispositivo móvel responsável por sua emissão (3). Além disto, o `GridProxy` é responsável por manter os arquivos de entrada utilizados na execução da aplicação. Estes arquivos podem ser fornecidos pelo dispositivo móvel (no corpo da solicitação) ou podem ser obtidos através de uma URL fornecida. Desta forma, o `GridProxy` formata os dados constantes da requisição de execução e os envia para a grade (4), de forma a tornar transparente o fato da requisição ter sido realizada através de um dispositivo móvel. A comunicação entre o `GridProxy` e a interface para submissão de aplicações do InteGrade é realizada através da tecnologia de objetos distribuídos CORBA (*Common Object Request Broker Architeture*).

Após constatar a disponibilidade de recursos para executar a aplicação solicitada, a grade retorna uma mensagem informando que a requisição foi aceita para execução (5). Esta mensagem é enviada ao `GridProxy`, que atualiza a entrada em seu banco de dados referente àquela requisição (6). Uma mensagem de notificação é, então, enviada ao dispositivo móvel através do `ProxyAdapter` (7). O resultado da computação realizada pela grade segue o mesmo percurso. Como citado anteriormente, o `ProxyAdapter` trata períodos de desconexão através de uma *cache* local e também realiza transformações nas mensagens de forma a adaptá-la ao dispositivo móvel específico ao qual ela se destina.

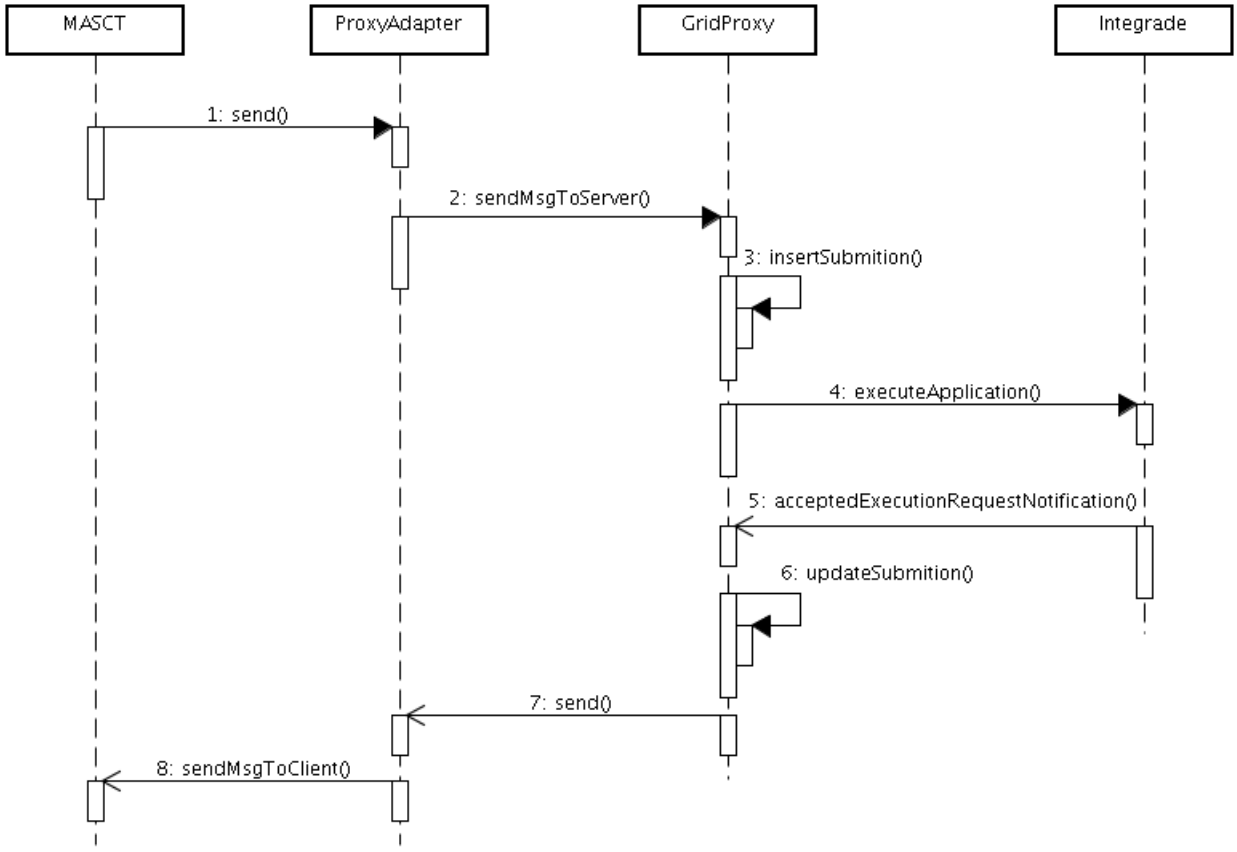

Figura 3.8: Submissão de uma aplicação à grade através de dispositivo móvel

3.4 Implemetação do Serviço

A arquitetura do MInteGrade é uma arquitetura orientada a objetos. Seus componentes foram escritos em Java, sendo que para sua implementação foram utilizados componentes da arquitetura MoCA. Foram utilizados tanto componentes já prontos desta arquitetura, para obtenção de contexto dos dispositivos móveis (`Monitor`)

e processamento das informações de contexto (`CIS`), quanto um novo componente foi criado (`ProxyAdapter`) a partir da extensão do *framework* `ProxyFramework` para criação de *proxies* de aplicação capazes de adaptação de conteúdo sensíveis a contexto. Nas subseções a seguir serão fornecidos os detalhes de implementação de todos os componentes constantes da arquitetura proposta.

3.4.1 Mobile Application Submission and Control Tool (MASCT)

O *Mobile Application Submission and Control Tool* (`MASCT`) é formado por cinco pacotes: `view`, `control`, `communication`, `messages` e `util`, como ilustrado no diagrama de classes da Figura 3.9.

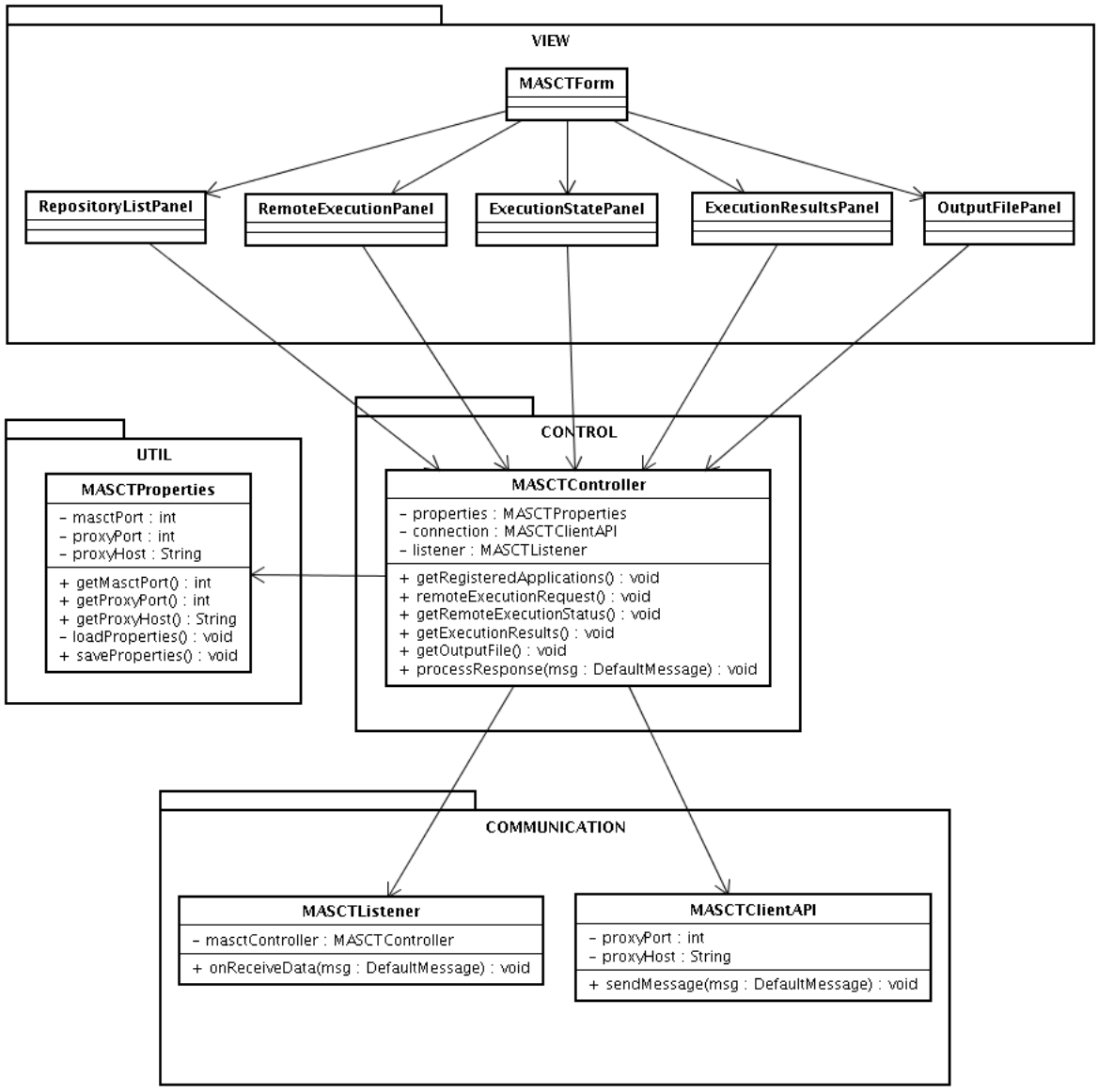

Figura 3.9: Diagrama de classes do MASCT

O pacote `view` possui as classes utilizadas para construir a interface gráfica da aplicação. Todas as telas da interface gráfica do `MASCT` foram construídas utilizando-se o *toolkit* gráfico AWT [5]. O `MASCT` é composto por um formulário principal, implementado através da classe `MASCTForm`, que disponibiliza através de um menu os tipos de requisições que podem ser feitas à grade. Através dos painéis `RepositoryListPanel`, `ExecutionStatePanel`, `ExecutionResultsPanel` e `OutputFilePanel`, os usuários podem, respectivamente, visualizar a lista de aplicações registradas, o estado de execução (*ACCEPTED*, *REFUSED*, *EXECUTING*, *FINISHED*) das aplicações enviadas a grade, os arquivos gerados como resultado de uma determinada computação e os próprios conteúdos dos arquivos de saída gerados, sejam eles textos ou imagens.

O `MASCT` também possui um conjunto de painéis onde o usuário pode fornecer os parâmetros a serem utilizados na execução de uma aplicação. O `RemoteExecutionPanel` permite que sejam definidos parâmetro gerais, utilizados em todas as classes de aplicações (regulares, paramétricas, paralelas). Entre estes parâmetros constam o nome da aplicação, o caminho até o binário da aplicação no repositório de aplicações e as preferências e restrições utilizados pelo escalonador da grade durante o processo de seleção dos recursos a serem utilizados para a execução da aplicação; o `RegularPanel` permite que o usuário defina os parâmetros para submissão de aplicações regulares, como a lista de argumentos da aplicação, os eventuais arquivos de entrada da aplicação e os eventuais arquivos de saída que devem ser gerados durante sua execução; o `ParametricPanel` é utilizado para definir os parâmetros para submissão de aplicações paramétricas. Este painel permite que sejam adicionadas, removidas ou editadas cópias da aplicação a serem executadas em paralelo e que para cada uma delas sejam definidos seus argumentos e eventuais arquivos de entrada e saída; o `BSPPanel` define os parâmetros de submissão de aplicações BSP, como a lista de argumentos da aplicação, o número de tarefas da aplicação BSP e uma última opção que permite ao usuário impor uma restrição adicional solicitando que cada tarefa que compõem a aplicação BSP seja executada em uma máquina diferente;

A Figura 3.10 mostra a tela principal (a) e a de visualização da lista de aplicações registradas (b). A Figura 3.11 apresenta a tela para submissão de uma

[5]Documentação do Java AWT: `http://java.sun.com/javame/reference/apis/jsr216/java/awt/package-summary.html`

aplicação regular (a) e a de submissão de uma aplicação BSP(b).

(a) (b)

Figura 3.10: Tela principal (a) e lista de aplicações registradas (b)

(a) (b)

Figura 3.11: Submissão de aplicação regular (a) e submissão de aplicação BSP (b)

A classe `MASCTController`, definida no pacote `control`, é responsável por tratar os eventos de submissão de requisição gerados nas classes de interface gráfica e montar as respectivas mensagens de solicitação a serem enviadas à grade, assim como invocar os métodos adequados das classes de interface para apresentação

das respostas recebidas pelo `MASCT`.

O pacote `communication` contém a classe `MASCTClientAPI`, utilizada para enviar as mensagens de solicitação de serviço à grade, através do estabelecimento de uma conexão TCP com o componente `ProxyAdapter` e a classe `MASCTListener`, uma `Thread` implementada para receber mensagens de resposta de requisições enviadas à grade.

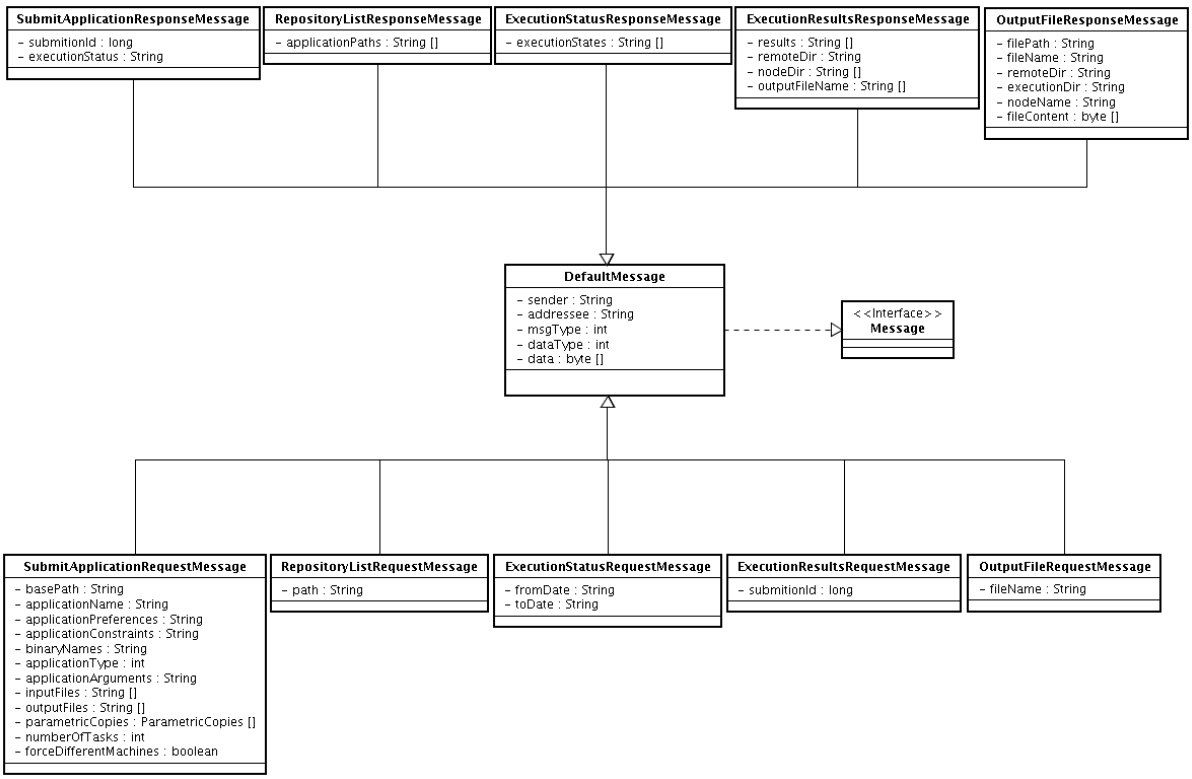

Figura 3.12: Diagrama de classes do pacote messages do MASCT

Já o pacote `messages`, cujo diagrama de classes é ilustrado na Figura 3.12, agrupa as classes de mensagens utilizadas para encapsular os dados necessários para realizar requisições de serviço à grade e as classes de mensagens de resposta às solicitações realizadas. Todas as mensagens definidas nesse pacote estendem da classe `DefaultMessage` do `ProxyFramework`. Esta classe é utilizada na arquitetura MoCA como mensagem padrão de comunicação dos clientes da rede móvel e servidores com os *proxies* que são desenvolvidos através da extensão do `ProxyFramework`. Entre as classes de mensagens definidas nesse pacote estão as classes `RepositoryListRequestMessage`, `SubmitApplicationRequestMessage`, `ExecutionStatusRequestMessage`, `ExecutionResultsResponseMessage` e `OutputFileRequestMessage`, utilizadas respectivamente para enviar requisições

de busca por aplicações registradas, execução de aplicações, busca pelo estado de execução de uma aplicação, busca pela lista de arquivos de saída gerados pela execução de uma aplicação e recuperação do conteúdo de um dos arquivos de saída gerados. Também foram inseridas nesse mesmo pacote as classes de resposta a cada uma dessas solicitações: `RepositoryListResponseMessage`, `SubmitApplicationResponseMessage`, `ExecutionStatusResponseMessage`, `ExecutionResultsRequestMessage` e `OutputFileRequestMessage`.

O pacote `util` contém a classe utilitária `MASCTProperties`, utilizada para persistir em um arquivo de propriedades informações sobre o endereço de rede e porta do componente `ProxyAdapter` com o qual `MASCT` se comunicará, além da porta que será utilizada pelo `MASCT` para escutar por respostas enviadas pelo `ProxyAdapter`. Essa classe também é utilizada para, na inicialização do `MASCT`, ler e carregar essas informações de configuração do arquivo de propriedades.

3.4.2 ProxyAdapter

O `ProxyAdapter` é o componente da arquitetura do MInteGrade responsável por intermediar toda a comunicação entre os clientes móveis e o `GridProxy`. Sua implementação foi realizada a partir da extensão do *framework* `ProxyFramework` da arquitetura MoCA. Este *framework* foi utilizado com o objetivo de fornecer ao `ProxyAdapter` a possibilidade de realizar adaptações sensíveis ao contexto dos clientes móveis e armazenamento temporário (*buffering*) de mensagens em momentos de desconexão ou baixa conectividade dos dispositivos.

Como ilustrado no diagrama de classes da Figura 3.13, o `ProxyAdapter` é formado por quatro classes: `ProxyFramework`, `ProxyAdapter`, `ClientMsgListener` e `ServerMsgListener`.

A classe `ProxyAdapter` é a classe principal do componente `ProxyAdapter`. Suas principais responsabilidades são receber as mensagens dos clientes móveis destinadas à grade, registrar estes clientes no `ProxyFramework`, informar ao `CIS` que está interessado em receber informações de contexto do cliente registrado, iniciar o processo de adaptação das mensagens de resposta retornadas pela grade e enviá-las ao dispositivo destinatário.

O registro dos clientes móveis no `ProxyFamework` e no `CIS` são neces-

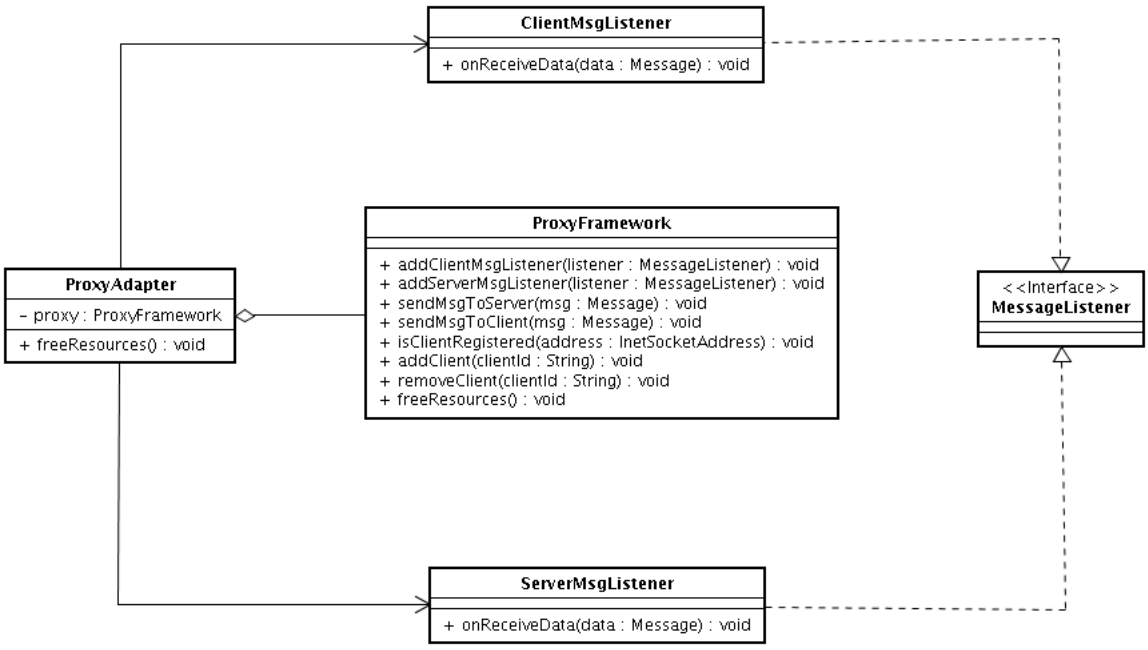

Figura 3.13: Diagrama de classes do ProxyAdapter

sários para que seja realizado o mapeamento entre o identificador do dispositivo móvel (*MAC Address*) e seu endereço de comunicação (*IP Address*), possibilitando que mensagens de resposta destinadas ao dispositivo possam ser corretamente entregues ao destinatário e para que notificações de alterações no contexto dos clientes móveis possam ser recebidos para realização das adaptações das imagens retornadas pela grade à resolução de tela do dispositivo.

Para utilizar as funcionalidades de comunicação, adaptação e armazenamento de mensagens do `ProxyFramework`, a classe `ProxyAdapter` instancia a classe `ProxyFramework` e utiliza seus métodos. Ao iniciar, o `ProxyAdapter` utiliza o método `init` da classe `ProxyFramework` para definir o protocolo de comunicação que será utilizado (TCP) e carregar os parâmetros de comunicação que indicam a porta para comunicação com o `MASCT`, o endereço IP e a porta onde o `GridProxy` aguardará por mensagens e o endereço IP e porta do serviço de provisão de contexto que será utilizado. Logo após, o `ProxyAdapter` registra as classes `ClientMsgListener` e `ServerMsgListener`, como os `MessageListeners` que ficarão responsáveis, respectivamente, por receber as mensagens oriundas dos clientes móveis e as respostas retornadas pelo `GridProxy`. Ao receber mensagens enviadas pelos clientes, o `ProxyAdapter` verifica se a mensagem é uma mensagem de controle para registro do cliente e utiliza o método `isClientRegistered` para verificar se o cliente já

se encontra registrado. Caso não esteja, é realizado o registro no `ProxyFramework` utilizando o método `addClient`. Ao receber mensagens de requisição de serviço destinadas à grade, o `ProxyAdapter` as encaminha ao `GridProxy` através do método `sendMsgToServer`.

Quando o `ProxyAdapter` recebe as mensagens de resposta que são enviadas pelo `GridProxy`, ele as encaminha ao `ProxyFramework` para que possam ser realizadas as adaptações necessárias, antes do seu envio para os clientes. A Figura 3.14 ilustra o percurso seguido por uma mensagem no `ProxyFramework`.

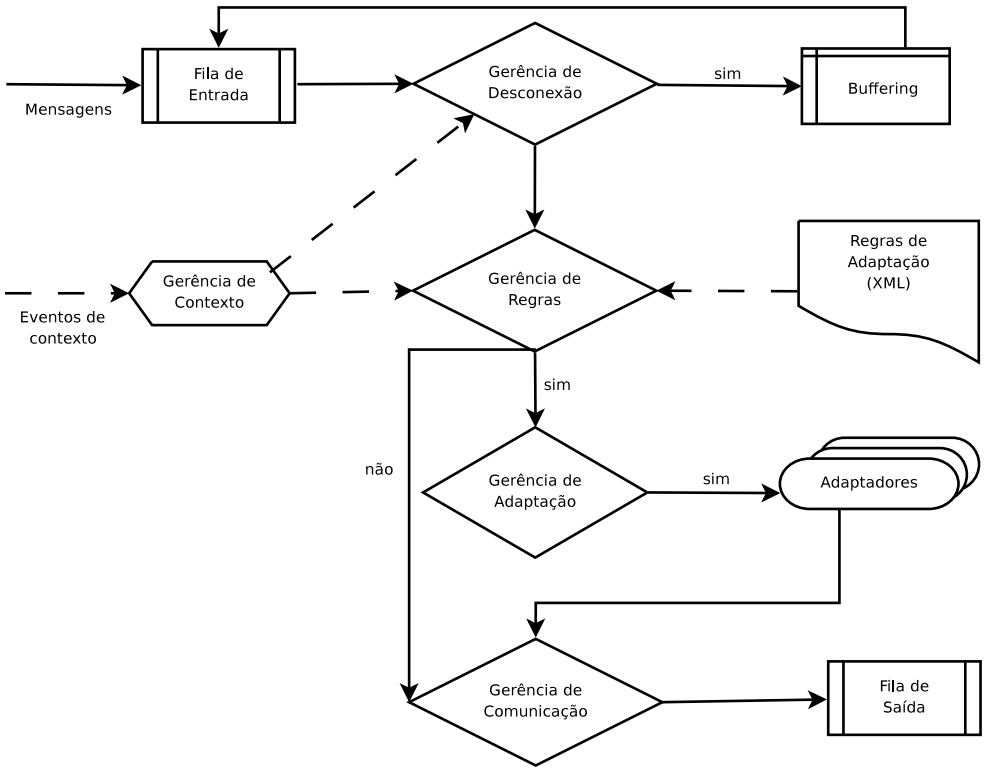

Figura 3.14: Fluxo de uma mensagem no ProxyFramework

Ao receber as mensagens, o `ProxyFramework` as insere em uma fila de entrada, de onde o componente Gerente de Desconexão as retira para armazenamento temporário caso o cliente esteja passando por um período de desconexão superior a dois segundos, de acordo a primeira regra do arquivo de configuração de regras definido para o `ProxyAdapter` (linhas 4 a 12), ilustrado na Figura 3.15. Esta regra também indica que as mensagens destinada aos clientes nesta condição devem ser armazenadas seguindo uma estratégia FIFO, definida através da classe `FIFOCacher`, uma das classes da biblioteca de adaptadores fornecida juntamente com o `ProxyFramework`.

No estágio seguinte, a mensagem é encaminhada ao componente Gerente

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <ProxyConf>
3
4 <State> <!-- Desconexão Temporária -->
5     <Expression>
6          <![CDATA[ OnLine = false AND DeltaT < 2000 ]]>
7     </Expression>
8     <Action class="moca.core.proxy.actions.listeners.DefaultCacheListener">
9         <Parameter name="cacheClassName">moca.core.proxy.cache.FIFOCacher
10        </Parameter>
11    </Action>
12 </State>
13
14 <State>
15        <Expression><![CDATA[ ALL_DEVICES ]]></Expression>
16        <Rule priority="1">
17                <Filter>
18                        <!-- tipo de dados da mensagem -->
19                        <StartWith>
20                                <FieldValue>
21                                        <Literal>datatype</Literal>
22                                </FieldValue>
23                                        <Literal>image/</Literal>
24                        </StartWith>
25                </Filter>
26                <Action class="moca.core.proxy.actions.adapters.ResizeImageAdapter">
27                </Action>
28        </Rule>
29 </State>
30
```

Figura 3.15: Arquivo de configuração de regras de adaptação do ProxyAdapter

de Regras que define quais adaptações devem ser aplicadas à mesma. No arquivo de configuração de regras do `ProxyAdapter`, a segunda regra (linhas 14 a 29) define que mensagens cujo conteúdo sejam uma imagem deverão ser redimensionadas utilizando-se como parâmetro o tamanho da tela do dispositivo para o qual a mensagem será enviada. Caso o conteúdo da mensagem seja uma imagem, a mensagem é repassada ao Gerente de Adaptações, que executará o adaptador `ResizeImageAdapter` definido pela regra. Este adaptador foi implementado e adicionado à biblioteca de adaptadores do `ProxyFramework` para que fosse possível realizar o redimensionamento da imagem de acordo com a altura e a largura da tela do dispositivo móvel. Sua implementação é baseada no adaptador `ScaleImageAdapter` presente na biblioteca de adaptadores do `ProxyFramework`, cuja funcionalidade consiste em ampliar ou reduzir a imagem de acordo com um fator pré-definido (se o fator estiver entre 0 e 1 a imagem é reduzida, se for maior que 1 é ampliada) indicado no próprio arquivo de configuração da regra de adaptação. Diferente da classe `ScaleImageAdapter`, a classe `ResizeImageAdapter`, consulta as informação de contexto estático do cliente `ScreenSizeWidth` e `ScreenSizeHeight` para definir as novas dimensões da imagem. Depois que a adaptação de resolução de imagem tiver sido aplicada, a mensagem é encaminhada ao Gerente de Comunicação que as coloca na fila de saída para ser enviada ao cliente.

Além das regras já definidas em nossa arquitetura para realização do *cache* de mensagens e adaptação do tamanho das imagens, o administrador da grade que utiliza o `ProxyAdapter` pode definir, se necessário, outros contextos de interesse no arquivo XML de configuração de regras e as respectivas adaptações associadas a eles.

3.4.3 GridProxy

O `GridProxy` é o componente responsável por tornar os dispositivos móveis transparentes aos demais componentes da grade, atuando como um conversor entre os protocolos de comunicação utilizado pelo *middleware* MoCA e o *middleware* InteGrade. O `GridProxy` também possui um banco de dados onde são registrados para cada solicitação de execução à grade, o identificador do dispositivo móvel que a emitiu, o estado da execução e a data de sua submissão. A Figura 3.16 apresenta o diagrama de classes do componente `GridProxy`.

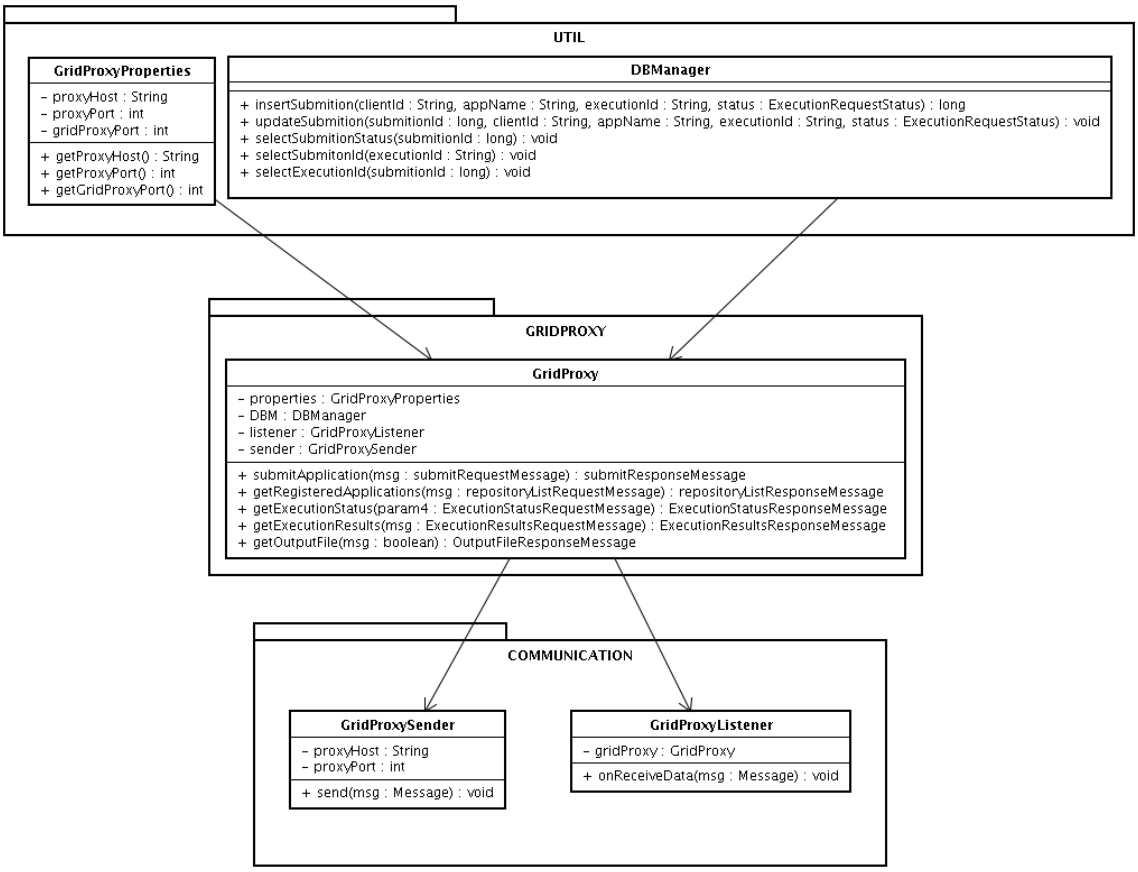

Figura 3.16: Diagrama de classes do GridProxy

A classe `GridProxy` é responsável por formatar as requisições que são repassadas pelos clientes móveis segundo o padrão de requisição de serviço utilizado

pelo InteGrade e repassá-las à grade. Também é responsável por, ao receber respostas às requisições realizadas, encapsular os resultados em mensagens que seguem o padrão de comunicação MoCA para que possam ser repassadas ao `ProxyAdapter`. Ao receber uma requisição de submissão de execução de aplicação à grade, o `GridProxy`, utiliza a classe utilitária `DBManager` para acessar o banco de dados embutido na aplicação e armazenar o identificador da submissão, o identificador do dispositivo móvel responsável pela requisição, o estado da execução e a data da submissão. Essa mesma classe é utilizada para atualizar o estado da execução quando notificações são enviadas pela grade e para recuperá-lo em resposta a uma solicitação de busca pelo estado de execução de uma aplicação por parte de um cliente. O `GridProxy` possui as classes de comunicação `GridProxyListener` e `GridProxySender`. A primeira é uma `Thread` que escuta por mensagens vindas do `ProxyAdapter`, que são então repassadas a classe `GridProxy` para serem processadas e enviadas à grade. A segunda tem a função de estabelecer uma conexão TCP como o componente `ProxyAdapter` para envio das mensagens de respostas que são retornadas pela grade.

3.5 Trabalhos Relacionados

Alguns outros projetos tratam o desafio de tornar seus serviços disponíveis para os dispositivos móveis. Esta seção descreve projetos relevantes na área de grades de computadores que integram dispositivos móveis à sua arquitetura. Após os resumos de cada projeto uma seção apresentará uma análise comparativa com a arquitetura proposta nesta dissertação.

3.5.1 Mobile OGSI.NET

Em Humprey e Chu [8] é descrito o projeto, implementação e avaliação de um *framework* chamado Mobile OGSI.NET. Este *framework* foi criado para promover o compartilhamento de recursos e colaboração entre dispositivos móveis, além de melhorar o acesso por parte do usuário móvel à ambientes de grade computacional já existentes. O Mobile OGSI.NET é uma extensão para dispositivos móveis do OGSI.NET [8], uma das implementações da especificação OGSI (*Open Grid Service Infrastructure*) [39]. Os objetivos deste *framework* são:

- Construir uma plataforma que forneça suporte à colaboração entre um ou mais dispositivos móveis na resolução de problemas;
- Promover a colaboração entre dispositivos móveis e computadores *desktops* ou servidores participantes de ambientes de grade já existentes, abrindo a possibilidade de maximizar a utilização de recursos limitados dos aparelhos portáteis;
- Operar em um grande número de dispositivos móveis;
- Tratar a limitação dos recursos dos dispositivos móveis, com o intuito de oferecer um uso destes recursos de forma otimizada.

O projeto segue os padrões da especificação OGSI, tanto para definir um conjunto de protocolos comuns que possibilitem a colaboração entre os dispositivos móveis, quanto para que o Mobile OGSI.NET possa interoperar com outras implementações tradicionais do OGSI para redes fixas, como o GTK (*Globus Toolkit*) [13] e o próprio OGSI.NET.

O Mobile OGSI.NET foi desenvolvido para o sistema operacional Microsoft PocketPC 2003, por ser esse um dos sistema operacionais mais populares entre os dispositivos móveis. Sua implementação foi realizada no topo do *framework* .NET Compact, cabendo a este último agir como uma camada de *software* intermediária entre a aplicação e o sistema operacional.

Para contornar o problema dos recursos limitados dos dispositivos móveis que podem levar rapidamente a sua indisponibilidade, o Mobile OGSI.NET propõem processos de migração e execução distribuída. Para otimizar a utilização dos recursos dos dispositivos, o *framework* permite que as *tarefas* do usuário sejam executadas em diversos dispositivos, com cada dispositivo responsável por alguma parte da tarefa.

Na implementação atual, o Mobile OGSI.NET, deixa a cargo do usuário iniciar o processo de migração de processos quando percebe que os recursos do seu dispositivos estão se esgotando ou quando quiser desligar o aparelho. Um objetivo futuro do projeto é migrar silenciosamente os processos assim que recursos, como por exemplo, a reserva da bateria se torne muito baixa.

3.5.2 UNICORE

A abordagem proposta por Brooke e Parkin [5] apresenta a implementação de um cliente PDA de grade computacional que permite segurança na submissão de tarefas assíncronas, monitoração de tarefas, entrega de resultados destas tarefas e uma facilidade para *upload* e *download* de arquivos a partir de um PDA, usando a infra-estrutura de rede *wireless* e o *middleware* UNICORE [12]. Dois requisitos nortearam a implementação do cliente PDA para o *middleware* de grade UNICORE:

- Devido ao aspecto de mobilidade dos clientes que irão acessar a grade, que podem mudar de rede e de endereço no decorrer do tempo, o *middleware* de grade não pode depender de referência fixa para encontrar os clientes. Desta forma, a submissão de tarefas a partir do dispositivo móvel deve ser feita sempre de forma assíncrona, ou seja, a troca de mensagens entre cliente e a grade deve ser sempre iniciada pelo cliente, pois caso o processo de envio de mensagens seja iniciado pelo lado servidor, o endereço do cliente pode ser desconhecido, indisponível ou ter mudado.
- Para acomodar os clientes móveis, o impacto em termos de modificação dos componente já implementados do *middleware* UNICORE ou inclusão de novos componentes deve ser o menor possível.

O *middleware* UNICORE segue uma arquitetura cliente-servidor em duas camadas. A camada cliente fornece a interface para o usuário e é usada para preparar e submeter as tarefas para a segunda camada, o UNICORE site (Usite), responsável pela execução das tarefas. O UNICORE obrigatoriamente exige que a submissão de tarefas seja realizada através da comunicação entre o cliente e um *gateway* seguro, através de uma conexão SSL. Como a interface fornecida por este *gateway* é a mesma para todos os clientes do UNICORE, não foi necessário a adição de nenhum componente adicional à arquitetura deste *middleware* para acomodar dispositivos móveis. A utilização do SSL garante que todas as comunicações entre os clientes e o *gateway* são autenticadas e encriptadas. A autenticação permite que os clientes possam mudar de rede ou de endereço e ainda assim conseguir recuperar de forma segura os resultados das computações submetidas de forma assíncrona.

O aspecto de suporte a segurança no cliente PDA é o principal foco deste projeto e norteou a escolha das ferramentas utilizadas no desenvolvimento do cliente PDA. A implementação da segurança especificada pelo UNICORE foi alcançada utilizando *J2ME CDC PersonalProfile*, garantindo uma comunicação segura entre o cliente PDA e o *middleware* UNICORE, através da criação de conexões SSL utilizando-se o conjunto de classes do pacote *JAVA Secure Socket Extension* (JSSE). A interface do usuário que permite preparar tarefas, submeter tarefas e visualizar os resultados obtidos na execução das tarefas foi desenvolvida utilizando o *toolkit* AWT, já que o *J2ME CDC PersonalProfile* não disponibiliza a API *Swing* para criação de interfaces avançadas com o usuário.

3.5.3 Condor

O sistema de grade Condor [28, 38] provê uma metodologia hierárquica para acesso à grade por dispositivos móveis descrita em [20]. Nesta abordagem, as plataformas móveis são classificadas em três categorias ou camadas hierárquicas, levando-se em conta características de *hardware* específicas de cada dispositivo, como pode ser observado na Figura 3.17.

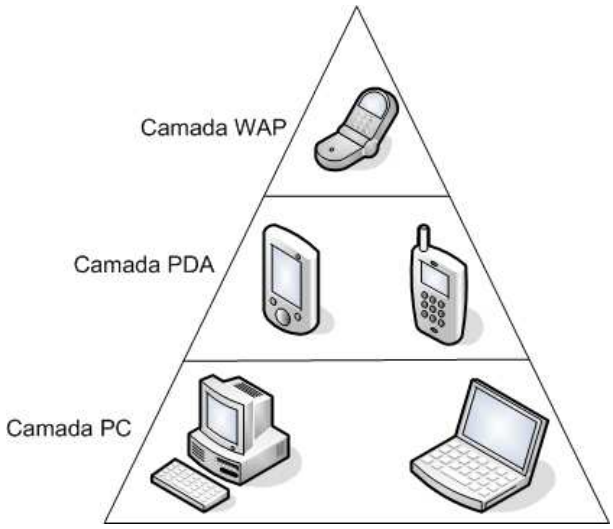

Figura 3.17: Camadas hierárquicas de dispositivos no Condor

Como resultado, foram determinadas que funções da grade podem ser disponibilizadas a cada camada da hierarquia e foram desenvolvidas interfaces específicas de interação dos usuários com o Condor para cada categoria.

Na base da pirâmide estão os *desktops* e *laptops*, que possuem acesso total

aos serviços do Condor através de uma interface de linha de comandos, que permite exibir a lista de tarefas atualmente em execução, listar estatísticas de tarefas concluídas, submeter novas tarefas para execução, parar, dar pausas, retomar a execução de uma tarefa e apresentar seus resultados.

Na camada intermediária estão localizados os PDAs, que podem checar o estado da execução de uma tarefa, parar, dar pausas e retomar a execução das tarefas, submeter novas tarefas, receber avisos via e-mail quando uma tarefa tiver terminado sua execução, assim como realizar pequenas modificações no código dos programas a serem submetidos e em seus arquivos de entrada.

No topo da pirâmide está a camada dos telefones celulares com suporte a WAP. Assim como os dispositivos das camadas inferiores, os telefones celulares podem checar o estado de execução de uma tarefa, parar, dar pausas e retomar a execução das tarefa, submeter novas tarefas e receber avisos via SMS quando uma tarefa tiver terminado sua execução. Contudo, devido a limitações de interface, as funcionalidades de edição do código dos programas e seus arquivos de entrada não são disponibilizadas.

Uma característica fundamental desta abordagem consiste no fato de que as funcionalidades providas para dispositivos localizados em uma determinada camada são disponibilizadas também para os dispositivos das camadas mais baixas. Por exemplo, todos os serviços que podem ser obtidos através de um telefone celular, também podem ser obtidos com um PDA, mas o contrário não é garantido.

Para as as duas camadas superiores (PDAs e telefones celulares WAP) foram desenvolvidos *front-ends* para acesso aos serviços do Condor, acessados através de navegadores *Web* com suporte a HTML, WML e XML, instalados nos próprios dispositivos. O protocolo HTTP foi o protocolo adotado para a arquitetura proposta por possuir a característica de ser independente de plataforma e por existir uma ampla gama de *browsers* para dispositivos móveis que implementam este protocolo nativamente ou que utilizam *gateways* que atuam como tradutores entre o protocolo HTTP e protocolos específicos como o WPA e o I-Mode.

Do lado servidor, um conjunto de *scripts* Pearl/CGI são utilizados para executar os comandos do Condor e enviar os resultados de volta para os clientes. O servidor pode produzir diversos padrões de páginas de saída de acordo com a

identificação do *browser* e do tipo do dispositivo cliente enviados pelo protocolo HTTP.

3.5.4 Resumo Comparativo

Esta seção apresenta uma análise comparativa dos trabalhos apresentados em relação ao MInteGrade. Tomamos como base os seguintes critérios de comparação: dispositivos alvo (celulares, PDA's, *smartphones* e outros), plataforma de desenvolvimento, suporte a mecanismos de segurança, suporte à adaptações baseadas em contexto e suporte a desconexão dos clientes móveis.

Podemos observar na Tabela 3.2 que a maioria dos projetos apresentados desenvolveram clientes para acesso à grade para dispositivos móveis com maior capacidade de recursos, como PDAs e *SmartPhones*. Com exceção do projeto Condor, que disponibiliza uma interface baseada em páginas ML, com funcionalidades limitadas, para telefones celulares com suporte a WPA. Vale ressaltar que apesar da interface gráfica `MASCT` do projeto MInteGrade ser voltada a PDAs e *SmartPhones*, uma extensão desta arquitetura, que será explicitada no Capítulo 4, foi desenvolvida para permitir a submissão de aplicações à grade, acompanhamento da execução e recuperação de resultados, por telefones celulares que possuam interface *bluetooth*.

É interessante observar que os projetos MInteGrade e UNICORE foram desenvolvidos em J2ME, em uma clara demonstração de preocupação com a portabilidade, diferentemente do projeto Mobile OGSI.NET, que é focando para dispositivos móveis com o sistema operacional Windows Mobile. O projeto Condor também garante portabilidade de suas interfaces móveis, ao desenvolvê-las utilizando linguagens de marcação de uso geral, HTML, WML, XML, que são interpretadas pela maioria dos navegadores presentes nos dispositivos móveis.

Apesar de não garantir que as comunicações entre os clientes e o *middleware* de grade sejam autenticadas e encriptadas como no UNICORE, dentre os projetos avaliados, o MInteGrade é o único com suporte à adaptações dos resultados retornados pela grade de acordo com informações de contexto monitorados no dispositivo móvel. Outro aspecto a ser levado em consideração é que todos esse projetos assumem que os dispositivos móveis têm acesso permanente a rede durante suas interações não tratando, como no MInteGrade, períodos de desconexão ou conectividade intermitente dos aparelhos.

Projeto	**MInteGrade**	**Mobile OGSI.NET**	**UNICORE**	**Condor**
Dispositivos	PDAs e Smartphones c/ J2ME CDC/PP	PDAs e Smartphones com Windows Mobile	PDA's c/ J2ME CDC/PP	PDAs e telefones celulares WPA c/ navegador web
Plataforma	J2ME	.NET	J2ME	Páginas HTML, WML, scripts Pearl/CGI
Suporte a mecanismos de segurança	Não	Não	Sim	Não
Suporte a adaptações baseadas em contexto	Sim	Não	Não	Não
Suporte a períodos de desconexão	Sim	Não	Não	Não

Tabela 3.2: Resumo comparativo dos trabalhos relacionados

3.6 Conclusões

Este capítulo apresentou uma infra-estrutura de *software* através da qual clientes de dispositivos móveis que se conectam à redes sem fio IEEE 802.11 em modo infra-estruturado podem ter acesso à grade computacional disponibilizada pelo projeto InteGrade. Através desta infra-estrutura, usuários de aparelhos móveis podem solicitar a execução de aplicações, realizar o acompanhamento da execução de aplicações e visualizar o resultado de computações já concluídas utilizando a ferramenta `MASCT`.

A arquitetura da infra-estrutura de *software* proposta é baseada no modelo cliente-*proxy*-servidor, tornado os problemas relacionados ao uso de dispositivos móveis e redes sem fio transparentes aos componentes já implementados do *middleware* InteGrade. A infra-estrutura utiliza componentes do *framework* MoCA para tratar períodos de desconexão e realizar adaptações no conteúdo dos resultados das computações retornados pela grade.

Além das adaptações já definidas na arquitetura do MInteGrade para realização de *cache* de mensagens e adaptação do tamanho das imagens, o administrador da grade pode definir, se necessário, outros contextos de interesse a serem monitorados nos dispositivos móveis (percentual de utilização da CPU, quantidade de memória disponível, nível de utilização da bateria, etc.) e as respectivas adaptações dinâmicas que devem ser realizadas quando houver alterações nas informações desses dados de contexto.

4 Compartilhamento do Acesso ao InteGrade em Redes Ad hoc

O objetivo deste capítulo é apresentar como a infra-estrutura de *software* apresentada no Capítulo 3 foi estendida para permitir que usuários móveis conectados através de redes sem fio em modo Ad hoc compartilhassem um serviço móvel de acesso aos serviços de grade oferecidos pelo InteGrade. Através deste mecanismo, os clientes móveis, a exemplo do que acontece para os clientes de redes sem fio infra-estruturadas, podem solicitar a execução de aplicações na grade, realizar o acompanhamento da execução das aplicações e visualizar o resultado de computações já concluídas. A arquitetura proposta leva em consideração a topologia altamente dinâmica das redes Ad hoc e possui mecanismos de ciência de contexto, provendo suporte a períodos de desconexão e baixa conectividade, bem como a adaptação de conteúdo dos resultados das computações realizadas pela grade.

Nas seções deste capítulo serão descritos os requisitos de projeto que nortearam o desenvolvimento do *software*, o SNU, um *middleware* voltado à construção de aplicações para redes Ad hoc de curto alcance, a arquitetura do serviço MInteGrade Ad hoc e detalhes de sua implementação, além de trabalhos relacionados ao mesmo.

4.1 Requisitos do Projeto

A solução arquitetural proposta para o MInteGrade em redes Ad hoc foi guiada por diversos requisitos impostos por características intrínsecas às redes móveis Ad hoc. Uma característica considerada é a alta dinamicidade da rede, cuja topologia é variável. Dispositivos podem entrar e sair da rede a qualquer momento e sem prévio aviso, passar por períodos de desconexão ou de baixa conectividade, ou até mesmo falharem. Assim, tanto provedores de serviço quanto seus consumidores podem inesperadamente deixar a rede no meio de uma sequência de interações. Outra característica levada em consideração é o suporte à grande heterogeneidade tecnológica dos dispositivos, tanto com relação a propriedades de *hardware* quanto de

software, decorrente das diversas famílias de equipamentos. Assim, a visualização dos resultados da execução de uma aplicação na grade pode requerer uma adaptação de seu conteúdo. Uma imagem, por exemplo, deve ser adaptada à resolução da tela do dispositivo. Um outro aspecto levado em consideração são eventuais períodos de desconexão que impossibilitem temporariamente a comunicação entre o *proxy* e os provedores do acesso a grade na rede Ad hoc.

4.2 Introdução ao SNU

A arquitetura do serviço de compartilhamento do acesso a grades de computadores por dispositivos móveis conectados através de uma rede Ad hoc foi implementada com o auxílio do *middleware* SNU (Spontaneous Network Utilities) [32]. O SNU é um *middleware* que tem por objetivo principal facilitar a construção de aplicações voltadas ao compartilhamento de conteúdo em redes Ad hoc espontâneas de curto alcance.

A arquitetura do SNU é estruturada em camadas, onde cada camada é composta por componentes que tratam de assuntos relacionados ao mesmo nível de abstração. Serviços fornecidos em uma camada podem ser acessados pela camada superior através de uma API bem definida. Isto fornece a possibilidade de substituir funcionalidades, mudando componentes sem que seus clientes sejam afetados.

A Figura 4.1 mostra as camadas que compõem o *middleware* SNU. A camada de rede realiza o gerenciamento dos dispositivos ao redor, a descoberta dos serviços por eles ofertados, o envio e recebimento de dados através da rede, assim como a compressão e criptografia dos mesmos. A implementação atual desta camada é baseada na tecnologia *bluetooth*, mas a forma como foi desenvolvida permite a utilização de outras tecnologias de rede, através da adoção de uma interface bem definida.

A Camada de Serviços contém todos os serviços disponibilizados pelo *middleware*. Como mostra a Figura 4.1, esta é a única camada visível para as aplicações, sendo responsável por fornecer todas as interfaces utilizadas pelos desenvolvedores ao criar aplicações para os usuários finais. Existem serviços que lidam com o gerenciamento de contatos e grupos, compartilhamento de dados que podem ser organizados em coleções, gerenciamento de informações de contexto da aplicação e

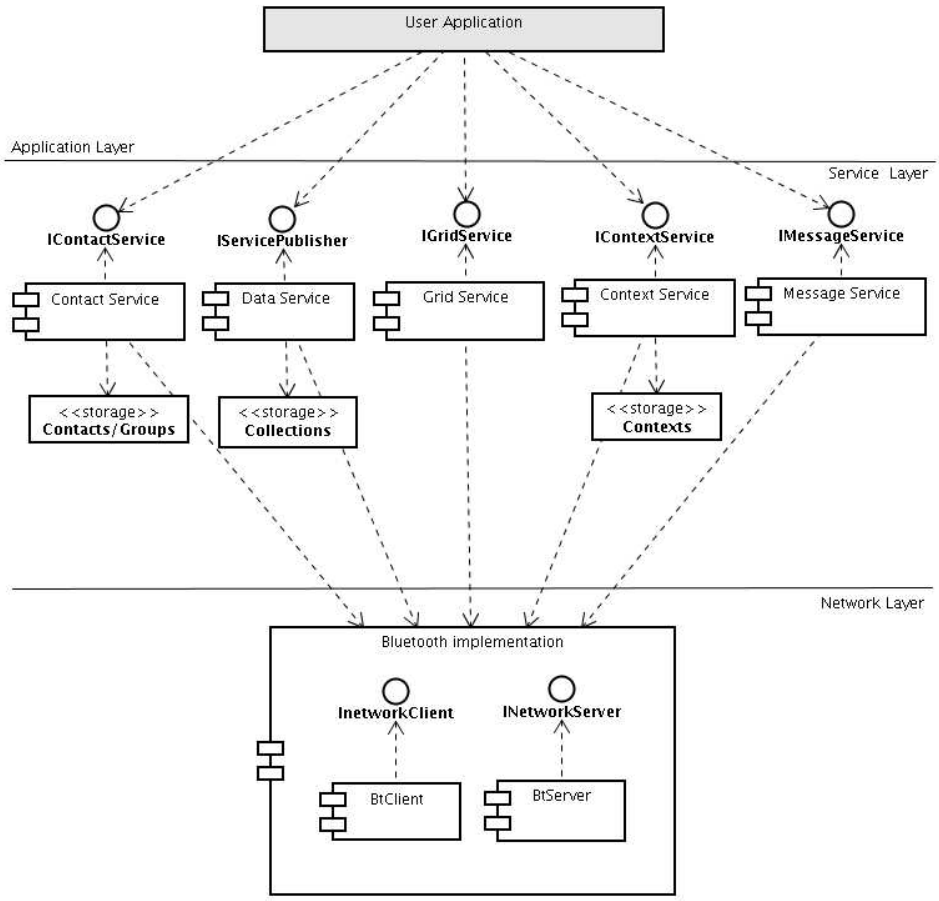

Figura 4.1: Diagrama de componentes do SNU.

troca de mensagens de texto. Nesta camada introduzimos o serviço de acesso à grade, um dos componentes da arquitetura proposta neste trabalho que será detalhada a seguir.

4.3 Acesso ao InteGrade por Dispositivos Móveis em Redes Ad hoc

A arquitetura de *software* proposta neste trabalho tem por objetivo o compartilhamento em redes Ad hoc de um serviço de acesso a grades de computadores estabelecido através do *middleware* InteGrade, conforme ilustrado na Figura 4.2.

Esta arquitetura permite a qualquer dispositivo em uma rede Ad hoc, solicitar a execução de aplicações à grade, acompanhar o estado de execuções solicitadas e visualizar os resultados das mesmas quando do seu término. Para tanto, pelo menos um dispositivo móvel pertencente à rede Ad hoc deve funcionar como provedor deste

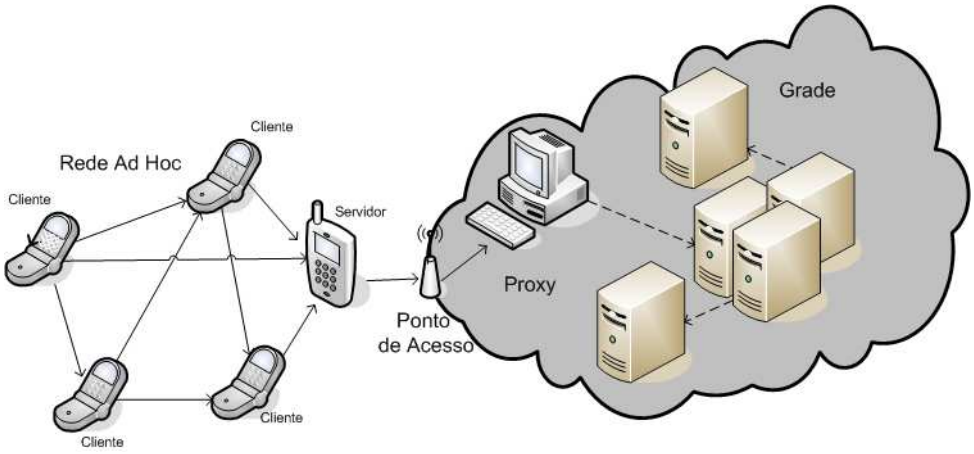

Figura 4.2: Acesso a grade através de redes Ad hoc.

serviço. Este provisionamento é baseado no trabalho para acesso ao serviços do *middleware* InteGrade por dispositivos móveis em redes sem fio IEEE 802.11 infra-estruturadas, apresentado no Capítulo 3, que adota o padrão arquitetural cliente / *proxy* / servidor [34]. O componente *proxy* executa em uma rede cabeada e é responsável por realizar a interação entre os clientes móveis e o servidor (a grade), tornando os problemas decorrentes do uso de enlaces sem fio e dispositivos móveis transparentes ao *middleware* da grade. Como ilustrado na Figura 4.2, a comunicação com o *proxy* é realizada através de uma rede infra-estruturada IEEE 802.11, portanto, o provedor do serviço de compartilhamento do acesso a grade em redes Ad hoc deve também possuir uma interface de rede IEEE 802.11, atuando como ponte entre a rede Ad hoc e a rede fixa.

4.4 Descrição da Arquitetura Proposta

A infra-estrutura de *software* proposta é composta pelos componentes ilustrados na Figura 4.3. Dois componentes foram implementados na camada de serviço do *middleware* SNU. O `GridClient` executa nos dispositivos que desejam obter acesso aos serviços do InteGrade. Utilizando os serviços providos pela camada de rede do SNU, este componente é responsável por descobrir, entre os dispositivos que compõem a rede Ad hoc, um nó que forneça o serviço de acesso à grade. Através de um protocolo de registro junto ao provedor do serviço, o cliente passa a poder requisitar os serviços do InteGrade. O outro componente implementado na camada de serviço do

SNU denomina-se `GridServer`. Ele é executado no dispositivo provedor do serviço de acesso à grade e sua função é receber as solicitações dos clientes, formatar as mensagens recebidas como requisições de serviço à grade e enviá-las ao *middleware* InteGrade através de uma rede infra-estruturada IEEE 802.11. O `GridServer` deve também retornar o resultado do processamento das solicitações ao cliente. Para tanto, ele deve manter um mapeamento requisição/resposta, associando-as ao cliente responsável pela requisição, já que o `GridServer` pode estar atuando em favor de diversos clientes.

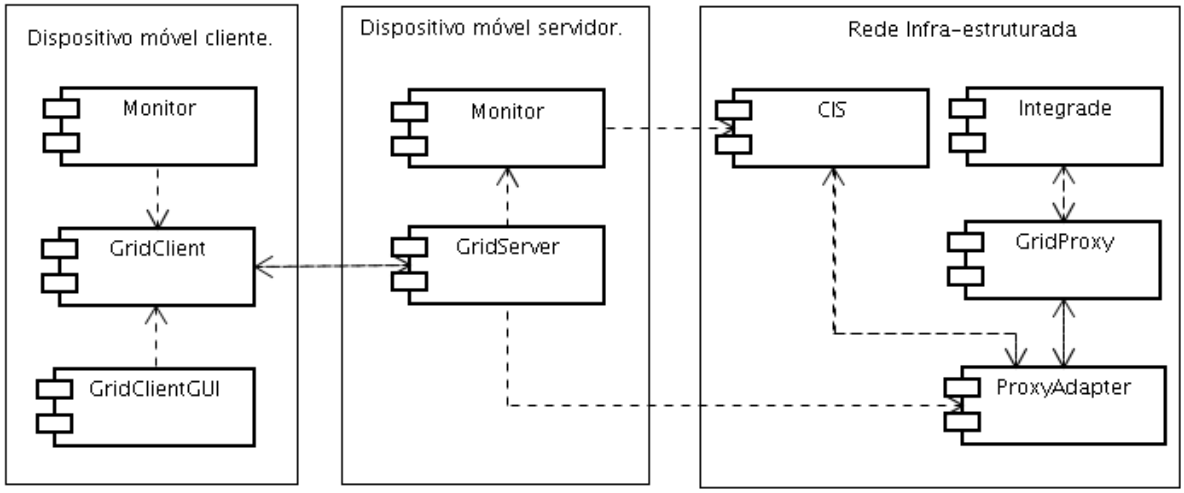

Figura 4.3: Componentes da arquitetura de acesso a grade em redes Ad hoc

A arquitetura proposta compreende também componentes que executam em nós fixos. O componente `ProxyAdapter` intermedeia toda a comunicação entre os dispositivos móveis que executam o serviço `GridServer` e a grade, tratando eventuais desconexões destes dispositivos através do armazenamento em uma *cache* das mensagens direcionadas a eles até que a conectividade seja restabelecida. Ele também é responsável pela adaptação do conteúdo dos resultados das computações submetidas à grade, de modo a permitir sua exibição nos dispositivos móveis de acordo com suas propriedades de *hardware* e *software*. O `GridProxy` é o componente responsável por tornar os dispositivos móveis transparentes aos demais componentes do *middleware* InteGrade. Este componente recebe as requisições enviadas pelos dispositivos móveis, formata estas requisições de acordo com o padrão de comunicação utilizado pelo *middleware* InteGrade e as encaminha para execução. As notificações de aplicações finalizadas são também encaminhadas pela grade a este componente para serem, então, enviadas aos dispositivos móveis que as requisitaram. Do ponto de vista da

grade, o `GridProxy` é visto como o solicitante das requisições, tornando os problemas decorrentes da mobilidade transparentes aos componentes do InteGrade.

4.5 Ciência de Contexto

Um dos requisitos da arquitetura proposta é permitir que o resultado das computações realizadas pelo InteGrade possam ser visualizadas em uma grande variedade de dispositivos. Para tanto, o conteúdo deve ser adaptado levando-se em consideração aspectos de contexto computacional dos dispositivos, como diferentes resoluções de tela.

Um *daemon* `Monitor` que executa nos dispositivos móveis clientes coleta informações estáticas, como o fabricante e o modelo do dispositivo. Estas informações de contexto estático dos nós clientes são repassadas ao nó provedor do serviço na rede Ad hoc como parte do processo de registro dos mesmos. Do lado servidor, um serviço `Monitor` recebe as informações de contexto estáticos dos clientes móveis que são registrados pelos provedor de serviço e as envia periodicamente para o serviço de provisão de contexto `CIS` (*Context Information Service*). O `CIS`, como descrito na Seção 3.2, utiliza as especificações CC/PP como fonte de informação das características estáticas dos dispositivos. Ao receber as informações de fabricante e modelo dos dispositivos móveis, o `CIS` busca o documento UAProf que descreve as características de *hardware* e *software* daquele dispositivo em seu repositório de arquivos, interpreta o documento, adequa as informações estáticas ali presentes (tipo do dispositivo, resolução de tela, capacidade de exibir cores e quantidades de bits por pixel) ao seu modelo de banco de dados e armazena estas informações.

Os monitores que executam nos nós clientes e servidores da rede Ad hoc, foram desenvolvidos como parte da nossa arquitetura, o serviço de provisão de contexto utilizado (`CIS`) faz parte da arquitetura MoCA [36] (*Mobile Collaboration Architecture*), desenvolvida na PUC-Rio.

O `CIS` fornece APIs de acesso a contexto que são utilizadas na arquitetura proposta pelo componente `ProxyAdapter` para obtenção do contexto dinâmico e estático dos dispositivos. O `ProxyAdapter` usa essas informações para adaptar o conteúdo das mensagens retornadas pela grade aos dispositivos móveis, conforme

descrito na Subseção 3.4.2.

4.6 Interação entre Componentes

Para ilustrar como os componentes da arquitetura proposta interagem, esta seção descreve o conjunto de passos que clientes e servidores da rede Ad hoc devem realizar para obter acesso aos serviços de grade. Isto inclui desde o processo de registro do cliente no provedor de serviço até como ocorre a submissão de uma requisição propriamente dita. Esta seção também descreve o que acontece caso clientes ou servidores se tornem permanentemente ou temporariamente indisponíveis durante estas sequências de interações.

4.6.1 Registro de Clientes na Rede Ad hoc

Para que os clientes da rede Ad hoc possam utilizar o serviço de acesso à grade é necessário que estes se registrem junto a provedores desse serviço na rede, que passarão a atuar como representantes desses dispositivos junto ao *middleware*. Para tanto, um protocolo de registro de clientes, ilustrado na Figura 4.4, foi definido. (1) Ao iniciar, os clientes da rede Ad hoc realizam uma busca por dispositivos ao redor que disponibilizem o serviço de acesso à grade; (2) Ao encontrar um desses dispositivos, o `GridClient` envia uma mensagem de solicitação de registro ao provedor; (3) Ao receber uma solicitação de registro de um cliente, o `GridServer` formata a mensagem de registro recebida segundo o padrão de comunicação utilizado no *middleware* MoCA e envia a solicitação de registro do cliente ao `ProxyAdapter` que executa na rede fixa; (4) O `GridServer` repassa, nesse momento, informações de contexto do cliente da rede Ad hoc (identificador da interface de rede, fabricante e modelo do dispositivo) a um serviço monitor, que fica responsável por enviá-las ao serviço de Contexto da MoCA a intervalos de tempo regulares (5). O envio dessas informações de contexto é necessário tanto para que o `ProxyAdapter` possa inferir quando os clientes da rede Ad hoc se tornam indisponível na rede e, portanto, mensagens destinadas a eles devem ser armazenadas em *cache*, quanto para obter informações de contexto estáticas dos clientes, como a resolução de tela e assim poder realizar o redimensionamento das mensagens contendo imagens retornadas pela grade; (6) Ao receber uma mensagem

de registro, o `ProxyAdapter` associará em sua base de clientes o identificador de rede do dispositivo da rede Ad hoc que realizou a requisição ao IP do dispositivo que executa o `GridServer`. Dessa forma, todas as mensagem oriundas da Grade serão repassadas pelo `ProxyAdapter` ao `GridServer` que atua em favor do cliente; (7) O `ProxyAdapter` informará ao `CIS` que está interessado nas informações de contexto enviadas para o dispositivo que acabou de ser registrado; (8) e enviará uma resposta ao `GridServer` informando que o dispositivo foi registrado; (9) Após receber a confirmação de registro do cliente no `ProxyAdapter`, o `GridServer` registrará o cliente em sua base de dados e informará ao `GridClient` que ele passará a atuar como seu representante no acesso a grade (10).

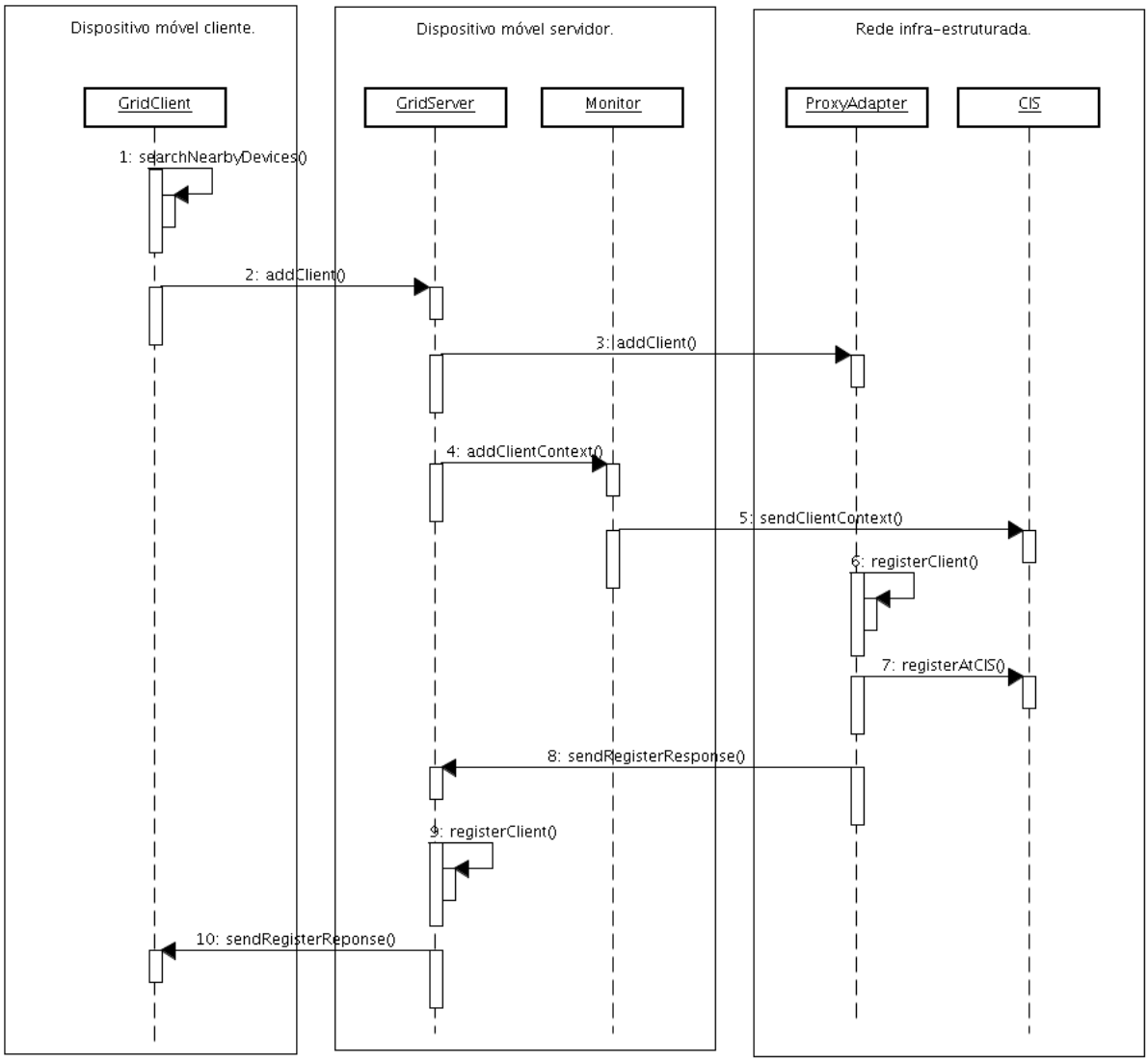

Figura 4.4: Diagrama de registro de um cliente da rede Ad hoc junto a um provedor de serviço de acesso a grade

Para se prevenir contra eventuais perdas de mensagens provenientes da rede fixa ou falhas no `ProxyAdapter`, o `GridServer` inicia um *timeout* ao enviar uma solicitação de registro de clientes. Caso a mensagem de resposta de registro do cliente não seja entregue pelo `ProxyAdapter` ao `GridServer` até o *timeout* definido expirar, este último informará ao cliente que este não pode ser registrado e informará ao serviço monitor que as informações de contexto para aquele dispositivo não necessitam mais serem enviadas.

Caso não consiga enviar uma requisição de registro ou qualquer outra requisição de serviço à grade devido a problemas de comunicação com a rede fixa, o `GridServer` irá mudar seu estado para desconectado. Quanto isto ocorrer, este componente irá remover todos os seus clientes registrados e os informará da situação, passando a recusar novas solicitações de registro que porventura venha a receber. A partir do momento em que percebe que perdeu comunicação com a rede fixa, o `GridServer` iniciará o envio de mensagens *"Are You Alive"* para o `ProxyAdapter` para que possa perceber quando sua conectividade for restabelecida e, portanto, estiver apto a aceitar novos registros de clientes.

4.6.2 Solicitação de Execução de uma Aplicação à Grade

Como ilustração de como uma solicitação de serviço à grade ocorre, o texto a seguir descreve a sequência de passos realizados quando um dispositivo móvel pertencente à rede Ad hoc solicita a execução de uma aplicação à grade. Esta sequência de interações é ilustrada pela Figura 4.6. (1) O usuário do dispositivo móvel indica, através da interface gráfica `GridClientGUI`, mostrada na Figura 4.5, qual aplicação deseja que seja executada, informando também parâmetros a serem utilizados na sua execução (como os argumentos, preferências, restrições, arquivos de entrada e saída);

(2) O componente `GridClient` recebe esta solicitação e a repassa para o `GridServer` no qual está registrado; (3) O `GridServer` formata a mensagem recebida e a encaminha, através da interface de rede 802.11, ao `ProxyAdapter`, que executa na rede fixa. (4) O `ProxyAdapter` repassa a mensagem ao `GridProxy`. Este último componente mantém um banco de dados que relaciona cada solicitação a um dos serviços da grade ao dispositivo móvel responsável por sua emissão. Além disso, o `GridProxy` é responsável por outras tarefas como, por exemplo, baixar

(a) (b)

Figura 4.5: Configuração dos parâmetros da aplicação (a) e dos arquivos de saída (b)

da rede os arquivos contendo os dados de entrada da aplicação a ser executada na grade, localizados através das URLs fornecidas pelo usuário do dispositivo móvel responsável pela submissão da aplicação; (5) O `GridProxy` insere as informações referentes à solicitação em seu banco de dados; (6) Ele também formata os dados constantes da requisição de execução de acordo com o padrão de comunicação utilizado pelo InteGrade e os envia para a grade, tornando transparente o fato da requisição ter sido realizada através de um dispositivo móvel; (7) Após constatar a disponibilidade de recursos para executar a aplicação solicitada, a grade retorna uma mensagem de notificação informando que a requisição foi aceita para execução. Esta mensagem é enviada ao `GridProxy`; (8) O estado da entrada no banco de dados do `GridProxy` referente àquela requisição é atualizado; (9) A mensagem de notificação é, então, repassada ao `ProxyAdapter` (10) e enviada ao dispositivo móvel que está provendo o serviço de acesso à grade; (11) Este, por sua vez, ao receber a mensagem verifica para qual dispositivo da rede Ad hoc ela é destinada, formata a mensagem de acordo com o padrão de comunicação adotado pelo SNU e a envia para o serviço `GridClient` do dispositivo cliente; (12) Finalmente, ao receber a mensagem de notificação, o `GridClient` chama o método da interface gráfica `GridClientGUI` responsável por mostrar ao usuário o estado da solicitação da execução.

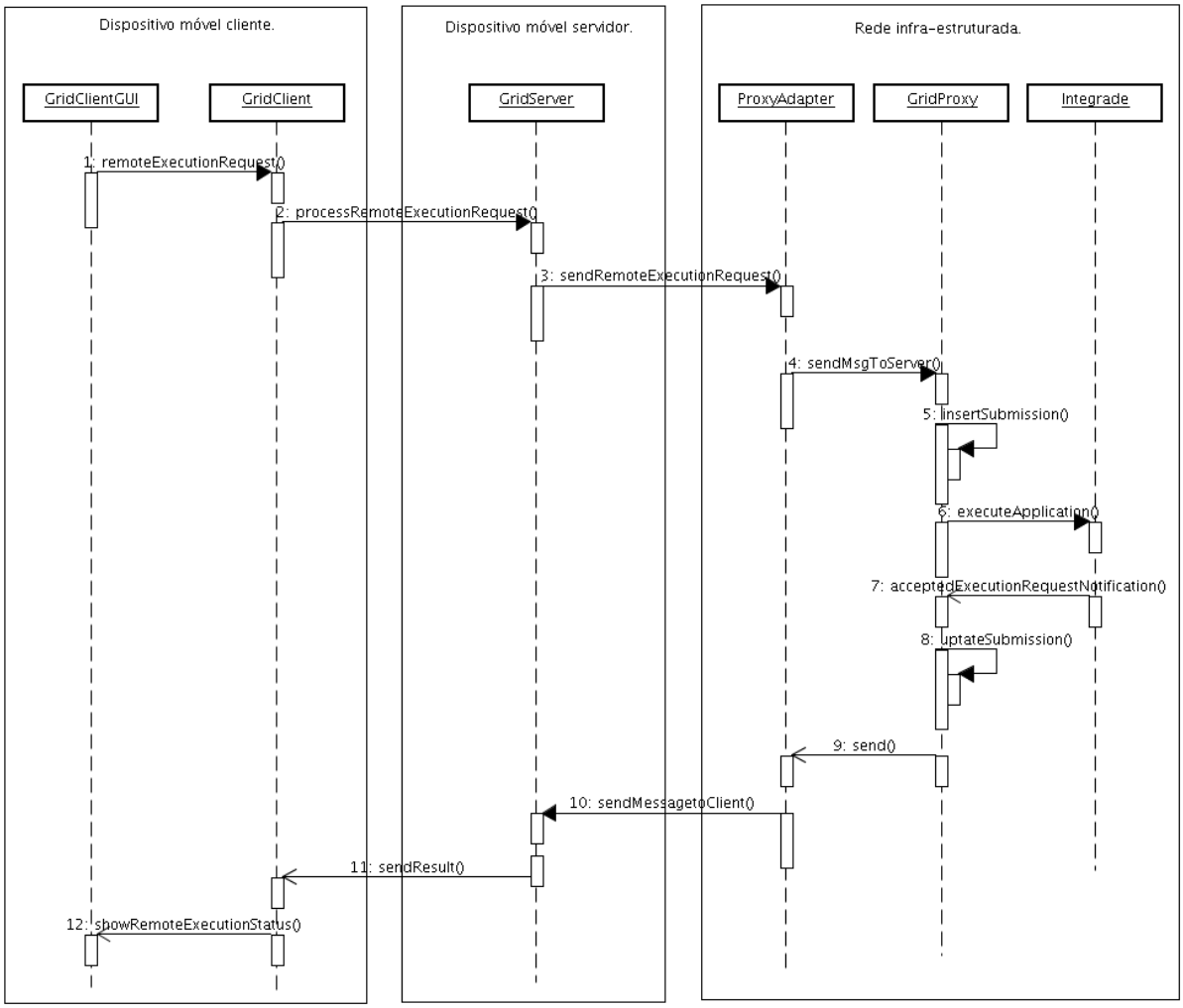

Figura 4.6: Diagrama de execução de uma aplicação na grade submetida utilizando dispositivos conectados a uma rede ad hoc.

Além da solicitação para execução de aplicações na grade, os clientes da rede Ad hoc podem realizar o acompanhamento da execução das aplicações e visualizar o resultado de computações já concluídas enviando requisições para grade que seguem o mesmo trajeto da requisição para execução de aplicações registradas.

4.6.3 Lidando com Variações na Topologia da Rede Ad hoc

Para lidar com a alta dinamicidade das redes Ad hoc, prevenindo-se contra eventuais desconexões, períodos de conectividade intermitente ou mesmo falha dos dispositivos, a arquitetura desenvolvida incorpora mecanismos de tolerância à falhas que tratam situações onde provedores do serviço de acesso à grade e os consumidores desse serviço podem inesperadamente deixar a rede no meio de uma sequência de

interações.

O `GridClient`, após obter registro em um provedor de serviço, verificará periodicamente a sua disponibilidade na rede através de uma busca por dispositivos vizinhos. Caso, ao realizar a busca, o `GridClient` note a ausência do dispositivo servidor a quem se encontra registrado, um novo provedor de serviço será selecionado, se houver, para que o acesso aos serviços de grade seja mantido. Ao selecionar um novo provedor de serviço, o `GridClient` enviará uma requisição para obter o registro do cliente no novo `GridServer`. Caso não consiga registrar-se no provedor de serviço selecionado, uma nova busca por dispositivos provedores do serviço de acesso à grade será realizada. Este processo será repetido até que se esgotem as possibilidade de provedores de serviço disponíveis na rede para realizar o registro.

Um ícone na interface gráfica do usuário indica a disponibilidade ou não do acesso aos serviços da grade, conforme ilustrado na Figura 4.7. Este ícone permanece verde enquanto o dispositivo estiver registrado em um provedor de serviço e muda para vermelho sempre que a conectividade com o mesmo for perdida.

(a) (b)

Figura 4.7: Interface do GridClient indicando que o cliente está registrado em um provedor de serviço (a) e indicando que a conectividade foi perdida (b)

Quando o cliente realiza a troca de provedor de serviço, o `GridServer` do

novo provedor informará ao `ProxyAdapter` na rede fixa que passará a atuar em nome do cliente da rede Ad hoc. O `ProxyAdapter` atualizará o registro deste cliente com o endereço de comunicação do novo `GridServer` e passará a encaminhar as mensagens destinadas ao cliente através desse endereço, inclusive as mensagens que porventura já estejam armazenadas em *cache*.

O `GridServer` também monitora a rede Ad hoc para verificar se seus clientes registrados ainda se encontram disponíveis, realizando uma busca periódica por este dispositivos a cada 30 segundos. Ao registrar um cliente, o provedor de serviço define um parâmetro, chamado a partir de agora de "vivacidade", que indicará a probabilidade do dispositivo estar permanentemente incomunicável com o servidor. A vivacidade do cliente é definida com um valor inicial de 10 unidades. A cada busca periódica realizada, o provedor de serviço verifica entre os seus clientes registrados quais não foram encontrados e decrementa sua vivacidade em uma unidade. Nesse momento, o `GridServer` também informa ao serviço monitor que as informações de contexto dos clientes indisponíveis devem parar de ser enviadas ao serviço de contexto da MoCA. Esta passo é necessário para que o `CIS`, ao deixar de receber as informações periódicas de contexto dos clientes, atualize seus *status* de conectividade para indisponível. De posse das informações sobre a conectividade dos clientes, obtidas junto ao `CIS`, o `ProxyAdapter` passa a enfileirar mensagens destinadas aos clientes indisponíveis na rede Ad hoc. Caso a vivacidade de um cliente após sucessivas buscas alcance o valor zero, este será dado como incomunicável pelo servidor e será desregistrado. Se o cliente for reencontrado antes que sua vivacidade atinja o valor 0, seu valor será atualizado novamente para 10 unidades.

Caso o próprio provedor de serviço venha a ficar indisponível, as informações de contexto de todos os seus clientes registrados deixarão de ser enviadas pelo serviço `Monitor`. O `CIS` informará ao `ProxyAdapter` que não há conectividade para estes clientes, que passará a armazenar em *cache* informações oriundas da grade que seriam enviadas aos clientes do provedor de serviço indisponível.

4.7 Trabalhos Relacionados

Diversos projetos abordam o uso de dispositivos móveis em grades fixas - tipicamente, a partir de redes infra-estruturadas [8, 5, 20, 6]. Contudo a interoperação entre dispositivos móveis conectados a uma rede Ad hoc com recursos de uma grade fixa é um problema ainda pouco abordado na literatura, problema este decorrente do fato de se tratarem de redes distintas, com padrões e protocolos de controle bem diferentes.

Em Bastos at al. [3] uma proposta para o acesso de dispositivos móveis participantes de uma grade móvel aos recursos de uma grade fixa é apresentada. O trabalho usa como base os *middlewares* MoGrid [26] para grades móveis Ad hoc, e o Globus Toolkit (GTK) [13] para grades fixas.

O *middleware* MoGrid permite a organização de dispositivos móveis como uma grade puramente Ad hoc. Esse *middleware* define dois papéis principais em uma grade móvel: iniciadores e colaboradores. Iniciadores são os dispositivos responsáveis pela submissão de tarefas, e colaboradores os responsáveis pela disponibilização de recursos computacionais para a execução dessas tarefas. Já o *middleware* Globus Toolkit (GTK) permite que tarefas computacionais sejam submetidas à máquinas de uma grade, de acordo com os recursos disponíveis nessa máquinas e com as necessidades específicas da tarefa em questão.

Na solução proposta neste projeto, os dispositivos móveis atuando como iniciadores no MoGrid podem também participar como clientes no contexto de uma ou mais grades fixas baseadas no GTK. Assim como em nossa abordagem, a solução proposta nesta caso segue o padrão cliente/*proxy*/servidor. Um *proxy* situado em uma máquina da rede fixa é responsável por intermediar a comunicação entre os dispositivos da rede Ad hoc com os dispositivos da rede fixa como ilustrado na Figura 4.8.

Contudo, como as máquinas da rede fixa onde os *proxies* estão situados frequentemente não possuem interface de comunicação com a rede Ad hoc, um dispositivo móvel tem de atuar como ponto de acesso entre as duas redes, encaminhando as requisições realizadas para que possa ocorrer a interoperação. Como o GTK trabalha com um sistema de segurança baseado em certificados digitais para autorizar a

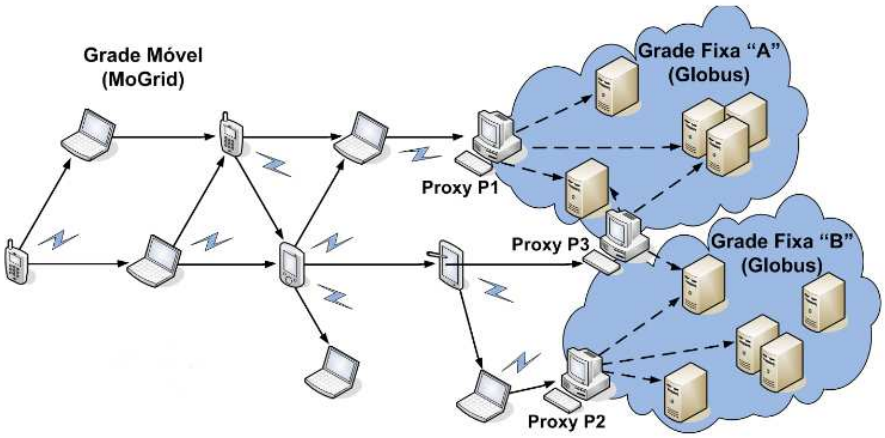

Figura 4.8: Interoperação entre uma grade MoGrid e uma grade GTK baseada em proxy

submissão de tarefas pelos usuários da grade, um sistema de autenticação foi criado no *proxy* dessa arquitetura, de modo a associar um dispositivo móvel da rede Ad hoc com um certificado de usuário válido da grade fixa. Entretanto, esse trabalho não se preocupa com questões relacionadas a tolerância a falhas dos dispositivos da rede Ad hoc que evitem a perda de resultados de execuções de tarefas devido a desconexões ou indisponibilidade dos dispositivos que atuam como ponto de acesso à rede fixa, ou a adaptação do conteúdo das respostas retornadas pela grade as características dos dispositivos.

4.8 Conclusões

Em contraste com demais abordagens encontradas na literatura, tipicamente focadas na integração de dispositivos móveis a grades através de redes infra-estruturadas, este capítulo apresentou uma infra-estrutura de *software* através da qual usuários móveis conectados através de uma rede Ad hoc podem compartilhar um serviço móvel de acesso aos serviços de grade oferecidos pelo *middleware* InteGrade. Através desta arquitetura, usuários de aparelhos móveis podem solicitar a execução de aplicações à grade, realizar o acompanhamento da execução de aplicações e visualizar o resultado de computações já concluídas.

Características intrínsecas a redes móveis Ad hoc, como a topologia altamente dinâmica da rede, onde dispositivos podem entrar e sair da rede a qualquer momento e sem prévio aviso, passar por períodos de baixa conectividade, ou até

mesmo falharem, guiaram a proposta arquitetural da solução. Outras características consideradas foram: o suporte a grande heterogeneidade tecnológica dos dispositivos decorrente das diversas famílias de equipamentos, através da adaptação do conteúdo de imagens retornadas pela grade as diferentes resoluções de telas dos aparelhos e o tratamento de eventuais períodos de desconexão que impossibilitem temporariamente a comunicação entre a grade e os dispositivos da rede Ad hoc.

5 Testes e Avaliação de Desempenho

Um dos grandes desafios no desenvolvimento de soluções de *software* para ambientes de redes Ad hoc é a necessidade de se avaliar seu desempenho considerando a dinamicidade da rede e um grande número potencial de dispositivos sem fio que podem compor a mesma. Mesmo que soluções desenvolvidas tenham sido testadas em pequena escala, é importante a sua avaliação em um ambiente de maior escala, devido à complexidade adicional introduzida pelo comportamento dinâmico de um grande número de dispositivos sem fio. Com base nestas considerações, a simulação apresenta-se como uma alternativa viável e largamente empregada para a avaliação de desempenho em cenários de larga escala.

O objetivo deste capítulo é descrever os testes funcionais realizados para validar a infra-estrutura de acesso ao serviços do InteGrade descrita nos capítulos 3 e 4, bem como apresentar os resultados obtidos através da simulação em cenários de larga escala dos protocolos que permitem aos dispositivos móveis em redes Ad hoc realizarem solicitações de serviço à grade. Como mencionado na Subseção 4.6.1, para que os clientes da rede Ad hoc possam realizar requisições de serviço à grade, estes necessitam registrar-se em um dispositivo que forneça o serviço de acesso à mesma. O objetivo das simulações é verificar a quantidade dessas requisições que são completadas quando diversos clientes tentam obter serviços da grade em diversos cenários marcados pela mobilidade dos nós.

5.1 Testes Funcionais

Para validação da funcionalidade da arquitetura proposta para acesso aos serviços do InteGrade utilizando-se dispositivos conectados a uma rede infra-estruturada 802.11, foram realizados testes utilizando-se um dispositivo real para realizar submissões de aplicações e recuperação das respostas geradas. Para estes testes foi utilizado um *smarthphone* HP iPAQ modelo hw6900 com sistema operacional Windows Mobile 5.0. Esse dispositivo foi escolhido porque além de apresentar uma interface 802.11

b/g, sua configuração de *hardware* permite a instalação de uma máquina virtual Java 2 ME com Configuração CDC 1.0 Personal Profile 1.1 [1], requisito mínimo para executar a interface de submissão de aplicações `MASCT` em dispositivos móveis. A máquina virtual utilizada no dispositivo foi a IBM J9 6.1 CDC 1.0 Personal Profile 1.1.

Para execução do *middleware* InteGrade foi utilizada uma das máquinas do Laboratório de Sistemas Distribuídos (LSD) da UFMA, onde foram executados os componentes `GRM`, `AR` e `LRM`. Foram registradas no Repositório de Aplicações (AR) do InteGrade duas aplicações do próprio sistema operacional Linux, a aplicação `ls` e a aplicação `convert`. A primeira têm a função de listar arquivos e diretórios e a segunda de realizar conversão de formato de uma imagem fornecida como entrada. Apesar da simplicidade, estas aplicações foram escolhidas por gerarem resultados textuais (`ls`) e imagens (`convert`), que correspondem aos dois formatos de resultado que podem ser exibidos pela interface gráfica `MASCT`. O teste realizado utilizando a aplicação `convert` também teve como objetivo testar o mecanismo de adaptação de imagens realizado pelo `ProxyAdapter`. Para isso, uma imagem JPG de resolução de 1024 x 768 *pixels* foi armazenada no servidor web do Laboratório de Sistemas Distribuídos. Indicou-se na interface de submissão de aplicações do `MASCT` a URL onde estava localizada a imagem JPG que deveria ser convertida para o formato PNG pela aplicação *convert*. A requisição foi enviada através da rede fixa e ao chegar ao `GridProxy`, a imagem JPG a ser utilizada como arquivo de entrada da aplicação foi baixada da URL indicada e enviada junto com a solicitacão de execução. A aplicação `convert` foi executada pelo InteGrade e um arquivo de saída contendo uma imagem PNG de resolução 1024 x 768 *pixels* foi gerado. Ao realizar a solicitacão da visualização da imagem PNG gerada pela execução, o arquivo passou pelo componente `ProxyAdapter`, que a adaptou à resolução de 240 x 320 *pixels* do dispositivo HP iPAQ modelo hw6900. A imagem convertida enviada pelo `ProxyAdapter` pôde, então, ser visualizada sem problemas através da interface gráfica `MASCT` no dispositivo móvel.

Em todos os teste realizados, os mecanismos de busca por aplicações registradas, execução de aplicações, exibição dos resultados da execução e recuperação dos resultados tanto textuais quanto de imagens foram executados sem erros e apresentaram o comportamento esperado.

[1] Java ME website: `http://java.sun.com/javame/technology/index.jsp`

Para a realização de testes funcionais da arquitetura para acesso a serviços da grade através de dispositivos móveis conectados em redes Ad hoc, foram realizados testes de busca por aplicações registradas na grade, execução de aplicações e obtenção de resultados utilizando-se dispositivos reais. Para executar o *middleware* SNU com o serviço `GridClient` e a interface gráfica para solicitação de serviços à grade, `GridClientGUI`, foi utilizado um celular NOKIA modelo N95 com sistema operacional Symbian OS S60. Esse aparelho celular foi escolhido por possuir a interface *bluetooth* necessária para que os clientes da rede Ad hoc pudessem enviar suas solicitações de serviço à grade através dos provedores de serviço presentes nesta rede, além de possuir uma máquina virtual Java 2 ME com suporte a configuração CLDC 1.0 MIDP 2.0, configuração mínima necessária para executar o *middleware* SNU com o serviço `GridClient`. Do lado servidor o *middleware* SNU com o serviço `GridServer` foi executado em um *smarthphone* modelo HP com sistema Operacional Windows Mobile. Esse dispositivo apresenta tanto uma interface *bluetooth*, que permite a comunicação com os dispositivos clientes da rede Ad hoc, quanto uma *interface* 802.11 que permite a comunicação com os componentes da grade que executam na rede infra-estruturada, servindo de ponte entre as duas redes. Para execução do *middleware* InteGrade foi utilizada uma das máquinas do Laboratório de Sistemas Distribuídos (LSD) da UFMA, onde foram executados os componentes `GRM`, `AR` e `LRM`. Os mesmos testes realizados com as aplicações `ls` e `convert` para o MInteGrade em modo infra-estruturado também foram realizados para a arquitetura do MInteGrade em modo Ad hoc.

Em todos os teste realizados, os mecanismos de busca por aplicações registradas, execução de aplicações, exibição dos resultados da execução e recuperação dos resultados tanto textuais quanto de imagem foram executados sem erros e apresentaram o comportamento esperado.

Devido a quantidade limitada de dispositivos móveis disponíveis no Laboratório de Sistemas Distribuídos da UFMA, além dos testes funcionais acima descritos foram realizados testes adicionais utilizando-se o simulador Impronto Simulator[2]. Este simulador permite o teste de aplicações Java que se comunicam utilizando a API *bluetooth* JSR82. Através desse simulador os programadores podem testar suas aplicações criando um ambiente de rede *bluetooth* composto por vários dispositivos virtuais sem a necessidade de possuir nenhum *hardware* ou pilha de protocolos *bluetooth*. Desta

[2]Impronto Simulator website: `http://www.rococosoft.com/blue_simulator.html`

forma, pôde-se avaliar a infra-estrutura de *software* desenvolvida em cenários com um número maior de dispositivos clientes e servidores, simulando situações onde tanto os dispositivos clientes quanto provedores de serviço se desconectavam da rede durante os testes. O objetivo destes testes foi o de verificar se os protocolos desenvolvidos para lidar com variações na topologia da rede Ad hoc, descritos na Subseção 4.6.3, eram realmente funcionais.

Nestes testes 5 clientes e 2 servidores eram iniciados. Requisições de busca por aplicações registradas, execução de aplicações, exibição dos resultados da execução e recuperação dos resultados foram enviados a partir dos clientes para o *middleware* InteGrade. Durante os testes alguns dos provedores de serviço eram desligados com o intuito de verificar se seus clientes conseguiriam detectar a indisponibilidade do mesmo e se registar em um novo provedor de serviço da rede para continuar enviando e recebendo requisições. De fato, ao detectar a indisponibilidade do provedor de serviço a quem se encontravam registrados, os clientes conseguiram encontrar na rede um novo provedor de serviço disponível, se registrar junto a ele e receber respostas da grade através desse novo provedor de serviço selecionado, assim como realizar novas requisições.

Também foram realizados testes onde alguns dos clientes eram desligados após algumas interações com a grade, para verificar se o provedor de serviço a quem se encontravam registrados conseguiria detectar sua ausência e notificar o serviço de provisionamento de contexto `CIS` da indisponibilidade dos mesmos na rede, conforme especificado na Subseção 4.6.3. Esse evento deveria ser o gatilho para que o `ProxyAdapter` desse início ao armazenamento das mensagens direcionadas a estes clientes . Nos teste realizados, os provedores de serviço conseguiram detectar a ausência de seus clientes logo após os mesmos terem sido desligados e informar ao `CIS` da indisponibilidade, o que levou o `ProxyAdapter` a enfileirar mensagens para estes clientes. Os provedores de serviço também foram capazes de detectar a volta dos clientes, quando estes eram reiniciados nos testes, e tornar a enviar suas informações de contexto ao `CIS` fazendo com o `ProxyAdapter` liberasse as mensagens enfileiradas. No caso em que os clientes não eram reiniciados nos testes, os provedores de serviço os desregistravam após 10 minutos de notada a ausência.

O uso do Impronto Simulator nos permitiu testar alguns dos protocolos desenvolvidos que lidam com a indisponibilidade dos dispositivos na rede Ad hoc.

No entanto, os testes realizados exigiram muita intervenção manual, dado que foi necessário iniciar os dispositivos e enviar as requisições utilizando a interface gráfica da aplicação `GridClientGUI`, o que limitava a quantidade de dispositivos que poderiam ser utilizados simultaneamente. Além disso, a simulação da indisponibilidade dos dispositivos consistia somente em torná-los conectáveis ou indisponíveis através da janela principal do simulador. Não era possível simular a situação onde os dispositivos se tornavam incomunicáveis devido ao seu distanciamento, pois nenhum padrão de mobilidade para os nós instanciados era utilizado.

Tendo isso em vista, complementamos a avaliação da arquitetura para acesso ao InteGrade em redes Ad hoc utilizando o simulador ns-2, que permitiu realizar simulações envolvendo um grande número de dispositivos, estipular padrões de mobilidade para os nós participantes da simulação e montar cenários onde os dispositivos clientes enviariam requisições aos servidores de forma automática, simulando o comportamento padrão de um usuário do serviço. A seção a seguir descreve uma introdução ao simulador ns-2 e de seus recursos utilizados na avaliação da arquitetura de *software* para acesso aos serviços do *middleware* InteGrade em redes Ad hoc, proposta nesta dissertação.

5.2 Introdução ao Simulador ns-2

O ns-2[3] é um simulador de redes de eventos discretos que implementa protocolos de transporte TCP e UDP, fontes de tráfego simulando aplicações FTP, Telnet, Web, CBR e VBR, mecanismos de gerenciamento de filas de roteadores como Drop Tail, RED e CBQ, algoritmos de roteamento, *multicasting* e alguns dos protocolos da camada MAC para simulação de redes locais. O ns-2 também permite a simulação de nós móveis conectados por interfaces de redes sem fio, incluindo a habilidade de simular redes Ad hoc e protocolos de roteamento *multihop*.

Dentre os aspectos que motivaram a escolha do simulador ns-2 para a realização deste trabalho estão a sua utilização ser bastante difundida na comunidade científica para simulação de redes fixas e sem fio, a grande quantidade de documen-

[3]ISI - Information Sciences Institute. The Network Simulator (ns-2): `http://www.isi.edu/nsnam/ns`

tação disponível e listas de discussão ativas sobre o mesmo; a versão do simulador utilizada foi a 2.29.

A programação no ns-2 é feita em duas linguagens: C++ para a implementação dos agentes, módulos do ns-2 que contêm a implementação dos protocolos a serem simulados, e OTCL (*Object-oriented Tool Command Language*) para descrever os cenários das simulações e seus parâmetros. O motivo para se utilizar duas linguagens de programação baseia-se em duas diferentes necessidades. Detalhadas simulações de protocolos requerem uma linguagem de programação que possa eficientemente manipular *bytes*, cabeçalhos de pacotes e implementar algoritmos que executem sobre uma grande quantidade de dados. Para estas tarefas, tempo de execução é mais importante do que o tempo gasto no processo de recompilação a cada alteração realizada no código. Nesse contexto, C++, que é uma linguagem compilada, mostrou-se a ferramenta mais eficaz. Por outro lado, uma grande parte das simulações de redes envolve variações em parâmetros ou configurações. Portanto, ajustes são necessários com certa frequência. Nesse contexto o uso da linguagem OTcl, que é interpretada, para a configuração das simulações torna mais eficiente o processo iterativo de modificar parâmetros do modelo da simulação e reexecutá-lo.

Para a realização da simulação no ns-2 foram desenvolvidos *scripts* implementados em OTcl, que reproduzem o cenário onde vários clientes de uma rede Ad hoc enviam requisições de registro e requisições de acesso à grade para servidores de acesso durante um determinado período de tempo. Também foi necessário a inclusão de dois novos agentes para o ns-2, escritos em C++, um representando o comportamento do `GridClient` e outro do `GridServer`.

5.3 Cenários de Simulação e Métricas de Avaliação

O objetivo das simulações é avaliar o desempenho do serviço de compartilhamento do acesso ao InteGrade em redes Ad hoc considerando diversos cenários que apresentam diferentes densidades de dispositivos móveis, mobilidade de seus usuários e diferentes proporções entre clientes e servidores presentes no ambiente. Nesta avaliação, as seguintes métricas foram levadas em consideração:

- **Quantidade de requisições de registro de clientes completadas**, que consiste

na contabilização da quantidade de vezes em que os clientes da rede Ad hoc obtiveram resposta às solicitações de registro junto a provedores do serviço de acesso ao InteGrade;

- **Quantidade de requisições de registro de clientes perdidas**, que consiste na contabilização da quantidade de vezes em que os clientes da rede Ad hoc não obtiveram resposta às solicitações de registro realizadas junto a provedores do serviço de acesso ao InteGrade;

- **Quantidade de requisições de solicitação de serviço à grade completadas**, que consiste na contabilização da quantidade de vezes em que os clientes da rede Ad hoc obtiveram respostas às solicitações de serviço enviadas à grade.

- **Quantidade de requisições de solicitação de serviço à grade perdidas**, que consiste na contabilização da quantidade de vezes em que os clientes da rede Ad hoc não obtiveram respostas às solicitações de serviço enviadas à grade.

A escolha das métricas citadas teve como objetivo avaliar se os clientes das redes Ad hoc, ao utilizarem os protocolos implementados para acesso à grade, conseguiriam obter resposta a maior parte das requisições efetuadas, levando-se em consideração que estes clientes ao se locomoverem pelo ambiente podem se afastar dos provedores de serviço a quem estão registrados e eventualmente ficar fora do raio de comunicação desses servidores, o que levaria a perda de mensagens tanto de requisições à grade quanto de respostas. Para as simulações realizadas não foram aferidos os tempos de resposta das requisições que podem ser feitas a grade, pois os métodos relativos a solicitação de execução de aplicações e recuperação das computações realizadas transferem um pequeno volume de dados e provocam pouco processamento, gerando tempos de respostas baixos.

Para avaliação das métricas citadas, imaginamos um conjunto de cenários onde a arquitetura para acesso aos serviços do InteGrade através de redes Ad hoc poderia ser utilizada. A elaboração do primeiro cenário teve por objetivo testar a arquitetura em um ambiente caracterizado por uma baixa densidade de dispositivos e por uma taxa de mobilidade média. Como ilustração, imaginamos um prédio de uma instituição universitária onde um de seus andares, correspondendo a uma área de 500 m^2, fosse ocupado por um conjunto de salas de professores ou departamento de cursos.

Considerando que cada sala ocupe um espaço de 25 m^2 e que cada sala acomode 4 professores, haveria nesta situação um total de 80 professores aptos a utilizar ou disponibilizar os serviços do MInteGrade em rede Ad hoc. Para este cenário, definimos que os participantes permanecem estáticos por 30 minutos na localidade em que se encontram atualmente e logo após esse tempo realizam um deslocamento para um ponto aleatório dentro da área de 500 m^2.

Em um segundo cenário tivemos por objetivo testar a arquitetura implementada em um ambiente caracterizado por uma densidade de dispositivos média e uma taxa de mobilidade baixa. De forma semelhante ao cenário anterior, imaginamos o mesmo andar do prédio da instituição universitária desta vez ocupado por laboratórios de pesquisa com área de 25 m^2 cada. Estabelecendo-se uma quantidade de 8 estudantes usuários de cada laboratório, teremos um total de 160 estudantes ocupando este andar. O tempo de permanência dos estudantes nos laboratórios foi estabelecido em 60 minutos, após os quais os estudantes se locomovem para um outro ponto da área definida.

Para um terceiro cenário definimos uma densidade de dispositivos alta com uma taxa de mobilidade baixa. Ainda no contexto acadêmico, vislumbramos um ambiente composto por salas de aula de 50 m^2, em uma área total de 500 m^2. Cada uma dessas salas seria ocupada por 32 alunos, totalizando um total de 320 alunos, usuários ou provedores do serviço de acesso à grade. O tempo de permanência nesse cenário foi definido baseado no tempo médio de duração de uma aula, 50 minutos.

No último cenário elaborado, procuramos investigar como a arquitetura elaborada reagiria a um cenário marcado por uma grade densidade de nós com uma alta taxa de mobilidade dos dispositivos. Para isso, imaginamos um ambiente de uma feira, disposta em uma área de 500 m^2, onde estariam localizadas uma quantidade média de 1,5 pessoas por m^2, totalizando um total de 320 pessoas participantes da feira. Neste cenário as pessoas se locomovem pelo espaço da feira realizando pausas de 5 minutos, o que equivaleria a visitas rápidas aos estandes disponíveis.

A Figura 5.1 ilustra uma das topologias iniciais da rede Ad hoc para o primeiro cenário descrito, composto por 80 nós dispostos em um área de 500 m^2.

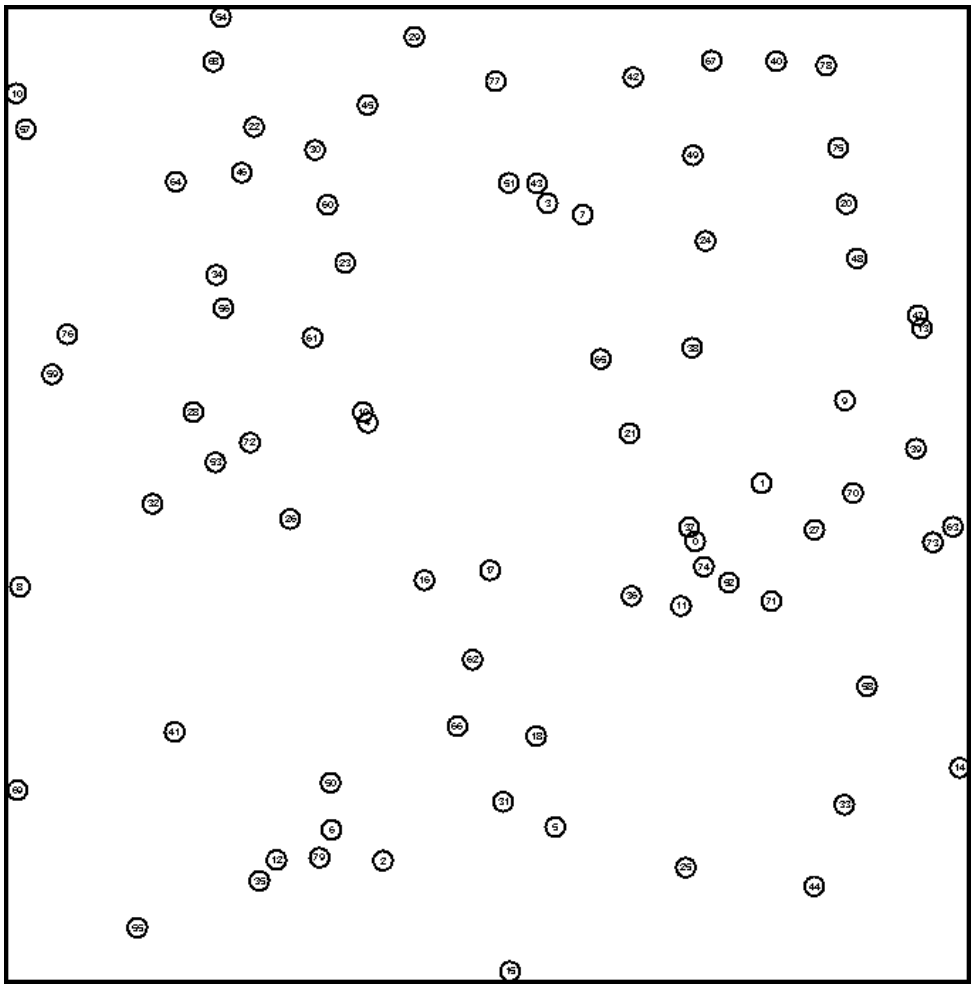

Figura 5.1: Exemplo da distribuição inicial dos dispositivos móveis em um cenário de 500 m^2

5.4 Implementação das Simulações

Para todos os cenários citados na Seção 5.3 foram gerados arquivos que especificam a posição inicial de cada nó e suas movimentações ao longo da simulação. Esses arquivos foram gerados utilizando-se um *script* fornecido pelo ns-2 que utiliza o modelo de mobilidade RWP (*Random Waypoint*) [4] para gerar os padrões de movimentação. Nesse modelo, quando a simulação inicia, cada nó seleciona aleatoriamente uma localização como destino dentro do campo definido. Ele então se desloca em direção a esse destino com uma velocidade constante escolhida uniformemente e aleatoriamente dentro do intervalo [0, 2,5], onde o último valor do intervalo representa a velocidade máxima que pode ser alcançada por cada nó. O valor máximo de 2,5 m/s foi escolhido para as simulações por representar a velocidade máxima de uma pessoa caminhando. Após alcançar o destino, o nó novamente escolhe um novo destino dentro do campo da simulação e move-se em direção a ele. O *script* também permite que seja configurado

um "tempo de pausa", que indica a quantidade de tempo em que o nó permanecerá estático entre seus deslocamentos.

No *script* OTcl utilizado para configurar a simulação são definidos a quantidade de nós participantes, a proporção entre clientes e servidores, o tempo total da simulação, o tamanho da área em que estarão dispostos os dispositivos e o arquivo que contém a descrição dos deslocamentos dos nós durante o tempo total da simulação, assim como a velocidade máxima e o tempo de "pausa" de cada nó. Para cada um dos cenários apresentados, foram executadas simulações considerando as seguintes proporções de clientes por servidor: 2 clientes para cada servidor, 4 clientes para cada servidor e 8 clientes para cada servidor. O valor do raio de transmissão foi configurado como 10 m para todos os nós, a fim de reproduzir o alcance dos dispositivos *bluetooth* com transmissores de classe 2, presentes nas maioria dos dispositivos celulares disponíveis no mercado e para os quais a arquitetura de *software* descrita nesta dissertação foi desenvolvida. O tempo total de simulação foi configurado para 4 horas nos cenários 1, 2 e 3 e de 1 hora no cenário 4.

Nas simulações não foi considerada a existência de obstáculos, como paredes por exemplo, no espaço onde os nós foram dispostos. Isso tanto limitaria o alcance dos enlaces *bluetooth* como restringiria a movimentação dos nós nos ambientes propostos. Assumiu-se ainda que os dispositivos móveis servidores na simulação possuem conectividade 802.11 irrestrita.

Como ilustrado no pseudo-código da figura 5.2, o *script* OTcl define os passos seguidos por um cliente durante a realização da simulação. Ao iniciar sua execução, o *script* de configuração cria os nós clientes e servidores que participarão da simulação. Para cada cliente é escolhido um valor de tempo aleatório, utilizando-se uma distribuição uniforme, com intervalo definido entre o momento em que o cliente é iniciado e o momento final da simulação. Este valor de tempo é utilizado para determinar o exato momento em que o cliente começará a participar da simulação. Estes passos são mostrados nas linha 2 e 3 do pseudo-código.

O cliente, ao iniciar na simulação, tentará realizar um conjunto de requisições de serviço à grade. Primeiramente ele selecionará um tempo aleatório entre o tempo atual e os 5 minutos futuros da simulação, utilizando uma distribuição uniforme, que servirá para indicar em que momento da simulação ele enviará a

1 **início**
2 Cliente c_i seleciona tempo aleatório T entre 0 e o tempo de conclusão da simulação;
3 Cliente c_i inicia sua participação na simulação no tempo T;
4 Cliente c_i seleciona tempo aleatório T entre o tempo atual e os 5 minutos futuros da simulação;
5 **enquanto** *O tempo de conclusão da simulação não é alcançado* **faça**
6 Evento de busca por aplicações registradas é escalonado no tempo T;
7 Ao receber resposta da requisição anterior, o cliente c_i seleciona tempo aleatório T entre 1 a 2 minutos do tempo atual da simulação;
8 Evento de submissão de execução é escalonado no tempo T;
9 Ao receber resposta da requisição anterior, o cliente c_i seleciona tempo aleatório T entre o tempo atual e os 20 minutos futuros da simulação;
10 Evento de consulta ao estado da aplicação é escalonado no tempo T;
11 Ao receber resposta da requisição anterior, evento de obtenção de resultados da execução é escalonado;
12 Ao receber resposta da requisição anterior, evento de recuperação de arquivos de saída é escalonado;
13 Ao receber resposta da requisição anterior, o cliente c_i seleciona tempo aleatório T entre o tempo atual e os 5 minutos futuros da simulação;
14 **fim**
15 **fim**

Figura 5.2: Passos seguidos por um cliente da rede Ad hoc durante a simulação

primeira requisição, uma solicitação de busca por aplicações registradas na grade. As linhas 4 e 6 do pseudo-código ilustram os passos que devem ser seguidos pelo cliente para enviar esta requisição.

Ao receber a confirmação do recebimento da resposta da requisição de consulta por aplicações registradas na grade, o cliente selecionará novo tempo aleatório entre 1 a 2 minutos a contar do tempo atual da simulação para enviar uma requisição de execução de aplicação à grade, conforme passos 7 e 8 do pseudo-código. O escalonamento dessas duas requisições uma após a outra, com curto espaço de tempo, tem como objetivo simular o comportamento usual do usuário que tende a efetuar uma requisição de busca por aplicações registradas e logo em seguida uma requisição de execução de aplicação ao começar a utilizar o *software* de acesso aos serviços da grade.

A seguir, será escalonado um segundo conjunto de requisições em rajada para simular o comportamento do usuário que tende a fazer requisições para verificação do estado da execução da aplicação, obtenção da lista de arquivos de saída gerados pela execução e visualização dos mesmos. O escalonamento da primeira requisição desse segundo conjunto de requisições será feita em até no máximo 20 minutos após a chegada da confirmação do recebimento da última requisição do primeiro conjunto. Os passos para a realização desse segundo conjunto de requisições estão definidos nas linhas 9 a 12 do pseudo-código.

Após a conclusão de um ciclo de requisições para um cliente, será escalonado um novo ciclo de requisições em até 5 minutos a partir do final do último ciclo, como mostrado na linha 13 do pseudo-código. Este processo é repetido até que o tempo estipulado para a simulação termine.

Um outro aspecto de implementação da simulação refere-se a um comportamento típico do usuário do serviço de acesso ao InteGrade em redes Ad hoc, que usualmente verifica na sua interface de submissão de aplicações à grade se o sistema está conectado a algum provedor de serviço disponível antes de enviar sua requisição. Um ícone da interface permanece verde caso exista conectividade com algum provedor de serviço e vermelho caso contrário, conforme descrito na Subseção 4.6.3. Caso não haja um servidor disponível, o usuário normalmente verificará o ícone de tempos em tempos até que seja estabelecida a conectividade. Para simular esse comportamento

nos experimentos, o cliente sempre verifica se está de fato registrado a algum provedor de serviço no momento imediatamente anterior a enviar uma requisição de serviço à grade e, caso não esteja, esperará um novo tempo aleatório entre 0 e 5 minutos para verificar se existe algum servidor disponível ao qual possa enviar sua requisição. Apesar de constar no *script* OTcl de configuração da simulação, este comportamento foi omitido do pseudo-código da Figura 5.2 para facilitar sua leitura.

Os agentes implementados em C++ no ns-2, têm o objetivo de simular o comportamento do `GridClient` e do `GridServer`, descritos no Capítulo 4. O agente `GridClient` é responsável por, no momento em que o cliente é iniciado, encontrar um provedor de serviço no seu raio de alcance e se registrar neste provedor de serviço. Outra função do agente `GridClient` é a de tentar manter o cliente sempre registrado a um provedor de serviço na rede. Isto é feito através da verificação, a cada 30 segundos, da disponibilidade do provedor de serviço ao qual o cliente se encontra registrado. Caso o `GridClient` constate nessa busca que o provedor de serviço não está alcançável, ele tentará realizar o processo de registro em novo provedor de serviço na rede, se houver. O agente `GridClient` também é responsável por enviar as requisições de serviço ao agente `GridServer` e tratar as respostas retornadas.

O agente `GridServer` têm a função de receber as requisições de registro e de serviço enviadas pelos agentes `GridClient` e respondê-las após um tempo de processamento de 260 ms para requisições de registro e de 5 segundos para as demais requisições, simulando com isso o tempo que a requisição levaria pra trafegar até o InteGrade e para que uma resposta à requisição retornasse ao `GridServer`. O agente `GridServer` também monitora a rede para verificar se seus clientes registrados ainda estão alcançáveis. Caso não estejam, ele passa a decrementar o atributo de vivacidade dos clientes em uma unidade a cada 30 segundos, desregistrando-os quando a vivacidade atingir o valor zero ou retornando a vivacidade ao seu valor original de 10 unidades caso o cliente seja novamente encontrado.

5.5 Resultados Obtidos

Para cada um dos cenários descritos na Seção 5.3 foram realizados 30 experimentos. Desta amostra de 30 experimentos, foram calculadas a quantidade média de

requisições de registro completadas e perdidas e a quantidade média de requisições de serviço à grade completadas e perdidas.

5.5.1 Cenário 1

As tabelas 5.1 e 5.2 apresentam respectivamente, a quantidade e o percentual médio de requisições de registro e serviço completadas e perdidas no cenário de baixa densidade (80 nós dispostos em uma área de 500 m^2) e mobilidade média (cada nó permanece em repouso durante 30 minutos entre seus deslocamentos). O tempo total simulado foi de 4 horas. Foram simuladas as seguintes proporções entre clientes e servidores presentes no ambiente: 2, 4, 8 e 16 clientes por servidor.

Proporção cliente/servidor	**Nº Médio de Req. Realizadas**	**Nº Médio de Req. Completadas**	**Nº Médio de Req. Perdidas**
53 clientes 27 servidores	268,21	265,26 (98,90%)	2,95 (1,10%)
64 clientes 16 servidores	332,26	327,47 (98,56%)	4,79 (1,44%)
71 clientes 9 servidores	361,74	353,63 (97,76%)	8,11 (2,24%)
75 clientes 5 servidores	342,33	332,11 (97,01%)	10,22 (2,99%)

Tabela 5.1: Quantidade média de requisições de registro realizadas, completadas e perdidas no Cenário 1, variando-se a proporção de clientes e servidores

Proporção cliente/servidor	**Nº Médio de Req. Realizadas**	**Nº Médio de Req. Completadas**	**Nº Médio de Req. Perdidas**
53 clientes 27 servidores	2169,42	2166,74 (99,88%)	3 (0,12%)
64 clientes 16 servidores	2648,95	2643,79 (99,81%)	5,16 (0,19%)
71 clientes 9 servidores	2847,42	2839,05 (99,71%)	8,37 (0,29%)
75 clientes 5 servidores	2801,11	2789,74 (99,59%)	11,37 (0,41%)

Tabela 5.2: Quantidade média de requisições de serviço realizadas, completadas e perdidas no Cenário 1, variando-se a proporção de clientes e servidores

Pela observação dos dados das tabelas 5.1 e 5.2, percebe-se que neste cenário marcado por uma baixa densidade e média mobilidades dos nós, o protocolo para acesso ao InteGrade em redes Ad hoc comporta-se bem, completando a maior parte das requisições emitidas pelos clientes, tanto de registro quanto de solicitação de serviços à grade. Nota-se também que mesmo aumentando-se a proporção de clientes por

servidor para este cenário, o crescimento da quantidade de requisições perdidas para os dois tipos de requisição não cresceu substancialmente.

A tabela A.1 do Apêndice A apresenta um conjunto de medidas estatísticas complementares, calculadas a partir dos dados obtidos com a simulação. Considerando a quantidade média de requisições de registro completadas e perdidas, bem como a quantidade média de requisições de serviço completadas e perdidas, relativos aos 30 experimentos realizados, foram calculados: valor médio obtido, valor mínimo, valor máximo, desvio padrão e intervalo de confiança de 95%.

5.5.2 Cenário 2

As tabelas 5.3 e 5.4 apresentam os resultados obtidos na simulação do cenário de média densidade (160 nós dispostos em uma área de 500 m^2) e mobilidade baixa (cada nó permanece em repouso durante 60 minutos entre seus deslocamentos). O tempo total simulado foi de 4 horas. Foram simulados as seguintes proporções entre clientes e servidores presentes no ambiente: 2, 4, 8 e 16 clientes por servidor.

Proporção cliente/servidor	**Nº Médio de Req. Realizadas**	**Nº Médio de Req. Completadas**	**Nº Médio de Req. Perdidas**
107 clientes 53 servidores	269,06	263,33 (97,87%)	5,72 (2,13%)
128 clientes 32 servidores	312,44	305,39 (97,74%)	7,06 (2,26%)
142 clientes 18 servidores	353,4	344,07 (97,36%)	9,33 (2,64%)
151 clientes 9 servidores	384,9	367,07 (95,37%)	17,83 (4,63%)

Tabela 5.3: Quantidade média de requisições de registro realizadas, completadas e perdidas no Cenário 2, variando-se a proporção de clientes e servidores

Proporção cliente/servidor	**Nº Médio de Req. Realizadas**	**Nº Médio de Req. Completadas**	**Nº Médio de Req. Perdidas**
107 clientes 53 servidores	4459,39	4451,11 (99,81%)	8,28 (0,19%)
128 clientes 32 servidores	5196,22	5185,83 (99,80%)	10,39 (0,20%)
142 clientes 18 servidores	5733,87	5710,8 (99,60%)	23,07 (0,40%)
151 clientes 9 servidores	5894,9	5861,67 (99,44%)	33,23 (0,56%)

Tabela 5.4: Quantidade média de requisições de serviço realizadas, completadas e perdidas no Cenário 2, variando-se a proporção de clientes e servidores

Pela observação dos dados das tabelas 5.3 e 5.4 percebe-se que, neste cenário

marcado por uma média densidade de nós e uma baixa mobilidade dos dispositivos, o protocolo para acesso ao InteGrade em redes Ad hoc também comporta-se de maneira adequada, completando a maior parte das requisições emitidas pelos clientes, tanto de registro de clientes quanto de solicitação de serviços à grade. Nota-se também que mesmo aumentando-se a proporção de clientes por servidor para este cenário, o crescimento do número de requisições perdidas para os dois tipos de requisição não cresceu substancialmente. De forma similar ao descrito na subseção anterior, a tabela A.2 do Apêndice A apresenta um conjunto de medidas estatísticas complementares, calculadas a partir dos dados obtidos com a simulação deste cenário.

5.5.3 Cenário 3

As tabelas 5.5 e 5.6 apresentam os resultados obtidos na simulação do cenário de alta densidade (320 nós dispostos em uma área de 500 m^2) e mobilidade baixa (cada nó permanece em repouso durante 50 minutos entre seus deslocamentos). O tempo total simulado foi de 4 horas. Foram simulados as seguintes proporções entre clientes e servidores presentes no ambiente: 2, 4, 8 e 16 clientes por servidor.

Proporção cliente/servidor	**Nº Médio de Req. Realizadas**	**Nº Médio de Req. Completadas**	**Nº Médio de Req. Perdidas**
213 clientes 107 servidores	744,4	710,1 (95,39%)	34,3 (4,61%)
256 clientes 64 servidores	920,03	879,57 (95,60%)	40,47 (4,40%)
284 clientes 36 servidores	996,63	952,93 (95,62%)	43,7 (4,38%)
301 clientes 19 servidores	1086,2	1021,6 (94,05%)	64,6 (5,95%)

Tabela 5.5: Quantidade média de requisições de registro realizadas, completadas e perdidas no Cenário 3, variando-se a proporção de clientes e servidores

Proporção cliente/servidor	**Nº Médio de Req. Realizadas**	**Nº Médio de Req. Completadas**	**Nº Médio de Req. Perdidas**
213 clientes 107 servidores	8745,43	8720,77 (99,72%)	24,67 (0,28%)
256 clientes 64 servidores	10570,73	10543,53 (99,74%)	27,2 (0,26%)
284 clientes 36 servidores	11387,53	11327,03 (99,47%)	60,5 (0,53%)
301 clientes 19 servidores	11686,57	11608 (99,33%)	78,57 (0,67%)

Tabela 5.6: Quantidade média de requisições de serviço realizadas, completadas e perdidas no Cenário 3, variando-se a proporção de clientes e servidores

Pela observação dos dados das tabelas 5.5 e 5.6, percebe-se que, neste cenário marcado por uma alta densidade de nós e uma baixa mobilidade dos dispositivos, o protocolo para acesso ao InteGrade em redes Ad hoc novamente comportou-se de forma adequada, completando a maior parte das requisições emitidas pelos clientes, tanto de registro de clientes quanto de solicitação de serviços à grade. Nota-se também que mesmo aumentando-se a proporção de clientes por servidor para este cenário, o crescimento do número de requisições perdidas para os dois tipos de requisição não cresceu substancialmente. A tabela A.3 do Apêndice A apresenta um conjunto de medidas estatísticas complementares, calculadas a partir dos dados obtidos com a simulação deste cenário.

5.5.4 Cenário 4

As tabelas 5.7 e 5.8 apresentam os resultados obtidos na simulação do cenário de alta densidade (320 nós dispostos em uma área de 500 m^2) e mobilidade alta (cada nó permanece em repouso durante 5 minutos entre seus deslocamentos). O tempo total simulado foi de 1 hora. Foram simulados as seguintes proporções entre clientes e servidores presentes no ambiente: 2, 4, 8 e 16 clientes por servidor.

Proporção cliente/servidor	**Nº Médio de Req. Realizadas**	**Nº Médio de Req. Completadas**	**Nº Médio de Req. Perdidas**
213 clientes 107 servidores	1613,77	1551,5 (96,14%)	62,27 (3,86%)
256 clientes 64 servidores	1880,87	1810,87 (96,28%)	70 (3,72%)
284 clientes 36 servidores	2116,47	2028,23 (95,83%)	88,23 (4,17%)
301 clientes 19 servidores	2306,3	2175,07 (94,31%)	131,22 (5,69%)

Tabela 5.7: Quantidade média de requisições de registro realizadas, completadas e perdidas no Cenário 4, variando-se a proporção de clientes e servidores

Pela observação dos dados das tabelas 5.7 e 5.8, percebe-se que mesmo no cenário marcado por uma alta densidade de nós e uma alta mobilidade dos dispositivos o protocolo para acesso ao InteGrade em redes Ad hoc consegue garantir que a maior parte das requisições, tanto de registo quanto de serviço, emitidas pelos clientes sejam completadas. Nota-se, como nos demais cenários, que mesmo aumentando-se a proporção de clientes por servidor para este ambiente, o crescimento do número de requisições perdidas para os dois tipos de requisição não cresceu substancialmente. A

Proporção cliente/servidor	Nº Médio de Req. Realizadas	Nº Médio de Req. Completadas	Nº Médio de Req. Perdidas
213 clientes 107 servidores	2094,77	2088,53 (99,70%)	6,23 (0,30%)
256 clientes 64 servidores	2474,73	2435,17 (98,40%)	39,57 (1,60%)
284 clientes 36 servidores	2746,27	2704,77 (98,49%)	41,5 (1,51%)
301 clientes 19 servidores	2907,37	2849,26 (98,00%)	58,11 (2,00%)

Tabela 5.8: Quantidade média de requisições de serviço realizadas, completadas e perdidas no Cenário 4, variando-se a proporção de clientes e servidores

tabela A.4 do Apêndice A apresenta um conjunto de medidas estatísticas complementares, calculadas a partir dos dados obtidos com a simulação deste cenário.

6 Conclusões e Trabalhos Futuros

Ambientes de grade computacionais são reconhecidos por disponibilizar um enorme poder computacional, de armazenamento e de transferência de dados, além de outros recursos compartilháveis. Por outro lado, dispositivos móveis são equipamentos que objetivam fornecer aos usuários acesso ubíquo a recursos, informações e serviços. Devido a estas características, a interação entre esse dois ambientes pode trazer como vantagens a possibilidade de se utilizar a infra-estrutura de grade como uma extensão dos recursos computacionais limitados de dispositivos móveis e fornecer aos usuários de serviços de grade meios mais rápidos e fáceis de acesso as informações produzidas por estes sistemas em qualquer hora e lugar.

Este trabalho apresentou o MInteGrade (*Mobile InteGrade*), uma infra-estrutura de *software* para acesso aos serviços do *middleware* de grade InteGrade a partir de dispositivos móveis conectados através de redes sem fio IEEE 802.11 em modo infra-estruturado e redes *bluetooth* em modo Ad hoc. Através deste mecanismo, clientes móveis podem solicitar a execução de aplicações na grade, realizar o acompanhamento da execução das aplicações e visualizar o resultado de computações já concluídas. O MInteGrade foi projetado para levar em consideração o dinamismo das redes sem fio, provendo suporte a períodos de desconexão e baixa conectividade, bem como a adaptação de conteúdo dos resultados das computações realizadas pela grade.

As principais contribuições deste trabalho foram:

- A definição da arquitetura e implementação da infra-estrutura de *software* através da qual é possível tanto a clientes móveis com maior capacidade de recursos e conectividade, como *smartphones* e PDAs, quanto a dispositivos mais limitados, como telefones celulares, solicitar a realização de computações a uma grade de computadores, acompanhar e obter seus resultados.

- Implementação de aspectos de ciência de contexto para cliente móveis de grade, aspecto pouco abordado em trabalhos do gênero, onde através do monitoramento de informações de contexto estáticos ou dinâmicos dos aparelhos móveis, como resolução de tela, energia disponível, uso de CPU, quantidade de memória

livre, podem ser realizadas adaptações no conteúdo de mensagens retornadas pelo *middleware* InteGrade.

- Implementação de mecanismos de tolerância a desconexões, conectividade intermitente ou baixa conectividade dos dispositivos móveis, principalmente aqueles conectados à redes Ad hoc, durante seus acessos aos serviços do *middleware* de grade InteGrade.

A partir dos resultados do nosso trabalho foram publicados os seguintes artigos:

- Integrando Dispositivos Móveis ao Middleware InteGrade: artigo completo publicado no *I Workshop on Pervasive and Ubiquitous Computing, (WPUC 2007)*, que descreve a arquitetura para acesso ao *middleware* de grade InteGrade por dispositivos móveis conectados em redes sem fio IEEE 802.11 em modo infra-estruturado [18].

- Compartilhamento do Acesso a Grades Computacionais em Redes Ad Hoc: artigo completo publicado no *I Simpósio Brasileiro de Computação Ubíqua e Pervasiva, (SBCUP 2009)*, que descreve o compartilhamento do serviço de acesso ao *middleware* de grade InteGrade entre dispositivos móveis conectados em redes sem fio em modo Ad hoc [19].

6.1 Trabalhos Futuros

Durante o desenvolvimento do MInteGrade identificamos diversas possibilidades interessantes de extensão deste trabalho que por restrição de tempo não puderam ser desenvolvidas, bem como identificamos alguns possíveis desdobramentos que podem advir deste esforço inicial. Entre estes possíveis trabalhos futuros podemos destacar:

- A inclusão de mecanismos de segurança ao MInteGrade que garantam que todas as comunicações entre os clientes móveis e o `ProxyAdapter` sejam autenticadas e encriptadas;

- Possibilitar a utilização do serviço de armazenamento distribuído de dados do *middleware* InteGrade a partir de clientes móveis;

- Permitir que clientes da rede Ad hoc possam utilizar provedores de serviço que estejam fora do seu raio de alcance de comunicação, através da utilização de protocolos de roteamento de múltiplos saltos;

- Permitir que dispositivos da rede Ad hoc ao encontrar provedores de serviço de acesso à grade possam baixar dinamicamente o código para interação com o serviço de acesso ao InteGrade.

Referências Bibliográficas

[1] O. M. Alliance. User agent profile-approved version 2.0. www.openmobilealliance.org/technical/release_program/docs/UAProf/V2_0-20060206A/OMATSUAProfV2_020060206A.pdf. Acessado em: 14/11/2009.

[2] M. Baker, R. Buyya, and D. Laforenza. The grid: International efforts in global computing. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2000)*, Rome, Italy, August 2000.

[3] B. Bastos, L. Lima, A. Gomes, and A. Ziviani. Interoperação de Grades Móveis Ad hoc com Grades Fixas. *Revista Eletrônica de Iniciação Científica (REIC)*, 8:8, 2008.

[4] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, 2003.

[5] J. Brooke and M. Parkin. A PDA client for the computational grid. In *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 325–330, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[6] D. Bruneo, M. Scarpa, A. Zaia, and A. Puliafito. Communication Paradigms for Mobile Grid Users. In *Proceedings of the 3st International Symposium on Cluster Computing and the Grid*, page 669, Washington, DC, USA, 2003. IEEE Computer Society.

[7] R. Buyya. Understanding the Grid. *Grid Computing Planet Conference*, 2002.

[8] D. C. Chu and M. Humphrey. Mobile OGSI.NET: Grid computing on mobile devices. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, pages 648–655, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[9] R. Y. de Camargo and F. Kon. Distributed data storage for opportunistic grids. In *Proceedings of the 3rd international Middleware doctoral symposium*. ACM New York, NY, USA, 2006.

[10] R. Y. de Camargo and F. Kon. Design and implementation of a middleware for data storage in opportunistic grids. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, Rio de Janeiro, Brazil, May 2007.

[11] P. Domingues, P. Marques, and L. Silva. Resource usage of windows computer laboratories. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW 2005)*, pages 469–476, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[12] D. Erwin and D. Snelling. UNICORE: A Grid computing environment. *Concurrency and Computation: Practice and Experience*, 14(13-15):1395–1410, 2002.

[13] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications*, 11(2):115, 1997.

[14] I. Foster and C. Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure*, chapter Globus: A Toolkit-Based Grid Architecture, pages 259–278. Morgan Kaufmann Publisher, 1999.

[15] I. Foster, C. Kesselman, and S. Tueke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputing Applications*, 2001.

[16] A. Goldchleger. InteGrade: Um Sistema de Middleware para Computação em Grade Oportunista. Master's thesis, Instituto de Matemática e Estatística (IME), Universidade de São Paulo (USP), Dezembro 2004.

[17] A. Goldchleger, F. Kon, A. Goldman, M. Finger, and G. Bezerra. InteGrade: object-oriented Grid middleware leveraging the idle computing power of desktop machines. *Concurrency and Computation: Practice & Experience*, 16(5):449–459, 2004.

[18] D. S. Gomes, F. J. da Silva e Silva, and M. Endler. Integrando dispositivos móveis ao middleware integrade. In *Proceedings of the I Workshop on Pervasive and Ubiquitous Computing (WPUC 2007)*, Gramado, RS, Brazil, October 2007. SBAC-PAD 2007.

[19] D. S. Gomes, F. J. da Silva e Silva, A. C. T. Vidal, D. L. Franco, and V. R. A. Silva. Compartilhamento do acesso a grades computacionais em rede ad hoc. In *Proceedings of the I Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP 2009)*, Bento Gonçalves, RS, Brasil, Julho 2009. CSBC 2009.

[20] F. González-Castaño, J. Vales-Alonso, M. Livny, E. Costa-Montenegro, and L. Anido-Rifón. Condor grid computing from mobile handheld devices. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(1):117–126, 2003.

[21] V. N. Gudivada and V. V. Raghavan. Content-based image retrieval systems. *IEEE Computer*, 28(9):18–22, September 1995.

[22] H. K. Huang. *PACS and Imaging Informatics: Basic Principles and Applications*. John Wiley & Sons, New Jersey, 2004.

[23] J. Hwang and P. Aravamudham. Middleware services for P2P computing in wireless grid networks. *IEEE Internet Computing*, 8(4):40–46, 2004.

[24] J. Jing, A. Helal, and A. Elmagarmid. Client-server computing in mobile environments. *ACM Computing Surveys*, 31(2):117–157, 1999.

[25] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. In *Software - Practice and Experience*, volume 32, pages 135–164. February 2002.

[26] L. d. S. Lima. *Um Protocolo Para Descoberta e Seleção de Recursos em Grades Móveis Ad Hoc*. PhD thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, PUC-Rio, 2007.

[27] A. Litke, D. Skoutas, and T. Varvarigou. Mobile grid computing: Changes and challenges of resource management in a mobile grid environment. In *Proceedings of the 5th International Conference on Practical Aspects of Knowledge Management (PAKM 2004)*, 2004.

[28] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[29] D. Marinescu, G. Marinescu, Y. Ji, L. Boloni, and H. Siegel. Ad hoc grids: Communication and computing in a power constrained environment. In *Workshop on Energy-Efficient Wireless Communications and Networks*, 2003.

[30] M. W. Mutka and M. Livny. Profiling workstations' available capacity for remote execution. In *Proceedings of the 12th IFIP WG 7.3 International Symposium on Computer Performance Modelling, Measurement and Evaluation*, pages 529–544, Amsterdam, The Netherlands, 1988. North-Holland Publishing Co.

[31] M. W. Mutka and M. Livny. The available capacity of a privately owned workstation environment. *Performance Evaluation*, 12(4):269–284, 1991.

[32] A. S. F. Palmeira Filho. SNU: Um framework para o desenvolvimento de aplicações voltadas às redes ad-hoc espontaneas. Master's thesis, Programa de Pós-Graduação em Engenharia de Eletricidade (PPGEE), Universidade Federal do Maranhão (UFMA), Novembro 2007.

[33] S. Park, Y. Ko, and J. Kim. Disconnected Operation Service in Mobile Grid Computing. In *Proceedings of the International Conference on Service Oriented Computing (ICSOC'2003)*, Trento, Italy, June 2003. Springer.

[34] E. Pitoura and G. Samaras. *Data Management for Mobile Computing*. Kluwer Academic Publisher, 1998.

[35] H. Rubinsztejn, M. Endler, and N. Rodriguez. A framework for building customized adaptation proxies. In *Proceedings of the IFIP conference on Intelligence in Communication Systems (INTELLCOMM 2005)*. Springer, Montreal, Canada, 2005.

[36] V. Sacramento, M. Endler, H. Rubinsztejn, L. Lima, K. Goncalves, F. Nascimento, and G. Bueno. MoCA: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10):2–2, 2004.

[37] W. Stallings. *Wireless Communications and Networks*. Pearson Prentice Hall, 2005.

[38] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[39] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, and P. Vanderbilt. Open Grid Service Infrastructure WG. In *Global Grid Forum*, volume 2, page 17. Citeseer, 2002.

[40] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 8(33):103–111, 1990.

A Medidas Estatísticas das Simulações Realizadas

Este apêndice apresenta um conjunto de medidas estatísticas calculadas a partir da média do número de requisições de registro completadas, número médio de requisições de registro perdidas, número médio de requisições de serviço completadas e número médio de requisições de serviço perdidas, considerando os 30 experimentos executados nas simulação dos Cenários 1, 2, 3, e 4, descritos na Seção 5.3, com proporções entre clientes e servidores de 2, 4, 8 e 16 clientes por servidor. As medidas estatísticas são: média, valor mínimo, valor máximo, desvio padrão e intervalo de confiança de 95%.

Tabela A.1: Medidas estatísticas geradas a partir da média do número de requisições de registro completadas, número de requisições de registro perdidas, número de requisições de serviço completadas e número de requisições de serviço perdidas nas simulações do cenário 1

Tipo de Requisição	Proporção cliente servidor	Média	Min	Max	Desvio Padrão	Intervalo de Confiança (95%)
Registro Completadas	53/27	265,26	208	303	24,86	[262,32 ; 268,21]
	64/16	327,47	271	397	33,56	[323,55 ; 331,4]
	71/9	353,63	271	445	41,12	[348,74 ; 358,53]
	75/5	332,11	231	387	43,16	[327,59 ;336,63]
Registro Perdidas	53/27	2,95	1	5	1,57	[0,77 ; 5,13]
	64/16	4,79	1	11	2,74	[0,59 ; 10,17]
	71/9	8,11	2	13	2,9	[5,96 ; 10,25]
	75/5	10,22	3	23	5,39	[5,91 ; 14,54]
Serviço completadas	53/27	2166,74	1860	2440	166,48	[2159,97 ; 2173,5]
	64/16	2643,79	2434	3016	150,98	[2637,9 ; 2649,67]
	71/9	2839,05	2499	3265	239,33	[2829,94 ; 2848,16]
	75/5	2789,74	2409	3385	209,2	[2782,24 ; 2797,24]
Serviço Perdidas	53/27	3	1	5	1,08	[1,94 ; 4,06]
	64/16	5,16	2	10	1,9	[3,49 ; 6,82]
	71/9	8,37	3	16	3,86	[5,51 ; 11,22]
	75/5	11,37	4	19	3,55	[9,17 ; 13,57]

Tabela A.2: Medidas estatísticas geradas a partir da média do número de requisições de registro completadas, número de requisições de registro perdidas, número de requisições de serviço completadas e número de requisições de serviço perdidas nas simulações do cenário 2

Tipo de Requisição	Proporção cliente servidor	Média	Min	Max	Desvio Padrão	Intervalo de Confiança (95%)
Registro Completadas	107/53	263,33	239	319	19,53	[260,99 ; 265,68]
	128/32	305,39	272	333	15,61	[303,59 ; 307,19]
	142/18	344,07	307	401	18,56	[342,07 ; 346,06]
	151/9	367,07	333	462	26,31	[364,42 ; 369,71]
Registro Perdidas	107/53	5,72	2	13	2,98	[3,34 ; 8,1]
	128/32	7,06	1	12	2,74	[4,66 ; 9,46]
	142/18	9,33	3	18	3,49	[7,27 ; 11,39]
	151/9	17,83	7	36	7,06	[14,48 ; 21,19]
Serviço Completadas	107/53	4451,11	3891	5064	300,35	[4441,96 ; 4460,26]
	128/32	5185,83	4860	5696	259,71	[5178,55 ; 5193,12]
	142/18	5710,8	5325	6344	226,86	[5704,86 ; 5716,74]
	151/9	5861,67	5339	6683	318,36	[5853,13 ; 5870,21]
Serviço Perdidas	107/53	8,28	3	1	3,05	[6,39 ; 10,17]
	128/32	10,39	6	14	1,86	[9,17 ; 11,6]
	142/18	23,07	14	32	4,3	[21,13 ; 25]
	151/9	33,23	24	47	5,94	[31,5 ; 34,97]

Tabela A.3: Medidas estatísticas geradas a partir da média do número de requisições de registro completadas, número de requisições de registro perdidas, número de requisições de serviço completadas e número de requisições de serviço perdidas nas simulações do cenário 3

Tipo de Requisição	Proporção cliente servidor	Média	Min	Max	Desvio Padrão	Intervalo de Confiança (95%)
Registro Completadas	213/107	710,1	675	766	22,43	[708,41 ; 711,79]
	256/64	879,57	818	973	35,51	[877,18 ; 881,95]
	284/36	952,93	635	1041	68,68	[948,6 ; 957,26]
	301/19	1021,6	958	1101	38,74	[1019,15 ; 1024,05]
Registro Perdidas	213/107	34,3	21	47	6,47	[32,39 ; 36,21]
	256/64	40,47	30	56	5,93	[38,77 ; 42,16]
	284/36	43,7	30	57	6,74	[41,79 ; 45,61]
	301/19	64,6	37	88	12,23	[60,66 ; 68,54]
Serviço Completadas	213/107	8720,77	7983	9313	304,73	[8714,58 ; 8726,96]
	256/64	10543,53	9647	11188	382,72	[10536,28 ; 10550,79]
	284/36	11327,03	9885	12244	439,32	[11319,16 ; 11334,91]
	301/19	11608	10569	12540	392,78	[11601,03 ; 11614,97]
Serviço Perdidas	213/107	24,67	13	40	5,82	[22,51 ; 26,82]
	256/64	27,2	17	45	6,11	[24,98 ; 29,42]
	284/36	60,5	35	90	11,38	[58,15 ; 62,85]
	301/19	78,57	59	107	12	[75,9 ; 81,23]

Tabela A.4: Medidas estatísticas geradas a partir da média do número de requisições de registro completadas, número de requisições de registro perdidas, número de requisições de serviço completadas e número de requisições de serviço perdidas nas simulações do cenário 4

Tipo de Requisição	Proporção cliente servidor	Média	Min	Max	Desvio Padrão	Intervalo de Confiança (95%)
Registro Completadas	213/107	1551,5	1419	1714	72,24	[1547,96 ; 1555,04]
	256/64	1810,87	1704	2024	69,07	[1807,68 ; 1814,05]
	284/36	2028,23	1859	2202	81,57	[2024,68 ; 2031,79]
	301/19	2175,07	2006	2365	114,71	[2170,38 ; 2179,77]
Registro Perdidas	213/107	62,27	44	93	10,83	[59,67 ; 64,86]
	256/64	70	54	87	8,8	[67,83 ; 72,17]
	284/36	88,23	67	109	9,98	[86,19 ; 90,27]
	301/19	131,22	89	175	19,32	[127,89 ; 134,56]
Serviço Completadas	213/107	2088,53	1929	2287	86,51	[2084,93 ; 2092,14]
	256/64	2435,17	2260	2754	109,42	[2430,79 ; 2439,54]
	284/36	2704,77	2425	3053	118,4	[2700,28 ; 2709,25]
	301/19	2849,26	2645	2987	81,26	[2846,25 ; 2852,27]
Serviço Perdidas	213/107	6,23	2	10	2,28	[4,24 ; 8,23]
	256/64	39,57	31	53	6,23	[37,82 ; 41,31]
	284/36	41,5	25	75	14,45	[36,41 ; 46,59]
	301/19	58,11	45	72	7,57	[56,15 ; 60,08]