Integrando grades móveis em uma arquitetura orientada a serviços

Danilo Costa Marim Segura

Danilo Costa Marim Segura

Integrando grades móveis em uma arquitetura orientada a serviços

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Marcos José Santana

USP – São Carlos
Agosto de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

S634i

Segura, Danilo Costa Marim
Integrando grades móveis em uma arquitetura orientada a serviços / Danilo Costa Marim Segura; orientador Marcos José Santana. - São Carlos - SP, 2016.
89 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2016.

1. *Fog computing*. 2. dispositivos móveis. 3. Qualidade de Serviço. I. Santana, Marcos José, orient. II. Título.

Danilo Costa Marim Segura

Integrating mobile grids into a service oriented architecture

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Marcos José Santana

USP – São Carlos
August 2016

Aos meus pais Silvano e Ana Cláudia,

à minha irmã Isabelle,

aos meus avós Leonor e Servando.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Ele, que continua me guiando pelas noites mais escuras e pelos caminhos mais perigosos, tendo como esperança o incompreensível e eterno.

Agradeço meus pais Silvano e Ana Cláudia que sempre acreditaram em mim, mesmo quando não pude corresponder à sua altura e ao seu exemplo. Agradeço à minha irmã Isabelle, que sentiu minha ausência quando tive de me despedir ao sair de casa.

Também gostaria de agradecer à minha avó Leonor, por ser a prova que o amor supera as dificuldades, e ao meu avô Servando, que perdi pouco antes do fim desta caminhada. Agradeço à Ana Raquel, pelo companheirismo e amor a mim dedicados.

Meus profundos agradecimentos à Prof.ª Sarita Bruschi, pela dedicação e atenção durante seus ensinamentos, e ao Prof. Marcos Santana pelo exemplo que sempre foi e será. Também agradeço aos professores Paulo Sérgio e Júlio Estrella por tudo que me ensinaram.

Não menos importantes, agradeço a todos os grandes amigos que tive a honra de conviver durante minha vida até hoje. Amigos do LaSDPC, amigos da USP, amigos da vida, amigos de infância... Enfim, amigos irmãos! São tantos que agradeço a todos com a minha alma. E caipirada, a empreitada não acaba aqui!

Agradeço à Universidade de São Paulo e ao CNPq pelo financiamento através de bolsa de estudos.

Quando eu era menino, falava como menino, sentia como menino, discorria como menino, mas, logo que cheguei a ser homem, percebi que ainda sonho como menino.

“Agora, pois, permanecem a fé,
a esperança e o amor, estes três,
mas o maior destes é o amor.”
(1 Coríntios 13:13)

RESUMO

SEGURA, D.C.M.. **Integrando grades móveis em uma arquitetura orientada a serviços**. 2016. 89 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

O aumento no número de dispositivos móveis, como *smartphones*, *tablets* e *laptops*, e o avanço em seu potencial computacional permitiu considerá-los como recursos computacionais. O uso de recursos computacionais com maior proximidade vem crescendo ano após ano, sendo chamado de *Fog computing*, em que os elementos na borda da *Internet* são explorados, uma vez que os serviços computacionais convencionais podem estar indisponíveis ou sobrecarregados. Dessa forma, este projeto de Mestrado tem como foco possibilitar o uso de dispositivos móveis no provimento de serviços computacionais entre si de forma colaborativa através da heurística *Maximum Regret* adaptada, que busca alocar tarefas computacionais em dispositivos locais de forma a minimizar o consumo de energia e evitar dispositivos não confiáveis. Também há uma meta-heurística em um nível global, que interconecta os diferentes aglomerados de dispositivos móveis na borda da *Internet*, e possui informações globais de *Quality of Service* (QoS). Foram realizados experimentos que mostraram que evitar dispositivos móveis como recursos com um baixo grau de confiabilidade possibilitou diminuir o impacto no consumo de energia, além de ser possível diminuir os tempos de resposta e de comunicação ao ajustar a política de seleção de aglomerados externos.

Palavras-chave: *Fog computing*, dispositivos móveis, Qualidade de Serviço.

ABSTRACT

SEGURA, D.C.M.. **Integrando grades móveis em uma arquitetura orientada a serviços**. 2016. 89 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

The increasing number of mobile devices, such as smartphones, tablets and laptops, as well as advances in their computing power have enabled us to consider them as resources, exploring the proximity. The use of near computing resources is growing year by year, being called as Fog computing, where the elements on the edge of the Internet are exploited, once the computer services providers could be unavailable or overloaded. Thus, this Master's project focuses on using mobile devices to provide computing services among them through a heuristic called Adapted Maximum Regret, which tries to minimize energy consumption and avoid untrustable devices. There is also top-level metaheuristic which interconnects different clusters of devices on the edge of the Internet with global information to guarantee Quality of Services (QoS). We conducted a set of experiments that showed us to avoid devices with a high degree of failures to save more energy when allocating tasks among them, as well as decreasing the applications response time and communication through adjusts in the selection algorithm of external agglomerates.

Key-words: Fog computing, mobile devices, Quality of Service.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BSS ..... *Basic Service Set*
CPU ..... *Computing Processing Unit*
DNS ..... *Domain Name Service*
ESS ..... *Extended Service Set*
ETD ..... Energia Total Dispendida
GRB ..... *Grid Resource Brokers*
IoT ..... *Internet of Things*
MCC ..... *Mobile Cloud Computing*
QoS ..... *Quality of Service*
TCM ..... Tempo de Comunicação Médio
TRM ..... Tempo de Reposta Médio
WAP ..... *Wireless Access Point*

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

1.1 Contextualização e Motivação

O conceito de grade computacional surgiu na década de 90, e foi inspirado na estrutura das redes elétricas. Entre os vários objetivos de sua criação, destaca-se a utilização de recursos computacionais geograficamente distribuídos e ociosos para formar um grande poder de processamento, de modo a tratar problemas cada vez maiores (BUYYA; ABRAMSON; VENUGOPAL, 2005).

As grades computacionais convencionais são caracterizadas por sistemas de computação (estações de trabalho, computadores pessoais, *clusters*, supercomputadores) distribuídos e interconectados através de infraestruturas de rede (BUYYA; ABRAMSON; VENUGOPAL, 2005). Com essa estrutura, tornou-se possível o compartilhamento de recursos de forma coordenada (processamento, armazenamento, rede), tornando factível a resolução de problemas de alta complexidade (FOSTER; KESSELMAN; TUECKE, 2001).

Com a evolução da *Internet* e dos modelos de grades computacionais, passou a haver a tendência de cada vez mais explorar serviços de computação e armazenamento oferecidos em infraestruturas de *hardware* e *software* remotas. Tal abordagem foi nomeada de *cloud computing*. Isto permitiu que problemas complexos e aplicações explorassem tal infraestrutura de forma simples e organizada, abstraindo os recursos computacionais físicos com técnicas de virtualização (DIKAIAKOS *et al.*, 2009).

Conforme os dispositivos móveis (*laptops, smartphones, tablets*) se desenvolveram a partir da diminuição da escala dos componentes, uma nova proposta de grade computacional surgiu: as grades móveis. Com cada vez mais poder de processamento, capacidade de armazenamento e acesso à *Internet*, os dispositivos móveis passaram a ser investigados como recursos computacionais, ainda que a capacidade computacional e de bateria destes sistemas sejam limitados quando comparados com sistemas de computação convencionais. (RODRíGUEZ; MATEOS; ZUNINO,

2012; CHUNLIN; LAYUAN, 2014; BORRO, 2013).

Existem esforços que buscam explorar o potencial computacional dos dispositivos móveis com políticas sensíveis às restrições energéticas, estruturando-os como elementos ativos na grade móvel (RODRíGUEZ; MATEOS; ZUNINO, 2012). O objetivo de tais propostas é permitir que os dispositivos contribuam para a resolução de problemas de forma conjunta em ambientes distribuídos, levando em consideração as restrições de energia e de processamento (BORRO, 2013; CHUNLIN; LAYUAN, 2014).

Em relação ao número de *smartphones* comercializados pelo mundo, a consultoria Gartner [1] realizou uma pesquisa referente ao ano de 2013, mostrando que foram vendidos aproximadamente 968 milhões de *smartphones* pelo mundo. Já em 2014, as vendas de *smartphones* atingiram a marca de 1.244.890.000 unidades[2], representando um aumento de 28,63% em relação ao ano anterior. Os dados mais recentes do último trimestre de 2015 reafirmaram a tendência de crescimento de vendas de *smartphones*, uma vez que houve um aumento de 9,7% em relação ao último trimestre de 2014 [3].

Como os dispositivos móveis estão em constante evolução e expansão, existe um grande potencial computacional apto a ser devidamente explorado, e que pode servir de alternativa complementar dos serviços de computação em nuvem.

Em países menos desenvolvidos, as infraestruturas precárias de *Internet* podem dificultar o acesso a serviços de computação em nuvem por dispositivos móveis. Com isso, investigar o uso de dispositivos móveis com maior proximidade e a possibilidade de processarem aplicações remotas de outros dispositivos torna-se importante nesses cenários. Explorar essa proximidade serve de motivação para esforços recentes, como no novo conceito de *Fog computing*, que surgiu como uma forma de utilizar recursos computacionais na borda da *Internet* (DASTJERDI *et al.*, 2016).

Além disso, os serviços em nuvem utilizados por aplicações móveis podem, de alguma forma, se tornar indisponíveis ou sobrecarregados, afetando diretamente a experiência de uso das aplicações móveis. Dessa maneira, deve haver uma maneira de expandir os recursos que a computação em nuvem disponibiliza, permitindo também que aplicações móveis não sejam prejudicadas caso algum serviço remoto se torne indisponível. Para isso, é necessário investigar o uso de dispositivos móveis como provedores de recursos computacionais ao explorar sua proximidade, reduzindo os impactos no consumo de energia.

[1] http://www.gartner.com/newsroom/id/2665715
[2] http://www.gartner.com/newsroom/id/2996817
[3] http://www.gartner.com/newsroom/id/3215217

1.2 Objetivos

O principal objetivo deste projeto de Mestrado é propor um modelo de computação móvel que permita que dispositivos móveis exportem serviços para aplicações compostas em tarefas, e utilizem serviços de computação de outros membros.

Os objetivos secundários deste projeto de Mestrado são:

1. Minimizar o consumo de energia nos dispositivos móveis, considerados como recursos computacionais;
2. Considerar a mobilidade dos dispositivos e, consequentemente, sua confiabilidade.

1.3 Metodologia

Para que o objetivo fosse atingido, foi definida primeiramente a arquitetura do sistema, a qual segue um modelo hierárquico, com dois níveis, cada um usando um algoritmo de escalonamento diferente. O nível 1 tem escopo local, e é composto por aglomerados de dispositivos móveis, sendo que cada aglomerado contém um ponto de acesso e um líder escolhido através de um algoritmo de escalonamento. O nível 0 integra e interconecta os diferentes aglomerados do nível 1, e é composto por um nó especializado chamado *Broker*, que possui informações de QoS globais. Nessa arquitetura, os dispositivos móveis requisitam serviços para as aplicações móveis internamente aos seus líderes, que verificam a possibilidade de execução interna. Dependendo do resultado dessa verificação, o líder é responsável por selecionar os recursos para atender aquele serviço através de uma heurística que considera aspectos de disponibilidade de energia e confiabilidade dos dispositivos.

Caso as aplicações não possam ser atendidas internamente no nível 1, são encaminhadas ao *Broker*, presente no nível 0, o qual utiliza as informações de QoS de todos os aglomerados para escolher qual deles pode atender as solicitações.

De maneira a obter uma avaliação da arquitetura proposta, após a especificação da mesma foi definido que a avaliação seria feita utilizando simulação. A escolha por simulação teve como objetivo permitir a avaliação de vários cenários, permitindo uma flexibilidade maior do que a construção de um protótipo da arquitetura. O principal desafio da escolha da simulação foi a validação do modelo, a qual foi realizada através da comparação da heurística do nível 1 com o modelo de programação linear compatível, passando pela validação dos custos estimados de comunicação com a literatura e, por fim, a validação estatística acerca das variáveis de resposta quando o modelo como um todo foi considerado.

Para o desenvolvimento da simulação, primeiramente foi definido um modelo de rede de filas, o qual foi implementado utilizando a biblioteca de simulação SimPack/Sim++ (FISHWICK, 1992). A avaliação considerou três cenários com objetivos de avaliação distintos, variando ajustes

internos das políticas de escalonamento de forma a verificar como o gasto de energia e os tempos de resposta e comunicação se comportam.

Após a coleta dos resultados da simulação, a última etapa foi a de análise dos resultados, em que foram realizados testes de hipótese para a validação estatística.

1.4 Estrutura da Dissertação

Esta dissertação está estruturada da seguinte forma:

Capítulo 2: Nesse capítulo são apresentados todos os conceitos importantes para este projeto de Mestrado. Primeiramente, os conceitos acerca dos sistemas distribuídos são elencados, assim como as grades computacionais e, consequentemente, da computação móvel, com as grades móveis e *Mobile Cloud Computing*. Por fim, conceitos de uma nova área chamada *Fog computing* são discutidos, uma vez que sua motivação e aplicabilidade são importantes para este projeto.

Capítulo 3: No Capítulo 3 são discutidos os trabalhos relacionados e a lacuna a ser preenchida, de forma a situar este projeto.

Capítulo 4: Nesse capítulo são apresentados todos os detalhes de desenvolvimento do projeto, começando com as modificações no modelo e heurística proposto por Borro (2013), seguindo com os detalhes de desenvolvimento do nível 0, o modelo de rede de filas criado para representar a proposta e as estimativas de gastos estimados de comunicação.

Capítulo 5: Nesse capítulo são apresentadas toda a metodologia de validação das hipóteses referentes ao projeto, assim como as definições dos três cenários e a carga gerada. Além do mais, é apresentado o planejamento de experimentos, com os fatores, níveis e as variáveis de resposta e, por fim, a discussão dos resultados de cada cenário de forma isolada.

Capítulo 6: Este capítulo contém todas as conclusões que foram obtidas em relação ao projeto, assim como as contribuições científicas e os possíveis trabalhos possíveis que podem se originar.

CAPÍTULO

2

FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações Iniciais

Neste capítulo é apresentada toda a fundamentação teórica importante para este projeto de Mestrado. Primeiramente, conceitos de sistemas distribuídos clássicos são debatidos, elencando seus desafios, modelos arquiteturais, algoritmos de eleição e as grades computacionais. Na sequência, conceitos de computação móvel são elencados, mostrando duas principais vertentes: as grades móveis e a *Mobile Cloud Computing* (MCC). Por fim, serão discutidos conceitos da *Fog computing*, que é um novo paradigma no qual o oferecimento de serviços computacionais é situado na borda da *Internet*.

2.2 Sistemas Distribuídos

Segundo Tanenbaum e Steen (2007), "Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente", ou seja, um sistema distribuído é composto por sistemas de computação que se comunicam por uma rede, se comportando de forma transparente.

A computação distribuída surgiu a partir da evolução das redes de computadores e dos sistemas distribuídos. Computadores e recursos computacionais, que antes eram caros e ocupavam muito espaço físico, se popularizaram e permitiram que novas formas de resolver problemas e aplicações se tornassem factíveis (TANENBAUM; STEEN, 2007).

A *Internet* pode ser vista como um grande e complexo sistema distribuído, com sistemas de computação heterogêneos (PCs, estações de trabalho, celulares, sensores, etc). A computação distribuída, por sua vez, pode fazer uso da *Internet* como possível meio de intercomunicação global, o que permite, por exemplo, que aplicações móveis distribuídas sejam factíveis (COULOURIS *et al.*, 2013).

2.2.1 Computação Distribuída, Computação Paralela e Sistemas Distribuídos

É importante destacar como os conceitos de computação distribuída, computação paralela e de sistemas distribuídos se relacionam, uma vez que a computação móvel, área foco deste trabalho, está inserida nesses nichos.

Os sistemas distribuídos podem ser vistos como a base necessária para a criação da computação distribuída, e possível meio para o desenvolvimento da computação paralela. É possível citar alguns exemplos de sistemas distribuídos: *clusters*, grades e estruturas em nuvem. O aspecto de *hardware* relaciona-se com o fato dos sistemas distribuídos serem constituídos fisicamente por sistemas de computação autônomos interligados. Já o aspecto de *software* está relacionado com a maneira que o sistema aparenta ao usuário, ou seja, de maneira única (TANENBAUM; STEEN, 2007; COULOURIS *et al.*, 2013).

A computação distribuída possui atualmente um novo sentido, devido à maturidade que os sistemas distribuídos atingiram, sendo amplamente empregada na forma de serviços através de empresas. Esta abordagem é análoga ao que acontece quando serviços públicos como água e energia são contratados. Os serviços podem ser recursos físicos (armazenamento, processamento) ou serviços de *software* (COULOURIS *et al.*, 2013).

Em Tanenbaum e Steen (2007), a computação distribuída é considerada como uma classe de sistemas distribuídos, utilizando sistemas de computação distribuídos (*clusters*, grades) na solução de problemas que envolvem computação de alto desempenho.

Já a distinção entre a computação distribuída e a computação paralela não é bem clara na literatura. A computação paralela está mais relacionada com a busca pelo ganho de desempenho que os sistemas de computação podem oferecer. Em sistemas distribuídos, a computação paralela se dá pela troca de mensagens por redes de intercomunicação, onde os múltiplos processos se comunicam e se sincronizam para resolver algum problema, definição esta muito próxima da computação distribuída. Porém, a computação paralela pode utilizar como base sistemas de memória compartilhada, com múltiplos processos ou *threads* em um único sistema de computação, diferentemente da computação distribuída (PACHECO, 2011; TANENBAUM; STEEN, 2007).

2.2.2 Desafios em Sistemas Distribuídos

Os sistemas distribuídos trazem uma série de desafios que devem ser considerados (COULOURIS *et al.*, 2013). Ao se definir um sistema distribuído baseado em dispositivos móveis, os desafios a seguir também devem ser considerados.

1. Heterogeneidade

A heterogeneidade em sistemas distribuídos permite que os usuários acessem aplicações e serviços independentemente dos sistemas de computação envolvidos e das tecnologias de rede empregadas para interconectá-los. A heterogeneidade se refere a: redes de computadores, *hardware*, sistemas operacionais, linguagens de programação e diferentes implementações (COULOURIS *et al.*, 2013).

2. Abertura

   Um sistema computacional que pode ser re-implementado e ampliado por diferentes organizações é um sistema aberto. Para que isto seja possível é necessário que todas as interfaces e documentação do *software* sejam bem definidas e divulgadas, permitindo que sua reprodução seja possível (COULOURIS *et al.*, 2013).

3. Segurança

   A segurança pode ser subdividida em três componentes: confidencialidade, integridade e disponibilidade. A confidencialidade se refere à proteção das informações e módulos do sistema contra acessos não autorizados. Já a integridade está relacionada com a proteção contra a alteração não autorizada de qualquer característica ou informação do sistema. A disponibilidade é um característica direta da proteção contra qualquer interferência que limite o acesso aos recursos, como um ataque de negação de serviços (COULOURIS *et al.*, 2013).

4. Escalabilidade

   Um sistema distribuído escalável consegue ser adaptável ao ponto de se ajustar adequadamente quando o número de sistemas de computação que o integra varia. Com isso, os serviços e aplicações em um sistema distribuído devem se comportar da mesma maneira em termos de desempenho, capacidade, etc, independente da escala do sistema (COULOURIS *et al.*, 2013).

5. Tratamento de falhas

   Os sistemas computacionais estão suscetíveis a falhas das mais diferentes naturezas. Seja por *hardware* ou *software*, não há como garantir que um sistema nunca irá falhar em algum momento. O mesmo pode ser estendido aos sistemas distribuídos, uma vez que são baseados em estruturas de rede de computadores. Dessa forma, os sistemas distribuídos devem ser sensíveis às possíveis falhas, lidando com isso de várias maneiras: detectando falhas, mascarando falhas, tolerando falhas, recuperando-se de falhas e aplicando redundância (COULOURIS *et al.*, 2013).

6. Concorrência

   Os clientes de sistemas distribuídos podem acessar recursos compartilhados através das aplicações ou serviços oferecidos. O sistema deve estar preparado, tratando a concorrência

adequadamente com técnicas de exclusão mútua e sincronização (COULOURIS *et al.*, 2013).

7. Transparência

   - Acesso: a maneira e as operações de acesso ao sistema independem da localização do cliente;
   - Localização: os recursos podem ser acessados independentemente de sua localização;
   - Concorrência: a concorrência deve ser tratada de forma imperceptível ao usuário;
   - Replicação: as aplicações devem gerenciar as réplicas de dados e componentes de forma transparente;
   - Falhas: o sistema deve operar de forma sólida e transparente ao usuário, superando as possíveis falhas;
   - Mobilidade: permitir que os clientes e os recursos sejam dinâmicos no sistema, evitando que isso afete as operações;
   - Desempenho: garantir o desempenho independentemente da carga aplicada no sistema;
   - Escalabilidade: o sistema deve ser capaz de se expandir em escala sem alterar seu funcionamento e sua estrutura.

2.2.3 Modelos Arquiteturais

Os modelos arquiteturais dos sistemas distribuídos refletem diretamente na maneira que os componentes do sistema se interagem, e serão apresentados a seguir (COULOURIS *et al.*, 2013).

O modelo cliente-servidor talvez seja o mais utilizado e citado. Sua estrutura conta com dois tipos de processos: cliente e servidor. Nesta arquitetura o cliente é responsável por iniciar a comunicação, solicitando algum serviço do servidor, que por sua vez conta com objetos e métodos para prover tal requisição, respondendo ao cliente. Um servidor também pode requisitar algum serviço de outro, sendo visto como cliente (COULOURIS *et al.*, 2013). Um exemplo real de tal estrutura são os servidores Web, que atendem requisições de páginas *Web* de clientes (navegadores) espalhados pelo mundo. Ao mesmo tempo, os servidores *Web* são clientes dos servidores *Domain Name Service* (DNS), que traduzem endereços em linguagem natural para endereços de rede (COULOURIS *et al.*, 2013). Outro exemplo que pode ser citado é o de um celular requisitando o processamento de tarefas para um *Web service*.

O modelo cliente-servidor pode ser estendido quando o serviço, e seus respectivos dados, oferecido aos clientes está espalhado por vários processos servidores, os quais devem se comunicar e sincronizar de forma a atender as requisições. Um exemplo para tal modelo

arquitetural é a própria *Web*, onde vários servidores atuam de forma conjunta para prover os recursos (COULOURIS *et al.*, 2013).

Técnicas de *caching* são amplamente utilizadas nos mais diversos sistemas computacionais, de forma a minimizar o impacto do alto custo em comunicação e tranferência de dados. Os servidores *proxy* são empregados com o objetivo de diminuirem o custo de comunicação entre os clientes e os servidores, armazenando informações ou resultados que estão sendo acessados, ou até que serão acessados em um determinado espaço de tempo (COULOURIS *et al.*, 2013).

No modelo de processos *Peer* todos os processos envolvidos se comportam tanto como provedores de serviços quanto clientes, operando cooperativamente de forma distribuída e sincronizada. Este modelo consegue aproveitar os recursos computacionais das estações que também são clientes, explorando o potencial computacional (processamento, armazenamento, etc) que as arquiteturas cliente-servidor não consideram (COULOURIS *et al.*, 2013).

O modelo arquitetural cliente-servidor originou outros modelos que são importantes, uma vez que tais variações abrangem novos paradigmas, como aplicações que são capazes de serem migradas pelos recursos e redes com dispositivos móveis e intermitentes. Tais modelos derivados são apresentados a seguir:

- O conceito de *web applets* está relacionado com a migração de código e dados de um servidor para algum recurso mais próximo do cliente. Com isso é possível evitar o uso de interconexões de rede lentas que prejudicam o desempenho geral da aplicação (COULOURIS *et al.*, 2013).
- Um agente móvel é um programa capaz de executar em um computador e, automaticamente, fazer a sua migração (código e dados) para outras estações através da rede (*Internet*). Tal modelo resulta em preocupações acerca dos possíveis problemas de segurança (COULOURIS *et al.*, 2013).
- O modelo arquitetural chamado computadores em rede é caracterizado por um sistema de computação que, através de uma rede e um servidor de arquivos remoto, é capaz de requisitar os dados de uma determinada aplicação e executá-la localmente. Apesar disso, a aplicação e seus dados continuam no servidor de arquivos (técnicas de coerência são empregadas neste modelo) (COULOURIS *et al.*, 2013).
- Os *thin clients* são camadas de *software* baseadas em interfaces gráficas que acessam as aplicações remotamente, executando inteiramente em computadores remotos com maiores capacidades computacionais (COULOURIS *et al.*, 2013).
- Com o aumento da quantidade de dispositivos móveis (PDAs, *smartphones*, etc) e das tecnologias de intercomunicação disponíveis (GSM, Bluetooth, etc), surgiu o conceito de "redes espontâneas", das quais os dispositivos móveis fazem parte e conseguem descobrir e utilizar serviços de outros recursos (COULOURIS *et al.*, 2013).

2.2.4 Eleição em Sistemas Distribuídos

Muitos algoritmos distribuídos requerem que, em determinados momentos, um dos processos que integram o sistema se torne coordenador. Para que isto ocorra de maneira sincronizada e correta, são propostos algoritmos que garantem que a eleição do coordenador seja iniciada, finalizada e que todos os processos estejam cientes e de acordo com o resultado (TANENBAUM; STEEN, 2007).

Os algoritmos tradicionais de eleição em sistemas distribuídos consideram que não há distinção de características entre os processos, assumindo somente que existe um número único de identificação atribuído a cada um (TANENBAUM; STEEN, 2007).

Para o presente projeto esta sub-seção é importante pois o modelo arquitetural a ser proposto requer que dispositivos móveis sejam organizados em aglomerados, e cada aglomerado precisa de um nó coordenador.

O algoritmo tradicional a ser discutido é o *Bully*, baseado na troca de mensagens entre os processos de forma a decidir qual deles possui o maior número de identificação, sendo análogo a uma "disputa de quem é o maior"(TANENBAUM; STEEN, 2007). Seu funcionamento é baseado em três etapas:

1. A partir do momento que um processo P detecta a ausência de um coordenador, ele envia uma mensagem de eleição aos processos de maior número de identificação;

2. Se não há nenhuma resposta, o processo P é o novo coordenador, enviando uma mensagem de eleição a todos processos;

3. Se houver uma resposta de um processo com maior número de identificação, o processo P encerra o algoritmo de eleição, vencendo o processo que se encaixar na condição da etapa 2.

Assim como o algoritmo Bully, o algoritmo Ring também necessita de um número de identificação para cada processo. Com isso, os processos são organizados em forma "circular"(TANENBAUM; STEEN, 2007).

Em ambientes de comunicação sem fio *ad-hoc* (ponto-a-ponto) existem trabalhos, como em Vasudevan, Kurose e Towsley (2004), que buscam eleger nós como coordenadores de forma a contornar os problemas de falhas em redes sem fio através de mensagens de confirmação e de propagação de índices computacionais, assim como estruturas que protegem o funcionamento da rede contra falhas arbitrárias de nós e suas possíveis reinserções.

Em (PARK *et al.*, 2009) é proposto um algoritmo de eleição para ambientes de rede sem fio *ad-hoc*, o qual surgiu através da readaptação de um algoritmo para redes baseadas em enlaces físicos, que funciona de forma similar aos algoritmos de eleição convencionais já discutidos.

2.2.5 Grades Computacionais

Com a popularização da *Internet* e das tecnologias de rede de alto desempenho, tornou-se possível a utilização de computadores geograficamente distribuídos, se comportando como um único sistema de computação poderoso (BAKER; BUYYA; LAFORENZA, 2002). Esta nova maneira de organização, denominada grade computacional, teve como inspiração as redes elétricas, sendo os recursos acessados por demanda (BUYYA; ABRAMSON; VENUGOPAL, 2005). As grades móveis são um tipo particular de grade computacional, com características particulares, como a mobilidade.

O acesso aos recursos oferecidos por uma grade computacional é realizado através de servidores especializados, nomeados como *Grid Resource Brokers* (GRB), sendo responsáveis por descobrir recursos, realizar o escalonamento e a atribuição de tarefas a estes recursos. Esta organização pode ser vista na Figura 1 (BAKER; BUYYA; LAFORENZA, 2002).

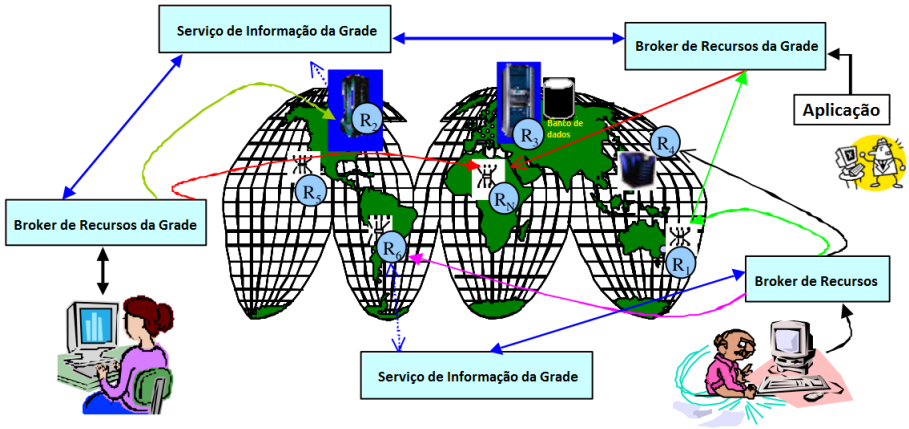

Figura 1 – Visão de alto nível de Grade e as interações entre as suas entidades, adaptada de (BAKER; BUYYA; LAFORENZA, 2002)

2.2.5.1 Classificações

De acordo com (BAKER; BUYYA; LAFORENZA, 2002), as grades são estruturas de provimento de recursos, oferecendo aos usuários alguns tipos de serviços:

- Serviços computacionais: provimento de serviços através da execução segura de tarefas oriundas de aplicações clientes, sendo os *brokers* responsáveis por gerenciar o provimento dos serviços com o uso colaborativo de recursos distribuídos;

- Serviços de dados: são responsáveis por oferecer o acesso seguro aos dados e seu gerenciamento, mantendo os arquivos em servidores distribuídos para garantir a disponibilidade através de técnicas de replicação;

- Serviços de aplicação: tais serviços se referem ao gerenciamento de aplicações e provimento ao acesso a *software* remoto e bibliotecas de forma transparente, utilizando tecnologias emergentes de *Web services*;
- Serviços de informação: são responsáveis pela extração e apresentação de dados através da utilização de serviços computacionais, de dados e de aplicação. Mais detalhadamente, os serviços de informação são responsáveis pela forma em que as informações são apresentadas, armazenadas, acessadas, compartilhadas e mantidas;
- Serviços de aprendizagem: estes serviços estão relacionados com a maneira que o conhecimento, que é a informação aplicada para atingir objetivos, é distribuído, armazenado e obtido.

2.2.5.2 Escalonamento

Os recursos de uma grade computacional são gerenciados por um sistema gerenciador de recursos, sendo o escalonador um dos seus principais componentes (KRAUTER; BUYYA; MAHESWARAN, 2002).

O escalonador, a partir de uma política interna, é responsável por promover a alocação das tarefas aos recursos da grade. Em (KRAUTER; BUYYA; MAHESWARAN, 2002) são definidos três modelos de escalonador de acordo com seu escopo, os quais são observados na Figura 2 e apresentados na sequência.

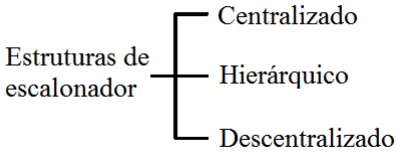

Figura 2 – Modelos de escalonador, adaptado de Krauter, Buyya e Maheswaran (2002)

1. Escalonamento centralizado: este modelo de escalonamento é caracterizado por possuir apenas um escalonador, o qual tem visão global de todos os recursos do sistema. Como vantagens pode-se citar a fácil manutenção, simples implantação e a possibilidade de co-alocar recursos. Porém, também é possível destacar algumas desvantagens: impossibilidade de escalabilidade, ausência de tolerância a falhas e dificuldade de criar políticas múltiplas (KRAUTER; BUYYA; MAHESWARAN, 2002);
2. Escalonamento hierárquico: os escalonadores são organizados em níveis. Os escalonadores nos níveis mais altos possuem uma visão mais genérica e simplificada dos recursos, e os de menores nível estão mais próximos dos recursos e alocam as tarefas de forma mais direta. Esta organização permite escalabilidade e tolerância a falhas, porém não provê autonomia e escalonamento de políticas múltiplas (KRAUTER; BUYYA; MAHESWARAN, 2002);

3. Escalonamento descentralizado: neste modelo, é possível organizar os escalonadores de forma a garantir tolerância a falhas, escalabilidade entre outros pontos positivos. Porém, o seu gerenciamento é mais complexo, podendo gerar sobrecargas no modelo por conta das sincronizações (KRAUTER; BUYYA; MAHESWARAN, 2002).

2.3 Computação Móvel

Os dispositivos móveis (*smartphones, tablets, notebooks, smartwatches*, etc) estão evoluindo rapidamente, mudando desde paradigmas da computação até o estilo de vida das pessoas. Os dispositivos móveis são sistemas computacionais restritos quando comparados com sistemas convencionais, apesar de expressivos avanços tecnológicos, como a melhoria de processadores *multi-core* e grande capacidade de memória. Assim sendo, os trabalhos acadêmicos na área de computação móvel podem explorar a capacidade computacional dos dispositivos móveis, sendo otimista em relação aos avanços, ou então focar apenas nas restrições que existem.

Dessa forma, há uma vertente de pesquisa que possui o foco apenas nas restrições (computacionais, conexão, energia) que os dispositivos móveis possuem em relação aos sistemas convencionais de computação (PCs, estações de trabalho, servidores, etc). Alguns trabalhos exploram a técnica de *offloading*, que se resume em transferir parte ou toda aplicação para sistemas remotos, como serviços de nuvem e servidores dedicados, economizando energia e expandindo a capacidade dos dispositivos móveis (ELGAZZAR; MARTIN; HASSANEIN, 2014; ZHANG; WEN; WU, 2013).

Também é possível encontrar trabalhos com outra abordagem, considerando os dispositivos como provedores de recursos. Para isso, estudos buscam utilizar os recursos computacionais de forma eficiente, reduzindo gastos desnecessários (comunicação, sincronização, recursos ociosos, etc) (HASSAN; ZHAO; YANG, 2010; ELGAZZAR; MARTIN; HASSANEIN, 2014; MORSY; EL-REWINI, 2013).

Independentemente do foco dos trabalhos acerca da computação móvel, existem desafios e dificuldades que são presentes sempre que dispositivos móveis são considerados: baixa capacidade de energia, custos de comunicação e alta mobilidade (MORSY; EL-REWINI, 2013; ZHANG; WEN; WU, 2013; HASSAN; ZHAO; YANG, 2010; ELGAZZAR; MARTIN; HASSANEIN, 2014). O presente trabalho considera o uso dos dispositivos móveis tanto como provedores de recursos através da disponibilização de serviços, quanto como usuário de recursos, numa abordagem colaborativa.

2.4 Grades Móveis

As grades móveis são formadas por dispositivos móveis e convencionais interconectados por rede, permitindo que a computação móvel se torne possível. As grades móveis podem

considerar que os dispositivos móveis são recursos computacionais, vistos como extensão natural das grades convencionais, ou ainda que os dispositivos móveis são clientes de outros elementos computacionais que fazem parte da infraestrutura, como estações de trabalho e servidores (LI *et al.*, 2009).

Os trabalhos que envolvem grades móveis geralmente tratam os dispositivos móveis de maneira genérica, não importando se os nós são *smartphones*, *tablets*, *laptops* ou sensores sem fio, sendo possível ver esta integração na Figura 4, em que os dispositivos se interconectam independentemente de sua natureza.

Esta maneira de explorar os dispositivos como provedores em sistemas distribuídos e móveis está relacionada com trabalhos que investigam os dispositivos móveis para o processamento de aplicações de alta demanda computacional (científicas, computação de alto desempenho). Em Rodríguez, Mateos e Zunino (2012) foi realizado um estudo comparativo que investigou o uso de *smartphones* para o processamento de problemas em computação científica. Os autores verificaram que é possível utilizar *smartphones* para realizar cálculos científicos, com maior eficiência em problemas iterativos e com preponderância de operações em inteiros.

2.4.1 Arquiteturas de Grades Móveis

É possível classificar as grades móveis de acordo com a forma em que os componentes são estruturados, apesar de em ambas serem utilizadas tecnologias de comunicação sem fio. A seguir serão apresentadas as duas classificações segundo (KATSAROS; POLYZOS, 2007).

2.4.1.1 On-site

A arquitetura *on-site* é caracterizada por possuir uma entidade central, responsável por agrupar os dispositivos de forma a interligá-los. Assim, para estruturar uma grade móvel dessa maneira é necessário existir uma infraestrutura de rede (KATSAROS; POLYZOS, 2007; GHOSH; DAS, 2010).

Um exemplo de arquitetura *on-site* pode ser vista na Figura 3. Esta arquitetura de grade móvel é baseada em redes de dispositivos móveis, sendo que cada aglomerado de rede de dispositivos, *Basic Service Set* (BSS), possui um *Wireless Access Point* (WAP), responsável por monitorar os recursos em sua área (BSS) e interligar ao controlador da grade através de roteadores de borda. A integração de dois ou mais BSSs é chamada de *Extended Service Set* (ESS) (GHOSH; DAS, 2010).

2.4.1.2 Ad hoc

As grades móveis *ad hoc* não dependem de uma infraestrutura de rede pré-estabelecida, e são implantadas em lugares sem um elemento central (KATSAROS; POLYZOS, 2007). Os

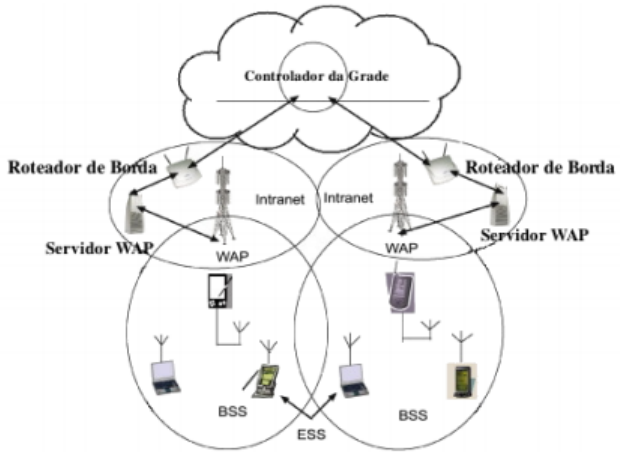

Figura 3 – Arquitetura de grade móvel *on-site* adaptada de Ghosh e Das (2010)

elementos dessa organização se comunicam diretamente, fazendo o repasse de mensagens de outros nós quando necessário (comunicação *multi-hop*) (SHAH, 2013).

Essa ausência de elemento central impõe desafios na descoberta de recursos, no escalonamento de tarefas e no gerenciamento dos componentes, uma vez que o sincronismo e a coordenação nesse cenário devem levar em consideração a maneira que as mensagens são trocadas e propagadas (KATSAROS; POLYZOS, 2007).

Na Figura 4 é possível ver diversos dispositivos que se comunicam de forma *ad hoc*, podendo ser utilizados como recursos (KATSAROS; POLYZOS, 2007).

Figura 4 – Arquitetura de grade móvel *ad hoc* adaptada de Katsaros e Polyzos (2007)

2.4.2 Escalonamento de Tarefas em Grades Móveis

O escalonamento de tarefas, tanto em grades convencionais quanto em grades móveis, é baseado em três principais componentes: consumidores, recursos e política. A política de escalonamento segue algum critério, resolvendo o problema de atribuição das tarefas dos consumidores aos recursos (CASAVANT; KUHL, 1988).

O trabalho de Casavant e Kuhl (1988) propõe uma classificação hierárquica dos diferentes tipos de escalonamento para máquinas massivamente paralelas, que pode ser visualizada na Figura 5.

A primeira diferenciação entre os modelos de escalonamento define-os como local ou global. O escalonamento local considera apenas um processador que recebe as tarefas divididas pelo tempo. Já o escalonamento global faz a alocação de tarefas em vários recursos (CASAVANT; KUHL, 1988).

O escalonamento global, por sua vez, pode ser sub-dividido em estático ou dinâmico. O escalonamento estático é realizado antes da execução real das tarefas, e pode ser ótimo, apenas se realizado com todas as informações dos recursos e tarefas, e de alta complexidade de resolução, ou sub-ótimo, que também é dividido entre heurístico (baseado em regras empíricas) ou aproximação (CASAVANT; KUHL, 1988).

Já o escalonamento dinâmico pode ser centralizado (realizado por um elemento central) ou distribuído. O escalonamento distribuído pode ser não-cooperativo (cada escalonador resolve o problema de atribuição localmente). Já o cooperativo é caracterizado pela comunicação e pelo sincronismo entre os escalonadores, de forma a atingir algum objetivo global (CASAVANT; KUHL, 1988).

As grades móveis herdam as características que as grades convencionais (estáticas) possuem, além de estendê-las a ambientes pervasivos e dinâmicos, com recursos computacionais limitados e recursos energéticos escassos (THENMOZHI; TAMILARASI, 2010).

O escalonamento de tarefas em grades móveis deve considerar a intermitência e a garantia de QoS que deve ser oferecida. O escalonador da grade deve priorizar a economia de energia da grade (CHANG-QIN *et al.*, 2006; BORRO, 2013).

2.4.3 Desafios

As grades móveis, por serem compostas por dispositivos portáteis e com algumas limitações, possuem desafios além dos já conhecidos em sistemas distribuídos.

Os nós de uma grade móvel podem ser movidos sem previsibilidade, sendo mais um fator a ser considerado. Tanto a comunicação quanto o escalonamento de tarefas para a grade devem considerar que os dispositivos não são confiáveis como nas grades computacionais convencionais (SHAH, 2013). Um nó da grade móvel pode simplesmente deixar de ser um recurso disponível

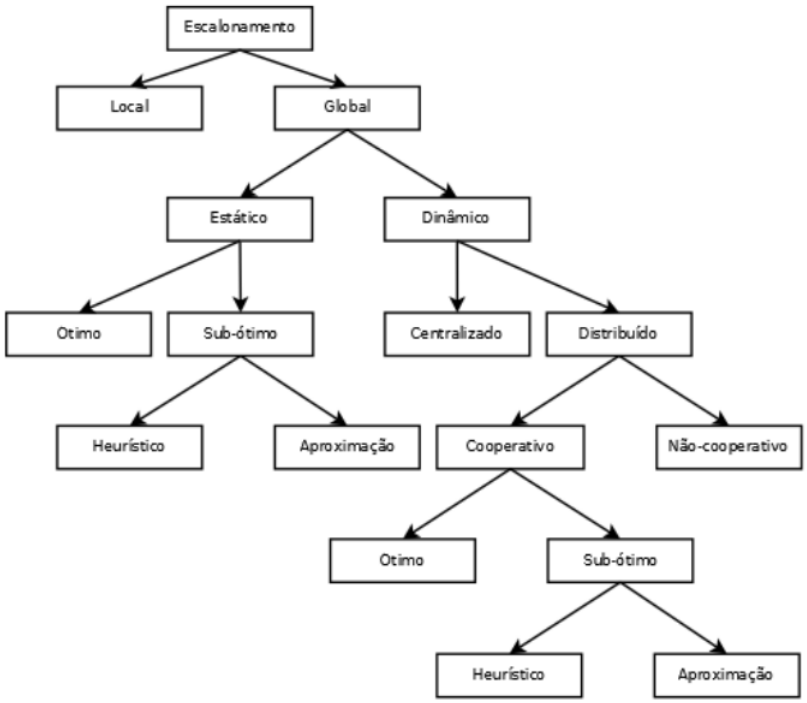

Figura 5 – Classificação hierárquica do problema de escalonamento proposta por Casavant e Kuhl (1988) e adaptada por Dantas (2005).

também por conta da intermitência das tecnologias sem fio (ZENG *et al.*, 2008).

Ao se atribuir tarefas para a execução nos nós é importante considerar o fator de mobilidade, adotando técnicas de migração de tarefas entre os nós e replicação, escalonando a mesma tarefa para recursos diferentes, de forma a aumentar a chance da mesma ser processada integralmente, além de priorizar os nós mais confiáveis (SHAH, 2013).

Outro grande desafio das grades móveis é a questão energética. De acordo com Shah (2013), os componentes que mais consomem energia são: *Computing Processing Unit* (CPU), memória e comunicação de rede. Os dispositivos são integrados como parte das grades computacionais, e a restrição que as baterias impõem faz com que os trabalhos nesta área investiguem como utilizá-los de forma eficiente (RODRíGUEZ; MATEOS; ZUNINO, 2012).

Assim como o fator de mobilidade, a energia pode ser vista como um fator ligado à QoS do sistema, uma vez que seu uso sem controle impacta diretamente na degradação dos serviços oferecidos pela grade (ZENG *et al.*, 2008).

As grades móveis são heterogêneas, ou seja, compostas por diferentes dispositivos, que por sua vez são propriedade de pessoas ou organizações. Além do mais, as grades móveis podem integrar cenários militares, cidades em recuperação pós-catástrofes e vigilância urbana, sendo requisito primário a segurança de toda a informação que a grade manipula (SHAH, 2013). As restrições computacionais que as grades móveis possuem tornam mais difícil agregar requisitos de segurança, sendo uma área de pesquisa em expansão (ROSADO *et al.*, 2011).

Outra dificuldade é convencer os proprietários dos recursos a colaborarem com a grade móvel, permitindo que serviços como ciclos de processamento ou o uso de algum sensor sejam executados em seus dispositivos em prol de outros usuários (SHAH, 2013).

Uma pesquisa realizada por Ma *et al.* (2012) mostrou que as maiores preocupações dos usuários em relação à participação na grade como provedor de recursos são o aumento do consumo de energia e a privacidade de suas informações.

Nesse sentido, alguns trabalhos buscam desenvolver mecanismos de recompensa como incentivo à colaboração. Em Duan *et al.* (2012) são criados dois mecanismos de recompensa para o incentivo dos usuários, sendo um para aplicações de aquisição de dados e outro para o processamento distribuído.

A qualidade de serviço nas grades móveis está relacionada com os requisitos que as aplicações demandam, como energia, garantias de banda de rede e tempo de atendimento (SHAH, 2013).

Os desafios de mobilidade e restrição energética também interferem na QoS. Se um recurso passa a ser indisponível é preciso que de alguma maneira o sistema se readapte, gerando uma sobrecarga que pode degradar a QoS (SHAH, 2013; ZENG *et al.*, 2008).

2.4.4 Mobile Cloud Computing

Alguns trabalhos argumentam que os dispositivos computacionais móveis, como os *smartphones* são restritos, e que os avanços nas redes de comunicação sem fio permitem mitigar as restrições locais e economizar energia utilizando serviços em sistemas de computação remotos, como por exemplo, serviços disponíveis na nuvem.

Esta integração entre dispositivos móveis e estruturas em nuvem é chamada de *Mobile Cloud Computing*. Dessa forma, a infraestrutura que os serviços de nuvem oferecem é considerada como uma extensão do dispositivo móvel (DEV; BAISHNAB, 2014; KOTWAL; SINGH, 2012; DINH *et al.*, 2013; SHARMA; KUMAR; TRIVEDI, 2013).

A *Internet* é essencial para *Mobile Cloud Computing*, interligando os dispositivos móveis com as infraestruturas em nuvem (*Data centers*, servidores de aplicação). Os dispositivos podem utilizar armazenamento, processamento e dados da nuvem, sendo esses serviços oferecidos através de técnicas de virtualização e elasticidade (DINH *et al.*, 2013). Esta arquitetura pode ser observada na Figura 6.

A integração entre os dispositivos móveis e a computação em nuvem permite que várias vantagens sejam exploradas, como discutidas a seguir:

- Economia de energia: dispositivos móveis possuem severas restrições energéticas, uma vez que a tecnologia empregada na produção de baterias não se desenvolveu na mesma proporção dos outros componentes. Uma das técnicas empregadas para a economia de

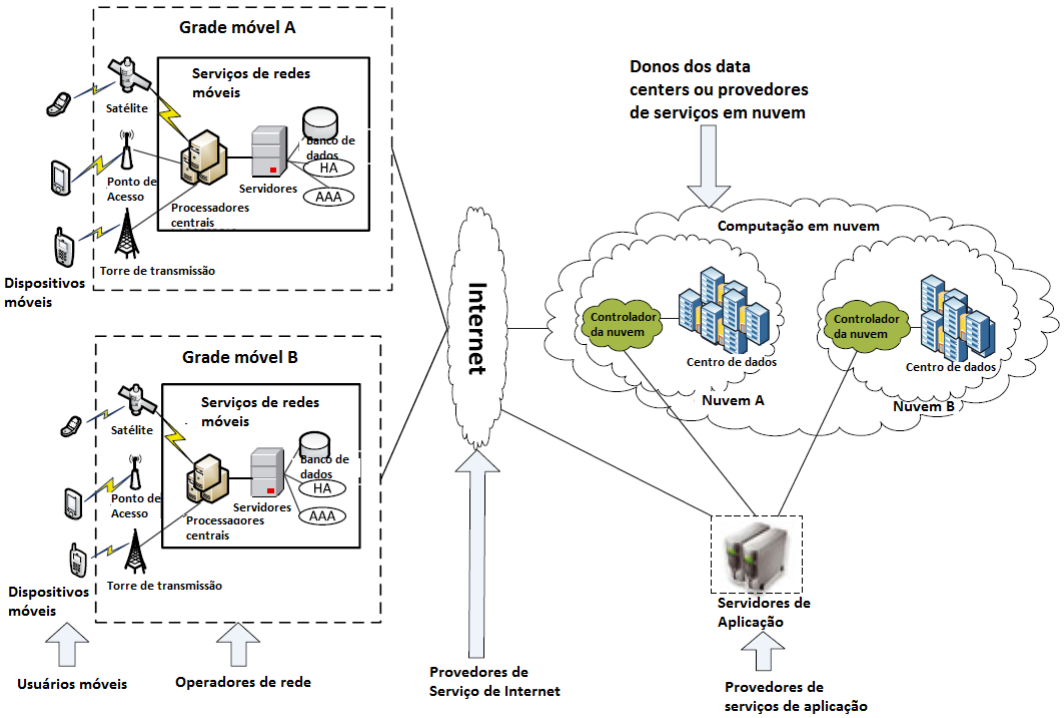

Figura 6 – Arquitetura de *Mobile Cloud Computing* adaptada de Dinh *et al.* (2013)

energia é o *offloading*, que resumidamente é a migração de processamento complexo e dispendioso dos dispositivos móveis e restritos para a nuvem (DEV; BAISHNAB, 2014; DINH *et al.*, 2013);

- Aumento da capacidade: outra restrição de dispositivos como *smartphones* e *tablets* é a capacidade de armazenamento e processamento em comparação com sistemas de computação convencionais. Os serviços de nuvem, como o *Amazon Simple Storage Service (Amazon S3)*, oferecem soluções que podem superar tais limitações (DEV; BAISHNAB, 2014; DINH *et al.*, 2013);

- Aumento da confiabilidade: o uso de serviços em nuvem permite que as informações das aplicações móveis sejam armazenadas em diversos computadores em infraestruturas com grande capacidade. Dessa forma, são exploradas implicitamente técnicas de replicação e serviços de segurança (DEV; BAISHNAB, 2014; DINH *et al.*, 2013).

2.4.4.1 Offloading

Offloading pode ser definido como a técnica de migrar processamento de dispositivos móveis para outros sistemas de computação, objetivando melhorar o desempenho e poupar energia (ELGAZZAR; MARTIN; HASSANEIN, 2014).

Existem três formas de realizar o *offloading*. A primeira é através da invocação remota de serviços implantados na nuvem através de mecanismos como o *Remote Procedure Call* (RPC)

e o *Remote Method Invocation* (RMI), permitindo que *Web services* sejam acessados de forma padronizada, oferecendo a invocação de serviços. A segunda consiste na migração de máquinas virtuais, que se resume em migrar a cópia da memória do dispositivo móvel (incluindo estado, processos, etc) para os servidores em nuvem. A terceira é a migração de código, transportando as instruções e os dados para servidores remotos para a execução fora do dispositivo móvel (ELGAZZAR; MARTIN; HASSANEIN, 2014).

Em Zhang, Wen e Wu (2013) os autores investigam a execução colaborativa entre o dispositivo local e sistemas de computação em nuvem. Para isto, a rede de comunicação e a aplicação, particionada em tarefas, formam um grafo acíclico direcionado que tem os pesos das arestas proporcionais à energia dispendida para ou migrar o código para a nuvem ou executar localmente. Assim, também utilizando técnicas de Relaxação Lagrangiana, a heurística proposta consegue uma solução aproximada que consegue poupar energia no dispositivo móvel e garantir que a aplicação seja processada em tempos viáveis.

2.4.4.2 Aplicações em Mobile Cloud Computing

Algumas aplicações que exploram as vantagens e características que o MCC possui são:

- *Mobile commerce*: Estas aplicações são integradas aos serviços em nuvem para superar dificuldades como conexões lentas, heterogeneidade dos dispositivos móveis, segurança, etc. MCC permite que os clientes de *m-commerce* realizem pagamentos, negociações, etc; de forma segura e simples, apesar das limitações dos dispositivos (DINH *et al.*, 2013; SHARMA; KUMAR; TRIVEDI, 2013).

- *Mobile learning*: *m-learning* é uma aplicação baseada no *eletronic learning*, permitindo que os usuários adquiram conhecimento através de dispositivos como celulares e *tablets*. MCC pode fornecer serviços de informação através de recursos de processamento e armazenamento, ampliando as capacidades dos dispositivos, o que possibilita a popularização da aplicação através de dispositivos baratos e com baixo potencial computacional (DINH *et al.*, 2013; SHARMA; KUMAR; TRIVEDI, 2013).

- *Mobile healthcare*: Esta aplicação utiliza a MCC também no sentido de mitigar as limitações e problemas dos tratamentos médicos convencionais, permitindo o acesso rápido e fácil às informações dos pacientes, evitando erros médicos, entre outras vantagens (DINH *et al.*, 2013).

- *Mobile gaming*: O mercado de jogos para dispositivos móveis possui um imenso potencial, e utiliza serviços em servidores para exportar desde partes das aplicações até motores gráficos completos, possibilitando a expansão do potencial computacional do dispositivo e a economia de energia (DINH *et al.*, 2013; SHARMA; KUMAR; TRIVEDI, 2013).

2.4.4.3 Tendências em Mobile Cloud Computing

Por ser uma área tecnicamente recente e em exploração é possível discutir algumas tendências e direções futuras de pesquisas em MCC.

Limitações na Comunicação: apesar do avanço das redes sem fio, como a popularização da quarta geração (4G), os dispositivos móveis são prejudicados por limitações de velocidade de banda de rede, influenciando diretamente na QoS das aplicações que dependem da computação em nuvem, resultando em pesquisas que tentam otimizar a alocação de banda (DINH *et al.*, 2013).

Qualidade de Serviço: a qualidade de serviço é diretamente afetada por problemas que as tecnologias de rede sem fio possuem, como congestionamento de canal, desconexões, atenuações de sinal, etc. Dessa forma, atrasos de comunicação com a nuvem causam redução de QoS (DINH *et al.*, 2013). Os trabalhos nesta vertente buscam, no geral, investigar maneiras de estruturar e aproximar os servidores remotos de forma a minimizar os atrasos de rede (DINH *et al.*, 2013).

Modelos de Mercado: toda a infraestrutura entre os dispositivos móveis e a nuvem gera custos e despesas, que devem ser levados em consideração. As pesquisas neste campo estão direcionadas em criar modelos de negócio para cobrança (DINH *et al.*, 2013).

Convergência de Serviços: esta lacuna a ser preenchida em MCC é a mais importante para este projeto. A tendência é que os serviços oferecidos através de MCC serão cada vez mais diferenciáveis dependendo de seu tipo, disponibilidade, QoS, demanda dos usuários, etc. Com isso, é necessário compor e esquematizar como diferentes serviços de nuvem serão compostos de maneira a atender esses requisitos (DINH *et al.*, 2013).

2.5 Fog Computing

A expansão da *Internet of Things* (IoT) fez com que uma grande quantidade de objetos fossem integrados à *Internet*, resultando em uma substancial quantidade de dados gerados. A estimativa é que até 2020 sejam 24 bilhões de dispositivos conectados à *Internet* (GUBBI *et al.*, 2013). Servidores em nuvem, que muitas vezes fornecem serviços computacionais às aplicações de IoT, podem prejudicá-las, uma vez que muitas delas são sensíveis a atrasos de comunicação (DASTJERDI *et al.*, 2016; BONOMI *et al.*, 2012).

Fog computing surgiu como uma forma de aproximar elementos de computação na borda da *Internet*, diminuindo a latência e os atrasos de comunicação em relação a serviços em nuvem distantes e muitas vezes sobrecarregados. Essa estratégia possibilita aumentar a QoS das aplicações na camada mais externa da *Internet*, uma vez que serviços distantes e centralizados são evitados (DASTJERDI *et al.*, 2016; BONOMI *et al.*, 2012; STOJMENOVIC; WEN, 2014; HONG *et al.*, 2013; MADSEN *et al.*, 2013; AAZAM; HUH, 2015; LEWIS *et al.*, 2014).

Essa abordagem foi, em um primeiro momento, proposta pela Cisco, que começou a

considerar elementos como roteadores e servidores dedicados próximos à borda da Internet para prover serviços aos elementos na extremidade da mesma. *Fog computing* não é um paradigma que busca substituir a *Cloud computing*, e sim servir como suporte e extensão (DASTJERDI *et al.*, 2016; STOJMENOVIC; WEN, 2014; HONG *et al.*, 2013).

Na Figura 7 é possível ver uma arquitetura de referência segundo Stojmenovic e Wen (2014) que representa a organização da *Fog computing*, que está situada entre a nuvem e os dispositivos de borda, como celulares, carros inteligentes, sensores, atuadores etc, os quais geram muitos dados que podem ser processados ainda na borda da *Internet*.

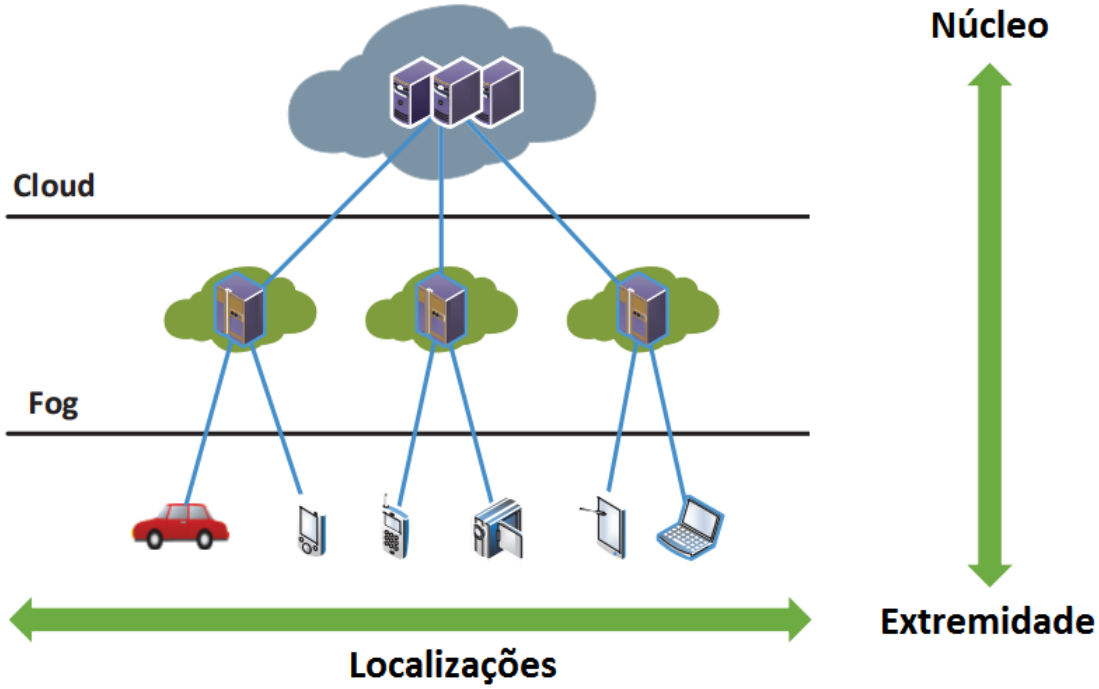

Figura 7 – Arquitetura de Referência de *Fog Computing* adaptada de Stojmenovic e Wen (2014)

Dessa forma, *Fog computing* possibilita que esta integração com a nuvem implique em algumas vantagens (DASTJERDI *et al.*, 2016):

- **Redução no tráfego de rede:** conforme o número de dispositivos geradores de dados aumenta na *Internet*, a quantidade de tráfego gerado também aumenta, e *Fog computing* possibilita reduzi-lo ao evitar recursos que estão distantes;
- **Mobilidade:** *Fog computing* deve suportar a mobilidade dos elementos que estão na borda da *Internet*;
- **Alta escalabilidade:** por extender os serviços que estão no núcleo da *Internet*, *Fog computing* possui um elevado grau de virtualização e elasticidade, se adaptando ao grande número de dispositivos e à massiva quantidade de dados gerados por eles;

- **Baixa latência:** a proximidade dos recursos computacionais implica em uma baixa latência nas proximidades da borda da rede;
- **QoS:** *Fog computing* pode possibilitar que os níveis de QoS das aplicações sejam melhorados, uma vez que a latência pode ser diminuída e os recursos expandidos.

Fog computing pode ser empregada em vários cenários e aplicações, como em sistemas de monitoramento da saúde de pessoas para evitar por exemplo problemas cardíacos, além de aplicações de realidade aumentada com sensibilidade à alta latência e elevada quantidade de dados gerados, sistemas de *caching* e armazenamento provisório de dados, suporte a soldados em campos de batalha com dispositivos inteligentes, entre muitas outras aplicações (DASTJERDI *et al.*, 2016; LEWIS *et al.*, 2014).

Porém, algumas dificuldades precisam ser superadas, indicando direções de pesquisa a serem exploradas:

- **Segurança:** a segurança é um aspecto crítico em *Fog computing*, uma vez que sua característica de ser um sistema distribuído em larga escala resulta em vários pontos com possíveis falhas de segurança, sendo necessário garantir níveis de segurança dos dados envolvidos, como controle de acesso (DASTJERDI *et al.*, 2016; STOJMENOVIC; WEN, 2014);
- **Heterogeneidade:** gerenciar os recursos na borda da *Internet* é um desafio, uma vez que pode haver alta heterogeneidade de *hardware* e *software* (DASTJERDI *et al.*, 2016);
- **Disponibilidade e Confiabilidade:** como os elementos que compõem um ambiente de *Fog computing* são heterogêneos, como servidores dedicados, sensores, dispositivos móveis, etc, sua mobilidade afeta incisivamente na disponibilidade, sendo um desafio desse conceito (DASTJERDI *et al.*, 2016; MADSEN *et al.*, 2013);
- **Minimização de energia:** como muitos nós são distribuídos e muitas vezes há restrição de energia, minimizar o impacto no consumo de energia dos elementos é primordial (DASTJERDI *et al.*, 2016);
- **Imprevisibilidade dos clientes:** alocar recursos para dispositivos com características de muitas desconexões é um desafio, uma vez que os clientes de *Fog computing* são instáveis e podem solicitar serviços e se tornar indisponíveis (AAZAM; HUH, 2015).

2.6 Considerações Finais

Conceitos fundamentais para este projeto de Mestrado foram apresentados, mostrando como o amadurecimento dos sistemas distribuídos e da *Internet* permitiu que novos paradigmas

de computação distribuída fossem criados, como as grades móveis, a posterior integração dos dispositivos móveis com a nuvem (MCC) e, mais recentemente, a *Fog computing*. No próximo capítulo são apresentados os trabalhos relacionados a este projeto, assim como o diferencial que este projeto de Mestrado tem em relação aos mesmos.

CAPÍTULO

3

TRABALHOS RELACIONADOS

3.1 Considerações Iniciais

Neste capítulo são discutidos os trabalhos relacionados a este projeto de Mestrado, sendo primeiramente detalhados na Seção 3.2 e depois comparados em relação às suas principais características na Seção 3.3, mostrando qual a lacuna que este projeto vem preencher, delimitando seu escopo de acordo com os objetivos já apresentados no Capítulo 1.

3.2 Apresentação dos Trabalhos Relacionados

Através do agrupamento dos dispositivos móveis em organizações virtuais, Isaiadis e Getov (2005) propuseram uma estrutura que permite o uso de dispositivos móveis, contornando aspectos de mobilidade e limitações. O agrupamento utilizado visa a superar a heterogeneidade e a dinamicidade que o ambiente móvel possui, mascarando e recuperando de falhas de forma a não degradar requisitos de QoS dos serviços que estão sendo oferecidos pela grade.

Para realizar esta virtualização do aglomerado de dispositivos os autores definiram um conjunto de servidores dedicados que atuam entre os dispositivos móveis e a infraestrutura da grade. Assim, estes servidores são responsáveis por tratar as falhas internamente, re-escalonando os recursos ou migrando dados quando necessário (ISAIADIS; GETOV, 2005).

Os autores ainda utilizaram a especificação OGSA/OGSI através da *Globus Toolkit* 3 (FOSTER; KESSELMAN, 1999), que é uma implementação dos padrões OGSA (Open Grid Services Architecture), sendo estes fundamentados em orientação a serviços e utilizados como *framework* para grades computacionais com mecanismos de segurança, gerenciamento de dados, etc. Com isso, são oferecidos dois tipos de serviços: funcionalidade específica (por exemplo acesso a bases de dados, acesso a sensores) e acesso a recursos controlados (ciclos de processamento, memória, armazenamento, etc) (ISAIADIS; GETOV, 2005).

Seu funcionamento se inicia na publicação de um serviço pelo dispositivo móvel, que envia uma mensagem com um padrão de descrição de *Web services* WSDL e um arquivo XML com o serviço e a informação do dispositivo. Essas mensagens são processadas e passam por vários processos, até o serviço ser propriamente publicado como registro na UDDI (*Universal Description, Discovery and Integration*) (ISAIADIS; GETOV, 2005).

O trabalho de Isaiadis e Getov (2005) concluiu através de técnicas de simulação que o modelo atendeu os requisitos de desempenho mesmo quando submetido a altas cargas de trabalho, não apresentando gargalos na rede com a estrutura de *proxies* simplificados. O que não foi considerado neste trabalho foi o aspecto de mobilidade dos dispositivos, sem avaliar o impacto do provimento de serviços nos dispositivos e na usabilidade do usuário, verificando como dispositivos não confiáveis interferem em métricas como tempos de resposta e gasto de energia.

Em (HUERTA-CANEPA; LEE, 2010) os autores argumentaram que dispositivos móveis são restritos e heterogêneos. Utilizar recursos remotos, como servidores em nuvem, nem sempre é possível, uma vez que é necessário primeiramente que haja acesso à *Internet*, devendo ser posteriormente levado em consideração o custo de comunicação.

O escopo do trabalho é definido em situações em que o dispositivo móvel precisa utilizar recursos externos, porém a comunicação com algum computador sem severas restrições de recurso não é possível ou é muito custosa. A partir disso foi elaborado um *framework* que permite que dispositivos na vizinhança sejam utilizados como recursos, criando grades móveis colaborativas e temporárias (HUERTA-CANEPA; LEE, 2010).

A arquitetura possui um mecanismo que verifica quando torna-se necessário executar uma tarefa (parte da aplicação) no dispositivo e os recursos locais não são suficientes. Então a vizinhança é monitorada e um módulo é responsável por instrumentar os *bytecodes* da aplicação para exportar tarefas para os recursos próximos (HUERTA-CANEPA; LEE, 2010).

Através de um protótipo desenvolvido baseado no Hadoop (plataforma para computação distribuída e processamento de grandes quantidades de dados desenvolvida pela Apache) os autores concluíram que a arquitetura consegue aproveitar o ambiente pervasivo que as grades móveis possuem (HUERTA-CANEPA; LEE, 2010). Porém os próprios autores reconhecem que este foi um trabalho preliminar, o qual não considerou os aspectos de energia e mobilidade, além de estar restrito a um contexto local.

Outro trabalho importante a ser debatido nesta seção é (HASSAN; ZHAO; YANG, 2010), que propõe um *framework* que permite que dispositivos móveis ofereçam serviços, e que provedores estáticos sejam utilizados quando há a necessidade de se processar mais do que a estação móvel pode oferecer.

Para tal é definida uma arquitetura responsável por recebe as requisições SOAP que são, a princípio, encapsuladas em mensagens HTTP. As mensagens SOAP são processadas,

repassando as requisições para o motor de execução, responsável por invocar o respectivo serviço (HASSAN; ZHAO; YANG, 2010).

O particionamento dos serviços é baseado em um provedor de serviços móvel (com restrições de recursos) e um provedor de serviços estático (servidor dedicado e com mais recursos). Esta configuração permite que partes custosas do serviço possam ser delegadas ao provedor estático, o qual não possui sérias restrições computacionais (HASSAN; ZHAO; YANG, 2010).

A decisão da estratégia de execução depende de um modelo chamado "*context manager*", gerando um coeficiente que considera o potencial dos recursos (memória e frequência de trabalho do processador) e a demanda computacional que a aplicação requer, além de pesos (HASSAN; ZHAO; YANG, 2010).

A avaliação foi baseada em protótipo, implementando dois serviços básicos para o modelo. Os autores concluíram que o modelo permitiu um ganho de desempenho das aplicações em dispositivos móveis. Porém nenhum aspecto de energia ou mobilidade foi levado em consideração, além do modelo estar limitado em duas partições e não permitir sua aplicação através da *Internet* (HASSAN; ZHAO; YANG, 2010).

O trabalho de (MORSY; EL-REWINI, 2013) propôs um escalonador de tarefas dinâmico e adaptativo capaz de lidar com tarefas de tempo-real, e que consiga lidar com variações de rede e de mobilidade do ambiente.

Primeiramente, foi definida a arquitetura alvo do trabalho. A arquitetura é baseada no modelo *on-site*, ou seja, há a necessidade de infraestrutura de rede. Os dispositivos são dispostos em aglomerados com um ponto de acesso sem fio. Em cada aglomerado, além dos dispositivos estarem conectados com o ponto de acesso, há um coordenador nomeado como WiGO (*Wireless Grid Operator*). O WiGO é responsável por monitorar os recursos do aglomerado e realizar o escalonamento de tarefas submetidas por um usuário. Esta arquitetura pode ser vista na Figura 8 (MORSY; EL-REWINI, 2013).

Para realizar o escalonamento, a arquitetura é mapeada na forma de um grafo, sendo os dispositivos móveis e os WiGOs representados por nós. São definidos pesos nas arestas determinados através de uma função de custo ponderada que considera tanto o fator de mobilidade dos nós quanto o fator de bateria restante. Uma vez construído o grafo, foram propostas duas heurísticas que definem a estratégia de escalonamento através de possíveis caminhos (MORSY; EL-REWINI, 2013).

A primeira heurística tenta achar o melhor caminho no grafo através de uma adaptação do algoritmo de fluxo máximo. A ideia dessa adaptação é conseguir maximizar o número de tarefas a serem executadas sem que aconteçam violações de *deadline*, também considerando a mobilidade e a energia dos nós (MORSY; EL-REWINI, 2013).

A segunda heurística também tenta escalonar o máximo de tarefas possível sem violar

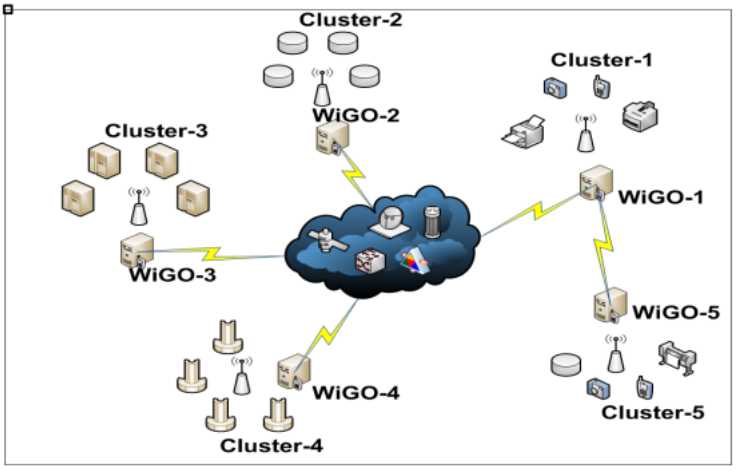

Figura 8 – Arquitetura considerada em (MORSY; EL-REWINI, 2013)

os *deadlines* e respeitando as características de mobilidade e de consumo de energia. Porém sua implementação é baseada na modificação do algoritmo de fluxo de rede de menor custo (MORSY; EL-REWINI, 2013).

A avaliação foi realizada através de se simulação no GridSim V4.1, comparando as duas heurísticas com os seguintes algoritmos de escalonamento:

- Escalonador aleatório: utilizado para servir de *baseline* nos testes, e faz o escalonamento sem considerar nenhuma informação dos recursos;
- NumrodG modificado: é um escalonador de aspecto "guloso", que faz o escalonamento das tarefas com os *deadlines* mais curtos para os recursos menos sobrecarregados (MORSY; EL-REWINI, 2013);
- PARM: realiza o escalonamento focando nas tarefas com os *deadlines* mais próximos, levando em consideração a energia e a mobilidade (MORSY; EL-REWINI, 2013).

Os autores concluíram que as heurísticas propostas possuem menor sobrecarga do que os algoritmos gulosos que serviram de comparação (MORSY; EL-REWINI, 2013). Porém, esta abordagem necessita de nós especializados geograficamente distribuídos, o que a tornaria pouco atrativa num contexto real. Além do mais, os algoritmos de alocação não buscam minimizar o gasto de energia, e sim maximizar a vazão dos usuários, não verificando como a alta mobilidade dos nós afeta no consumo de energia dos recursos e nas métricas de desempenho.

Outro trabalho relacionado é o apresentado em (ELGAZZAR; MARTIN; HASSANEIN, 2014), que tem como objetivo construir um *framework* capaz de dar suporte ao provisionamento de serviços móveis. Seu funcionamento permite que um dispositivo móvel provedor de serviços faça o *offloading* dinâmico quando o processamento exigido é alto.

A arquitetura proposta é baseada em quatro entidades: usuário, dispositivo móvel, nuvem e provedor de dados. O usuário é o consumidor dos serviços, oferecidos pelo dispositivo móvel, que pode utilizar a nuvem como apoio para o *offloading*. O provedor de dados é responsável por fornecer dados quando se faz necessário, como informações metereológicas. Esta estrutura pode ser vista na Figura 9.

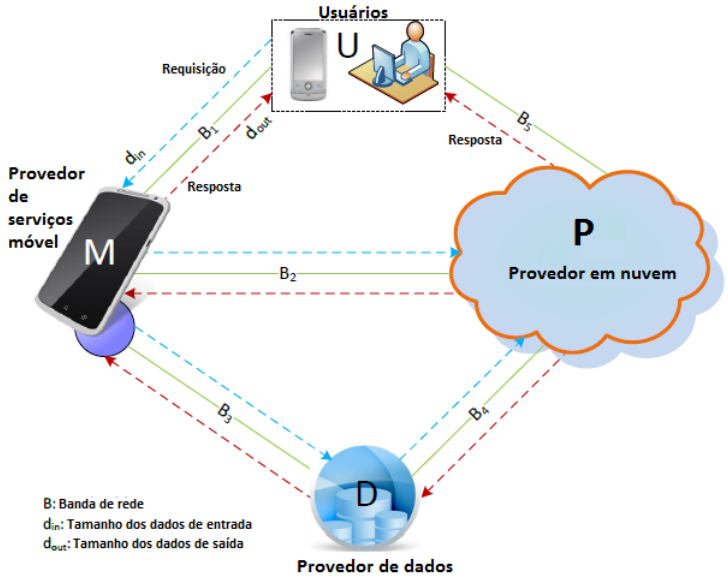

Figura 9 – Arquitetura abstrata de serviços móveis, adaptada de (ELGAZZAR; MARTIN; HASSANEIN, 2014)

A decisão de *offloading* amplia o cenário descrito na Figura 9 para três níveis: nuvem pública, *cloudlets* e nuvem móvel. A nuvem pública oferece serviços de computação dedicados, porém com grandes atrasos de comunicação. As *cloudlets* são estações computacionais mais simples porém mais próximas, com menos custos de comunicação. Por fim, a nuvem móvel é composta por dispositivos restritos e próximos, que podem oferecer serviços de computação. A decisão leva em conta vários fatores, como a situação da rede e informações de contexto (ELGAZZAR; MARTIN; HASSANEIN, 2014).

Através de protótipo, os autores desenvolveram serviços para avaliar a proposta com vários cenários e planos de execução, e puderam concluir que a abordagem foi capaz de contornar os problemas de latência de rede e economizar energia no dispositivo móvel (ELGAZZAR; MARTIN; HASSANEIN, 2014). O que este projeto não contempla em seu contexto é considerar como a mobilidade dos dispositivos móveis interfere na QoS das aplicações, assim como na energia dispendida.

Em Hong *et al.* (2013), os autores utilizaram como argumento o aumento de dispositivos de qualquer natureza integrados à Internet (IoT), e a inevitável sobrecarga de *Data centers* e sistemas de nuvem que integram e suportam suas aplicações. Assim sendo, Hong *et al.* (2013) propuseram um modelo de programação baseado nos paradigmas da *Fog computing*.

Para isso, Hong *et al.* (2013) consideraram utilizar diferentes elementos na "borda"da

Internet, sendo estes desde pequenos servidores dedicados até *smartphones* e computadores pessoais, explorando a sua baixa latência pela possível proximidade física.

Foi elaborada uma API com primitivas para o suporte à computação colaborativa, o que permitiu o desenvolvimento de duas aplicações para explorar esse paradigma. A primeira aplicação é para o monitoramento de veículos através do uso de sensores (câmeras), e a segunda aplicação é utilizada no monitoramento de tráfego em vias públicas, também utilizando sensores (HONG *et al.*, 2013).

A avaliação foi possível através de simulação no OMNeT++ com cenários em larga escala, e estudaram o comportamento tanto da latência quanto do tráfego de rede gerado para ambas as aplicações. Apesar dos esforços, Hong *et al.* (2013) não avaliaram nenhum aspecto em relação ao gasto de energia dos elementos envolvidos, além de não verificar o impacto nos tempos de resposta das aplicações.

Por fim, o último trabalho relacionado discutido neste trabalho é o Borro (2013). Este trabalho partiu de de um modelo de programação linear para o problema de escalonamento de tarefas do tipo *bag-of-tasks* (independentes) em uma grade móvel *on-site*, minimizando o consumo de energia global dos dispositivos e buscando garantir o atendimento das tarefas em seus *deadlines*.

A arquitetura considerada pelo trabalho é composta por uma grade local, onde usuários submetem tarefas a um nó *Proxy* que realiza o escalonamento nos dispositivos localmente.

O problema de escalonamento foi representado por um modelo de programação matemática, que está descrito no Capítulo 4, sendo expressado nas Equações **??**, 4.2, 4.3 e 4.4.

O modelo de programação matemática proposto, por ser NP-Difícil, possui um alto custo computacional para ser resolvido. Assim foram desenvolvidas duas heurísticas com menor complexidade, resolvendo o mesmo problema através de aproximações (BORRO, 2013).

O primeiro algoritmo heurístico chama-se *Maximum regret*. A ideia desse algoritmo é verificar para cada tarefa o quanto ela seria prejudicada caso não fosse atribuída para o recurso mais energeticamente "desejável", minimizando o consumo de energia global. O segundo algoritmo, *Greedy*, também visa minimizar o consumo de energia global, alocando as tarefas para os recursos mais energeticamente eficientes. Seu funcionamento é realizado em duas etapas, onde a primeira é a seleção dos melhores recursos e a segunda é a atribuição das tarefas respeitando as restrições de *deadline* e energia. A segunda etapa é realizada por outro algoritmo heurístico, uma vez que esse subproblema também é NP-difícil (BORRO, 2013).

Através de simulações que compararam o modelo de programação matemática resolvido por um *solver* com as heurísticas, o autor pôde concluir que os dois algoritmos propostos são computacionalmente eficientes, pois seu tempo de solução é pequeno em relação ao tempo para resolver o modelo de programação matemática. Além disso, as soluções geradas são muito próximas das encontradas através do referencial ótimo através do modelo de programação

matemática (BORRO, 2013). Porém, este trabalho não considera nenhum fator de mobilidade dos dispositivos, além de seu escopo estar fixado em uma rede simples e local. Além disso nenhum aspecto de rede foi considerado, permitindo verificar como a comunicação interfere nas aplicações e no custo de energia da abordagem.

3.3 Resumo dos Trabalhos Relacionados

Como é possível observar na Tabela 1 e na Tabela 2 de características, este projeto de Mestrado propõe uma arquitetura global que permita que dispositivos móveis sejam utilizados como recursos computacionais colaborativamente. Além disso, este projeto busca minimizar os gastos de energia e considerar a confiabilidade dos nós em relação à sua alta mobilidade, e investiga como associar isso com os gastos de energia, considerando também os custos de comunicação.

Na Tabela 2 estão descritas as características relacionadas aos trabalhos relacionados.

Tabela 1 – Resumo dos Trabalhos Relacionados

Relacionados / Características	**A**	**B**	**C**	**D**	**E**	**F**
(ISAIADIS; GETOV, 2005)	√	√	√			
(HUERTA-CANEPA; LEE, 2010)	√	√				
(HASSAN; ZHAO; YANG, 2010)	√	√	√	√		
(MORSY; EL-REWINI, 2013)	√	√		√	√	
(ELGAZZAR; MARTIN; HASSANEIN, 2014)	√	√	√	√		
(HONG *et al.*, 2013)	√	√	√			
(BORRO, 2013)	√			√		
Projeto de Mestrado	√	√	√	√	√	√

Tabela 2 – Características dos Trabalhos Relacionados

Característica	**Descrição**
A	Considera dispositivos móveis como recursos computacionais
B	Considera custos de comunicação
C	Possui um contexto global
D	Busca minimizar o custo de energia
E	Considera os aspectos de mobilidade dos recursos
F	Associa a confiabilidade dos recursos com o consumo de energia

3.4 Considerações Finais

Neste capítulo foram apresentados os trabalhos relacionados a este projeto de Mestrado para situá-lo. Nenhum deles contempla todas as características que este trabalho vem oferecer: uma arquitetura global e colaborativa para o provimento de serviços computacionais com foco

na economia de energia, na mobilidade, e com ciência no gasto em comunicação. No próximo capítulo são apresentados todos os detalhes de desenvolvimento do projeto.

CAPÍTULO

4

DESENVOLVIMENTO DO PROJETO

4.1 Considerações Iniciais

Neste capítulo são abordados os detalhes de desenvolvimento acerca do projeto. Primeiramente, é discutido o escopo inicial do projeto, que foi expandido a partir do trabalho de Borro (2013). Na sequência, todas as considerações do desenvolvimento do projeto são detalhadas, como a implementação dos níveis e das estimativas de custo de comunicação. Por fim, a metodologia de validação e verificação do modelo foi descrita.

4.2 Escopo Inicial do Trabalho

Este trabalho teve como ponto de partida o modelo de programação linear para o problema de escalonamento de tarefas do tipo *bag-of-tasks* (independentes) em uma grade móvel *on-site* proposto por Borro (2013), o qual busca minimizar o consumo de energia global dos dispositivos e garantir o atendimento das tarefas em seus *deadlines*.

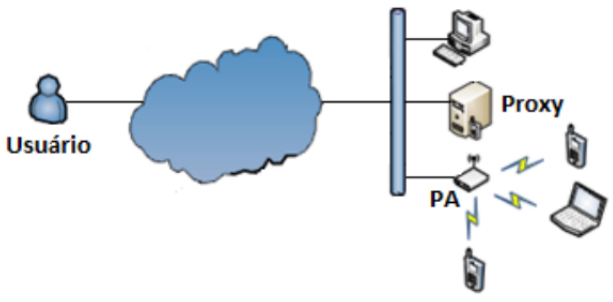

Figura 10 – Arquitetura considerada por Borro (2013)

A arquitetura considerada pelo trabalho de Borro (2013) é composta por uma grade local,

em que usuários submetem tarefas a um nó *proxy* que realiza o escalonamento nos dispositivos localmente. Tal organização pode ser vista na Figura 10.

Este trabalho elaborou um modelo de otimização, cuja função objetivo está representada pela Equação 4.1, e busca minimizar a energia global consumida pelos dispositivos para processar todas as tarefas. A restrição representada pela Equação 4.2 serve para garantir que a capacidade de bateria dos dispositivos e o *deadline* não sejam violados. A restrição que a Equação 4.3 impõe não permite que uma mesma tarefa seja atribuída a mais de um recurso, e a restrição representada pela Equação 4.4 define o domínio da variável binária "x", que é a resposta do modelo. Se x_{ij} se encontra igual a 0, significa que a tarefa j foi não alocada ao recurso i, e o contrário caso $x_{ij} = 1$ (BORRO, 2013). Ainda é possível ver as notações utilizadas na Tabela 3.

$$Minimizar \quad E = \sum_{i=1}^{m} \sum_{j=1}^{n} e_{ij} x_{ij} \tag{4.1}$$

$$s.a. \quad \sum_{j=1}^{n} t_{ij} x_{ij} \leq min\{d, c_i/p_i\}, \forall i \tag{4.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j \tag{4.3}$$

$$x_{ij} \in \{0,1\} \forall i, j \tag{4.4}$$

Tabela 3 – Notação do Modelo de Programação Linear Proposto por Borro (2013)

Parâmetro	Descrição
q_i	Potência de processamento do recurso i (MIPS, Milhões de Instruções por Segundo)
p_i	Estimativa do consumo de energia do recurso i (Watt)
c_i	Capacidade de bateria do recurso i (Joule)
w_j	Carga de trabalho da j-ésima tarefa da aplicação (MI, Milhões de Instruções)
d	*Deadline* da aplicação (segundos)
t_{ij}	Tempo de execução da tarefa j no recurso i (segundos)
e_{ij}	Consumo de energia relativo à execução da tarefa j no recursos i (Joule)

O modelo de programação matemática proposto, por ser NP-difícil, possui um alto custo computacional para ser resolvido. Assim, foram desenvolvidas duas heurísticas com menor complexidade, resolvendo o mesmo problema através de aproximações (BORRO, 2013).

O primeiro algoritmo heurístico chama-se *Maximum Regret*. A ideia desse algoritmo é verificar para cada tarefa o quanto ela seria prejudicada caso não fosse atribuída para o recurso mais energeticamente "desejável", minimizando o consumo de energia global. O segundo algoritmo, *Greedy*, também visa minimizar o consumo de energia global, alocando as tarefas para os recursos mais energeticamente eficientes. Seu funcionamento é realizado em duas etapas, onde

a primeira é a seleção dos melhores recursos e a segunda é a atribuição das tarefas respeitando as restrições de *deadline* e energia. A segunda etapa é realizada por outro algoritmo heurístico, uma vez que esse sub-problema também é NP-difícil (BORRO, 2013).

O presente projeto de Mestrado optou por utilizar a heurística *Maximum Regret*, pelo fato da mesma possuir uma melhor adaptação em relação às novas restrições em seu funcionamento, como será explicado posteriormente.

Sendo assim, o Algoritmo 1 representa o funcionamento da heurística *Maximum Regret*, a qual será detalhada a seguir.

Algoritmo 1: *Maximum Regret* original

Entrada: Um conjunto de recursos $R = \{R_1 ... R_m\}$ sendo $R_i = (q_i, p_i, c_i)$; Uma aplicação $A = (d, W = \{w_1 ... w_n\})$

1 $d_i \leftarrow d$
2 $t_{ij} \leftarrow w_j / q_i$
3 $e_{ij} \leftarrow t_{ij} p_i$
4 $f_{ij} \leftarrow -e_{ij}$
5 **repita**
6 **para cada** *tarefa j não atribuída* **faça**
7 $F_j = \{i : t_{ij} \leq min\{d_i, c_i/p_i\}\}$
8 **se** $F_j = \oslash$ **então**
9 $r_j \leftarrow -\infty$
10 **senão**
11 **se** $|F_j| = 1$ **então**
12 $r_j \leftarrow \infty$
13 **senão**
14 $r_j = max_1\{f_{ij} : i \in F_j\} - max_2\{f_{ij} : i \in F_j\}$
15 $j^* \leftarrow arg\_max\{r_j\}$
16 $i^* \leftarrow arg\_max\{f_{ij^*} : i \in F_{j^*}\}$
17 **se** $r_{j^*} \neq -\infty$ **então**
18 atribua a tarefa j^* ao recurso i^*
19 $d_{i^*} \leftarrow d_{i^*} - t_{i^* j^*}$
20 $c_{i^*} \leftarrow c_{i^*} - e_{i^* j^*}$
21 **até** *que todas as tarefas tenham sido atribuídas OU mais nenhuma tarefa possa ser alocada nos recursos disponíveis*;

Os valores contidos em f_{ij} expressam o "desejo"que uma tarefa j tem de ser alocada a um recurso i. Sendo assim, na linha 7 do Algoritmo 1 é criado um sub-conjunto F_j de recursos que não violam o *deadline* e que têm capacidade para processar determinada tarefa i.

Se o subconjunto de recursos para uma tarefa j é vazio (linha 8), ela recebe um valor muito pequeno de *"regret"* (linha 9). Agora, se há apenas um elemento no sub-conjunto (linha 11), o valor de *"regret"* é muito grande. Caso contrário aos anteriores, o valor de *"regret"* é a diferença entre os dois maiores valores de f_{ij}, sendo escolhida a tarefa j com maior valor r_j.

4.3 Expansão do Modelo

A organização lógica da arquitetura proposta pode ser vista na Figura 11. Como é possível observar, o nível 1, que está na base da estrutura, é formado por aglomerados, onde cada aglomerado é composto por dispositivos móveis interconectados por redes locais, como redes públicas e domésticas. É importante observar que não há limitações na combinação dos elementos que podem compor cada aglomerado, e cada aglomerado necessariamente precisa ter um nó coordenador como líder, responsável por resolver a heurística *Maximum Regret* com algumas adaptações que serão posteriormente apresentadas.

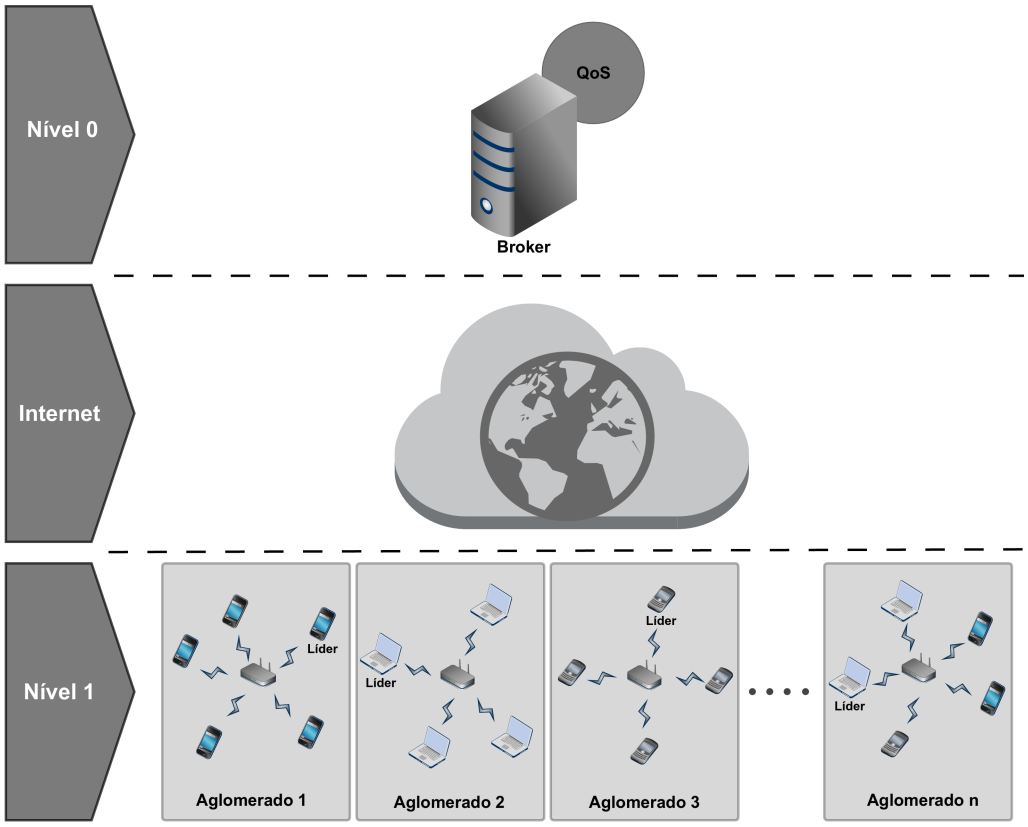

Figura 11 – Arquitetura Prosposta neste trabalho

Os aglomerados no nível 1, por sua vez, se comunicam com o nível 0 através da *Internet*. O nível 0 é composto por um nó especial chamado *Broker*, que tem informações globais dos aglomerados do nível 1, e é o responsável por possibilitar que diferentes aglomerados exportem aplicações entre si.

Vale destacar que os aglomerados no nível 1 não possuem comunicação direta entre si, e não necessariamente estão na mesma sub-rede. Isso significa que a camada de *Internet* é o meio que interconecta todos os aglomerados entre si e ao *Broker*.

Os elementos no nível 1 podem ser tanto provedores de serviços quanto clientes de serviços. Ou seja, quando um elemento pertencente a um aglomerado no nível 1 precisa utilizar um serviço, há a possibilidade de checar se ele é oferecido dentro de seu aglomerado, sendo necessário delegar ao nível 0 encontrar outro aglomerado caso contrário.

Para demonstrar melhor como ocorre essa dinâmica do momento da requisição de uma aplicação, composta por tarefas, e o seu serviço de atendimento, foram construídos dois cenários baseados no diagrama da Figura 11.

Cada aglomerado no nível 1 oferece, internamente, um conjunto de aplicações. Assim sendo, é preferível que uma determinada requisição seja atendida internamente, evitando utilizar um outro recurso que, provavelmente, está fisicamente longe.

Existem duas situações que podem acontecer quando uma requisição é gerada em um aglomerado:

1. O serviço para o atendimento desta aplicação é oferecido internamente;

2. O serviço é exportado ao nível 0 pois não é oferecido internamente ou não há recursos suficientes para atendê-lo (energia, processamento).

A primeira situação está representada na Figura 12, e as setas em vermelho mostram a ordem dos eventos quando um determinado nó precisa exportar processamento através da técnica de *offloading*.

Na Figura 12, a seta 1 no Passo 1 mostra quando um dispositivo encaminha a requisição ao líder de seu aglomerado, ficando esse responsável por verificar se essa aplicação está disponível internamente e se pode ser processada em seu *deadline*.

Neste caso, a heurística *Maximum Regret*, com adaptações a serem debatidas, escolhe os recursos na grade de forma a receberem sub-conjuntos de tarefas a serem processadas, sendo essa escolha representada pelas setas 2.a, 2.b e 2.c na Figura 12

Depois do processamento, cada nó devolve seu sub-conjunto de tarefas processadas ao líder (setas 3.a, 3.b e 3.c), o qual devolve o resultado da requisição ao nó solicitante, sendo essa última etapa representada pela seta 4 na Figura 12.

Mas, como explicado anteriormente, nem sempre uma determinada requisição pode ser atendida dentro do mesmo aglomerado que a requisitou, sendo necessário encaminhá-la ao nível 0 (*Broker*). Esse caso pode ser observado na Figura 13: em 1, o nó solicita uma requisição ao líder, que verificando a impossibilidade do oferecimento local desse serviço, o encaminha ao

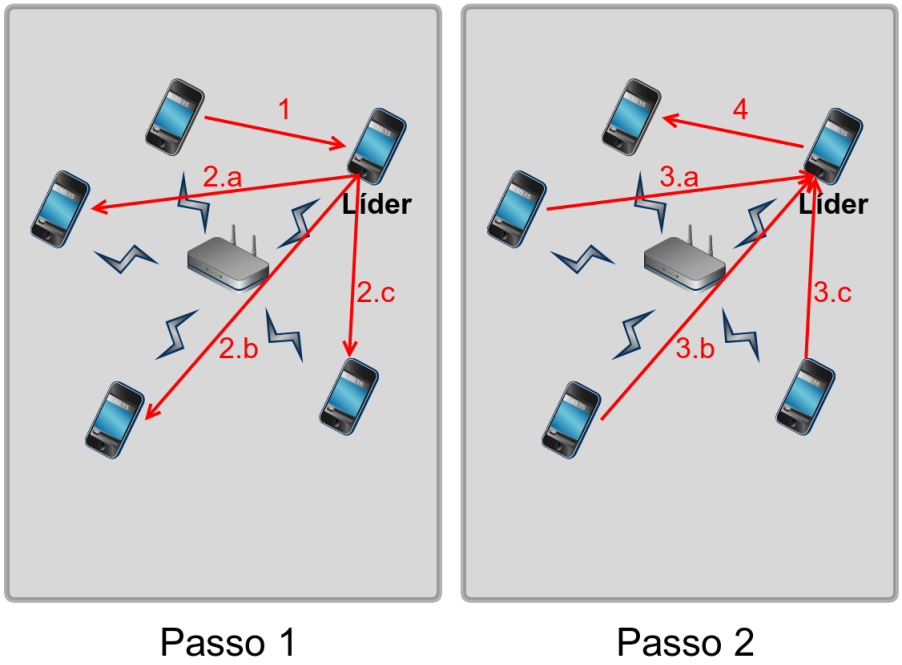

Figura 12 – Atendimento interno de uma aplicação

nível 0, representado pela seta 2. Consequentemente, o *Broker*, através de uma heurística global, a ser apresentada na Seção 4, seleciona outro aglomerado que consiga atender externamente essa requisição (seta 3).

Conforme a requisição é encaminhada para outro aglomerado, o líder desse outro conjunto de recursos, através da heurística *Maximum Regret* adaptada, atribui as tarefas internamente ao seu conjunto, representado na Figura 14 pelos arcos 4.a, 4.b, 4.c e 4.d.

Após o processamento da aplicação exportada, é possível ver na Figura 15 que os nós escolhidos na heurística no aglomerado selecionado anteriormente devolvem o resultado para o líder de seu aglomerado, sendo esse passo representado pelas setas 5.a, 5.b, 5.c e 5.d. O líder, por sua vez, devolve ao nível 0 o resultado da requisição (seta 6). O *Broker* repassa o resultado para o aglomerado que originou a requisição, e representado pelo seu líder (seta 7). Por fim, o líder repassa ao nó requisitante o resultado de sua requisição (seta 8).

Os detalhes de projeto serão apresentados no decorrer deste capítulo, começando pelo nível 1, depois nível 0, a representação completa no modelo de rede de filas desenvolvido para sua avaliação e, por fim, os custos estimados de comunicação.

4.4 Implementação do Nível 1

O nível 1, como descrito anteriormente, é composto por aglomerados de dispositivos móveis, que estão interligados através da *Internet*. Cada aglomerado é, na verdade, a representação da arquitetura local definida por Borro (2013), e já mostrada na Figura 10.

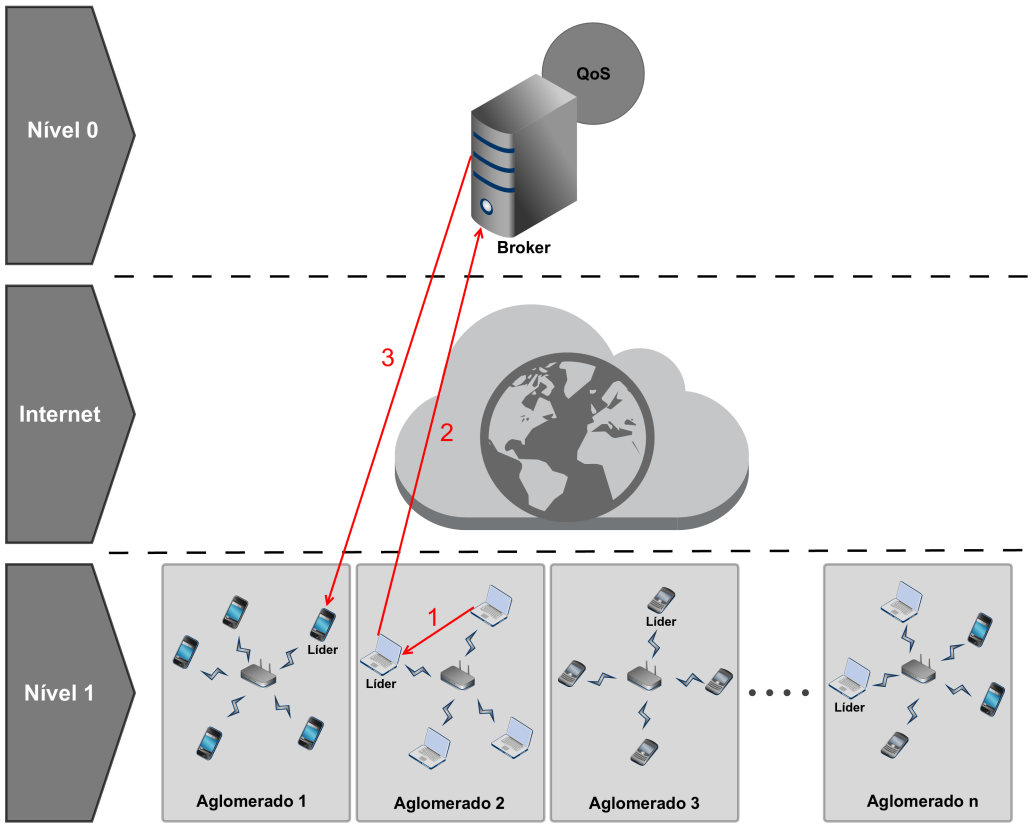

Figura 13 – Requisição sendo exportada do nível 1 para o nível 0

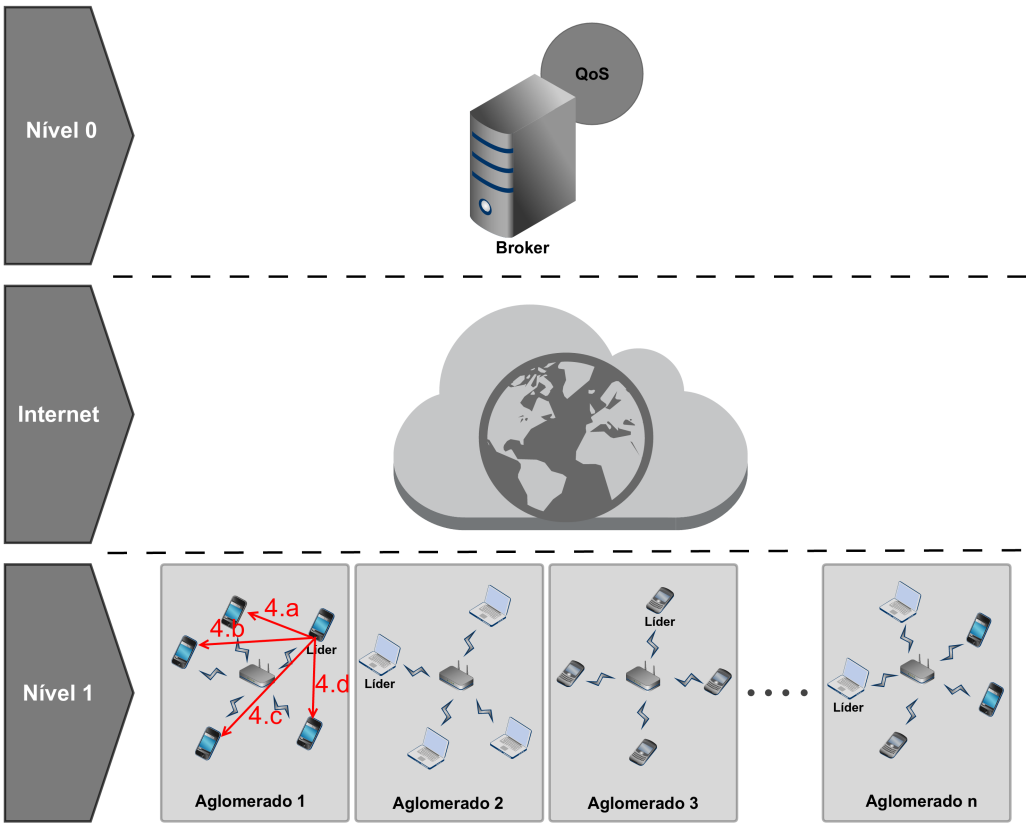

Figura 14 – Requisição processada externamente por outro aglomerado no nível 1

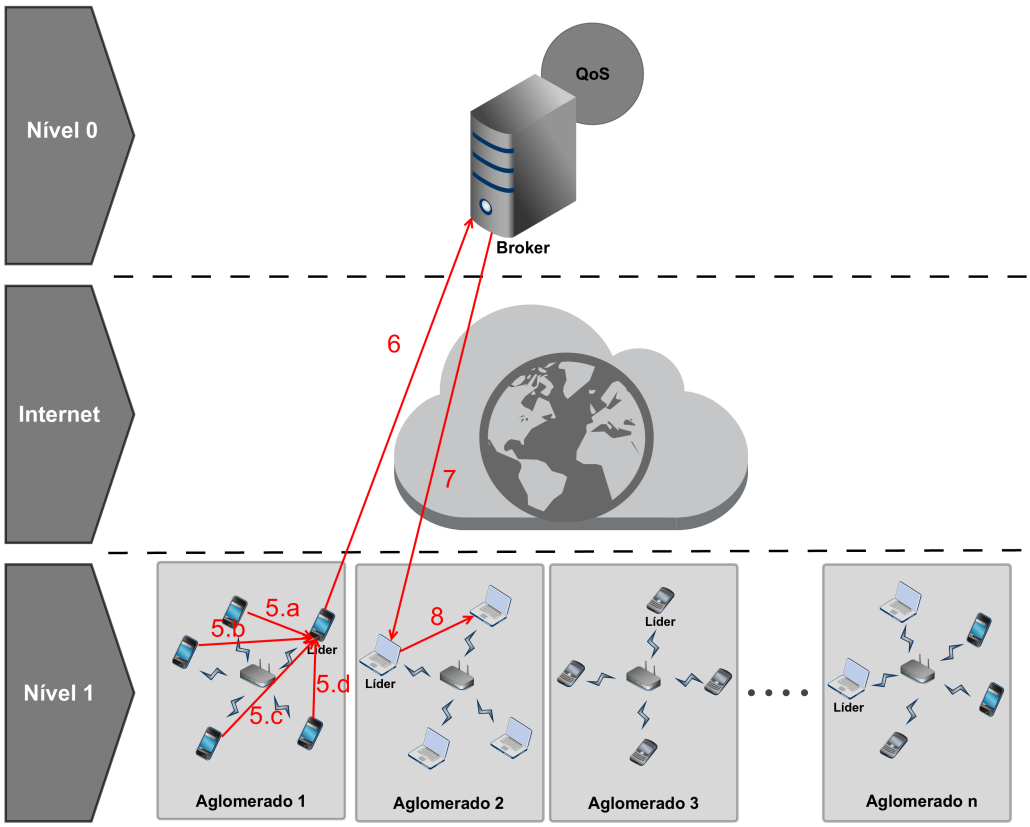

Figura 15 – Requisição sendo Reencaminhada ao Solicitante

Porém, é possível perceber que o ambiente genérico que Borro (2013) considerou conta com um nó especializado chamado *Proxy*, responsável por executar a heurística *Maximum Regret*. Isso torna-se um fator limitante ao desejar expandir esse modelo, uma vez que é necessário implantar um elemento especializado em cada aglomerado, tornando impraticável.

Por isso, a responsabilidade de selecionar os recursos através da heurística foi delegada a um dispositivo móvel sem nenhuma característica especial, tornando esse nível aplicável a qualquer rede sem-fio, como uma residência ou um campus de universidade com vários pontos de acesso.

Portanto, é preciso definir um algoritmo de eleição adaptado à realidade dessa proposta.

4.4.1 Eleição de Líder

Para escolher um líder em cada aglomerado, foi necessário estabelecer um algoritmo de eleição baseado em sistemas distribuídos convencionais. Neste caso, o algoritmo base utilizado foi o *Bully*, ou algoritmo do valentão (TANENBAUM; STEEN, 2007).

O algoritmo original é baseado em três etapas (TANENBAUM; STEEN, 2007):

1. A partir do momento que um processo P detecta a ausência de um coordenador, ele envia uma mensagem de eleição aos processos de maior número de identificação;

2. Se não há nenhuma resposta, o processo P é o novo coordenador, enviando uma mensagem de eleição a todos processos;

3. Se houver uma resposta de um processo com maior número de identificação, o processo P encerra o algoritmo de eleição, vencendo o processo que se encaixar na condição da etapa 2.

Sendo assim, para adaptá-lo ao contexto da computação móvel, foi estabelecido que o valor de *rank* no algoritmo seja representado pela quantidade de energia residual de cada dispositivo. Essa decisão tem como objetivo eleger o dispositivo com maior capacidade de energia em cada aglomerado, uma vez que quando há o esgotamento total de energia, o dispositivo deixa de ser o líder, sendo necessário escolher novamente um novo elemento.

O algoritmo adaptado está representado no Algoritmo 2

Algoritmo 2: Algoritmo de eleição nos aglomerados

Entrada: *Ranks* dos dispositivos
Saída: Dispositivo eleito

```
1 inicio
2     Ausência de coordenador detectada;
3     Solicita o valor de rank dos outros dispositivos;
4     repita
5         se Rank recebido é maior que o meu então
6             Encerra a eleição;
7     até Enquanto houver resposta;
8     Sou o líder;
9     Avisando a todos da eleição;
```

O Algoritmo 2 é executado em todos os dispositivos distribuídos na rede local, e elege o dispositivo que tem o maior valor de *rank*, ou seja, o que tem mais energia. O processo de eleição é repetido apenas quando o líder tem toda a sua energia drenada, se tornando indisponível.

4.4.2 Inserção de Falhas e Restrição no Fator de Confiabilidade

A heurística *Maximum Regret* definida por Borro (2013) não possui em sua concepção original aspectos relativos à possibilidade de algum nó falhar durante o escalonamento de tarefas. Por isso, a heurística foi adaptada de forma a contemplar a possibilidade dos nós falharem ao serem utilizados para processar tarefas.

O Algoritmo 3 contempla as alterações que foram necessárias para considerar que os nós podem falhar durante o escalonamento. As linhas 21 e 22 remetem à checagem responsável por verificar se a tarefa atual teve sucesso ao ser escalonada ao recurso em questão. Caso haja uma

falha, a tarefa volta ao escalonamento, porém a energia que foi gasta nesse processo não pode mais ser reposta.

Algoritmo 3: *Maximum Regret* adaptado

Entrada: Um conjunto de recursos $R = \{R_1...R_m\}$ sendo $R_i = (q_i, p_i, c_i, m_i)$; Uma aplicação $A = (d, W = \{w_1...w_n\})$; ε

1 $d_i \leftarrow d$
2 $t_{ij} \leftarrow w_j/q_i$
3 $e_{ij} \leftarrow t_{ij}p_i$
4 $f_{ij} \leftarrow -e_{ij}$
5 **repita**
6 **para cada** *tarefa j não atribuída* **faça**
7 $F_j = \{i : t_{ij} \leq min\{d_i, c_i/p_i\}, m_i \geq \varepsilon\}$
8 **se** $F_j = \oslash$ **então**
9 $r_j \leftarrow -\infty$
10 **senão**
11 **se** $|F_j| = 1$ **então**
12 $r_j \leftarrow \infty$
13 **senão**
14 $r_j = max_1\{f_{ij} : i \in F_j\} - max_2\{f_{ij} : i \in F_j\}$
15 $j^* \leftarrow arg\_max\{r_j\}$
16 $i^* \leftarrow arg\_max\{f_{ij^*} : i \in F_{j^*}\}$
17 **se** $r_{j^*} \neq -\infty$ **então**
18 atribua a tarefa j^* ao recurso i^*
19 $d_{i^*} \leftarrow d_{i^*} - t_{i^*j^*}$
20 $c_{i^*} \leftarrow c_{i^*} - e_{i^*j^*}$
21 **se** *recurso* j^* *falhou ao ser atribuída a tarefa* j^* **então**
22 tarefa j^* volta para a lista de não escalonadas
23 $m_i = m_i - 0.01$
24 **senão**
25 $m_i = m_i + 0.01$
26 **até** *que todas as tarefas tenham sido atribuídas OU mais nenhuma tarefa possa ser alocada nos recursos disponíveis*;

Para as probabilidades de que uma falha aconteça, foi utilizado como referência o trabalho de Litke *et al.* (2007), que sintetiza formalmente as chances de um nó móvel falhar durante a requisição de uma tarefa. As probabilidades são apresentadas no Capítulo 5.

Os recursos também passaram a ter um novo parâmetro: fator de confiabilidade m_i. Esse parâmetro é uma estimativa de como um recurso se comporta em relação à chance de falhar ao ser escolhido para processar uma tarefa. O fator de confiabilidade foi adaptado na linha 7 do Algoritmo 3. Isso faz com que a lista de recursos possíveis F_j para uma tarefa j receba somente os dispositivos com o fator de confiabilidade m_i acima de um valor ε, sendo que $0 \leq \varepsilon \leq m$. Quando $\varepsilon = 0$, não há restrição nos recursos disponíveis para a alocação, sendo que a heurística

não considera o m_i, e quando $\varepsilon = m$, os recursos oferecidos são exatamente os suficientes para que a aplicação seja atendida no seu *deadline*, ou seja, para que o problema seja factível.

Todo dispositivo recebe um valor de fator de confiabilidade gerado com uma distribuição normal com média 0.50 e desvio padrão 0.10. Esse valor inicial é uma estimativa "neutra", visto que o valor de probabilidade de falhas não é conhecido. Caso a heurística adaptada verifique uma falha ou sucesso na alocação, o m_i é alterado, sendo decrescido em 0.01 quando há uma falha (linha 23 do Algoritmo 3) ou acrescido em 0.01 quando há uma atribuição bem sucedida (linha 25). Essas modificações em seus valores servem de ajustes para que o corte dos dispositivos com m_i menor que o limiar ε seja efetivo e realmente evite dispositivos com características de ser pouco confiáveis.

4.4.3 Implementação do Nível 0

O nível 0, como explicado anteriormente, recebe as requisições exportadas do nível 1 e, através de um critério já estabelecido, escolhe outro aglomerado de forma a atender essa requisição. Para escolher esse novo aglomerado, é importante perceber que existem dois interesses conflitantes: garantir QoS para quem gerou a solicitação e minimizar o impacto em quem vai atender esse serviço.

Para balancear entre esses dois objetivos distintos nesse tipo de ambiente foram elaboradas duas métricas distintas que podem ser utilizadas, por exemplo, em uma SOA como o trabalho de Estrella (2010), onde o algoritmo de seleção de recursos pode ser, a priori, redefinido e adaptado. As métricas propostas são apresentadas a seguir.

4.4.4 Métricas de Seleção para Ambientes Móveis

A primeira métrica tenta garantir QoS dos nós que solicitaram certo serviço externamente, e será denominada a partir desse ponto de **métrica A**. Para esta métrica, são consideradas as seguintes características de cada aglomerado: capacidade de processamento em MIPS e capacidade de comunicação interna e externa.

A **métrica A** é expressa por: $A_x = (2 * NP_x + 2 * NE_x + NI_x)/5$, onde:

- A_x é a **métrica A** para o aglomerado x;
- NP_x é o valor normalizado da média de processamento do aglomerado x em relação aos n aglomerados possíveis;
- NE_x é o valor normalizado da capacidade de comunicação externa em Mbps do aglomerado x em relação aos n aglomerados possíveis;
- NI_x é o valor normalizado da capacidade de comunicação interna em Mbps do aglomerado x em relação aos n aglomerados possíveis;

A **métrica A** é representada pela média dos fatores NP_x, NE_x e NI_x, dando peso maior ao NP_x e NE_x, uma vez que são o que mais influenciam no QoS. Além do mais, como todas as partes integrantes de A_x estão entre os valores 0 e 1, a **métrica A** também está neste intervalo.

Já a **métrica** B busca sintetizar os aglomerados que ao mesmo tempo são mais energeticamente eficientes e confiáveis. Com isso, a **métrica B** é expressa por: $B_x = ((1 - NEE_x) + NC_x)/2$, onde:

- B_x é a **métrica B** para o aglomerado x;
- NEE_x é o valor normalizado do gasto médio de energia do aglomerado x em relação aos n aglomerados possíveis;
- NC_x é a média dos fatores de confiabilidade dos nós do aglomerado x.

Tem-se que a **métrica B** também é um valor entre 0 e 1, visto que $0 \leq NEE_x, NC_x \leq 1$.

Apresentadas as **métricas A** e **B**, é definida então a heurística de seleção global de aglomerados que pode considerar apenas uma delas ou então ambas dependendo do ajuste de uma variável pré-definida K a ser discutida.

4.4.5 Heurística Global de Seleção de Aglomerados

A heurística global para seleção dos aglomerados pode ser observada no Algoritmo 4.

No conjunto de entrada tem-se K, que é uma variável de ajuste entre as **métricas A** e **B**, e app_id, que é o tipo de serviço que está sendo requisitado. O valor de K varia entre $1 \leq K \leq n$, onde n é o número total de aglomerados que podem atender a requisição em questão.

As linhas 1 a 3 do Algoritmo 4 pré-selecionam os K maiores elementos em relação à **métrica A**, já definida anteriormente. Já as linhas 4 a 7 verificam dentre os K elementos qual tem a maior **métrica B**, sendo este o escolhido.

Quando $K = 1$, o aglomerado escolhido é o que tem a maior **métrica A** entre todos, ou seja, o algoritmo prioriza estritamente o aglomerado com maior capacidade de processamento e melhores condições de rede.

Já quando K é acrescido até o seu valor máximo n, a **métrica B** passa gradativamente a ser considerada, uma vez que entre os K melhores elementos pré-selecionados a partir da **métrica A** são novamente verificados. Entre esses K elementos, é escolhido aquele que tem a maior **métrica B**.

O valor de K é importante no sentido de ponderar entre as duas métricas e escolher o melhor aglomerado em todos os sentidos, seja na economia de energia ou seja em relação à QoS das aplicações.

Algoritmo 4: Algoritmo de Seleção de Aglomerados no Nível 0

Entrada: K, app_id
Saída: Aglomerado escolhido X

para *cada aglomerado x que pode atender a requisição app_id* **faça**
    $A_x = (2 * NP_x + 2 * NE_x + NI_x)/5$;
    lista_pre_selecionados ← insere_ordenado_decrescente(A_x, x);
para *os K primeiros aglomerados de lista_pre_selecionados* **faça**
    $B_x = ((1 - NEE_x) + NC_x)/2$;
    se $(B_x > B_k)\ \forall\ k \in K$ *primeiros aglomerados* **então**
        X ← x;
retorna (X);

Depois de definido o algoritmo de seleção global, será apresentado o modelo de rede de filas desenvolvido, e a forma em que foram integrados os níveis 0 e 1 a partir dele, além das estimativas de gasto de comunicação e de energia.

4.5 Modelo de Rede de Filas

Para que a arquitetura proposta fosse validada, foram utilizadas técnicas de avaliação de desempenho. Um modelo foi definido de modo a representar o comportamento dessa arquitetura, e posteriormente esse modelo foi simulado, provendo métricas de desempenho que foram analisadas.

O modelo de rede de filas, representado na Figura 16, mostra os nós dos aglomerados e o *Broker* que são centros de atendimento. No modelo, o Nó(i,j), ou seja, o nó *j* do aglomerado *i* é diretamente interligado com todos os nós que estão no mesmo aglomerado, o que representa a rede local.

Todos os nós têm comunicação com o *Broker* no modelo, apesar de apenas os líderes dos aglomerados se comunicarem diretamente com tal elemento. O *Broker*, por sua vez, possui sua via de comunicação com todos os nós de todos os aglomerados, apesar de ter contato de fato apenas com os líderes que representam cada aglomerado. Todos os nós dos aglomerados possuem a possibilidade de gerar novas requisições, representadas pelas setas de entrada sem ligação inicial. As setas para cima, sem nenhum elemento terminal, representam o final de atendimento de alguma requisição que transitou no modelo.

As requisições podem partir de qualquer nó, sendo uma possibilidade de chegada no modelo em cada centro de atendimento.

Quando um aglomerado está processando uma requisição - desde a heurística Maximum Regret até a alocação das tarefas - não é possível iniciar o atendimento de outras de forma simultânea. Ou seja, as requisições são atendidas internamente aos aglomerados de forma única

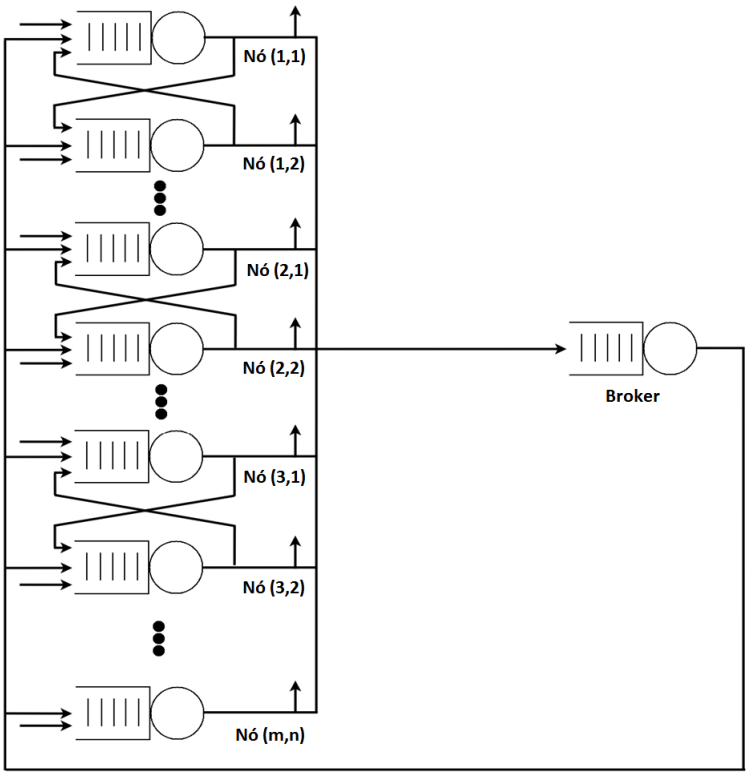

Figura 16 – Modelo de Rede de Filas Projetado

e exclusiva, utilizando todo o potencial computacional da grade local (capacidade máxima em MIPS e enlace de comunicação).

4.5.1 Modelo de Simulação

Com o modelo de rede de filas definido, o próximo passo foi a criação de um programa de simulação de modo a simular o comportamento do modelo. Para isso, foi utilizado o SimPack/Sim++ Simulation Toolkit (FISHWICK, 1992), uma vez que essa plataforma é consolidada e permite um maior controle nos elementos simulados, permitindo acoplar com certa facilidade os tempos de comunicação e gastos de tempo para fazer o escalonamento local de tarefas.

Na Figura 17 é possível ter uma visão global do funcionamento da simulação, através da definição de 4 fluxos principais.

Cada requisição de serviço para o atendimento de uma aplicação, dentro do ambiente de simulação, é representada por um *token* simbólico. Todo *token* deve passar, necessariamente, entre os estados de chegada, atendimento e saída.

O que diferencia o estado geral de cada *token* são os sinais lógicos de fluxo e de identificação do responsável atual pela requisição. Os estados de fluxo indicam se uma requisição está sendo atendida internamente ou externamente, e o de responsável indica quem está processando atualmente a requisição.

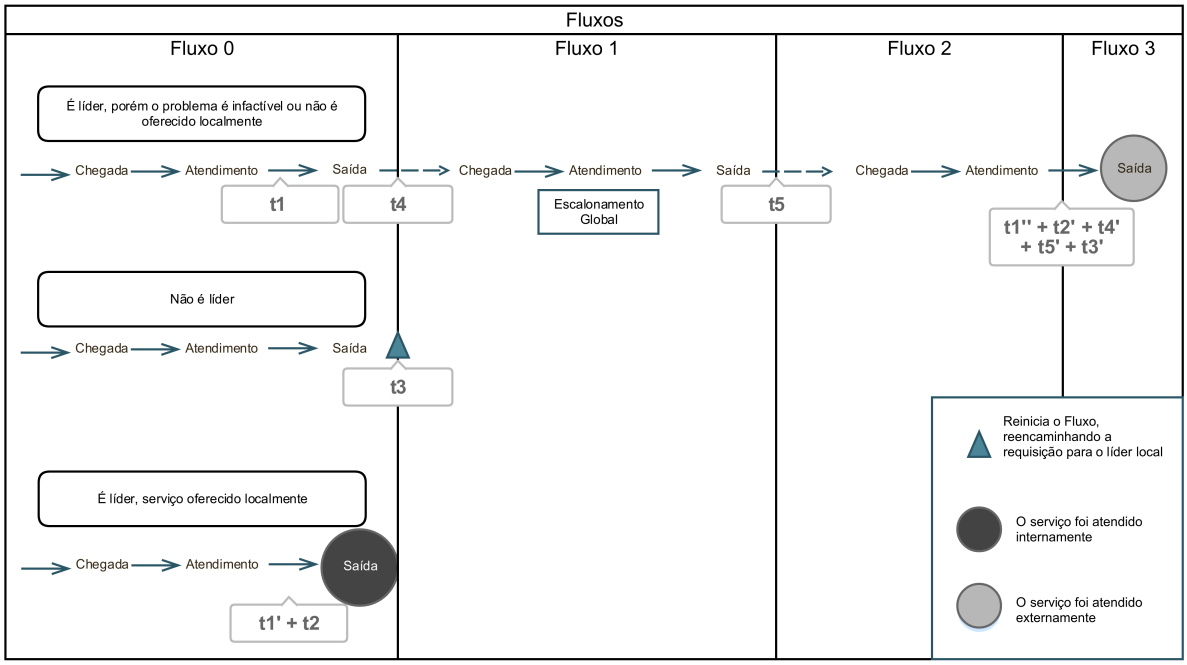

Figura 17 – Ilustração dos Fluxos Simulados

Tabela 4 – Tempos Dispendidos no Modelo de Rede de Filas

Tempo	Descrição
t1	Tempo para solucionar a heurística MRA
t2	Tempo para processar as tarefas alocadas
t3	Tempo para comunicar na rede local
t4	Tempo para comunicar do nível 1 para o nível 0
t5	Tempo para comunicar do nível 0 para o nível 1

Um vetor de requisições é previamente gerado de acordo com um arquivo de *traces*, e cada requisição é inicializada com seu valor de fluxo em 0. Assim sendo, todas as requisições necessariamente passam pelo fluxo 0. As mudanças entre estados de chegada, atendimento e saída são realizadas através do agendamento de eventos, e nesses instantes é possível definir o tempo de atraso em relação ao relógio atual de simulação, sendo possíveis tipos sumarizados na Tabela 4.

Quando uma requisição está no fluxo 0, primeiramente é preciso verificar se o nó que a detém é líder de seu aglomerado. Se não for líder, o *token* é novamente reescalonado para o fluxo 0, e a requisição é delegada ao líder, acrescentando um tempo t3.

Uma vez que o nó que recebe a solicitação é líder local e o fluxo continua em 0, é preciso verificar a factibilidade do problema e a disponibilidade do aglomerado em questão para atender localmente. Caso seja possível atender localmente, o líder resolve o escalonamento através da heurística *Maximum Regret* Adaptada e contabiliza um tempo t1’ que se remete ao tempo total para resolver a heurística e um tempo t2, que foi o tempo que os nós locais demoraram para processar as tarefas.

Porém, nem sempre o serviço pode ser atendido localmente, sendo necessário reencaminhar a solicitação ao nível 0, contabilizando o tempo t1 que foi dispendido e o tempo t4, gasto em comunicação para exportar a requisição. Além do mais, o sinal lógico de fluxo é ajustado para 1.

Já no fluxo 1, o escalonamento global é realizado sem acrescentar nenhum tempo de processamento, uma vez que considera-se que o *Broker* não é um gargalo na arquitetura. Após decidir qual aglomerado externo vai receber essa requisição, o tempo t5 para comunicar entre os níveis é acrescentado, e o fluxo passa a ser igual a 2.

No fluxo 2, o líder do aglomerado selecionado resolve a heurística e aloca as tarefas localmente, acrescentando os tempos t1", t2', t3', t4' e t5', que quando somados representam o tempo para a requisição ser processada e entregue ao aglomerado requisitante. O fluxo passa a ser 3, o que simboliza que a requisição foi completamente atendida externamente.

O tempo t2 foi obtido dentro da heurística *Maximum Regret* adaptada, sendo igual ao maior tempo de processamento dentre os recursos que foram escolhidos no escalonamento, uma vez que o conjunto de tarefas é alocado de forma paralela. Já as classes de tempo t3, t4 e t5 se remetem estritamente à comunicação, e sua forma de cálculo, assim o gasto de energia envolvido, são apresentados na próxima seção.

4.5.2 Estruturas de Dados da Simulação

Para viabilizar o modelo de simulação no SimPack/Sim++ foi necessário criar estruturas de dados para cada tipo de elemento presente. Para isso, foram criadas estruturas genéricas para definir as características de cada nó (dispositivo móvel), de cada aplicação, de cada evento de simulação e de cada aglomerado.

4.5.2.1 Estrutura dos Nós

Para a criação dos nós, foi definida a seguinte estrutura de dados que cada dispositivo simulado possui:

- my_id: número de identificação do nó dentro de seu aglomerado;
- cluster_id: número de identificação do aglomerado a que este nó pertence;
- MIPS: capacidade em milhões de instruções por segundo deste nó;
- residual_energy: energia residual atual que o nó possui em Joules;
- initial_residual_energy: energia residual que o nó iniciou na simulação em Joules;
- confiability_factor: fator de confiabilidade do nó entre 0 e 1;

- energy_expenditure: gasto de energia estimado do nó em mW por segundo;
- probability: probabilidade do nó falhar ao ser atribuída uma tarefa, entre 0 e 1.

4.5.2.2 Estrutura das Aplicações

As aplicações, que são exportadas em forma de requisições, têm informações básicas que a definem:

- num_tarefas: número de tarefas para atender requisições desse tipo;
- tamanho_tarefa: tamanho de cada tarefa em milhões de instruções;
- deadline_aplicacao: *deadline* da aplicação em segundos;
- tamanho_aplicacao: tamanho da aplicação em milhões de instruções;
- comunicacao: quantidade de dados para exportar a aplicação em Mb.

4.5.2.3 Estrutura dos Eventos

Já o vetor de eventos que é previamente criado no início da simulação possui os seguintes termos:

- identificador: identificador único do evento;
- inicio_requisicao: instante de tempo da submissão da requisição;
- termino_atendimento: instante de tempo do término de atendimento da requisição;
- tipo_servico: tipo do serviço necessário para atender a requisição;
- fluxo: fluxo atual do evento;
- aglomerado: aglomerado que está processando a requisição;
- no: nó que está processando a requisição;
- externo: aglomerado externo a atender a requisição;
- tempo_execucao: tempo que levou para executar as tarefas depois de alocadas em segundos;
- tempo_comunicacao: tempo gasto em comunicação em segundos;

4.5.2.4 Estrutura dos Aglomerados

Cada aglomerado possui sua estrutura de dados com informações acerca de seus nós e seus enlaces de rede simulados, e esta estrutura está descrita a seguir:

- num_nos: número de nós deste aglomerado;
- MIPS_normalizado: valor normalizado de capacidade em MIPS que este aglomerado tem em relação aos outros;
- comunicacao_interna_normalizada: valor normalizado de taxa de transmissão interna que este aglomerado tem em relação aos outros;
- comunicacao_externa_normalizada: valor normalizado de taxa de transmissão externa que este aglomerado tem em relação aos outros;
- gasto_energia_medio: gasto médio de energia dos nós deste aglomerado;
- confiabilidade_media: confiabilidade média dos nós deste aglomerado;
- servicos_oferecidos: serviços oferecidos por este aglomerado;
- transmissao_interna: taxa de transmissão interna deste aglomerado em Mb/s;
- latencia_interna: latência interna deste aglomerado em ms;
- transmissao_externa: taxa de transmissão externa deste aglomerado em Mb/s;
- latencia_externa: latência externa deste aglomerado em ms;

4.6 Gastos de Comunicação

Os gastos de comunicação, tanto de tempo quanto de energia, não foram considerados no modelo que o trabalho de Borro (2013) propôs, sendo uma informação importante em se tratando de ambientes móveis. Para simular os tempos de comunicação das aplicações, foi utilizada a teoria de atraso em redes de computadores. Para isso, foram considerados os seguintes atrasos segundo as teorias de redes de computadores em Kurose *et al.* (2013):

- Atraso de processamento;
- Atraso de fila;
- Atraso de transmissão;
- Atraso de propagação.

Os atrasos de processamento de fila foram considerados como 0.

Dessa forma, foi criada uma função (calcula_atraso_rede) que recebe o tamanho dos dados a serem transmitidos, a taxa de transmissão do enlace e a latência entre os pontos. O valor retornado se dá por:

$$(tamanho\_comunicao/taxa) + latencia;$$

A latência está em segundos, o tamanho do pacote em *kilobytes* e a taxa em *kilobytes* por segundo, e simplesmente é a soma dos tempos de transmissão (tamanho do pacote pela taxa do enlace) e de propagação (latência do enlace).

Obtido o tempo de comunicação, o próximo passo foi verificar o custo energético de cada nó para transmitir / receber os dados. Para isso, foi utilizado o modelo de Zhang, Wen e Wu (2013) de gastos de comunicação em dispositivos com conexão sem-fio. Para isso, o trabalho propõe o gasto de energia para enviar e o gasto de energia para receber:

$$E_s(k) = d_s(k)p_s \tag{4.5}$$

$$E_r(k) = d_r(k)p_r \tag{4.6}$$

Onde:

- $E_s(k)$ = energia em Joules dispendida para enviar um conjunto de dados k;
- $E_r(k)$ = energia em Joules dispendida para receber um conjunto de dados k;
- k = conjunto de dados;
- d_s = atraso total de comunicação no envio;
- d_r = atraso total de comunicação no recebimento;
- p_s = constante de gasto no envio = 0.1W;
- p_r = constante de gasto no recebimento = 0.05W.

Por fim, foi construída uma função que retorna a energia gasta na comunicação que recebe como parâmetros de entrada o tempo que um determinado nó gastou em comunicação e se está enviando ou recebendo os dados, sendo uma estimativa aproximada de gasto energético.

4.7 Verificação e Validação

Nesta seção será apresentad a metodologia que envolve a verificação e validação das partes que compõem o modelo.

Foram identificados três principais pontos de verificação e validação: heurística *Maximum Regret*, gastos de comunicação e o modelo como um todo. Para cada ponto foi definido qual a melhor estratégia de V&V, sendo cada ponto e sua forma de V&V representados na Figura 18.

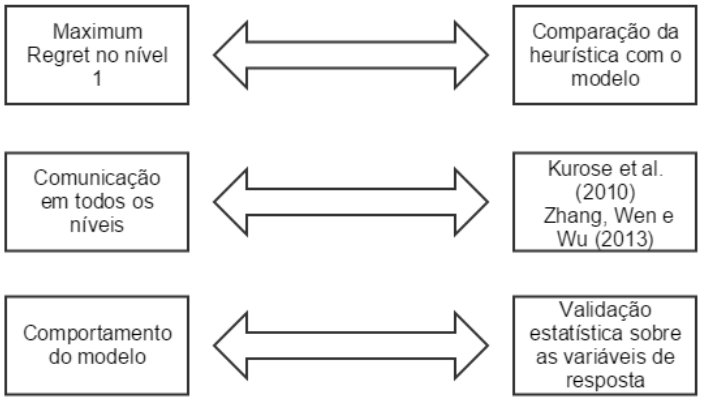

Figura 18 – Pontos de Verificação e Validação

Para verificar e validar se a implementação realizada da heurística *Maximum Regret* para este projeto de Mestrado está de acordo com o que se espera, e se seus resultados estão válidos em relação ao modelo ótimo (programação linear), foi realizado um estudo preliminar da energia total gasta para atribuir um determinado conjunto de tarefas em um ambiente.

Os custos de comunicação, tanto de tempo quanto de energia, foram validados conforme as teorias de Kurose *et al.* (2013) e Zhang, Wen e Wu (2013), e toda metodologia acerca desse ponto já foi debatida na Seção 4.6.

Por fim, o modelo completo foi validado com ferramentas estatísticas de teste de hipótese com as variáveis de resposta definidas no planejamento de experimentos, uma vez que não foi possível encontrar algum modelo já existente que possibilitasse uma comparação justa.

4.7.1 Maximum Regret

Para verificar e validar a implementação do Maximum Regret, o modelo de programação linear descrito no início desse capítulo foi implementado na linguagem do CPlex[1], que é um *solver* para modelos de programação matemática.

Foi definida uma grade com 30 dispositivos móveis, sendo que foram gerados valores em MIPS com média de 300 MIPS de forma uniforme, variando de 200 a 400 MIPS. Já o gasto de

[1] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer

energia de cada elemento foi gerado uniformemente entre 200mW e 400mW, com média igual a 300mW. Os valores também usaram como referência o que foi adotado em Borro (2013).

Em seguida, um conjunto de tarefas foi definido baseando-se nos experimentos realizados em Borro (2013). Ao todo foram definidas 50 tarefas sendo que:

- Média de 3000 MI para 10 tarefas;
- Média de 4000 MI para 10 tarefas;
- Média de 5000 MI para 10 tarefas;
- Média de 6000 MI para 10 tarefas;
- Média de 7000 MI para 10 tarefas.

O conjunto todo de tarefas resultou em uma média de 5000 MI (milhões de instruções). Com isso, foi definido um *deadline* seguindo o mesmo critério de Borro (2013) para o *deadline* estendido, sendo fixado em 66,66s (4 * (média em MIPS das tarefas) / (média em MIPS dos recursos)).

Após 10 replicações, foram colhidos os gastos totais de energia em ambas implementações, e comparadas suas médias. Tal comparação está ilustrada no gráfico da Figura 19, no qual observa-se que o modelo ótimo foi ligeiramente melhor que a heurística implementada, havendo até certo grau de incerteza se os intervalos de confiança forem considerados.

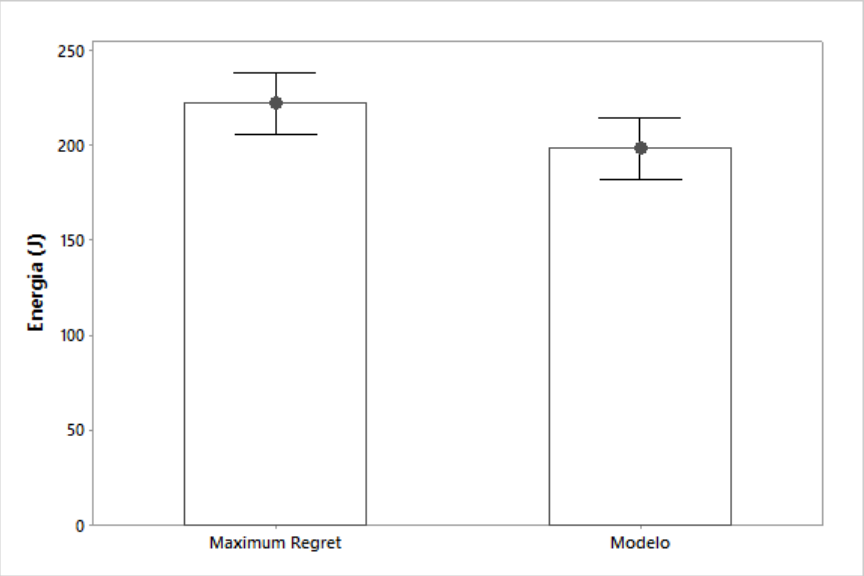

Figura 19 – Gasto de Energia para o Maximum Regret implementado e o Modelo de Programação Linear

Os valores médios obtidos estão muito próximos com os que foram obtidos em Borro (2013), assim como a diferença entre o modelo ótimo e a heurística. Sendo assim, a implementação da heurística *Maximum Regret* está condizente com o que se esperava.

4.7.2 Validação Estatística

A validação do comportamento do modelo completo foi realizada diretamente sobre as variáveis de resposta, através de ferramentas estatísticas que testam hipóteses e embasam as afirmações acerca do comportamento do modelo. Os testes de hipótese foram aplicados no Capítulo 5, e a metodologia será explicada a seguir.

Cada conjunto de replicações de um experimento que possuía certa incerteza em relação aos intervalos de confiança, seguindo a metodologia de avaliação de desempenho de Bukh e Jain (1992), foi submetido a um teste de normalidade, mais especificamente o teste Shapiro-Wilk (SHAPIRO; WILK, 1965). Com isso, foi possível saber se os valores possuíam normalidade, possibilitando escolher o teste de hipótese mais adequado para cada caso. Lembrando que as amostras eram independentes, ou seja, não havia correlação entre seus indivíduos.

Como os valores apresentaram normalidade, ou seja, o teste Shapiro-Wilk resultou em um *p_valor* maior ou igual a 0.05, o teste t foi aplicado (BOX, 1987). O teste t permite, por exemplo, refutar ou aceitar hipóteses como igualdade entre os valores ou diferença através do *p_valor* correspondente.

Porém, nos casos em que os valores analisados não apresentaram normalidade, observou-se a necessidade de aplicar outros testes estatísticos para amostras independentes. Neste caso, o teste utilizado foi o Mann-Whitney (RUXTON, 2006).

Na Figura 20 é possível ver um fluxograma que sintetiza toda a metodologia de validação estatística. Neste caso, nos testes t e Mann-Whitney a hipótese de nulidade significa que as amostras são estatisticamente diferentes, e a hipótese alternativa implica que as amostras são estatisticamente iguais. Desta forma, é possível afirmar se as variáveis de resposta possuem comportamentos de acordo com o que se espera em cada caso. Novamente, é importante ressaltar que esta metodologia foi aplicada nos casos em que restam dúvidas em relação às diferenças estatísticas dos resultados, fundamentando as conclusões obtidas.

4.8 Considerações Finais

Neste capítulo foram apresentados detalhes de desenvolvimento do projeto, onde o trabalho de Borro (2013) foi expandido a um modelo mais complexo e realista. O próximo capítulo contém a metodologia para a validação estatística dos resultados, assim como todas as definições de cenários e seus objetivos de avaliação, o planejamento de experimentos, a avaliação de desempenho, os resultados obtidos e sua análise.

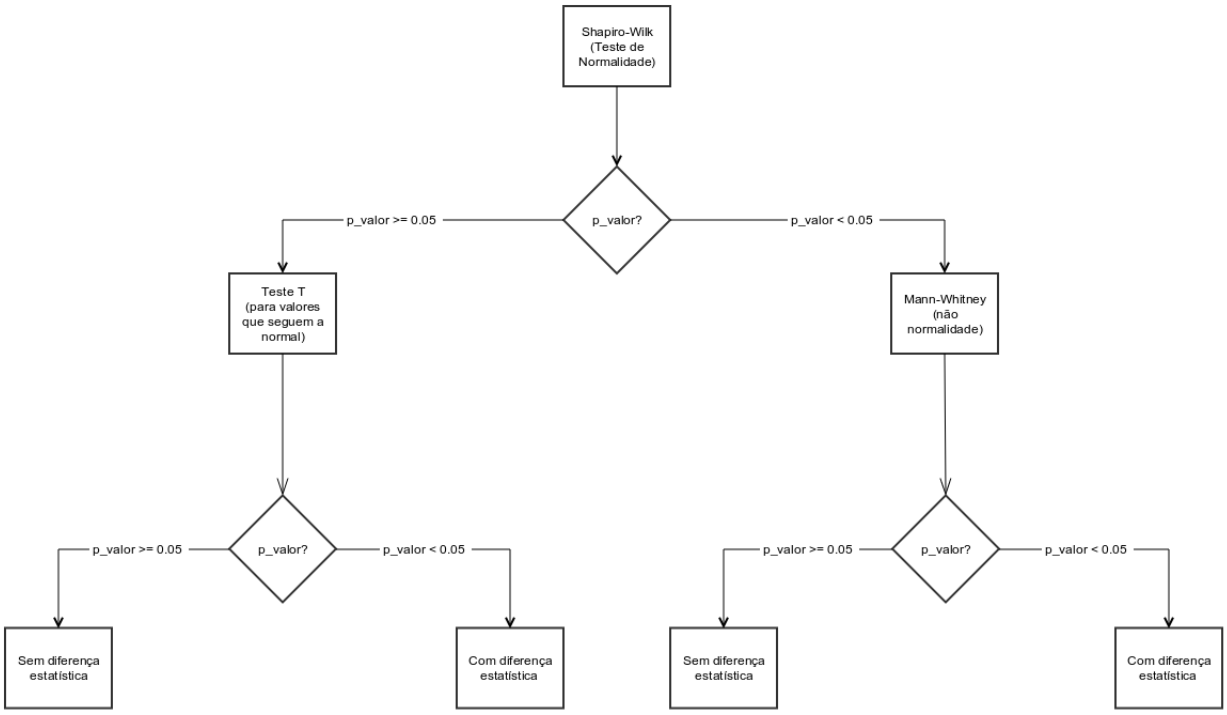

Figura 20 – Fluxograma da Metodologia para Validação do Modelo Completo

CAPÍTULO

5

AVALIAÇÃO EXPERIMENTAL

5.1 Considerações Iniciais

Este capítulo apresenta a avaliação do projeto, partindo em primeiro momento no detalhamento de três cenários com características específicas, os quais possuem objetivos de avaliação distintos, além de cinco classes de aplicação que serão impostas nos cenários avaliados. Após isso, será apresentado o planejamento de experimentos com os fatores, níveis e variáveis de resposta e, por fim, os resultados dos experimentos para cada cenário, assim como suas análises.

5.2 Cenários

Foram definidos três principais cenários para a realização dos experimentos com o objetivo de verificar como o modelo se comporta em diferentes situações, uma vez que existem muitas possibilidades de combinação para avaliar o comportamento da arquitetura. Na Tabela 5 estão representadas as características que definem as infraestruturas dos três cenários em questão.

Tanto os valores de **MIPS médio** quanto de **Gasto de energia médio**, mesmo que fixados, possuem variações estatísticas para cada nó dentro da simulação. Ambos são gerados através de uma distribuição normal, sendo a média o valor definido para cada cenário, e o desvio padrão igual à raiz quadrada da média.

Já os valores de probabilidades de algum nó falhar ao ser atribuído a ele uma ou mais tarefas foram definidos de acordo com valores de referência levantados por (LITKE *et al.*, 2007). Na criação de cada nó, é atribuído um valor de probabilidade de falha, que pode ser 0.02 (pequena), 0.15 (média) ou 0.25 (grande). A escolha do valor para cada nó é feita de forma arbitrária para todos os três cenários.

A seguir, são apresentados os três cenários em relação aos seus objetivos e características determinantes.

Tabela 5 – Fatores que definem os cenários

Infraestrutura
Característica dos enlaces externos
Número de aglomerados
Número de nós por aglomerado
MIPS médio
Gasto de energia médio
Energia residual por nó
Probabilidade de falha

5.2.1 Cenário 1

O cenário 1 foi planejado de forma a priorizar a execução interna das aplicações, ou seja, os aglomerados oferecem os cinco serviços simulados internamente, e possuem capacidade suficiente de atender as requisições que são internamente geradas.

Assim, os valores associados a cada fator foram definido de acordo com a Tabela 6.

Tabela 6 – Características do Cenário 1

Intraestrutura	**Valores**
Característica dos enlaces externos	128kbps e 200ms (lento)
Número de aglomerados	5 (poucos)
Número de nós por aglomerado	30 (muitos)
MIPS médio	900 (alto)
Gasto de energia médio em mW	100 (pouco)
Energia residual por nó em J	20000000 (muito)

Como é possível observar na Tabela 6, os aglomerados possuem muitos nós com muita capacidade de processamento média e com baixo consumo de energia. Além do mais, os enlaces não favorecem a comunicação com o nível 0 e, consequentemente, com os outros aglomerados.

5.2.2 Cenário 2

O cenário 2 foi criado com dois grupos de dispositivos, de forma a criar um grupo com menores capacidades em geral e outro grupo com melhores capacidades. Sendo assim, a ideia é que os dispositivos menos favorecidos exportem a maioria das aplicações para o nível 0, e assim a heurística global tenha influência no consumo de energia dos dispositivos e no tempo de resposta das aplicações. As características estão resumidas na Tabela 7.

Vale reforçar que os elementos do Grupo A explicitados na Tabela 7 são os menos favorecidos, pois possuem poucos elementos com pouco processamento, pouca energia residual por nó, além de gastarem muita energia e oferecem só parte das aplicações. Já os dispositivos do Grupo B são caracterizados por ter muitos nós com maior capacidade de processamento, pouco gasto médio de energia e muita energia residual, além de oferecerem todas as aplicações.

Tabela 7 – Características do Cenário 2

Intraestrutura	Valores Grupo A	Valores Grupo B
Característica dos enlaces externos	10Mbps e 20ms (rápido)	10Mbps e 20ms (rápido)
Número de aglomerados	7	8
Número de nós por aglomerado	10 (poucos)	30 (muitos)
MIPS médio	300 (baixo)	900 (alto)
Gasto de energia médio (mW)	500 (alto)	100 (baixo)
Energia residual por nó (J)	200000 (pouco)	20000000 (muito)

Porém, é importante notar que os enlaces possuem boas características, o que favorece a comunicação entre o nível 0 e o nível 1.

5.2.3 Cenário 3

O cenário 3 foi definido de forma a extrapolar as dimensões dos outros dois primeiros. Dessa forma, foram definidos três grupos de aglomerados. Cada grupo possui 10 aglomerados, sendo que o primeiro grupo possui 10 nós, o segundo 20 nós e o terceiro 30 nós, sendo 30 aglomerados ao todo.

Já as características de cada grupo de aglomerados variaram de forma arbitrária entre três possíveis níveis de cada fator, que estão representados na Tabela 8.

Tabela 8 – Valores considerados no cenário 3

Intraestrutura	Nível 1	Nível 2	Nível 3
Característica dos enlaces externos	128kbps e 200ms (lento)	2Mbps e 80ms (médio)	10Mbps e 20ms (rápido)
MIPS médio	300 (pequeno)	600 (médio)	900 (grande)
Gasto de energia médio (mW)	100 (pequeno)	300 (médio)	500 (grande)
Energia residual por nó (J)	200000 (pequeno)	2000000 (médio)	20000000 (grande)

Esta variação permite que tanto a heurística interna no nível 1 quanto a heurística global no nível 0 tenham efeitos sobre as aplicações, mostrando como se dá o comportamento das mesmas em um ambiente heterogêneo e variado, o que é muito similar ao mundo real.

Novamente citando, em todos os cenários, os valores de probabilidade de um dispositivo falhar ao receber uma tarefa para o processamento foram gerados seguindo valores de referência de (LITKE *et al.*, 2007). Um dispositivo pode, com a mesma chance, ser criado com probabilidade de 0.02 (baixa), 0.15 (média) ou 0.25 (alta).

5.3 Classes de Aplicações

Além dos cenários, foram definidos 5 classes diferentes de aplicações, sendo estas decompostas em tarefas.

Essa diferenciação em classes faz com que o ambiente como um todo represente o que acontece no mundo real, onde diferentes classes (ou tipos) de aplicações móveis com suas características específicas são executadas pelos usuários.

As características que definem as aplicações e os possíveis valores que cada característica pode assumir estão resumidos na Tabela 9. O valor em **MI da aplicação** é a quantidade em milhões de instruções que uma determinada aplicação tem para ser processada, assim como o **número de tarefas** que esta aplicação é composta, o ***deadline*** que as heurísticas devem assumir (não considera o tempo de comunicação nem de fila para o processamento em uma grade local) e o **tamanho da comunicação** necessária para deslocar uma aplicação entre os níveis e os nós no nível 1.

Tabela 9 – Características que definem cada aplicação e seus possíveis valores

Característica	**Possíveis valores assumidos**
MI da aplicação	1000, 3000, 6000
Número de tarefas	20, 60, 90
Deadline da aplicação	30 s, 50 s, 70 s
Tamanho da comunicação	100 kB, 1 MB, 10 MB

Os valores que cada aplicação assume, assim como suas definições e objetivos, estão descritos a seguir.

5.3.1 Classe de Aplicação 1

Esta aplicação tem o perfil de ser a mais computacionalmente "leve". Ou seja, seus valores em MIPS, número de tarefas e de comunicação são os menores entre todas, e seu *deadline* é o com maior "folga". Estas definições servem para simular uma aplicação móvel simples, e que não exija muito dos recursos remotos, e que não tenha muitas prioridades em relação à responsividade, como uma rotina de cálculo de localização por GPS de um *smartphone*.

Os valores assumidos para esta aplicação são:

- MI da aplicação: 1000;
- Número de tarefas: 20;
- *Deadline* da aplicação: 70s;
- Tamanho da comunicação: 100 kB.

5.3.2 Classe de Aplicação 2

A aplicação 2 têm predominância, dentre suas características, de requisitar muito processamento. Isso se deve à quantidade de MIPS que a aplicação requer dos recursos. Estas

definições têm o propósito de simular aplicações que são *CPU-bound*, como o solucionamento de um sistema linear.

Os valores assumidos para esta aplicação são:

- MI da aplicação: 6000;
- Número de tarefas: 60;
- *Deadline* da aplicação: 50s;
- Tamanho da comunicação: 1 MB.

5.3.3 Classe de Aplicação 3

A aplicação 3 possui muitas tarefas, o que põe à prova como o ajuste de corte de ε impacta no consumo de energia, uma vez que com mais tarefas o número de falhas pode aumentar. A ideia é que esta aplicação represente, como exemplo, aplicações móveis que precisem aplicar transformações e processamento intensivo em matrizes e vetores com grandes dimensões, e que a granularidade faça com que pequenas porções de processamento componham a aplicação como um todo.

Os valores assumidos para esta aplicação são:

- MI da aplicação: 3000;
- Número de tarefas: 90;
- *Deadline* da aplicação: 50s;
- Tamanho da comunicação: 1 MB.

5.3.4 Classe de Aplicação 4

Esta aplicação possui uma proporção muito estreita ao considerar a quantidade em MIPS e o seu *deadline*. Esta característica principal faz com que a heurística interna ao nível 1 necessite alocar as tarefas aos recursos de forma adequada, além de forçar com que esta aplicação seja, por vezes, exportada ao nível 0, uma vez que o *deadline* impeça o processamento no aglomerado de origem por escassez de recursos. A aplicação 4 pode representar aplicações reais como partes de motores gráficos e renderização de cenários em jogos eletrônicos, que requerem um processamento elevado com alta responsividade.

Os valores assumidos para esta aplicação são:

- MI da aplicação: 6000;

- Número de tarefas: 60;
- *Deadline* da aplicação: 20s;
- Tamanho da comunicação: 1 MB.

5.3.5 Classe de Aplicação 5

Esta aplicação tem predominância de requisitar muita comunicação, o que permite avaliar qual a relação do tempo de resposta com o tempo de comunicação. O objetivo ao modelar essa aplicação é simular aplicações reais como transformações em imagens e vídeos, as quais precisam de um processamento razoável e muita comunicação ao transportar os dados pelos enlaces.

Os valores assumidos para esta aplicação são:

- MI da aplicação: 3000;
- Número de tarefas: 60;
- *Deadline* da aplicação: 50s;
- Tamanho da comunicação: 10 MB.

5.4 Planejamento de Experimentos

O planejamento de experimentos, assim como a execução e a análise dos resultados foram os mesmos para cada um dos três cenários, seguindo a metodologia de (BUKH; JAIN, 1992).

Assim sendo, para cada cenário foi definido um modelo de experimentos fatorial completo, com dois fatores e três níveis cada fator, levando a uma combinação de $3^2 = 9$ experimentos. Além disso, foram realizadas 30 replicações para cada experimento, de forma a estabelecer um nível de confiança de 95%.

O primeiro fator é o ajuste da restrição que envolve o fator de confiabilidade na heurística Maximum Regret, chamada de epsilon (ε) e apresentada no Capítulo 4. Seus três níveis são:

- $\varepsilon = 0$: todos os recursos com fator de confiabilidade maior ou igual que 0 são considerados internamente, ou seja, nenhuma restrição é imposta à heurística;
- $\varepsilon = m/2$, onde m é o número máximo de elementos que podem ser desconsiderados sem comprometer a factibilidade da alocação;
- $\varepsilon = m$, onde m é o número máximo de elementos que podem ser desconsiderados sem comprometer a factibilidade da alocação.

Quando o $\varepsilon = m/2$, os $(m/2) - 1$ elementos com menores fatores de confiabilidade dos m possíveis são evitados, sendo considerados todos os outros que não estão nesse grupo.

Já o segundo fator está relacionado com o valor K, que ajusta quantos elementos são pré-selecionados dentro da heurística global apresentada no Capítulo 4, e seus níveis são:

- $K = 1$: somente o primeiro aglomerado da lista de pré-seleção é considerado;
- $K = n/2$, onde n é o número total de aglomerados possíveis;
- $K = n$, onde n é o número total de aglomerados possíveis.

A combinação dos fatores e níveis definidos na avaliação foi resumida na Tabela 10, onde o fator 1 corresponde à combinação de $\varepsilon = 0$ e $K = 1$, e assim por diante, de forma a simplificar a notação nas posteriores análises.

Tabela 10 – Combinação de fatores e níveis

Experimento	Combinação
1	$\varepsilon = 0, K = 1$
2	$\varepsilon = 0, K = n/2$
3	$\varepsilon = 0, K = n$
4	$\varepsilon = m/2, K = 1$
5	$\varepsilon = m/2, K = n/2$
6	$\varepsilon = m/2, K = n$
7	$\varepsilon = m, K = 1$
8	$\varepsilon = m, K = n/2$
9	$\varepsilon = m, K = n$

Já as variáveis de resposta adotadas nesta avaliação foram escolhidas de forma a permitir uma melhor análise de como a variação de ε e K as afeta. São:

- **Energia Total Dispendida (ETD):** é a média aritmética da soma de toda energia dispendida em Joules por todos os dispositivos de todos os aglomerados, do começo ao fim da simulação;
- **Tempo de Reposta Médio (TRM):** é a média aritmética do tempo de resposta em segundos de cada classe de aplicação, desde a sua requisição até sua conclusão. Cada valor que compõe a média é a soma do tempo de processamento / escalonamento com o tempo gasto em comunicação e o tempo de fila quando o aglomerado recebe requisições simultâneas. Cada classe de aplicação possui sua média de tempo de resposta, sendo denotados por TRM-1 para o tempo de reposta da aplicação 1, e assim por diante;
- **Tempo de Comunicação Médio (TCM):** é a média aritmética do tempo gasto em comunicação em segundos de cada classe de aplicação. Cada classe de aplicação possui sua média

de tempo de comunicação, sendo denotados por TCM-1 para o tempo de comunicação da aplicação 1, e assim por diante;

A geração de carga de trabalho foi realizada de forma a garantir que todos os aglomerados gerassem requisições de todas as classes de aplicações. Para tanto, nos cenários 1 e 2 cada aglomerado gerou quatro requisições de cada classe de aplicação durante a simulação. Já no cenário 3 foram geradas três requisições de cada classe de aplicação em cada aglomerado, sendo o número de requisições neste cenário menor que nos dois outros uma vez que neste cenário foram considerados 30 aglomerados, o que resultaria em um número total de requisições elevado caso as requisições fossem geradas da mesma forma que nos outros cenários.

5.5 Experimentos e Resultados

Antes de analisar separadamente os cenários, foram realizados os testes de normalidade Shapiro-Wilk para cada variação de fatores e cada variável de resposta.

As Tabelas 14, 15, 16, 17, 18, 19, que estão contidas no Apêndice A, contêm os *p_valores*, que servirão como referência para decidir qual teste é o mais apropriado em cada caso.

5.5.1 Resultados no Cenário 1

A avaliação no cenário 1, já definido anteriormente, prioriza estritamente a execução interna aos aglomerados das aplicações. Ou seja, é garantido que nenhuma aplicação que é submetida no nível 1 é repassada ao nível 0.

Dessa forma, avaliar como a variação nos níveis do fator K interfere nas variáveis de resposta torna-se desnecessário. Apenas o fator ε tem interferência nos resultados, uma vez que é aplicado internamente aos aglomerados no nível 1. Assim, a análise para este cenário considerou apenas os resultados para a variação de ε, ou seja, para cada resultado acerca de ε, o fator K assume todos seus níveis simultaneamente. Quando $\varepsilon = 0$, o K é igual a $1, K = n/2$ e n, e o resultado final é a média dos três níveis de K para o mesmo ε. O mesmo foi feito para $\varepsilon = m/2$ e $\varepsilon = m$.

Primeiramente, apresenta-se a análise em torno da Energia Total Dispendida em todos os aglomerados. O gráfico com as médias e intervalo de confiança em 95%, considerando somente o fator ε, pode ser visto na Figura 21.

O que fica evidente no gráfico da Figura 21 é a significativa redução nos custos de energia quando a restrição de corte imposta por ε passa de 0 (nenhuma) para o valor médio do conjunto de elementos que garantem a factibilidade das aplicações, dispensando qualquer análise mais aprofundada. Ou seja, a primeira metade do conjunto de factibilidade com os $(m/2) - 1$ dispositivos com os menores valores de confiabilidade em relação aos $m/2$ maiores gera a

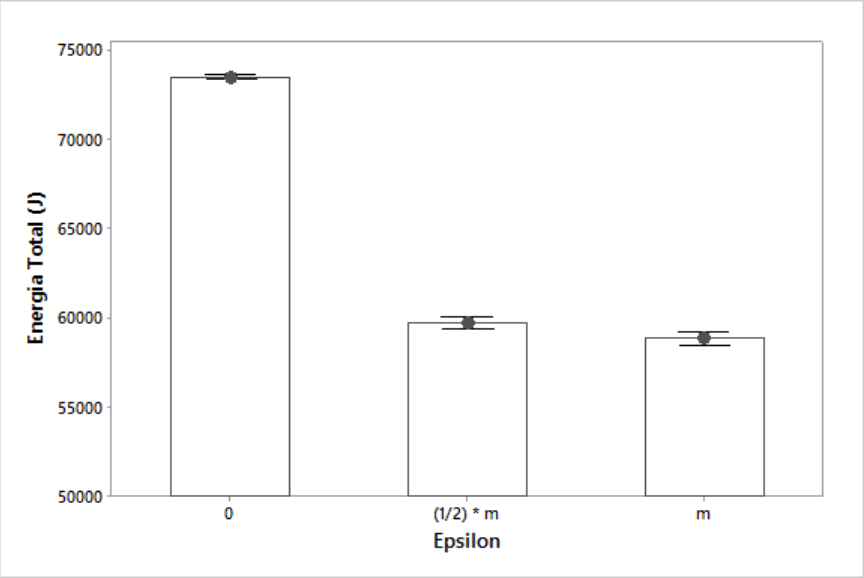

Figura 21 – Energia Total Dispendida no Cenário 1

maioria das falhas, o que aumenta o número de tarefas a serem realocadas a outros recursos, dispendendo mais energia.

Ao comparar a variação entre $\varepsilon = m/2$ e $\varepsilon = m$, é possível ver uma pequena diferença. Isto pode ser explicado pelo *trade-off* entre o custo extra gerado pelas falhas e a limitação nos recursos selecionáveis pela heurística interna. Bons dispositivos, com menores gastos de energia por unidade de tempo, podem estar sendo eliminados, resultando em uma possível estagnação na economia quando $\varepsilon = m/2$. Porém, como os intervalos de confiança desses níveis estão muito próximos, foi realizado um teste de hipótese seguindo a metodologia apresentada na Figura 20.

Apesar da análise estar voltada somente para o fator ε com os níveis de K estarem representados através de sua média, a verificação de normalidade foi realizada considerando cada experimento com os níveis de ε e K definidos, e o teste de hipóteses também foram realizados comparando os experimentos um a um.

Como todos os valores não seguem a distribuição normal de acordo com a Tabela 15 com os *p_valores* para o teste de Shapiro-Wilk, foram aplicados nos três níveis comparados o teste Mann-Whitney.

Tomou-se como hipótese nula (H_0) que os valores da energia total dispendida são iguais, sendo a hipótese alternativa (H_1) que são diferentes.

Tabela 11 – Teste de Mann-Whitney com *p_valores* - Cenário 1

$K = n/2$	$K = n$	*p_valor*
4	7	0.0095
5	8	0.6309
6	9	0.0022

Ao verificar os *p_valores* obtidos pelo teste Mann-Whitney na Tabela 11, é possível

ver que somente um deles resultou em não rejeitar a H_0 pois o *p_valor* é maior que o nível de significância 0.05. Ou seja, o ajuste de ε de m/2 para m não melhorou a ETD. Porém, na comparação com os outros fatores, todos rejeitaram a H_0.

Como a maioria dos testes resultou em diferenças estatísticas, será considerado que o ajuste de ε melhorou o consumo de energia, apesar de pouco, como já visto na Figura 21.

Ao analisar os tempos de resposta médios para cada aplicação em torno do ajuste de ε na Figura 22, percebe-se que os tempos de resposta não são afetados mesmo que o número de recursos disponíveis para a heurística seja reduzido quando o ε aumenta. Isso se justifica pelo fato do número de aglomerados ser somente cinco, o que não acarreta em filas de espera nos aglomerados nem sobrecarga nos aglomerados, uma vez que cada um atende suas requisições internamente.

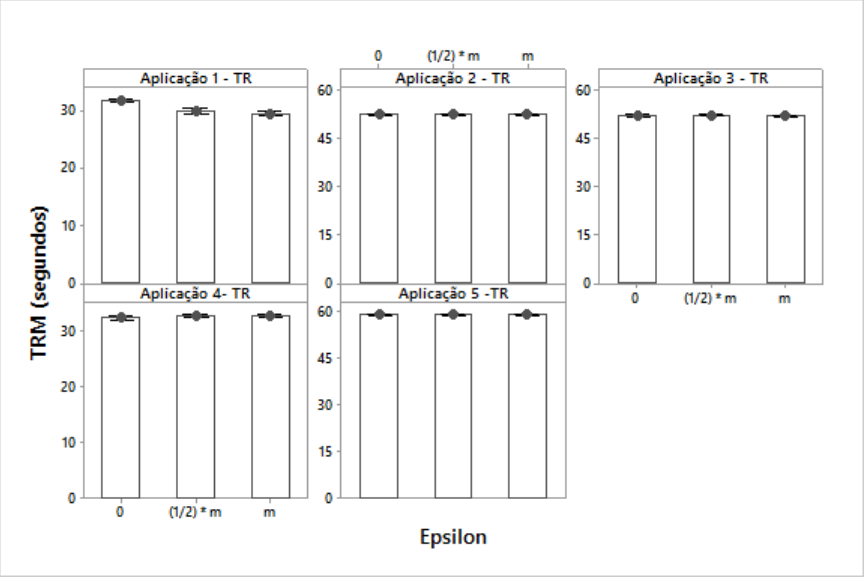

Figura 22 – Tempos de Resposta Médios no Cenário 1

Outro aspecto que contribuiu para os tempos de resposta médios não sofressem modificações conforme ε varia foram os tempos de comunicação médios, presentes na Figura 23, uma vez que também compõem os tempos de resposta. Isso se deve ao fato da comunicação ficar restrita dentro do nível 1, não sendo necessário transportar dados pelos diferentes enlaces que interligam o nível 1 com o nível 0.

5.5.2 Resultados no Cenário 2

O cenário 2 foi projetado de forma a ter dois conjunto distintos de aglomerados: um com dispositivos que possuíssem maior capacidade de processamento e outro com dispositivos que possuíssem capacidades intencionalmente reduzidas. Esta abordagem foi conduzida de forma a forçar com que parte das aplicações - média de 19,44% segundo os *logs* de simulação - fossem

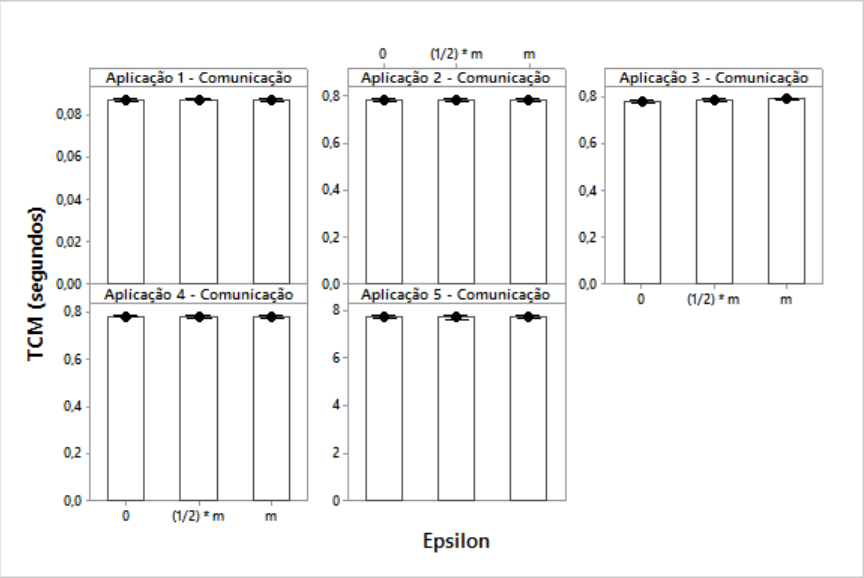

Figura 23 – Tempos de Comunicação Médios no Cenário 1

encaminhadas ao nível 0 e atendidas por outro aglomerado, verificando como o ajuste do fator K influencia as variáveis de resposta.

A análise desse cenário focou separadamente em cada fator, facilitando a compreensão do comportamento das variáveis de resposta em relação aos fatores e seus respectivos níveis.

No primeiro momento, a análise será em torno da ETD em relação ao ajuste do fator ε, sendo que para cada ε foi realizada a média para $K = 1, K = n/1$ e $K = n$. É possível ver na Figura 24 os valores da ETD para este caso.

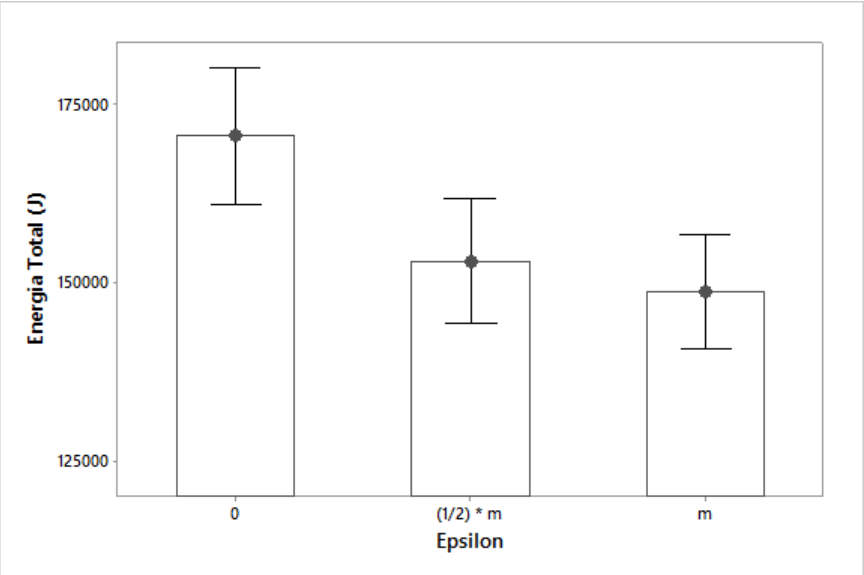

Figura 24 – Energia Total Dispendida no Cenário 2 para o fator ε

Similar com o que aconteceu no cenário 1, conforme o valor de ε aumenta de 0 para m/2, a ETD diminui, uma vez que os dispositivos com mais chances de falhar são evitados. Como os

intervalos de confiança estão próximos, fez-se a necessidade de aplicar um teste de hipóteses para entender melhor o que os resultados significam.

Como os valores de ETD pertinentes aos fatores e níveis abordados não seguem a normal segundo os *p_valores* já obtidos na Tabela 17 pelo teste de normalidade Shapiro-Wilk, o teste de hipótese utilizado será novamente o Mann-Whitney.

Foi fixado o nível de K para cada variação de ε, e a hipótese nula H_0 é que os valores comparados são iguais, e a hipótese alternativa H_1 que os valores são diferentes. Os *p_valores* estão representados na Tabela 12.

Tabela 12 – Teste de Mann-Whitney com *p_valores* - Cenário 2 Teste 1

$K = 1$	$K = n/2$	**p_valor**
1	4	0.7958
2	5	0.0001
3	6	0.0207

Somente o *p_valor* dos experimentos 1 e 4 é maior do que o nível de significância (0.05). Ou seja, somente um dos testes aceitou a H_0, e os outros a rejeitaram. Sendo assim, considerou-se que a hipótese alternativa H_1, que significa que os valores são diferentes, ou seja, quando ε muda de 0 para $m/2$, há uma diminuição na ETD.

Já quando o ε se torna ainda mais restritivo, passando de $m/2$ para m, os intervalos de confiança de ambos sobrepõem as médias, sendo estatisticamente equivalentes segundo a teoria contida em (BUKH; JAIN, 1992). Ou seja, restringir ainda mais os dispositivos em relação ao seu fator de confiabilidade não resultou em melhoras na economia de energia. Isso pode ser explicado pelo fato dos aglomerados com melhores capacidades estarem limitando bons recursos internos ao adicionar a restrição de ε, gastando energia por alocar tarefas a recursos com maior gasto de energia médio e menor capacidade de processamento.

Mudando o enfoque, é possível observar na Figura 25 como a ETD se comporta com a variação em torno de K, sendo que para cada valor de K foi realizada a média dos valores para $\varepsilon = 0, \varepsilon = m/2$ e $\varepsilon = m$.

Quando $K = 0$, ou seja, somente a métrica de QoS que considera capacidade de processamento média e características de comunicação dos aglomerados é utilizada, um maior consumo total foi observado em relação aos outros níveis. Conforme a heurística do nível 0 começa a priorizar as métricas que remetem à economia de energia e à confiabilidade média, aumentando para $K = n/2$, é possível ver uma possível melhora na economia de energia. Mesmo assim, observou-se a necessidade de aplicar o teste de hipótese pois os intervalos de confiança desses níveis estão próximos.

De modo semelhante aos dois primeiros testes, foi aplicado o teste de Mann-Whitney pois os valores não são parametrizáveis, sendo os *p_valores* representados na Tabela 19. Novamente, tomou-se a H_0 pela afirmação de que os valores são iguais, e H_1 como diferentes.

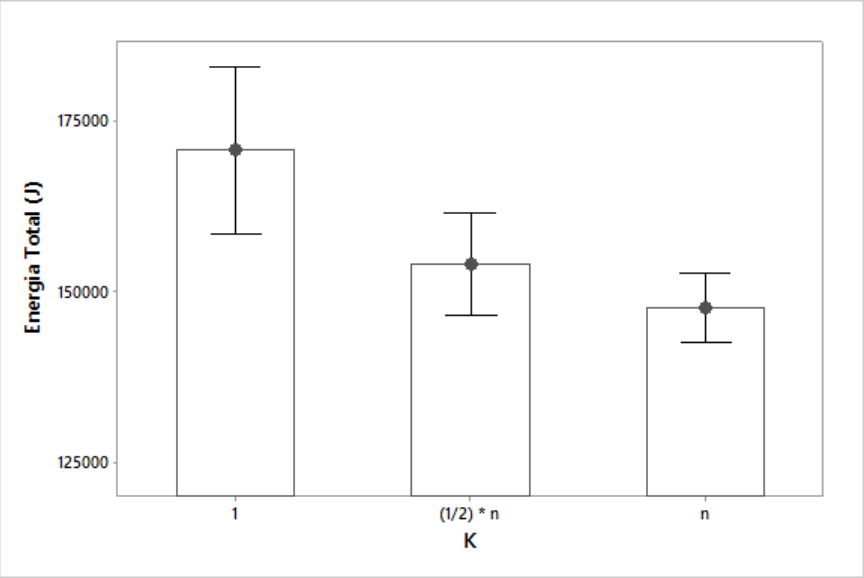

Figura 25 – Energia Total Dispendida no Cenário 2 para o fator K

Tabela 13 – Teste de Mann-Whitney com *p_valores* Cenário 2 Teste 2

$K = 1$	$K = n/2$	**p_valor**
1	2	0.0436
4	5	0.1907
7	8	0.0993

Como é possível ver na Tabela 13, somente o *p_valor* referente aos experimentos 1 e 2 foi menor que o nível de significância adotado (0.05), ou seja, somente um dos testes rejeitou a H_0. Com isso, considerou-se que não houve melhora na ETD quando K passa de 1 para $n/2$.

Já quando K aumenta de $K = n/2$ para $K = n$, a ETD praticamente foi a mesma, já que o intervalo de confiança de um nível contém a média do outro. Ou seja, a métrica que prioriza a economia de energia prevista e a confiabilidade média dos aglomerados não foi capaz de os selecionar de forma a diminuir o consumo.

Para concluir a discussão em torno da ETD, ficou evidente que só é possível ver melhora nesta variável de resposta quando comparamos os níveis $K = 1$ e $K = n$, visto que seus intervalos de confiança na Figura 24 estão bem separados.

Já em relação aos tempos de resposta médios, constatou-se previamente que o fator ε não afeta seus valores, o que permite evitar uma análise considerando puramente suas variações.

Assim sendo, a análise dos TRMs serão em torno da variação do fator K, uma vez que foi o que realmente teve influência.

Como é possível ver na Figura 26, todas as aplicações tiveram praticamente o mesmo comportamento, sendo $K = 0$ e $K = n/2$ os níveis com valores mais próximos, sendo este último ligeiramente melhor em algumas aplicações.

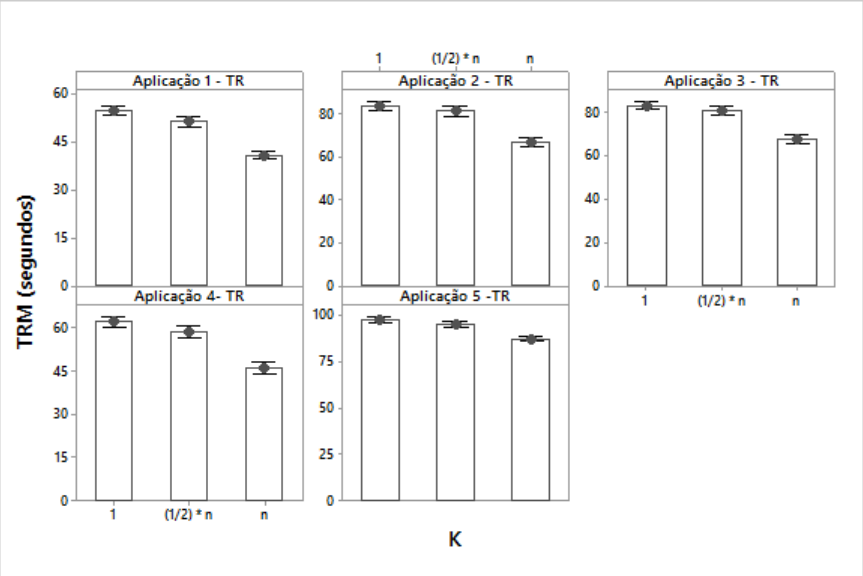

Figura 26 – Tempos de Resposta Médios no Cenário 2 para o fator K

Já quando a heurística global prioriza estritamente a diminuição do consumo de energia e a confiabilidade média ($K = n$), os TRMs apresentam uma diminuição. Esse comportamento, apesar de não ser esperado, pode ser explicado pois o aumento de K faz com que os melhores aglomerados em relação às capacidades de processamento e de comunicação (enlace) não sejam sempre escolhidos e, consequentemente, não sejam sobrecarregados, aumentando o tempo de fila e deteriorando os TRMs. Porém, aglomerados que não têm boas características computacionais podem ser escolhidos, o que também degrada os TRMs, apesar dessa degradação ser menor pois os tempos de fila tiveram um maior impacto. A métrica da heurística global que considera a capacidade de processamento dos aglomerados e as características de comunicação acabou sobrecarregando os melhores aglomerados relacionados à *métrica A*.

Apesar desse aumento nos TRMs, é possível perceber, através da Figura 27, que quando K passa de 1 para $n/2$, há apenas um pequeno aumento nos tempos de comunicação, sendo praticamente desprezíveis. Porém, quando K passa de $n/2$ para n, os TCMs aumentam de forma evidente. Ou seja, conforme apenas a métrica que tenta minimizar o uso de energia e maximizar a confiabilidade média é considerada ($K = n$), aglomerados com enlaces mais lentos podem ser selecionados, degradando o tempo de comunicação para todas as aplicações.

Resumindo, apesar desse aumento nos TCMs, os TRMs melhoram, ficando evidente uma limitação na avaliação pois o número de aglomerados / recursos é pequeno. O ajuste de K para $K = n$ evidencia uma sobrecarga nos aglomerados com melhor capacidade de processamento e de rede, aumentando os TRMs, mesmo que os TCMs diminuam, justificando a criação de um cenário com maiores variações e com mais elementos para a avaliação.

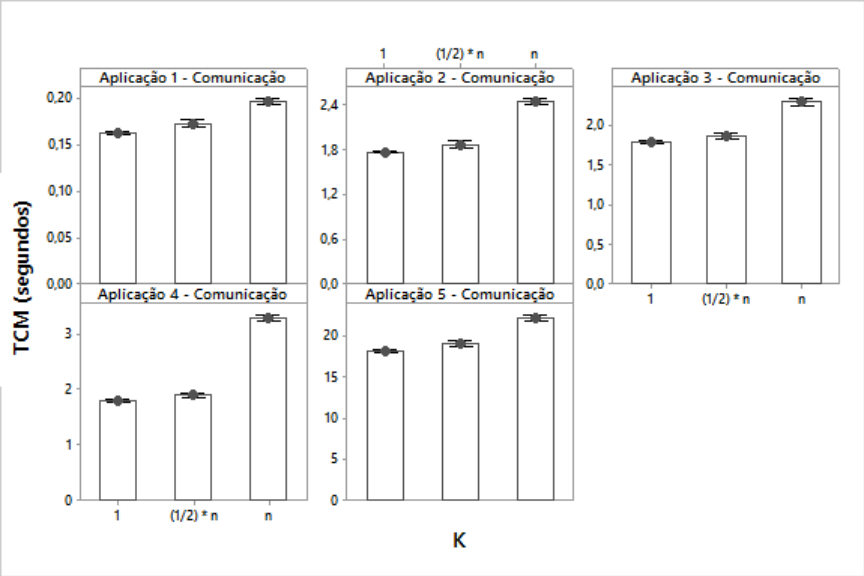

Figura 27 – Tempos de Comunicação Médios no Cenário 2 para o fator K

5.5.3 Resultados no Cenário 3

O cenário 3 possui o maior grau de heterogeneidade entre os aglomerados, uma vez que características como capacidade de processamento média, fator de confiabilidade médio, características de enlace, etc; não foram organizados seguindo qualquer critério ou objetivo de avaliação.

O número de aglomerados também foi extrapolado em 30 elementos, permitindo que a heurística global tenha mais elementos para tomar sua decisão, tentando averiguar se as duas métricas internas têm algum *trade-off* entre si em relação às variáveis de resposta.

A geração da carga de trabalho para esse cenário resultou em 22,28% (em média) das requisições exportadas, ou seja, que saíssem do nível 1 e fossem reescalonadas pelo nível 0. Essa proporção foi obtida através de ajustes propositais na simulação.

Primeiramente, a análise será em torno da ETD quando as heurísticas sofrem variações com os parâmetros ε e K.

Na Figura 28 estão descritos os valores de ETD para a variação dos níveis dos fatores considerados na avaliação. É possível perceber que nos experimentos relativos a $\varepsilon = m/2$ e $K = n/2$ ou $\varepsilon = m$ e $K = n/2$, houve a maior economia de energia. Ou seja, restringir os dispositivos com baixo fator de confiabilidade resultou em economizar energia.

Quando o enfoque é dado totalmente na variação de ε, ou seja, quando para cada nível de ε é realizada a média com todos os valores de K, a ETD isolada com esse fator, representada na Figura 29, atingiu um maior valor quando $\varepsilon = 0$ (sem restrições) em relação às outras duas médias.

Quando $\varepsilon = m/2$, a ETD cai cerca de 50000 J, sendo muito inferior ao nível inicial,

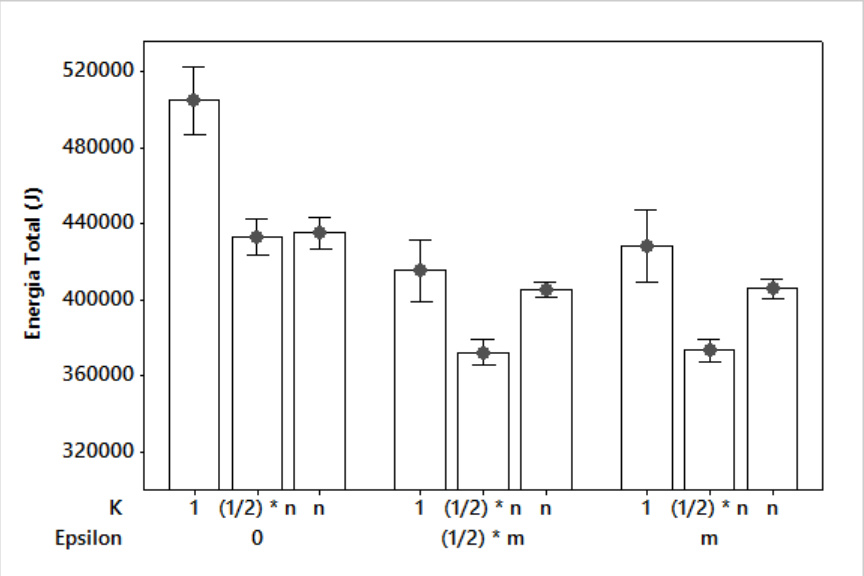

Figura 28 – Energia Total Dispendida no Cenário 3

o que mostra que o ajuste no fator de confiabilidade impediu que um elevado número de falhas degradasse o consumo de energia. Porém, quando o $\varepsilon = m$, a ETD não diminuiu e foi estatisticamente equivalente, uma vez que a restrição máxima faz com que nós com maiores capacidades de processamento em MIPS e com menores consumos de energia deixem de ser opção na heurística do nível 1. Esse comportamento já foi observado nos outros cenários já avaliados.

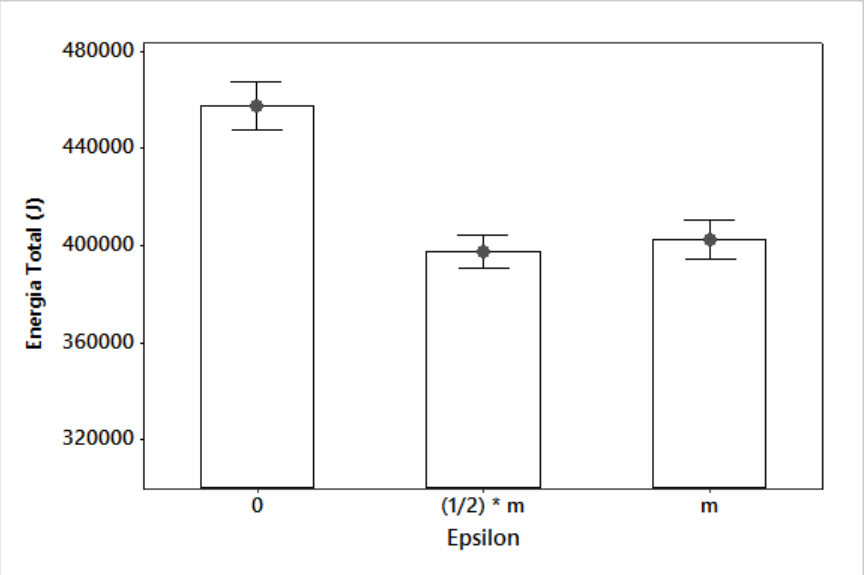

Figura 29 – Energia Total Dispendida no Cenário 3 para o fator ε

Quando isolado o fator K, sendo que para cada K é feita a média com todos os possíveis valores de ε, representado na Figura 30, é possível evidenciar o comportamento já observado na Figura 28: quando $K = n/2$, a ETD é a menor entre os níveis. Com $K = 1$, um maior gasto é

observado, uma vez que apenas a métrica de QoS que considera capacidade de processamento e características de comunicação é utilizada.

Em contrapartida, quando $K = n$, ou seja, somente a métrica que prioriza os recursos mais eficientes e com maiores fatores de confiabilidade, a ETD é menor em relação aos valores obtidos quando $K = 1$, porém pior nos casos quando $K = n/2$.

Esse comportamento evidencia um *trade-off* interessante no consumo de energia: conforme K se torna maior, a heurística de seleção global prioriza mais os recursos com menor gasto médio e mais confiáveis. Porém, aglomerados com maiores capacidades de processamento e com melhores enlaces deixam de ser opção, o que resulta em um aumento de energia, pois as aplicações decompostas em tarefas levam mais tempo dentro dos aglomerados para serem processadas e a comunicação começa a levar mais tempo e ser mais custosa.

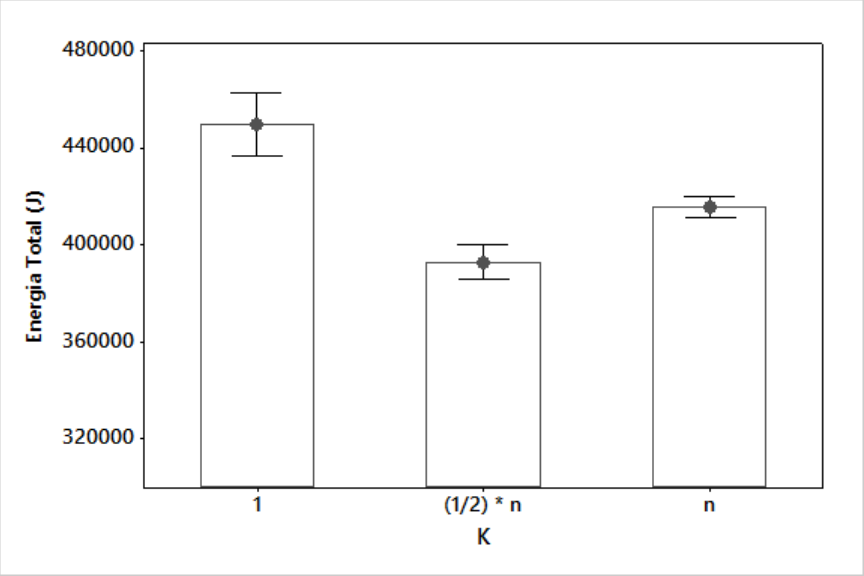

Figura 30 – Energia Total Dispendida no Cenário 3 para o fator K

Já em relação aos TRMs, novamente será dado enfoque apenas na variação do fator K, uma vez que o fator ε não tem muita influência neste caso.

Como é possível ver na Figura 31, todas as aplicações possuem um comportamento similar: os TRMs são menores quando $K = n/2$. Ou seja, também para os tempos de resposta, foi possível verificar um *trade-off* quando K restringe os elementos pré-selecionados na heurística global pela metade.

Quando $K = 1$, os melhores aglomerados em relação à sua capacidade de processamento e aos seus enlaces internos e externos recebem a maioria das aplicações exportadas, ficando sobrecarregados e aumentando o tempo de fila das requisições, sendo esse tempo maior ainda nesse cenário, uma vez que a carga de trabalho foi gerada de forma a saturar em muito os recursos.

Em contrapartida, quando K = n, além dos recursos com menor consumo energético e

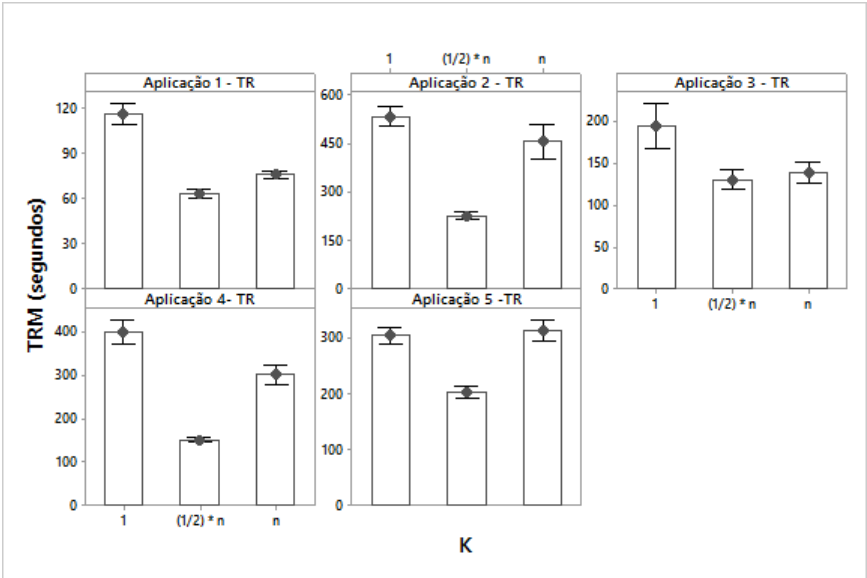

Figura 31 – Tempos de Resposta Médios no Cenário 3 para o fator K

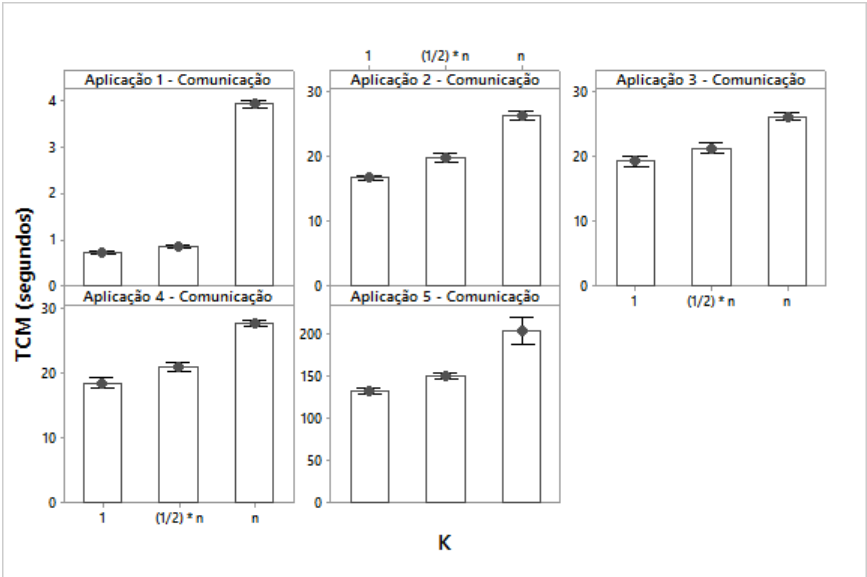

Figura 32 – Tempos de Comunicação Médios no Cenário 3 para o fator K

maior fator de confiabilidade serem sobrecarregados pela heurística global, os TCMs, representados na Figura 32, também contribuem para esta degradação nos TRMs.

Esse fato fica mais evidente ainda ao comparar o TRM-5 e o TCM-5 (aplicação com muita comunicação). Quando $K = 1$ e $K = n$, somente no TRM-5 existe equivalência estatística verificada pelos intervalos de confiança, e nos outros TRMs o $K = 1$ resulta em maiores tempos em relação a quando $K = n$. Isso é explicado pelo aumento no TCM-5, onde a comunicação tem um maior aumento absoluto conforme o fator K aumenta devido à característica da aplicação.

Concluindo, quando $K = n/2$, existe o melhor balanço entre as duas métricas da heurística global, onde tanto a ETD quanto os TRMs e TCMs são os melhores para todas as aplicações.

5.6 Considerações Finais

Este capítulo apresentou a avaliação, validação e análise dos resultados obtidos em cenários distintos, sendo possível chegar a conclusões interessantes em cada caso. As conclusões e o encerramento dos debates estão no próximo capítulo, assim como as contribuições científicas deste projeto de Mestrado e os possíveis trabalhos futuros.

CAPÍTULO

6

CONCLUSÃO

Os dispositivos móveis, como *smartphones* e *tablets*, estão cada vez mais tornando-se populares, sendo um vasto potencial computacional a ser explorado, surgindo como uma alternativa a serviços oferecidos pela computação em nuvem, porém com uma maior proximidade. Apesar disso, existem desafios importantes que devem ser considerados: a restrição energética e a alta mobilidade dos elementos. Ou seja, é importante que a energia dispendida ao utilizar dispositivos móveis como recursos computacionais seja a menor possível, uma vez que tais recursos possuem severas restrições. Além disso, a alta mobilidade dos recursos faz a necessidade de utilizá-los de forma consciente, priorizando os dispositivos com maior confiabilidade.

Este projeto de Mestrado propôs uma arquitetura com dois níveis. O nível 1 é composto por dispositivos móveis organizados em aglomerados que, através de uma heurística que considera tanto a minimização do gasto de energia quanto restrições de confiabilidade dos recursos, e permite que tais dispositivos compartilhem serviços entre si e aproveitem sua proximidade de forma a evitar comunicações dispendiosas com recursos que estão distantes. Já o nível 0 busca integrar diferentes aglomerados, uma vez que nem sempre é possível utilizar os recursos que são oferecidos em suas proximidades colaborativamente. O nível 0 possui seu algoritmo de seleção, que leva em consideração características próprias do ambiente móvel: capacidade de processamento, rede, energia e confiabilidade.

A avaliação desse projeto se deu através de três cenários distintos, e possibilitou ver como os ajustes nos fatores ε e K impactaram nas variáveis de resposta, sendo ε o ajuste na restrição de confiabilidade da heurística *Maximum Regret* e K o ajuste no algoritmo de seleção no nível 0. No cenário 1, em que todas as requisições foram intencionalmente atendidas internamente, foi possível ver que conforme ε aumentava, menos energia era gasta, uma vez que dispositivos que falham muito eram evitados pela restrição, o que fez com que reescalonamentos fossem evitados. Já em relação ao cenário 2, em que uma média de 19,44% das aplicações eram exportadas ao nível 0, foi possível ver uma melhora no consumo de energia acerca de ε

menos expressiva, o que mostrou um *trade-off* na economia de energia quando recursos menos confiáveis são evitados, uma vez que dispositivos com boas capacidades de processamento e baixo de consumo de energia podem também ser desconsiderados. Além do mais, o aumento de K também reduziu o consumo de energia global, uma vez que a métrica que prioriza aspectos de consumo estimado e confiabilidade dos aglomerados é priorizada. Já os tempos de resposta foram menores quando $K = 1$ pois os melhores recursos computacionais são escolhidos segundo a métrica A no algoritmo de seleção de aglomerados externos. Já o cenário 3, por ser o mais complexo e heterogêneo, permitiu ver que tanto ε quando K apresentassem *trade-offs* em seus valores médios $n/2$ e $m/2$, fato também observado nas variáveis de resposta que se referem aos tempos de resposta e de comunicação.

6.1 Contribuições Científicas

As contribuições científicas deste projeto de Mestrado são:

1. A proposta de uma arquitetura que permite que dispositivos móveis sejam explorados como recursos computacionais com ciência de suas restrições de bateria e sua alta mobilidade;
2. Permitir que dispositivos móveis próximos colaborem entre si de forma a aproveitar a baixa latência e evitar serviços em nuvem que podem, muitas vezes, estar sobrecarregados ou indisponíveis;
3. Relacionar a minimização do consumo de energia com a mobilidade dos dispositivos;
4. Propor duas métricas com informações globais que podem ser combinadas e adaptadas em arquiteturas orientadas a serviço de forma a permitir que dispositivos móveis colaborem entre si mesmo que não saibam de sua proximidade.

Com relação à produção científica, foi publicado um resumo expandido nos anais da VI Escola de Alto Desempenho de São Paulo (ERAD-SP/2015) com o título "Integrando Grades Móveis em uma Arquitetura Orientada a Serviço". Este trabalho foi apresentado oralmente por meio de pôster.

Além disso, será submetido um *paper* entitulado "Providing Computing Services through Mobile Devices Collaboratively - A Fog Computing Study Case"provavelmente para o periódico Future Generation Computing Systems, com Qualis Capes A2 e fator de impacto 2.786.

6.2 Trabalhos Futuros

Este projeto de Mestrado, pelo seu escopo e seu detalhamento, permite que outros aspectos sejam abordados e explorados. Os possíveis trabalhos futuros são:

- Considerar o uso de outra heurística adaptada, a *Greedy* (BORRO, 2013), buscando melhorar a economia de energia que a heurística *Maximum Regret* adaptada atinge;
- Considerar outras heurísticas com menores complexidades computacionais, além de heurísticas que não exigem o conhecimento prévio de todas as características dos recursos;
- Construção de um modelo multi-objetivo, cujo fator de confiabilidade deixaria de ser uma restrição e passaria a ser um segundo objetivo;
- Criação de um protótipo com aplicações reais em dispositivos móveis reais;
- Ajuste otimizado da restrição ε na heurística *Maximum Regret* adaptada;
- Unificação das métricas de QoS no nível 0, adicionando mais informações relativas ao contexto de computação móvel.

REFERÊNCIAS

AAZAM, M.; HUH, E.-N. Dynamic resource provisioning through fog micro datacenter. In: IEEE. **Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on**. [S.l.], 2015. p. 105–110. Citado 2 vezes nas páginas 21 e 23.

BAKER, M.; BUYYA, R.; LAFORENZA, D. Grids and grid technologies for wide-area distributed computing. **Softw. Pract. Exper.**, John Wiley & Sons, Inc., New York, NY, USA, v. 32, n. 15, p. 1437–1466, dez. 2002. ISSN 0038-0644. Disponível em: <http://dx.doi.org/10.1002/spe.488>. Citado 2 vezes nas páginas 15 e 11.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: ACM. **Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. [S.l.], 2012. p. 13–16. Citado na página 21.

BORRO, L. C. **Escalonamento em grades móveis: uma abordagem ciente do consumo de energia**. Dissertação (Mestrado) — Universidade de São Paulo, 2013. Citado 17 vezes nas páginas 15, 19, 2, 4, 16, 30, 31, 33, 34, 35, 38, 40, 41, 50, 53, 54 e 79.

BOX, J. F. Guinness, gosset, fisher, and small samples. **Statistical Science**, JSTOR, p. 45–52, 1987. Citado na página 54.

BUKH, P. N. D.; JAIN, R. **The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling**. [S.l.]: JSTOR, 1992. Citado 3 vezes nas páginas 54, 62 e 68.

BUYYA, R.; ABRAMSON, D.; VENUGOPAL, S. The grid economy. **Proceedings of the IEEE**, IEEE, v. 93, n. 3, p. 698–714, 2005. Citado 2 vezes nas páginas 1 e 11.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. **Software Engineering, IEEE Transactions on**, IEEE, v. 14, n. 2, p. 141–154, 1988. Citado 3 vezes nas páginas 15, 16 e 17.

CHANG-QIN, H.; ZHU, Z.-T.; WU, Y.-H.; XIAO, Z.-H. Power-aware hierarchical scheduling with respect to resource intermittence in wireless grids. In: IEEE. **Machine Learning and Cybernetics, 2006 International Conference on**. [S.l.], 2006. p. 693–698. Citado na página 16.

CHUNLIN, L.; LAYUAN, L. Exploiting composition of mobile devices for maximizing user qos under energy constraints in mobile grid. **Information Sciences**, Elsevier, v. 279, p. 654–670, 2014. Citado na página 2.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. **Sistemas Distribuídos-: Conceitos e Projeto**. [S.l.]: Bookman Editora, 2013. Citado 5 vezes nas páginas 5, 6, 7, 8 e 9.

DANTAS, M. A. **Computação distribuída de alto desempenho: redes, clusters e grids computacionais**. [S.l.]: Axcel Books, 2005. Citado 2 vezes nas páginas 15 e 17.

DASTJERDI, A. V.; GUPTA, H.; CALHEIROS, R. N.; GHOSH, S. K.; BUYYA, R. Fog computing: Principals, architectures, and applications. **arXiv preprint arXiv:1601.02752**, 2016. Citado 4 vezes nas páginas 2, 21, 22 e 23.

DEV, D.; BAISHNAB, K. A review and research towards mobile cloud computing. In: **Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on**. [S.l.: s.n.], 2014. p. 252–256. Citado 2 vezes nas páginas 18 e 19.

DIKAIAKOS, M.; KATSAROS, D.; MEHRA, P.; PALLIS, G.; VAKALI, A. Cloud computing: Distributed internet computing for it and scientific research. **Internet Computing, IEEE**, v. 13, n. 5, p. 10–13, Sept 2009. ISSN 1089-7801. Citado na página 1.

DINH, H. T.; LEE, C.; NIYATO, D.; WANG, P. A survey of mobile cloud computing: architecture, applications, and approaches. **Wireless communications and mobile computing**, Wiley Online Library, v. 13, n. 18, p. 1587–1611, 2013. Citado 5 vezes nas páginas 15, 18, 19, 20 e 21.

DUAN, L.; KUBO, T.; SUGIYAMA, K.; HUANG, J.; HASEGAWA, T.; WALRAND, J. Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing. In: IEEE. **INFOCOM, 2012 Proceedings IEEE**. [S.l.], 2012. p. 1701–1709. Citado na página 18.

ELGAZZAR, K.; MARTIN, P.; HASSANEIN, H. Cloud-assisted computation offloading to support mobile services. In: IEEE. [S.l.], 2014. PP, n. 99, p. 1–1. ISSN 2168-7161. Citado 7 vezes nas páginas 15, 13, 19, 20, 28, 29 e 31.

ESTRELLA, J. C. **WSARCH: Uma arquitetura para a provisão de web services com qualidade de serviço**. Tese (Doutorado) — Universidade de São Paulo, 2010. Citado na página 43.

FISHWICK, P. A. Simpack: getting started with simulation programming in c and c++. In: ACM. **Proceedings of the 24th conference on Winter simulation**. [S.l.], 1992. p. 154–162. Citado 2 vezes nas páginas 3 e 46.

FOSTER, I.; KESSELMAN, C. The globus toolkit. **The grid: blueprint for a new computing infrastructure**, p. 259–278, 1999. Citado na página 25.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. The anatomy of the grid: Enabling scalable virtual organizations. **International journal of high performance computing applications**, Sage Publications, v. 15, n. 3, p. 200–222, 2001. Citado na página 1.

GHOSH, P.; DAS, S. K. Mobility-aware cost-efficient job scheduling for single-class grid jobs in a generic mobile grid architecture. **Future Generation Computer Systems**, Elsevier, v. 26, n. 8, p. 1356–1367, 2010. Citado 2 vezes nas páginas 15 e 14.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of things (iot): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, Elsevier, v. 29, n. 7, p. 1645–1660, 2013. Citado na página 21.

HASSAN, M.; ZHAO, W.; YANG, J. Provisioning web services from resource constrained mobile devices. In: IEEE. **Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on**. [S.l.], 2010. p. 490–497. Citado 4 vezes nas páginas 13, 26, 27 e 31.

HONG, K.; LILLETHUN, D.; RAMACHANDRAN, U.; OTTENWÄLDER, B.; KOLDEHOFE, B. Mobile fog: A programming model for large-scale applications on the internet of things. In: ACM. **Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing**. [S.l.], 2013. p. 15–20. Citado 5 vezes nas páginas 21, 22, 29, 30 e 31.

HUERTA-CANEPA, G.; LEE, D. A virtual cloud computing provider for mobile devices. In: ACM. **Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond**. [S.l.], 2010. p. 6. Citado 2 vezes nas páginas 26 e 31.

ISAIADIS, S.; GETOV, V. Integrating mobile devices into the grid: Design considerations and evaluation. In: **Euro-Par 2005 Parallel Processing**. [S.l.]: Springer, 2005. p. 1080–1088. Citado 3 vezes nas páginas 25, 26 e 31.

KATSAROS, K.; POLYZOS, G. C. Optimizing operation of a hierarchical campus-wide mobile grid for intermittent wireless connectivity. In: IEEE. **Local & Metropolitan Area Networks, 2007. LANMAN 2007. 15th IEEE Workshop on**. [S.l.], 2007. p. 111–116. Citado 2 vezes nas páginas 15 e 14.

KOTWAL, P. A.; SINGH, A. R. Evolution and effects of mobile cloud computing, middleware services on cloud, future prospects: A peek into the mobile cloud operating systems. In: IEEE. **Computational Intelligence & Computing Research (ICCIC), 2012 IEEE International Conference on**. [S.l.], 2012. p. 1–5. Citado na página 18.

KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing. **Software: Practice and Experience**, Wiley Online Library, v. 32, n. 2, p. 135–164, 2002. Citado 3 vezes nas páginas 15, 12 e 13.

KUROSE, J. F.; ROSS, K. W.; MARQUES, A. S.; ZUCCHI, W. L. **Redes de Computadores ea Internet: uma abordagem top-down**. [S.l.]: Pearson, 2013. Citado 2 vezes nas páginas 50 e 52.

LEWIS, G.; ECHEVERRÍA, S.; SIMANTA, S.; BRADSHAW, B.; ROOT, J. Tactical cloudlets: Moving cloud computing to the edge. In: IEEE. **Military Communications Conference (MILCOM), 2014 IEEE**. [S.l.], 2014. p. 1440–1446. Citado 2 vezes nas páginas 21 e 23.

LI, G.; SUN, H.; GAO, H.; YU, H.; CAI, Y. A survey on wireless grids and clouds. In: IEEE. **Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on**. [S.l.], 2009. p. 261–267. Citado na página 14.

LITKE, A.; SKOUTAS, D.; TSERPES, K.; VARVARIGOU, T. Efficient task replication and management for adaptive fault tolerance in mobile grid environments. **Future Generation Computer Systems**, Elsevier, v. 23, n. 2, p. 163–178, 2007. Citado 3 vezes nas páginas 42, 57 e 59.

MA, Y.; GONG, B.; SUGIHARA, R.; GUPTA, R. Energy-efficient deadline scheduling for heterogeneous systems. **Journal of Parallel and Distributed Computing**, Elsevier, v. 72, n. 12, p. 1725–1740, 2012. Citado na página 18.

MADSEN, H.; ALBEANU, G.; BURTSCHY, B.; POPENTIU-VLADICESCU, F. Reliability in the utility computing era: Towards reliable fog computing. In: IEEE. **Systems, Signals and Image Processing (IWSSIP), 2013 20th International Conference on**. [S.l.], 2013. p. 43–46. Citado 2 vezes nas páginas 21 e 23.

MORSY, H.; EL-REWINI, H. Adaptive scheduling in a mobile ad-hoc grid for time-sensitive computing. In: IEEE. **Computer Systems and Applications (AICCSA), 2013 ACS International Conference on**. [S.l.], 2013. p. 1–8. Citado 5 vezes nas páginas 15, 13, 27, 28 e 31.

PACHECO, P. **An introduction to parallel programming**. [S.l.]: Elsevier, 2011. Citado na página 6.

PARK, S.-H.; LEE, T.-G.; SEO, H.-S.; KWON, S.-J.; HAN, J.-H. An election protocol in mobile ad hoc distributed systems. In: **Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on**. [S.l.: s.n.], 2009. p. 628–633. Citado na página 10.

RODRíGUEZ, J.; MATEOS, C.; ZUNINO, A. Are smartphones really useful for scientific computing? In: CIPOLLA-FICARRA, F.; VELTMAN, K.; VERBER, D.; CIPOLLA-FICARRA, M.; KAMMüLLER, F. (Ed.). **Advances in New Technologies, Interactive Interfaces and Communicability**. Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7547). p. 38–47. ISBN 978-3-642-34009-3. Disponível em: <http://dx.doi.org/10.1007/978-3-642-34010-9_4>. Citado 3 vezes nas páginas 2, 14 e 17.

ROSADO, D. G.; FERNÁNDEZ-MEDINA, E.; LÓPEZ, J.; PIATTINI, M. Systematic design of secure mobile grid systems. **Journal of Network and Computer Applications**, Elsevier, v. 34, n. 4, p. 1168–1183, 2011. Citado na página 17.

RUXTON, G. D. The unequal variance t-test is an underused alternative to student's t-test and the mann–whitney u test. **Behavioral Ecology**, ISBE, v. 17, n. 4, p. 688–690, 2006. Citado na página 54.

SHAH, S. C. Mobile ad hoc computational grid: Opportunities and challenges. In: IEEE. **Military Communications Conference, MILCOM 2013-2013 IEEE**. [S.l.], 2013. p. 848–857. Citado 4 vezes nas páginas 15, 16, 17 e 18.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, JSTOR, v. 52, n. 3/4, p. 591–611, 1965. Citado na página 54.

SHARMA, R.; KUMAR, S.; TRIVEDI, M. C. Mobile cloud computing: A needed shift from cloud to mobile cloud. In: IEEE. **Computational Intelligence and Communication Networks (CICN), 2013 5th International Conference on**. [S.l.], 2013. p. 536–539. Citado 2 vezes nas páginas 18 e 20.

STOJMENOVIC, I.; WEN, S. The fog computing paradigm: Scenarios and security issues. In: IEEE. **Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on**. [S.l.], 2014. p. 1–8. Citado 4 vezes nas páginas 15, 21, 22 e 23.

TANENBAUM, A.; STEEN, M. V. **Distributed systems**. [S.l.]: Pearson Prentice Hall, 2007. Citado 4 vezes nas páginas 5, 6, 10 e 40.

THENMOZHI, S.; TAMILARASI, A. A cluster based resource allocation architecture for mobile grid environments. In: IEEE. **Computing Communication and Networking Technologies (ICCCNT), 2010 International Conference on**. [S.l.], 2010. p. 1–5. Citado na página 16.

VASUDEVAN, S.; KUROSE, J.; TOWSLEY, D. Design and analysis of a leader election algorithm for mobile ad hoc networks. In: **Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on**. [S.l.: s.n.], 2004. p. 350–360. ISSN 1092-1648. Citado na página 10.

ZENG, W.-Y.; ZHAO, Y.-L.; ZENG, J.-W.; SONG, W. Mobile grid architecture design and application. In: IEEE. **Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on**. [S.l.], 2008. p. 1–4. Citado 2 vezes nas páginas 17 e 18.

ZHANG, W.; WEN, Y.; WU, D. O. Energy-efficient scheduling policy for collaborative execution in mobile cloud computing. In: IEEE. **INFOCOM, 2013 Proceedings IEEE**. [S.l.], 2013. p. 190–194. Citado 4 vezes nas páginas 13, 20, 51 e 52.

APÊNDICE

A

TESTE DE NORMALIDADE PARA OS CENÁRIOS 1, 2 E 3

Tabela 14 – Tabelas com p_valores para Cenário 1 - Parte 1

Experimento	TRM-1	TRM-2	TRM-3	TRM-4	TRM-5
ε = 0, K = 1	0.01569	0.0007295	0.002473	0.01328	0.09987
ε = 0, K = n/2	0.01708	0.009148	0.0003223	0.001445	0.004823
ε = 0, K = n	0.0449	0.002842	0.003533	0.0314	0.006618
ε = m/2, K = 1	0.1165	0.001725	0.007105	0.001217	0.07902
ε = m/2, K = n/2	0.004262	0.003179	0.01689	0.007712	0.001404
ε = m/2, K = n	0.06042	0.00237	0.01769	0.01163	0.04873
ε = m, K = 1	0.4386	0.000791	0.00126	0.02678	0.0153
ε = m, K = n/2	0.0001705	0.0002645	0.009258	0.07117	0.000667
ε = m, K = n	0.1679	8.52E-05	0.000333	0.01189	0.05103

Tabela 15 – Tabelas com p_valores para Cenário 1 - Parte 2

Experimento	TCM-1	TCM-2	TCM-3	TCM-4	TCM-5	ETD
ε = 0, K = 1	5.26E-06	8.34E-05	5.23E-06	0.0002147	3.84E-05	0.2194
ε = 0, K = n/2	0.0004923	8.48E-05	0.0006192	7.24E-05	6.39E-06	0.03082
ε = 0, K = n	1.81E-05	0.0004133	0.0005537	0.0001677	0.0004054	7.98E-06
ε = m/2, K = 1	0.0001788	0.0004723	1.97E-06	8.88E-05	4.05E-05	0.02394
ε = m/2, K = n/2	0.001521	3.87E-06	0.001519	0.0003703	0.0016	0.0978
ε = m/2, K = n	0.000104	4.64E-06	0.0001363	0.0001069	1.37E-05	0.003453
ε = m, K = 1	2.98E-05	8.88E-05	0.0001495	0.0001971	0.000126	0.08134
ε = m, K = n/2	6.31E-05	0.0008319	0.001136	0.0003246	0.0002452	0.007427
ε = m, K = n	0.0005161	0.0002068	1.36E-06	6.11E-06	1.41E-06	0.007874

Tabela 16 – Tabelas com p_valores para Cenário 2 - Parte 1

Experimento	TRM-1	TRM-2	TRM-3	TRM-4	TRM-5
ε = 0, K = 1	0.4065	0.0143	8.21E-02	0.006076	0.2181
ε = 0, K = n/2	0.3911	0.2916	0.04503	0.4347	0.09129
ε = 0, K = n	0.0009114	0.003801	3.67E-06	1.35E-03	0.0006571
ε = m/2, K = 1	0.441	0.00616	0.05495	0.9088	0.0121
ε = m/2, K = n/2	0.2209	0.7833	4.51E-01	3.36E-01	0.3597
ε = m/2, K = n	2.47E-01	0.2562	2.43E-02	0.5456	4.39E-01
ε = m, K = 1	0.0002252	0.006119	0.214	0.0007304	0.7974
ε = m, K = n/2	0.03919	4.31E-01	4.12E-01	3.20E-01	0.3201
ε = m, K = n	0.1509	1.29E-01	8.24E-02	3.69E-01	1.19E-01

Tabela 17 – Tabelas com p_valores para Cenário 2 - Parte 2

Experimento	TCM-1	TCM-2	TCM-3	TCM-4	TCM-5	ETD
ε = 0, K = 1	4.35E-01	3.89E-02	3.27E-01	0.5197	7.64E-01	1.29E-06
ε = 0, K = n/2	0.01498	2.64E-02	0.1489	0.0019	1.17E-01	6.59E-07
ε = 0, K = n	2.74E-02	0.01333	0.1196	0.6077	0.002068	1.20E-06
ε = m/2, K = 1	4.92E-01	7.24E-02	1.71E-01	7.04E-01	6.11E-02	0.01379
ε = m/2, K = n/2	0.01097	2.94E-01	0.07715	0.8986	0.021	1.27E-05
ε = m/2, K = n	0.6076	7.95E-01	0.7013	0.4078	9.74E-01	3.58E-07
ε = m, K = 1	2.24E-01	2.90E-01	0.3108	0.7489	0.3391	0.0002102
ε = m, K = n/2	8.13E-02	0.0009128	0.1073	0.3335	0.3436	2.20E-05
ε = m, K = n	0.4307	0.9542	3.60E-01	6.14E-01	8.18E-01	0.001355

Tabela 18 – Tabelas com p_valores para Cenário 3 - Parte 1

Experimento	TRM-1	TRM-2	TRM-3	TRM-4	TRM-5
ε = 0, K = 1	0.01942	0.07386	8.06E-06	0.0006937	0.03211
ε = 0, K = n/2	0.3526	0.000172	0.001097	0.0003712	0.751
ε = 0, K = n	0.1653	0.00437	0.0009183	3.18E-07	0.05903
ε = m/2, K = 1	0.2398	0.111	0.0001702	0.00103	0.03435
ε = m/2, K = n/2	0.08489	0.000145	7.19E-05	2.95E-06	0.6517
ε = m/2, K = n	1.18E-08	0.0001615	5.03E-05	0.004155	7.79E-05
ε = m, K = 1	0.4748	0.05578	0.0001312	0.005621	0.07562
ε = m, K = n/2	0.0001236	3.03E-06	2.88E-09	6.37E-06	0.009317
ε = m, K = n	0.001809	8.46E-04	2.86E-07	1.66E-09	8.65E-06

Tabela 19 – Tabelas com p_valores para Cenário 3 - Parte 2

Experimento	**TCM-1**	**TCM-2**	**TCM-3**	**TCM-4**	**TCM-5**	**ETD**
ε = 0, K = 1	4.30E-06	2.68E-05	4.16E-03	0.0001228	9.96E-01	0.0002466
ε = 0, K = n/2	0.0005017	1.41E-03	0.005827	0.4843	4.58E-01	3.18E-05
ε = 0, K = n	4.58E-01	0.1275	0.1363	0.08436	0.6528	2.85E-03
ε = m/2, K = 1	7.18E-06	7.18E-06	3.25E-04	2.19E-05	2.84E-01	0.002055
ε = m/2, K = n/2	0.2728	1.06E-02	0.02759	0.311	0.4733	0.008732
ε = m/2, K = n	0.1455	5.52E-02	0.9226	0.09541	6.44E-07	0.08623
ε = m, K = 1	5.09E-06	2.03E-05	0.0005383	0.0001782	0.1586	0.00045
ε = m, K = n/2	5.97E-02	0.04958	0.08902	0.7625	0.3371	0.0004993
ε = m, K = n	0.8268	0.7909	1.13E-01	4.54E-01	1.27E-07	0.1031