Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Ciência da Computação

Política Adaptativa de Gerenciamento Dinâmico de Energia Baseada em *Timeout* para Interfaces de Rede Sem Fio

Fabiano de Miranda Silva

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Hyggo Oliveira de Almeida
(Orientador)
Angelo Perkusich
(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S586p Silva, Fabiano de Miranda.

Política adaptativa de gerenciamento dinâmico de energia baseada em *Timeout* para interfaces de rede sem fio / Fabiano de Miranda Silva. – Campina Grande, 2015.

64 f.: il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2015.

"Orientação: Prof. Dr. Hyggo Oliveira de Almeida e Prof. Dr. Angelo Perkusich".

Referências.

1. Gerenciamento de Energia. 2. Interfaces de Rede. 3. Dispositivos Móveis. I. Almeida, Hyggo Oliveira de. II. Perkusich, Angelo. III. Título.

CDU 004.41 (043.3)

Resumo

As interfaces de rede estão presentes em uma variedade de sistemas computacionais alimentados à bateria, tais como *notebooks*, *ultrabooks*, *tablets* e *smartphones*. Considerando-se que a vida útil da bateria é uma limitação, há oportunidades para o gerenciamento de energia reduzindo-se o consumo de energia de interfaces de rede ociosas. Neste trabalho, uma política adaptativa de gerenciamento de energia baseada em *timeout* para interfaces de rede é apresentada, incorporando a estimativa da carga de trabalho e um problema de otimização em tempo de execução, proporcionando economia de energia e desempenho adequado.

Abstract

Network interfaces are present in a variety of battery powered computer systems, such as notebooks, ultrabooks, tablets and mobile phones. Considering that the battery life is still a constraint, there are opportunities for power management by reducing the power consumption of idle network interfaces. In this work, a dynamic timeout power policy for network interfaces is presented, incorporating workload estimation and a problem optimization at runtime, thus providing power savings and suitable performance.

Agradecimentos

Agradeço primeiramente a Deus por tudo.

Agradeço aos meus orientadores Hyggo e Angelo. Ao meu co-orientador Saulo, cuja contribuição foi fundamental para o desenvolvimento deste trabalho.

Agradeço aos meus pais Claudete e Espedito, aos meus irmãos Fabiana e Flaviano, e a Sara pelo apoio durante toda a caminhada.

Conteúdo

Lista de Símbolos

API - *Application Programming Interface*

DPM - *Dynamic Power Management*

DVFS - *Dynamic Voltage Frequency Scaling*

DVS - *Dynamic Voltage Scaling*

GDE - *Gerenciamento Dinâmico de Energia*

GQM - *Goal Question Metrics*

L - *Restrição de Penalidade de Desempenho*

TD - *Timeout Dinâmico*

TF - *Timeout Fixo*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

1.1 A Necessidade do Gerenciamento de Energia em Dispositivos Móveis

Dispositivos móveis em geral como *notebooks*, *smartphones*, *tablets*, entre outros, são dependentes de bateria. Com a evolução tecnológica, o desempenho dos seus componentes internos tende a aumentar (capacidade de processamento, memória, etc.), aumentando também o seu consumo de energia. Em contrapartida, a densidade de energia das baterias não tem aumentado o suficiente para manter o desempenho dos dispositivos por um período prolongado [3], tornando o seu tempo de utilização um fator limitador. Do ponto de vista do usuário, é comum só perceber que há pouca carga na bateria de forma tardia, tornando a recarga do aparelho uma tarefa urgente.

Sem a possibilidade de aumentar a densidade de energia nas baterias em curto prazo, são realizados esforços para aumentar a eficiência energética dos dispositivos a fim de evitar consumo desnecessário, e com este objetivo são desenvolvidas técnicas aplicadas tanto no projeto do produto quanto durante sua utilização [2]. As técnicas aplicadas durante a utilização dos dispositivos são chamadas Políticas de Gerenciamento Dinâmico de Energia (GDE) [2], que desativam componentes ociosos ou os colocam em estado de baixo consumo.

O processo de desativação e reativação dos componentes é um efeito colateral das políticas de GDE, pois se realizado excessivamente pode causar aumento no consumo de energia, além de que o seu tempo de reativação pode ser perceptível ao usuário. Portanto, as políti-

cas de GDE buscam economizar o máximo de energia com uma penalidade de desempenho aceitável [2].

As interfaces de rede estão presentes em quase todos os dispositivos móveis e são utilizadas por muitos de seus aplicativos. Os tipos de uso são diversos, como por exemplo, para localização, comunicação, acesso a dados e demais serviços na nuvem. Portanto, o tráfego de dados aumentou consideravelmente em interfaces de rede de dispositivos móveis. Para os usuários que acessam a Internet diariamente, as interfaces de rede são geralmente deixadas ligadas, consumindo energia desnecessariamente quando ociosas. Assim, há ainda muitas oportunidades não exploradas para reduzir o consumo de energia neste tipo de componente.

Políticas de gerenciamento de energia com base em *timeout* constante são comuns em dispositivos móveis devido à sua simplicidade. Estas políticas podem ser encontradas nas opções de gerenciamento de energia em sistemas operacionais na forma de intervalo de tempo para desligar a tela, por exemplo. Tais políticas são baseadas na identificação da ociosidade do componente por um intervalo de tempo maior do que o *timeout* selecionado. Neste caso, o componente é considerado ocioso e colocado em um estado de baixo consumo.

Luiz et al. [12] apresentaram uma metodologia para a seleção de um *timeout* por meio de um problema de otimização. Uma política baseada em *timeout* constante para finalidades gerais foi projetada usando esta metodologia. Há muitas possibilidades de utilização da Internet, tais como navegação, ouvir músicas, assistir vídeos e jogar jogos. Assim, há muitos tipos de usuários e, portanto, muitos tipos de cargas de trabalho. Considerando esta diversidade, a política proposta por Luiz et al. [12] pode apresentar um melhor desempenho e economia de energia para determinados usuários, mas um desempenho e economia de energia reduzidos para usuários específicos, como aqueles que geram cargas de trabalho intensas. Neste trabalho, uma política de gerenciamento de energia com base no trabalho de Luiz et al. [12] é proposta, incorporando a estimativa da carga de trabalho e problema de otimização em tempo de execução, proporcionando menor consumo de energia e desempenho aceitável para um grupo mais amplo de usuários.

1.2 Objetivos

Este trabalho descreve a elaboração de uma Política de Gerenciamento Dinâmico de Energia para interfaces de rede que leva em consideração a relação entre economia de energia e perda de desempenho, identificando a ociosidade da interface de rede de acordo com o tipo de utilização da mesma.

A política de gerenciamento de energia proposta é baseada em *timeout*. Ou seja, a interface de rede é desligada (ou colocada em estado de baixo consumo, caso haja esta opção) quando identificada como ociosa por um intervalo de tempo maior que o valor de *timeout* adotado. O *timeout* escolhido deve ser tal que haja máxima economia de energia, respeitando uma restrição de penalidade de desempenho L [12].

A escolha do valor de timeout é feita em tempo de execução, ou seja, a política é adaptativa. Desta forma, o *timeout* calculado é o mais adequado para o tipo de utilização do momento. Uma das dificuldades encontradas no cálculo do *timeout* em tempo de execução é encontrar um valor adequado em cargas de trabalho com característica não-estacionária.

1.3 Relevância

Com o desenvolvimento deste trabalho, as principais contribuições são apresentar uma política de GDE baseada em *timeout* que:

- Seja adaptativa, se adequando ao perfil de utilização da interface de rede no momento;
- Permita sua utilização em dispositivos com relativas baixa capacidade de processamento e pouca memória disponível;
- Tenha desempenho satisfatório (relacionado à economia de energia e penalidades de desempenho) em cargas de trabalho reais, não-estacionárias.

1.4 Estrutura da Dissertação

Os próximos capítulos da dissertação estão organizados da segunte forma: o Capítulo 2 contém um levantamento do estado da arte em gerenciamento dinâmico de energia, classificando

as políticas apresentadas quanto ao seu tipo; o Capítulo 3 apresenta a política proposta neste trabalho, citando seus módulos príncipais e descrevendo seu funcionamento; o Capítulo 4 descreve o estudo da eficácia do método de classificação que utiliza regressão logística; o Capítulo 5 contém o planejamento e execução de um experimento comparativo na forma de simulação entre a política proposta neste trabalho e sua versão com timeout fixo; o Capítulo 6 descreve o planejamento e execução de outro experimento comparando as mesmas políticas de GDE do capítulo anterior, com o diferencial de que as políticas estão implementadas em notebooks e as medições de consumos são reais; por fim, o sétimo capítulo discute os resultados obtidos nos experimentos dos capítulos anteriores, elucidando a diferença no desempenho das duas políticas e suas limitações, bem como a possibilidade de evolução do trabalho.

Capítulo 2

Fundamentação Teórica

Técnicas de redução do consumo de energia são classificadas como Estáticas e Dinâmicas. As técnicas estáticas, como escolha de materiais que possuem como característica o baixo consumo energético, e técnicas de compilação otimizada para garantir baixo consumo, são aplicadas durante o desenvolvimento do produto. De forma complementar, as técnicas dinâmicas (também conhecidas como políticas de Gerenciamento Dinâmico de Energia - GDE) são aplicadas em tempo de execução, reduzindo o consumo de energia quando componentes estão ociosos ou com pouca utilização [11].

As políticas de GDE utilizam diferentes estratégias para economizar energia. Dynamic Voltage Scaling (DVS) [11] é uma técnica que permite aumentar ou diminuir a tensão de um componente de acordo com determinadas circunstâncias. Derivadas do conceito de DVS, políticas baseadas em Dynamic Voltage and Frequency Scaling (DVFS), tais como as citadas em Devadas et al. [5] por exemplo, alteram a tensão, e por sua vez a frequência do núcleo do processador dependendo da sua carga de trabalho, diminuindo o seu consumo.

Políticas que tratam de outros componentes que não sejam o processador utilizam outras abordagens, e em sua maioria podem ser subdivididas em três categorias:

- Preditivas;
- Estocásticas;
- Baseadas em timeout.

2.1 Gerenciamento Dinâmico de Energia

2.1.1 Políticas Preditivas

As políticas de GDE preditivas tentam identificar o momento em que o componente ficará em estado ocioso. Baseado num histórico criado dinamicamente, a política faz uma predição de quando o componente entrará num estado com grande probabilidade de ficar ocioso, e sendo esta condição verdadeira, o gerenciador de energia aplica o procedimento para redução do consumo naquele componente.

Huang et al. [7] propuzeram uma política de predição adaptativa que utiliza um modelo autoregressivo (AR) e a aplicou em discos rígidos. Já Yue et al. [16] priorizaram o seu trabalho no aumento da precisão e da acurácia das predições quando há mudança de tipo de utilização da CPU, conseguindo uma economia de até 40% com relação a outras políticas preditivas.

2.1.2 Políticas Estocásticas

Políticas estocásticas modelam as mudanças de estados dos componentes e chegada de requisições como processos estocásticos, tais como processos de Markov. Com a modelagem, o problema de diminuir o consumo de energia passa a ser tratado como um problema de otimização estocástico.

Zafer et al. [17] propuzeram uma política de GDE estocástica para interfaces de rede sem fio, onde o problema de otimização trata de encontrar a taxa de transferência de dados ótima (mais econômica), dado que os seguintes requisitos sejam verdadeiros:

- A quantidade de dados que serão transmitidos é conhecida;
- Há um tempo limite para a transmissão dos dados;
- A transmissão é sem fio;
- O canal de transmissão possui qualidade variável.

A taxa de transferência ótima T_o é dada por:

$$T_o = R \times U \tag{2.1}$$

Onde R é a quantidade de dados restante e U é a urgência da transmissão. A quantidade restante de dados na transferência é um parâmetro conhecido, e o problema constitui-se em encontrar o valor que corresponde à urgência da transmissão.

2.1.3 Políticas Baseadas em *Timeout*

Políticas de GDE baseadas em *timeout* são as mais comuns em dispositivos móveis devido à sua simplicidade. Estas políticas podem ser encontradas na forma de intervalos de tempo previstos para esmaecimento da tela, desligamento da tela, entrar em *stand-by* ou em modo de hibernação, dentre outras opções. O funcionamento destas políticas baseia-se em identificar a ociosidade do componente por um intervalo de tempo maior que o valor de *timeout* estabelecido. Caso isto aconteça, o componente é considerado ocioso e colocado em modo econômico.

Na política de *timeout* fixo, o cálculo do *timeout* é feito previamente através de experimentos, e o valor obtido é um valor médio para diferentes tipos de usuário. Enquanto alguns tipos de usuário são bem favorecidos pela política, outros podem ter um baixo aproveitamento, de forma que a utilização da política chega a se tornar inviável para alguns usuários devido à penalidade de desempenho causada pela desativação e reativação do componente.

Criadas para apresentar melhor desempenho do que as políticas de *timeout* fixo, as políticas de *timeout* variável, também conhecidas como políticas adaptativas baseadas em *timeout*, calculam em tempo de execução o valor mais adequado de *timeout* de acordo com o tipo de utilização do dispositivo, favorecendo diferentes tipos de usuários, e evitando altas penalidades de desempenho.

Apesar de terem natureza mais simples, políticas baseadas em *timeout* podem apresentar desempenho equivalente às políticas estocásticas em termos de *tradeoff* entre economia de energia e penalidade de desempenho [9].

Wang et al. [15] propuzeram uma política de *timeout* combinando as vantagens de *timeout* fixo e variável e a aplicaram em discos rígidos. Embora seja mencionado que sua política poderia ser utilizada em outros componentes como interface de rede, seu estudo levou em

consideração apenas duas cargas de trabalho conhecidas, do tipo *cluster* e do tipo *bursty*, deixando desconhecido o desempenho da política em cargas de trabalho não-estacionárias. Kveton et al. [10] aplica aprendizado de máquina (*machine learning*) num algoritmo adaptativo baseado em *timeout*, e exibe resultados experimentais da implementação da política.

Xi et al. [9] fizeram uma otimização adaptativa de uma política de GDE baseada em *timeout* para interfaces de rede, utilizando controle de processos semi-markovianos.

2.1.4 Políticas Mistas

Apesar de haver uma classificação dos tipos de política de GDE, há a possibilidade de combinar abordagens de tipos diferentes e obter como resultado uma política mista. Recentemente alguns trabalhos apresentados possuem esta característica.

Devadas et al. [5] observaram que há uma relação entre DVFS e DPM, e apresentou uma política que tenta aumentar a economia de energia levando em consideração a interação entre a frequência do processador e o estado de alguns componentes. A política proposta tem como limitação o fato de que a alteração de estado dos componentes se dá na escala de milissegundos, e nem todos os componentes de um dispositivo móvel podem ter seu estado alterado nesta frequência.

Park et al. [13] apresentaram uma política adaptativa que também trata da interação entre DVFS e DPM e é voltada para dispositivos multimídia. No seu experimento, a carga de trabalho simulada foi a execução de arquivos MP3, e é proposto para trabalhos futuros melhorar a sua política para utilização em condições de carga de trabalho não-determinística.

Cai et al. [4] criaram um esquema de gerenciamento de energia para interfaces de rede baseado no modelo Produtor-Consumidor, tratando dados como um produto que pode ser armazenado num *buffer*. Com o *buffer* cheio, o dispositivo de transmissão é ativado e os dados são transmitidos (consumidos no modelo), permitindo que o dispositivo passe longos períodos de tempo com a interface de rede em modo econômico. Um requisito para esta política é que a transferência de dados não precisa ser imediata, e o estudo de caso utilizado foi um dispositivo que coleta amostras de ar e transmite os dados para uma base conectada por uma rede sem fio.

Pettis et al. [14] modelaram o problema de forma semelhante a Cai et al. [4], utilizando o esquema produtor-consumidor, mas sua solução é voltada para *streaming* de dados. Dhiman

et al. [6] utilizaram políticas de GDE num algoritmo com aprendizado que seleciona a politica adequada para o momento. É feito um experimento com as politicas aplicadas no disco rígido e na interface de rede. Das políticas utilizadas, há uma de *timeout* fixo e outra de *timeout* adaptativo. No experimento, a carga de trabalho utilizada corresponde a um período de cerca de 5000 segundos.

Capítulo 3

Modelagem de Política de GDE Baseada em *Timeout* para Interfaces de Rede Sem Fio

3.1 Aspectos Gerais

Neste trabalho, uma interface de rede foi modelada com duas características: o seu estado (ligado ou em baixo consumo), e a sua atividade (ocioso ou ativo). Quando o seu estado atual é "ligado", os dois tipos de uso são possíveis, mas quando o seu estado atual é "baixo consumo", assume-se que a interface de rede esteja ociosa, pois para que se torne ativa é necessário o seu religamento. A associação entre os possíveis valores de estado e atividade das interfaces de rede, conforme descrito neste parágrafo, está ilustrada na tabela 3.1.

Estado	Atividade
Ligado	Ocioso
Ligado	Ativo
Baixo Consumo	Ocioso

Tabela 3.1: Possíveis combinações de valores de estado e atividade das interfaces de rede.

Para escolha do valor do *timeout* δ, uma política de GDE baseada em *timeout* para interfaces de rede deve levar em consideração a potência consumida pela interface tanto no estado

ligado quanto no estado de baixo consumo. Durante o seu funcionamento, caso a interface de rede fique ociosa por um intervalo i, onde $i < \delta$, nada acontece e o consumo da interface de rede continua o mesmo, conforme observado na Figura 3.1.

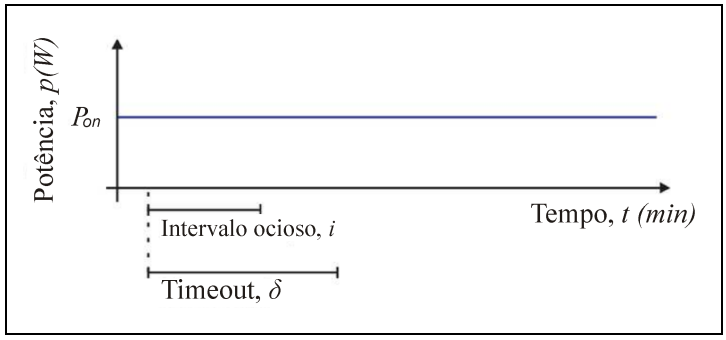

Figura 3.1: Consumo de potência quando o intervalo ocioso t_i é menor que o *timeout* δ escolhido.

Quando a interface de rede ficar ociosa por um intervalo $i \geq \delta$, a mesma tem o seu estado alterado para baixo consumo, e observa-se uma diminuição no consumo de potência. Este comportamento está ilustrado na Figura 3.2. Verifica-se também que durante as transições de estado há oscilações no consumo de potência, e caso o *timeout* seja pequeno demais, estas transições de estado podem causar aumento no consumo de potência, ao invés de economia.

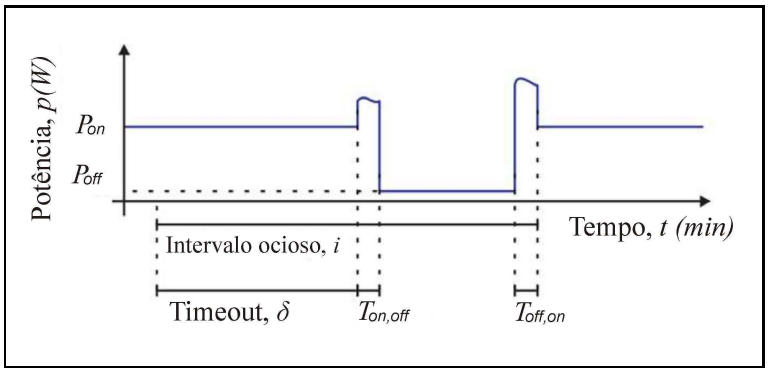

Figura 3.2: Consumo de potência quando o intervalo ocioso t_i é maior ou igual ao *timeout* δ escolhido.

Para medir a atividade da interface de rede quando a mesma se encontra em estado ligado, a política de GDE coleta valores médios de *upload* e *download* periodicamente. Estes dados são avaliados por um módulo classificador e a atividade da interface de rede (ocioso ou ativo)

é enviada para o seu módulo tomador de decisões. Neste módulo, se a interface de rede for considerada ociosa, é verificada a viabilidade de colocá-la em estado de baixo consumo de energia.

A política de *timeout* fixo (TF) é composta por basicamente três módulos. São eles:

- Módulo de comunicação com o Sistema Operacional - coleta os dados de *upload* e *download* periodicamente para que sejam analizados no módulo classificador;
- Módulo classificador - verifica os dados de *upload* e *download* e classifica o nível de atividade da interface de rede, repassando esta informação para o módulo tomador de decisões;
- Módulo tomador de decisões - verifica se a ociosidade da interface de rede continua por um intervalo de tempo maior ou igual que o valor de *timeout*. Em caso positivo, e se a interface de rede estiver ligada, um evento é enviado para o Sistema Operacional solicitando a mudança de estado da interface de rede para o estado de baixo consumo.

A política de *timeout* dinâmico (TD) também possui os três módulos da política TF. Porém, devido à necessidade de atualização do valor do *timeout* em tempo de execução, esta tarefa é implementada num quarto módulo. As seções seguintes descrevem detalhadamente cada módulo da política de TD proposta, e os compara com a implementação na política de TF.

A Figura 3.3 ilustra a arquitetura da política de GDE com os seus módulos internos, os quais são descritos nas subseções seguintes.

3.2 Módulo de Comunicação com o Sistema Operacional

Neste módulo, a coleta de dados enviados e recebidos pela interface de rede é feita de forma periódica, a cada minuto, controlada por um *timer*. Com o auxílio da API do Sistema Operacional, são obtidos valores referentes quantidade de *bytes* recebidos e enviados desde a última obtenção dos dados. Em seguida os dados são tratados para que o módulo classificador receba um valor médio de *download* e *upload* correspondente ao último minuto.

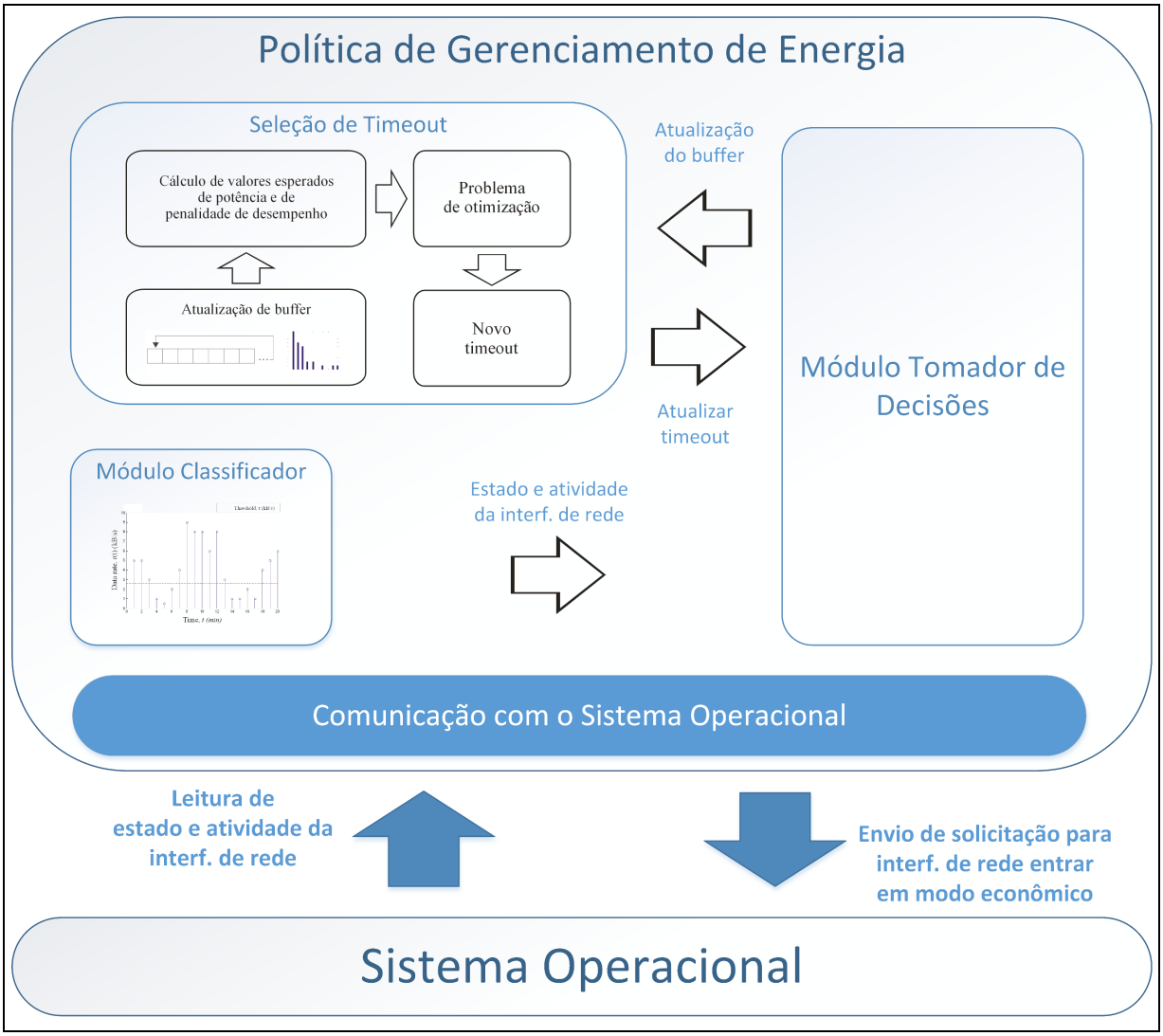

Figura 3.3: Arquitetura da política de GDE.

3.3 Módulo Classificador das Políticas de GDE

Para a classificação da atividade da interface de rede, verifica-se periodicamente as suas taxas médias de *download* e *upload*. Portanto, sejam $\{t_0, t_1, t_2 \in R | t_0 < t_1 < t_2\}$ instantes no tempo, se houve uma verificação v_1 no intervalo $i_1 = [t_0; t_1[$, a verificação seguinte v_2 levará em conta as taxas médias de *download* e *upload* no intervalo $i_2 = [t_1; t_2[$. Os módulos classificadores das políticas de GDE TD e TF diferem entre si e suas características serão detalhadas nas subseções seguintes. O resultado da classificação indica a atividade da interface de rede e seu resultado é enviado para o módulo tomador de decisões da política de GDE.

3.3.1 Módulo Classificador na Política TF

O método apresentado por Luiz et al. [12] para detectar intervalos ociosos foi representado como um problema de classificação, no qual há um fator de entrada caracterizado pela soma das taxas de *upload* e *download* no intervalo t, e como saída têm-se duas classes: ocioso e ativo. A classe positiva para este problema (interface de rede ociosa) pode ser descrita como uma confirmação para a seguinte pergunta: *a interface de rede está ociosa no instante* t*?*

Na política TF, o módulo classificador define a atividade a da interface de rede com base num limiar τ, dada a sua taxa de dados r. Caso a taxa de dados seja menor que o limiar, a interface de rede é considerada ociosa. Caso contrário, a interface de rede é considerada ativa.

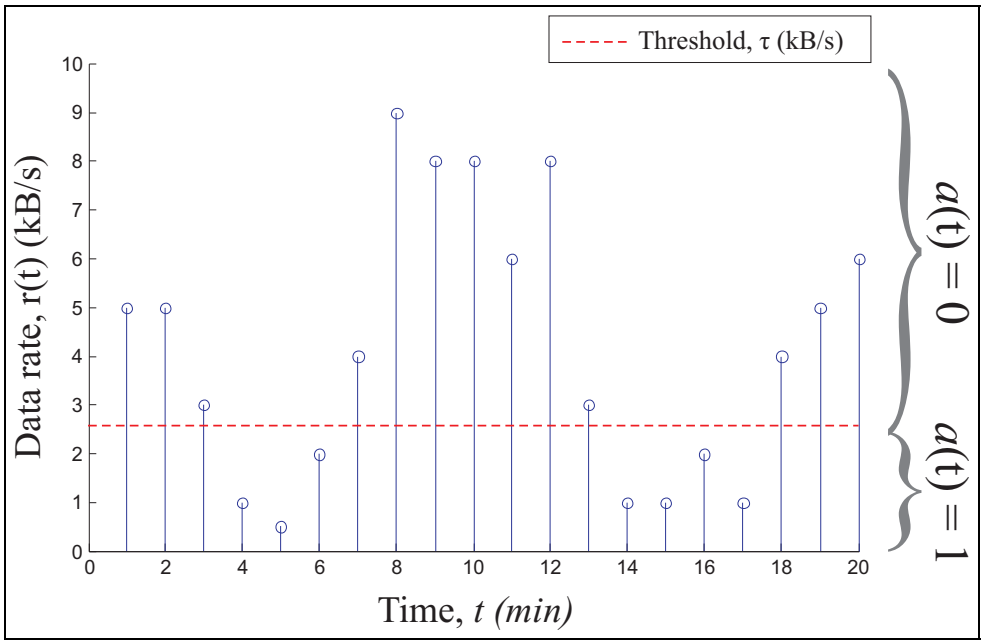

Figura 3.4: Divisão de taxa de $upload + download$ em duas categorias por um limiar.

Portanto, seja $r(t) = d(t) + u(t)$ a taxa de dados da interface de rede no instante t equivalente à soma das taxas de *download* e *upload* no respectivo instante, então a atividade estimada $\hat{a}$ da interface de rede é definida segundo a Equação (3.1).

$$\hat{a}(t|\tau) = \begin{cases} 0, & \text{se } r(t) > \tau \\ 1, & \text{se } r(t) \leq \tau \end{cases} \tag{3.1}$$

O problema de classificação utiliza aprendizagem supervisionada para encontrar o melhor valor de limiar τ, que minimiza a função de custo $J_{t_o,t_f}(\tau)$ no intervalo $[t_o\ t_f]$, conforme descrito na eq. (3.2).

$$J_{t_o,t_f}(\tau) = \sum_{t=t_o}^{t_f} \varepsilon^2(t) = \sum_{t=t_o}^{t_f} \left(a(t) - \hat{a}(t|\tau)\right)^2 \tag{3.2}$$

3.3.2 Módulo Classificador na Política TD

Buscando obter maior êxito em classificar corretamente a atividade da interface de rede, um método alternativo foi implementado no módulo classificador da política TD. Ao invés de utilizar o limiar como fator classificador e levar em consideração taxas de *download* e *upload* com o mesmo peso, utilizou-se uma regressão logística que leva em consideração a influência que estas taxas têm na atividade da interface de rede. Na regressão, é utilizada a função logística também conhecida como função *sigmoid*, representada graficamente na Figura 3.5, que se faz útil na determinação da probabilidade p do dado fornecido ser classificado como sendo de classe positiva.

O estudo da eficácia do método de classificação que utiliza regressão logística está detalhado no Capítulo 4.

Enquanto o método baseado no limiar encontra um valor de limiar que minimiza o erro de estimação, o método baseado na regressão logística encontra os coeficientes da função de estimação $\hat{a}_l$ descrita em (3.3). Sendo θ_d o coeficiente associado à variável *download*, θ_u o coeficiente associado à variável *upload*, θ_o um coeficiente livre, e g a função sigmoid. A função de estimação está relacionada à função sigmoid segundo as equações (3.3) e (3.4). Portanto:

$$\hat{a}_l(t|\theta) = g(\theta_o + \theta_d d(t) + \theta_u u(t)) \tag{3.3}$$

onde

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.4}$$

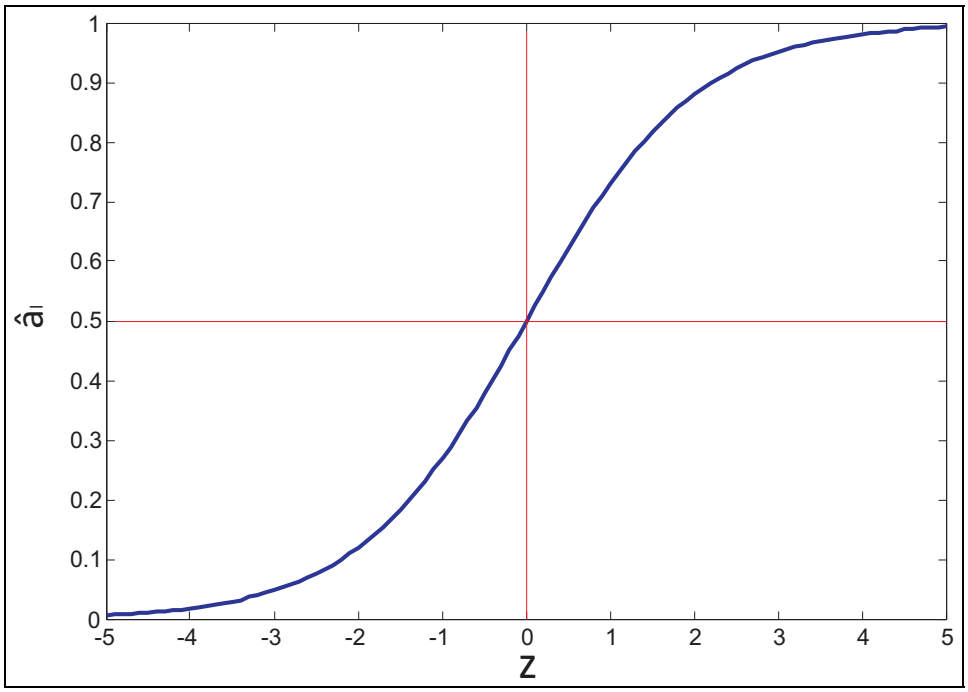

Figura 3.5: Função sigmoid.

3.4 Módulo Tomador de Decisões

Este módulo é constituído de um algoritmo que avalia a ociosidade da interface de rede. Dado um valor de *timeout* armazenado na variável *TIMEOUT* e uma variável *intervaloOcioso* que armazena a duração do intervalo ocioso, a interface de rede é colocada em estado de baixo consumo quando o intervalo ocioso é maior ou igual ao *timeout* escolhido. O pseudocódigo 3.1 descreve este procedimento.

Código Fonte 3.1: Procedimento principal do Módulo Tomador de Decisão (pseudocódigo)

```
1  avaliarMelhorEstadoWiFi(estadoWiFI, atividadeWifi) {
2          SE estadoWifi == "LIGADO" {
3                  SE atividadeWiFi == "OCIOSO" {
4                          intervaloOcioso=intervaloOcioso+1;
5                          SE intervaloOcioso >= TIMEOUT {
6                                  ativarBaixoConsumoWiFi();
7                          }
8                  } SENÃO {
9                          intervaloOcioso = 0;
10                 }
```

```
11              }
12  }
```

3.5 O Módulo de Seleção de *Timeout*

O objetivo deste módulo é calcular em tempo de execução um valor de *timeout* que corresponda ao tipo de utilização da interface de rede. Caso o usuário esteja utilizando pouco a interface de rede, o *timeout* tende a diminuir e ela será colocada em estado de baixo consumo com mais facilidade. Por outro lado, se o usuário estiver utilizando mais efetivamente a interface de rede, o *timeout* tende a aumentar.

O método proposto por Luiz et al. [12] consiste em encontrar o menor valor esperado para a potência média $\bar{p}$, dado um *timeout* δ. O cálculo do valor esperado da potência em função do *timeout* está descrito em (3.5) e (3.6).

$$E[\bar{p}(\delta)] = \sum_{t_i=1}^{\infty} Pr[T_i = t_i]\bar{p}(\delta, t_i) \tag{3.5}$$

$$\begin{aligned} E[\bar{p}(\delta)] &= \sum_{t_i=1}^{\delta-1} Pr[T_i = t_i]P_{on} + \\ &+ \sum_{t_i=\delta}^{\infty} Pr[T_i = t_i]\frac{P_{on}\delta + P_{off}(t_i - \delta - T_s) + E_s}{t_i} \end{aligned} \tag{3.6}$$

onde:

- P_{on} é a potência média do dispositivo móvel com a interface de rede ligada e ociosa;
- P_{off} é a potência média do dispositivo móvel com a interface de rede em modo de baixo consumo;
- E_s é a energia de transição usada entre as mudanças de estado da interface de rede, de ligada para baixo consumo, e de baixo consumo para ligada;
- T_s é o tempo de transição entre as mudanças de estado da interface de rede.

Além disso, precisa-se levar em conta uma penalidade de desempenho $l(\delta, t_i)$, pois o usuário pode ficar insatisfeito se houver interesse em utilizar a interface de rede e a mesma

estiver em estado de baixo consumo. Neste caso, há uma penalidade de desempenho, pois o usuário irá aguardar a interface de rede ser religada durante o intervalo de transição $T_{off/on}$. A probabilidade deste evento é $Pr[T_i \geq \delta]$, e a penalidade de desempenho esperada durante os intervalos ociosos é mostrada em (3.7).

$$E[l(\delta)] = Pr[T_i \geq \delta] = \sum_{t_i=\delta}^{\infty} Pr[T_i = t_i] \tag{3.7}$$

Portanto, o problema de otimização (3.8) consiste em escolher um *timeout* δ tal que minimize o valor esperado da potência média, obedecendo um limite L para a penalidade de desempenho chamado restrição de penalidade de desempenho.

$$\begin{gathered} \min_{\delta} E[\bar{p}(\delta)] \\ \textit{tal que } E[l(\delta)] < L \end{gathered} \tag{3.8}$$

O cálculo do valor de *timeout* requer alguns passos e alguns cálculos estatísticos. Sabendo que é necessário causar o mínimo de impacto no consumo de energia da política de GDE, este módulo foi implementado visando o baixo consumo de recursos computacionais.

A Figura 3.6 ilustra o passo-a-passo para se obter um novo valor de *timeout*. Cada uma das etapas ilustradas está descrita nos tópicos seguintes.

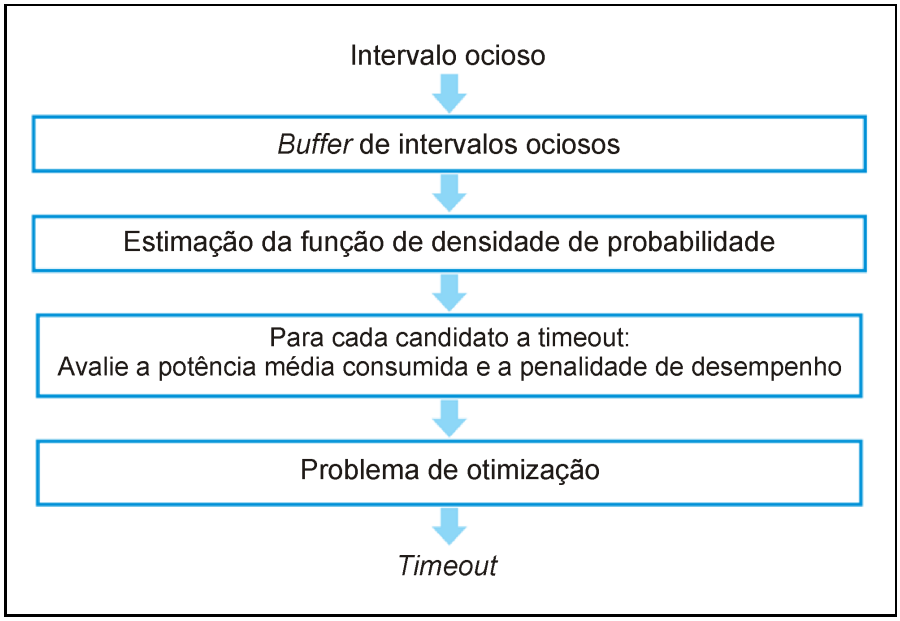

Figura 3.6: Etapas para seleção de um *timeout*.

3.5.1 Detecção de Intervalo Ocioso Completo

A escolha de um novo valor de *timeout* é feita sempre que um novo intervalo ocioso completo é detectado. Um intervalo ocioso completo corresponde a uma sequência de períodos de amostragem consecutivos, os quais terminam no início de um intervalo ativo. Após a detecção de um intervalo ocioso completo, este dado é armazenado num *buffer* de intervalos ociosos recentes.

3.5.2 *Buffer* de Intervalos Ociosos Recentes

Para armazenagem dos intervalos ociosos mais recentes sem a necessidade de alocação dinâmica de memória, foi criada uma lista circular de tamanho fixo, a qual foi denominada *buffer* de intervalos ociosos. Neste *buffer*, os intervalos ociosos completos são armazenados e assim que a lista é completamente preenchida, atinge-se a meta de quantidade mínima de valores necessários para se calcular um novo valor de *timeout*. Os intervalos ociosos seguintes são armazenados nas posições correspondentes aos valores mais antigos da lista, mantendo a característica principal do conjunto de dados que é a de representarem um comportamento recente do usuário.

A importância de efetuar o cálculo do *timeout* apenas quando o *buffer* for completamente preenchido se dá pela garantia de que a quantidade de intervalos ociosos será uma amostra representativa do comportamento do usuário. Caso contrário, corre-se o risco de se ter dados insuficientes e não haver confiabilidade no *timeout* calculado.

3.5.3 Estimação da Função de Densidade de Probabilidade

Com o *buffer* de intervalos ociosos, é possível calcular a sua distribuição de probabilidades para que seja aplicada na Equação (3.6). É estipulado um valor máximo i_{max} para os intervalos ociosos e gerado um *array* com i_{max} elementos. Este *array* deve conter as frequências relativas de cada intervalo ocioso no intervalo $[1, i_{max}]$.

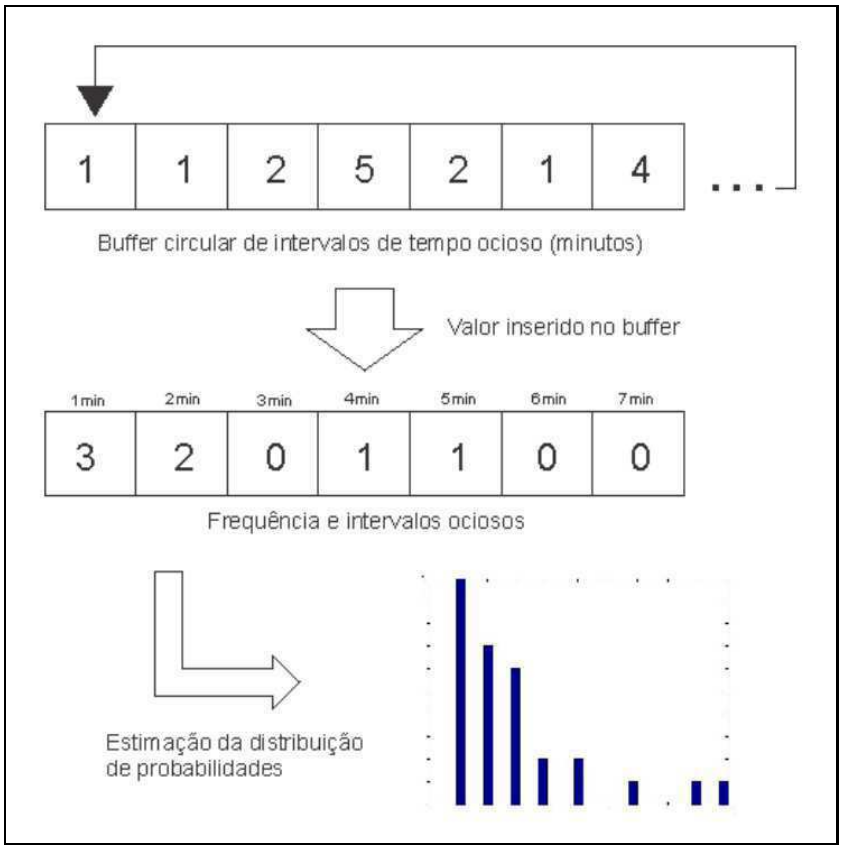

Figura 3.7: Estimação de função de densidade de probabilidade.

3.5.4 Avaliação dos Candidatos a *Timeout*

Após aplicar os valores obtidos na função de densidade de probabilidade na Equação (3.6), são obtidos os candidatos a *timeout*. Na implementação dos experimentos, o valor de *timeout* máximo $[1, i_{max}]$ estipulado foi 30 minutos, com variação de 1 minuto para cada valor, totalizando 30 valores da função de densidade de probabilidade.

Para o conjunto de 30 valores candidatos a *timeout*, espera-se que a resolução do problema de otimização descrito na Equação (3.8) não gere um aumento no consumo de energia que inviabilize o uso da política de TD.

Capítulo 4

Comparação entre métodos de classificação de ociosidade da interface de rede

Os métodos comparados são os seguintes: o método baseado em limiar (método L), apresentado por Luiz et al. [11], e o método baseado em regressão logística (método R).

Para comparar os métodos de classificação citados, quatro tipos de cargas de trabalho diferentes foram utilizados:

- Usuário - representa a carga de trabalho gerada por um usuário acessando a interface de rede durante atividades tais como bate-papo, reprodução de vídeos e navegação;
- Sistema Operacional - representa a carga de trabalho gerada pelo sistema operacional ao fazer *download* de atualizações ou outras atividades não inicializadas pelo usuário;
- Ambos - representa a carga de trabalho gerada por ambos usuário e sistema operacional utilizando a interface de rede simultaneamente;
- Nenhum - representa praticamente nenhum acesso à interface de rede.

Os tipos de carga de trabalho "usuário"e "ambos"são consideradas como dados de classe positiva no problema de classificação para ambos os métodos L e R. A carga de trabalho do tipo "sistema operacional"e "nenhum"são consideradas coom dados de classe negativa no problema de classificação.

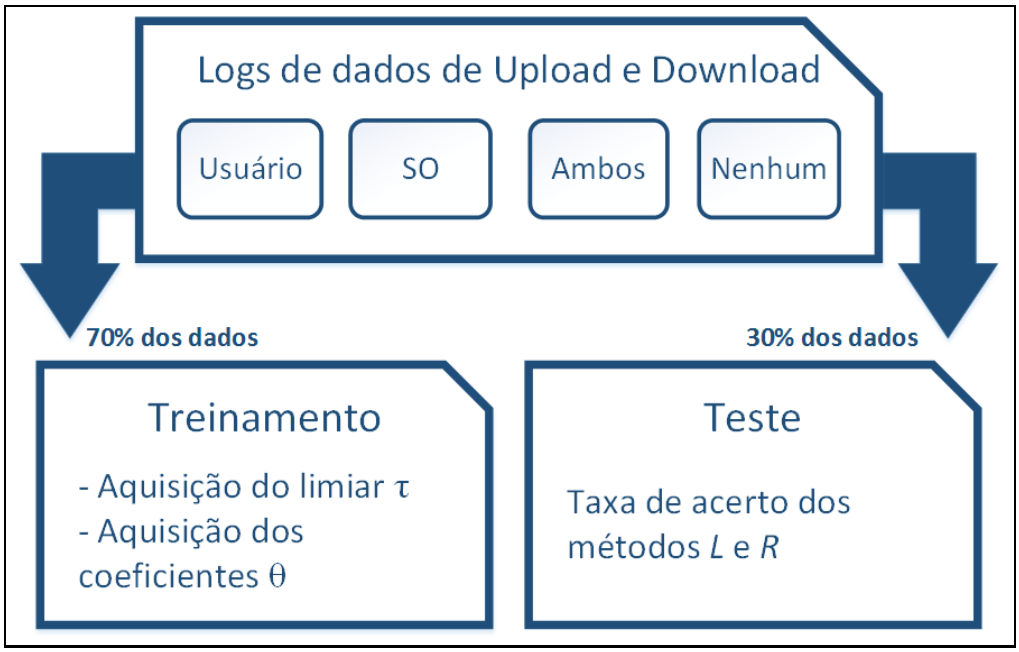

Figura 4.1: Proporção dos arquivos de log usados e resultados obtidos em cada etapa da simulação.

As taxas de *download* e *upload* da interface de rede foram medidas sob os quatro tipos de carga de trabalho apresentados e armazenados em arquivos de log. Estes arquivos representando as cargas de trabalho foram divididos em dois grupos: o conjunto de treinamento e o conjunto de teste. O conjunto de treinamento possui 70% de dos dados de log e é utilizado para executar a etapa de aprendizagem supervisionada, enquanto que os 30% restantes são utilizados pelo conjunto de teste. A taxa de acerto obtida quando o conjunto de teste é aplicado nos métodos de classificação é usada para validação de seu desempenho e realização de comparações.

Os *scripts* para aprendizado e teste foram implementados em MATLAB, e o método L teve seu limiar reavaliado por conta de que o computador de onde os *logs* foram obtidos é diferente daquele utilizado por Luiz et al. [11]. Após o passo de aprendizagem supervisionada, o limiar para o método L foi escolhido, e os coeficientes *teta* foram selecionados para o método R.

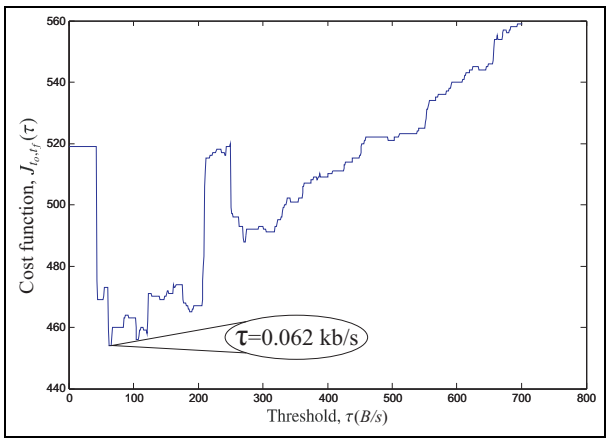

Figura 4.2: Função de custo para a seleção do limiar.

Coeficiente	Valor
θ_o	0.4039
θ_d	8.0422e-06
θ_u	0.0010

Tabela 4.1: Coeficienes escolhidos com o método de regressão logística.

4.1 Resultados da Simulação

Ambos os métodos de classificação foram executados com o conjunto de testes como parâmetro de entrada, e a taxa de acerto foi avaliada. Os resultados estão na Tabela 4.2.

Método	Taxa de acerto
Baseado no limiar (L)	69.861%
Baseado na regressão logística (R)	75.00%

Tabela 4.2: Taxas de acerto para cada método.

Conforme mostrado na TABELA 4.2, o método de regressão logística teve uma taxa de acertos maior. Possíveis causas para este resultado são:

- Utilização de taxas de *download* e *upload* como dois fatores diferentes, permitindo diferentes pesos para cada um pelos coeficientes θ;
- A regressão logística tira vantagem da função logística, a qual é apropriada para o uso em problemas de classificação.

4.2 Resultados obtidos

Como o método de regressão logística utiliza um total de três coeficientes e o resultado é obtido por calculos simples, este método de classificação para detecção de intervalos ociosos em interfaces de rede pode ser implementado em dispositivos móveis e aumentar a economia de energia devido à estimações mais acurada de cargas de trabalho para políticas de gerenciamento de energia.

Capítulo 5

Comparação Entre Modelos por Meio de Simulação

Neste capítulo, é descrito o experimento realizado para comparar as políticas de *timeout* dinâmico e fixo. O experimento foi realizado na forma de simulação, onde ambas as políticas foram implementadas no ambiente MATLAB com os seus parâmetros pré-calculados, extraídos de um *netbook*. O resultado deste experimento permitiu que fosse realizado um segundo experimento (descrito no Capítulo 5) com medição real de consumo de um dispositivo móvel com as políticas instaladas.

5.1 Definição dos Objetivos

Os objetivos deste experimento foram expressos seguindo o *template* GQM (*Goal Question Metrics*), que consiste num "mecanismo para definição e interpretação de medições de *software*" [1].

5.1.1 Objetivos no *template* GQM

Analisar dois algoritmos de gerenciamento dinâmico de energia para interfaces de rede baseados em *timeout*

com a intenção de compará-los

com respeito à sua eficácia

do ponto de vista dos usuários de aparelhos eletrônicos de consumo que dependem de bateria

no contexto de simulação representativa da utilização de notebooks por integrantes do laboratório Embedded, na UFCG.

5.1.2 Objetivos no *Design* do Experimento

Encontrar o melhor algoritmo: Experimento Comparativo.

5.2 Definição de Hipóteses

Para saber se o algoritmo é eficaz ou não iremos observar se houve economia de energia, dada uma penalidade de desempenho L. Logo, as métricas utilizadas para medição da eficácia de um algoritmo são dadas em Watts, para a potência média, e sem unidade, para a penalidade de desempenho média. As questões de pesquisa são:

P1: Os algoritmos original e otimizado apresentam diferenças significativas de desempenho quando temos como entrada *scripts* de utilização que representam diferentes perfis de usuário?

Como temos duas variáveis de resposta que identificam o desempenho do algoritmo, duas hipóteses nulas foram formuladas:

- Hipótese Nula 1 ($H1_0$): o consumo de energia dos algoritmos é igual;
- Hipótese Nula 2 ($H2_0$): a penalidade de desempenho média dos algoritmos é igual.

Da mesma forma, as hipóteses alternativas também foram formuladas:

- Hipótese Alternativa 1 ($H1_1$): o consumo de energia dos algoritmos é diferente;
- Hipótese Alternativa 2 ($H2_1$): a penalidade de desempenho média dos algoritmos é diferente.

Caso as hipóteses nulas sejam rejeitadas, pretende-se identificar o algoritmo com melhor desempenho com o auxílio do projeto de experimento descrito a seguir.

5.3 Planejamento do Experimento

5.3.1 Fatores e variáveis de resposta

Fatores principais

- Algoritmo (A): algoritmos estado-da-arte em gerenciamento dinâmico de energia para interfaces de rede baseada em *timeout*
  - 2 níveis: Algoritmo de Timeout Dinâmico (TD), Algoritmo de Timeout Fixo (TF)
- Restrição de penalidade de desempenho (L): limiar que indica o máximo de penalidades de desempenho que a política deve alcançar
  - 10 níveis: valores pertencentes ao conjunto 0.1, 0.2, ..., 1

Fator de blocagem

- Taxas de acesso à Internet
  - 40 níveis: arquivos de dados contendo, cada um, taxas de *upload* e *download* referentes a um período de utilização do computador.

Variáveis de resposta

- Potência média (Watt): potência média calculada durante a simulação.
- Penalidade de desempenho média: penalidade de desempenho média calculada em cada ensaio.

Planejamento experimental

O *design* escolhido para o experimento foi o *Design* Fatorial Completo com Blocagem, devido às seguintes características do problema:

- Há mais de um fator analizado e cada um deles possui vários níveis que podem ser comparados através de um experimento;

- Cada execução tem custo mínimo e não há problemas em executarmos todas as combinações possíveis dos fatores.

As taxas de *upload* e *download* presentes nos arquivos de *log* se comportam de forma não estacionária. Esta característica implica que a variação deste fator provavelmente irá afetar as variáveis de resposta de forma indesejada para o nosso estudo, e por isso, o *design* com blocagem foi escolhido a fim de ofuscar a influência do fator mencionado nas variáveis resposta.

O experimento foi realizado na forma de uma simulação, onde os algoritmos implementados serão executados com os fatores sendo passados como parâmetros predefinidos. O ambiente escolhido para simulação foi o *software* Matlab, pois este é um ambiente de desenvolvimento de alto nível.

Dado o design escolhido para o experimento, teremos no total 20 ensaios a serem realizados variando os fatores de interesse, e cada ensaio será repetido em cada um dos 40 arquivos de *log* que possuem dados reais correspondentes a taxas de acesso à Internet (*upload* e *download*). A tabela 5.1 descreve cada um dos tratamentos:

Redução de erros

Para aumentar o poder do experimento, serão tomadas as seguintes medidas:

- As taxas de acesso à Internet serão obtidas de logs de dados de utilização real de *notebooks*.

- As taxas de acesso à Internet se comportam de forma não estacionária. Para minimizar a influência da variação deste fator, iremos utilizá-lo como fator de blocagem. Portanto, cada arquivo de *log* de utilização representará um nível do fator de blocagem.

- Já que a simulação retornará os valores das variáveis de resposta de forma determinística, a randomização da ordem dos fatores na execução do experimento não se faz necessária.

Nº do Tratamento	Algoritmo	Restrição de Penalidade de Desempenho
01	TF	0,1
02	TF	0,2
03	TF	0,3
04	TF	0,4
05	TF	0,5
06	TF	0,6
07	TF	0,7
08	TF	0,8
09	TF	0,9
10	TF	1,0
11	TD	0,1
12	TD	0,2
13	TD	0,3
14	TD	0,4
15	TD	0,5
16	TD	0,6
17	TD	0,7
18	TD	0,8
19	TD	0,9
20	TD	1,0

Tabela 5.1: Tratamentos realizados no experimento. É exibido na tabela a variação dos fatores principais. Foi utilizado o fator de blocagem para delimitar o conjunto de 20 tratamentos como um bloco, e o mesmo não está exibido na tabela para não torná-la demasiadamente extensa.

5.3.2 Forma de coleta dos dados e análise

Coleta dos dados

Os valores de potência média e penalidade de desempenho média serão obtidos através da função *fprintf* do Matlab.

Análise dos dados

Sabendo que trata-se de um experimento onde existem mais de dois fatores principais controlados e analizados, e que pretende-se fazer uma comparação entre dois algoritmos, será feito:

- Estimação de erros;
- Estimar intervalos de confiança provenientes da eficácia de cada algoritmo;
- Verificar a normalidade dos dados para identificar que prova estatística será mais adequada;
- Determinar se a variação encontrada entre os valores de potência média e penalidade de desempenho média para cada uma das técnicas é significativa ou não.

5.3.3 Análise de ameaças à validade

Está sendo considerado que os seguintes fatores podem gerar ameaças e influenciar nas conclusões deste trabalho. São eles:

- Ausência da estimação do consumo de energia da própria política de GDE;
- Dados de *upload* e *download* conterem ações de programas de terceiros, como atualizações automáticas de *software*;
- A diferença entre os dispositivos móveis poder resultar em maior viabilidade de aplicação da política de GDE em alguns do que em outros aparelhos.

5.4 Preparação do Experimento

Para a simulação dos algoritmos durante o experimento, os dois algoritmos comparados foram implementados e preparados para simulação em *scripts* para o ambiente MATLAB. A escolha do ambiente foi feita pelos seguintes motivos:

- Um dos algoritmos já estava implementado nesta plataforma;

- O MATLAB é uma ferramenta poderosa que pode ser utilizada no nosso contexto sem riscos de não oferer suporte para tal simulação, já que algo semelhante foi feito anteriormente com o algoritmo original de *timeout* fixo.

Os resultados da simulação foram armazenados em arquivo[1] , e seus dados estão organizados na forma descrita abaixo:

- O arquivo contém 7 colunas, separadas entre si por tabulação;
- As colunas representam os seguintes atributos:
  - Coluna 1 - número do arquivo de *log*: indica o número correspondente a um arquivo de log específico com taxas de *upload* e *download*, variando de 01 a 40. Cada arquivo de *log* representa um nível do fator de blocagem;
  - Coluna 2 - restrição de penalidade de desempenho: indica o nível escolhido do fator Restrição de Penalidade de Desempenho;
  - Coluna 3 - potência média do algoritmo com *timeout* dinâmico (em Watt): contém uma das variáveis de resposta do experimento com o nível do fator algoritmo igual a timeout dinâmico e níveis dos demais fatores indicados na respectiva linha;
  - Coluna 4 - potência média do algoritmo com *timeout* fixo (em Watt): contém uma das variáveis de resposta do experimento com o nível do fator algoritmo igual a *timeout* fixo e níveis dos demais fatores indicados na respectiva linha;
  - Coluna 5 - potência média "on"(watt): valor da potência média de uma máquina com a interface de rede ligada e ociosa;
  - Coluna 6 - penalidade de desempenho média com *timeout* dinâmico: contém uma das variáveis de resposta do experimento com o nível do fator algoritmo igual a *timeout* dinâmico (a=TD) e níveis dos demais fatores indicados na respectiva linha;
  - Coluna 7 - penalidade de desempenho média com *timeout* fixo: contém uma das variáveis de resposta do experimento com o nível do fator algoritmo igual a timeout fixo (a=TF) e níveis dos demais fatores indicados na respectiva linha.

[1] https://docs.google.com/document/d/1Xm2ftli3dXvsVBktyEn6NHNbhuK29gXEXJQNXp3xLx8/edit?usp=sharing

A análise estatística realizada durante a execução do experimento foi feita utilizando a ferramenta R. Em alguns casos, para melhor aprendizado e entendimento de como é feita a implementação de alguns passos da análise (como a criação do modelo que explica a variação dos dados, por exemplo), foram utilizadas planilhas eletrônicas, e a fim de evitar erros de implementação, os valores obtidos nas planilhas foram comparados com valores obtidos através do cálculo no R. Estes casos estão descritos no decorrer do capítulo.

5.5 Análise Descritiva dos Dados

Esta análise foi feita visando a familiarização com os dados, nos permitindo observar o comportamento das variáveis de resposta e nos ajudando a visualizar padrões/inconsistências nos valores das mesmas.

A seguir encontra-se a análise dos dados de cada variável de resposta estudada e também de um dos fatores do experimento, o qual é usado como fator de blocagem.

5.5.1 Taxas de acesso à Internet (fator de blocagem)

Os dados correspondentes às taxas de acesso à Internet são valores de *download* e *upload* coletados a cada segundo e armazenados num arquivo de log. Cada arquivo de log corresponde a um período de utilização do computador, onde o usuário ligou a máquina, iniciou o log dos dados, e antes de desligar a máquina, finalizou o processo de *log* dos dados.

Os tipos de acesso à Internet podem ser bastante variados, como simples trocas de mensagens, *downloads* de grandes arquivos, exibição de vídeos, jogos *online*, entre outras atividades. Devido a essa grande variedade de comportamentos diferentes que um usuário pode assumir ao acessar a Internet, verifica-se que nos dados coletados, para o período de tempo que estudamos, a distribuição dos intervalos de tempo em que a interface de rede está ociosa não é bem definida, e varia com o tempo.

As figuras 5.1 e 5.2 exibem os histogramas dos intervalos de tempo inativos da interface de rede para cada arquivo de log. Nestas figuras, observa-se que em alguns casos temos uma distribuição bem definida com uma cauda longa à direita, mas como esta distribuição não é encontrada muitas vezes e nos casos restantes não há distribuição bem definida, não podemos assumir que os intervalos de tempo ociosos são uma variável aleatória independente do

tempo. Por este motivo, assumimos que os dados coletados no experimento estão divididos em blocos, que são delimitados pelos arquivos de dados.

5.5.2 Variáveis de resposta Potência média e Penalidade de Desempenho média

Para o estudo das variáveis de resposta, decidiu-se ver as suas características para vários tipos de agrupamentos diferentes, de acordo com os fatores do experimento.

Na Figura 5.3 temos os *boxplot* e histogramas das variáveis de resposta agrupadas para cada algoritmo.

Nos gráficos abaixo identificamos assimetria e a presença de vários candidatos a *outliers*. Nota-se também que o algoritmo TF parece ter variância um pouco maior que o algoritmo TD para a variável de resposta Penalidade de Desempenho média.

A seguir foi feito o agrupamento de acordo com os níveis do fator Restrição de Penalidade de Desempenho (L).

Nos gráficos da Figura 5.5, assim como apresentados na Figura 5.4, também identificamos assimetria e vários candidatos a *outliers*. Além disso observamos um aumento na dispersão dos dados à medida que o valor de L aumenta.

Na Figura 5.6 são exibidos grupos de dois *boxplots*, onde cada *boxplot* exibe os dados de um algoritmo, e cada grupo delimita os dados para um nível específico do fator L. Desta forma, podemos identificar visualmente a influência da mudança de nível do fator L para cada nível do fator Algoritmo.

Vemos que a variância dos dados aumenta à medida que aumentamos o nível do fator L, e além disso, o algoritmo TD aparenta maior resistência do que o algoritmo TF a este aumento de dispersão nos dados. O aumento na dispersão da amostras é previsível porque o aumento do nível do fator L indica que o algoritmo terá menor tolerância à ociosidade da rede. Ou seja, qualquer indício de que a rede está ociosa será mais "percebido"se o valor de L for maior, havendo mais desligamentos na interface de rede, e consequentemente mais penalidades de desempenho, como veremos nas análises da variável Penalidade de Desempenho Média.

O problema nestas análises é que dados de blocos diferentes foram agrupados sem o cuidado de tentar eliminar a influência que cada bloco pode estar exercendo nos valores. Nas

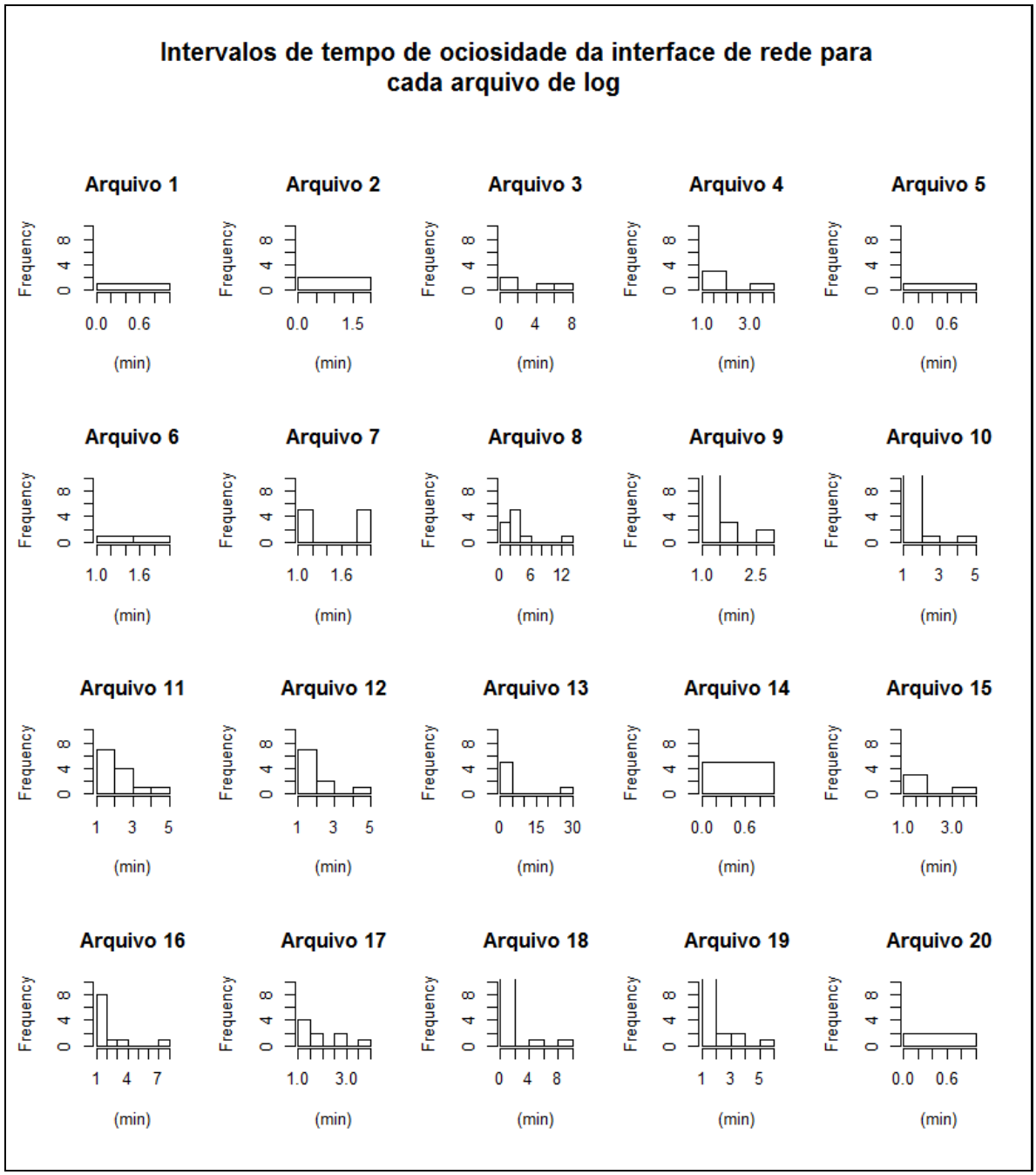

Figura 5.1: Histogramas dos intervalos de tempo de ociosidade da interface de rede para os arquivos de log (blocos) 1 a 20.

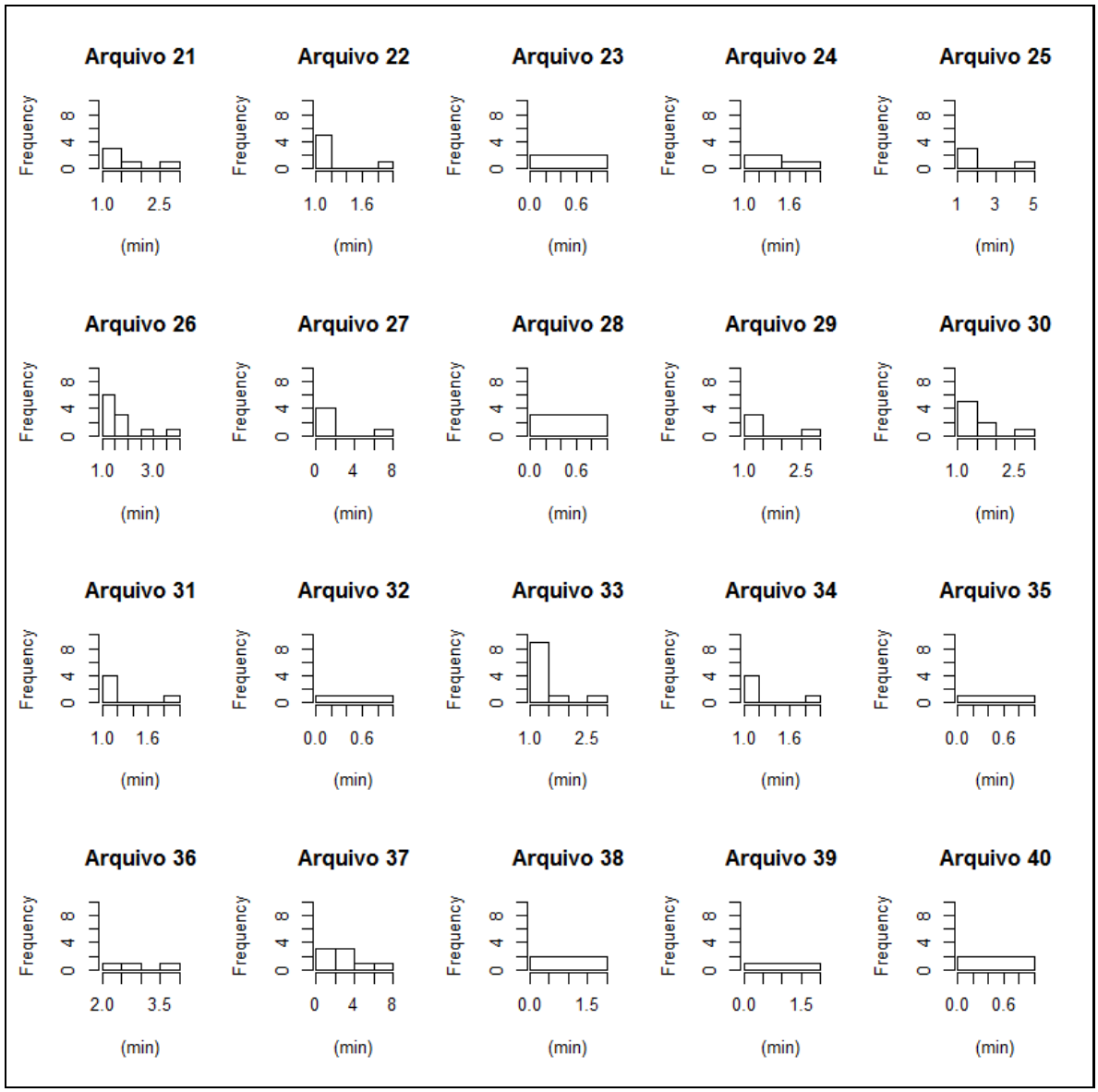

Figura 5.2: Histogramas dos intervalos de tempo de ociosidade da interface de rede para os arquivos de log (blocos) 21 a 40.

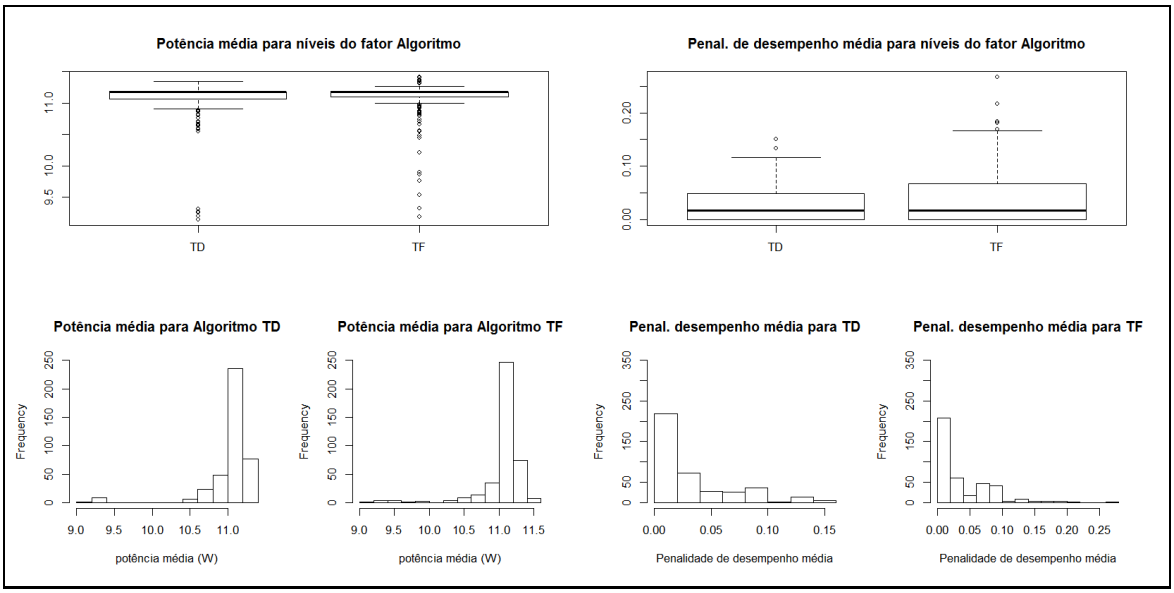

Figura 5.3: Boxplots e histogramas das variáveis de resposta, com amostras de cada algoritmo.

figuras 5.8 e 5.9 podemos ter uma idéia do quanto cada bloco deve estar afetando nos valores desta variável de resposta.

Com os dados agrupados por blocos, há uma grande variação nos valores para cada nível do fator de blocagem (tanto em simetria quanto em dispersão), e nos gráficos exibidos anteriormente não houve tratamento para eliminar a influência deste fator, portanto, não podemos analisar os possíveis *outliers* apontados nos agrupamentos que independem de blocos. É preciso utilizar outra técnica para análise dos dados. Quanto aos possíveis *outliers* exibidos nos gráficos por blocos das Figuras 5.8 e 5.9, eles são valores correspondentes ao nível "1"do fator L, e em alguns casos também o nível "0.9". Nestes níveis de L há a probabilidade de que, devido à alta taxa de penalidade de desempenho média, a quantidade de religamentos da interface de rede acabe aumentando consideravelmente a potência média consumida, prejudicando a eficácia do algoritmo. Vale salientar que estes casos não são frequentes, dado que o algoritmo calcula o valor de *timeout* considerando a probabilidade de o usuário voltar a religar a interface de rede em um dado momento futuro.

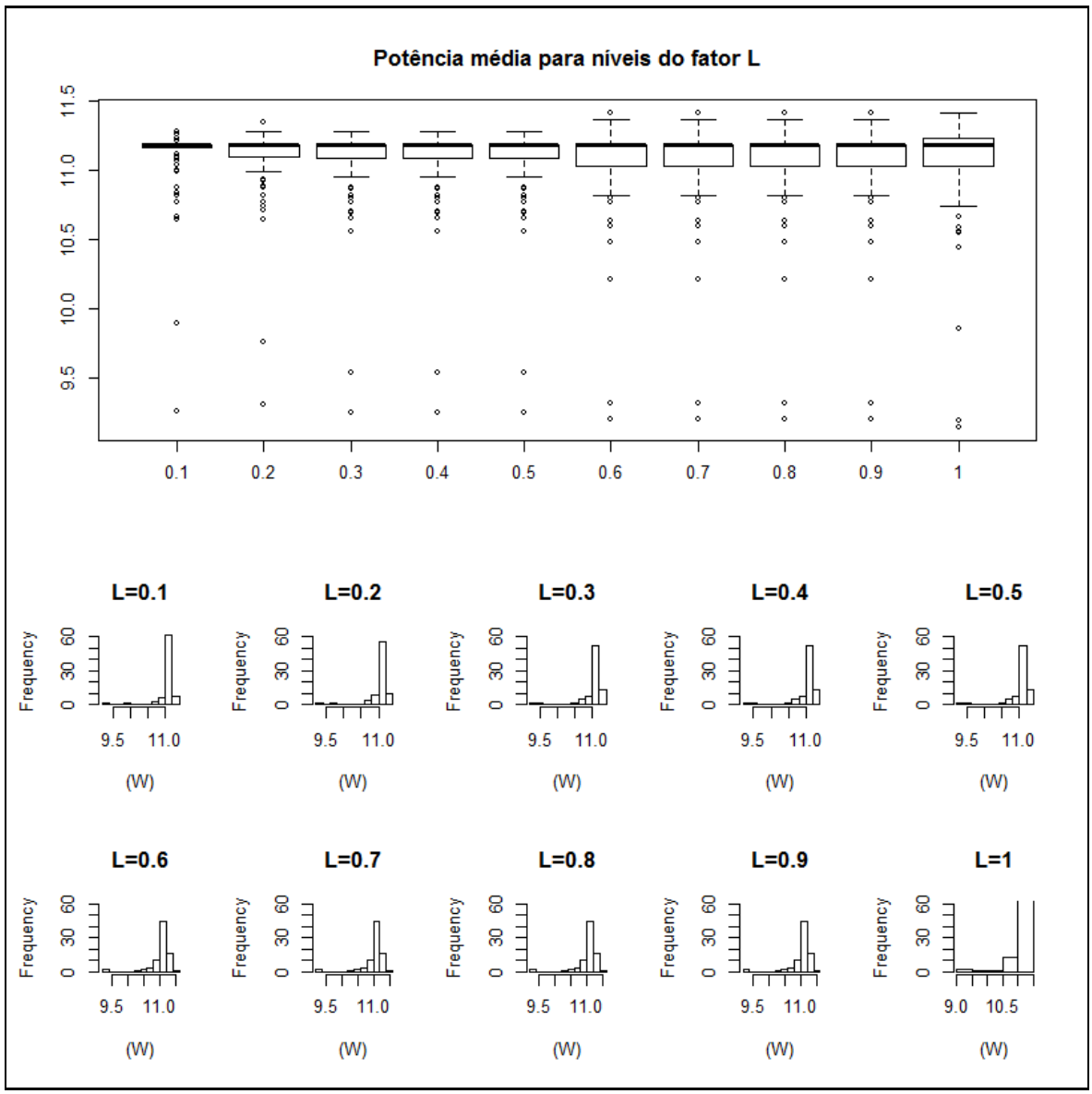

Figura 5.4: Histogramas da potência média, com amostras de cada nível do fator L.

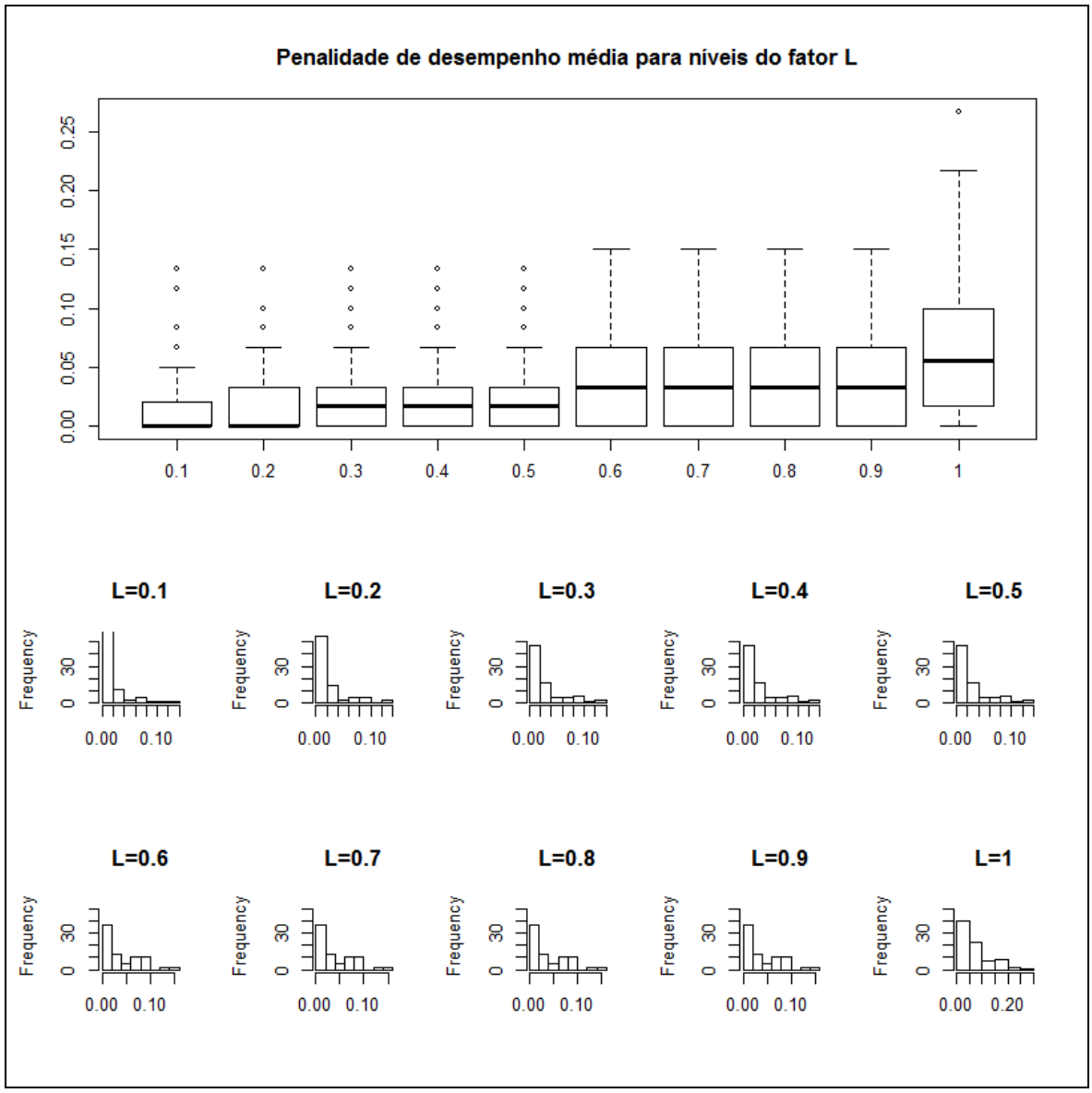

Figura 5.5: Histogramas das penalidade de desempenho média com amostras de cada nível do fator L.

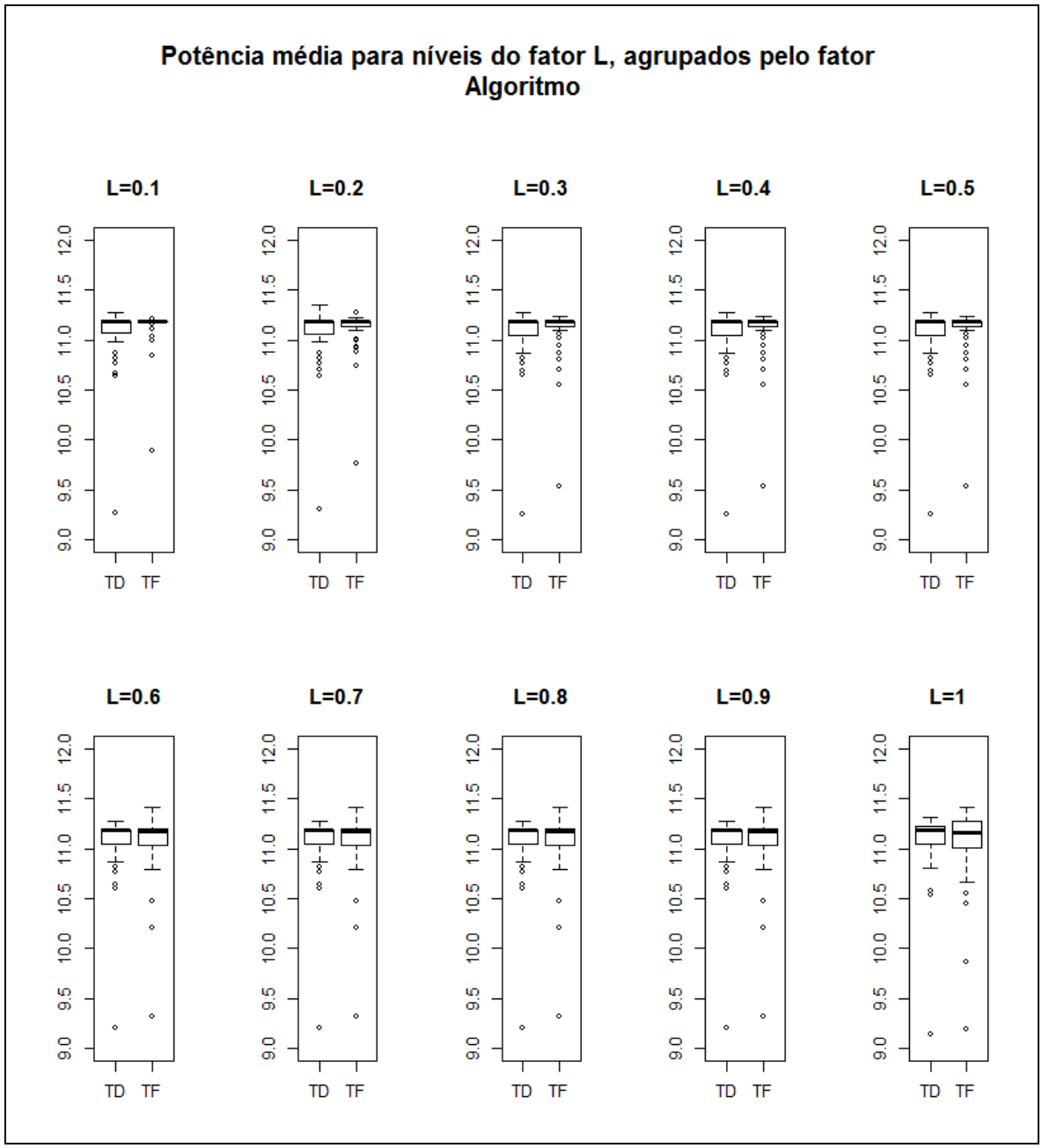

Figura 5.6: Boxplots da potência média, com amostras dos algoritmos para cada nível do fator L.

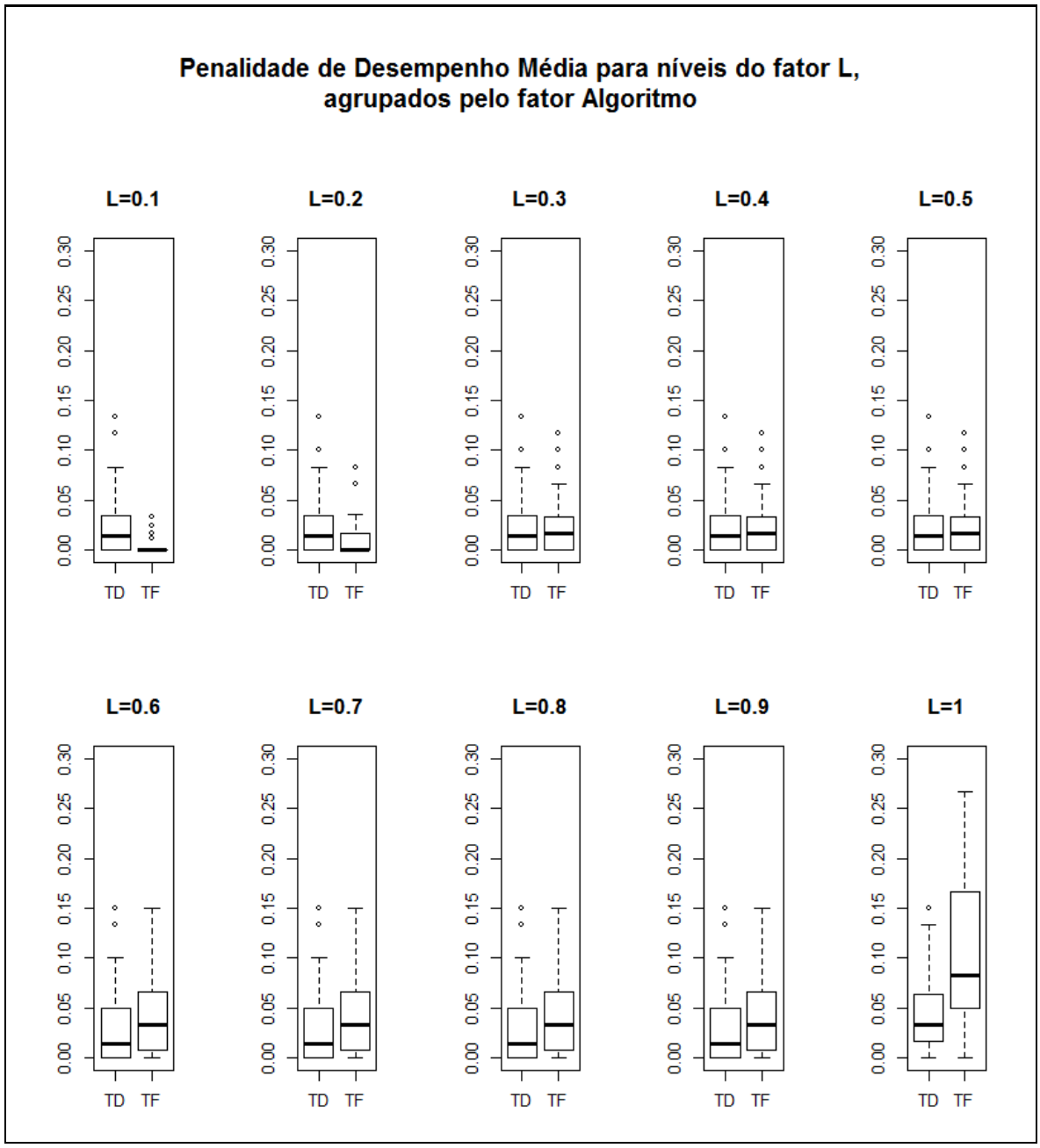

Figura 5.7: Boxplots da penalidade de desempenho média, com amostras dos algoritmos para cada nível do fator L.

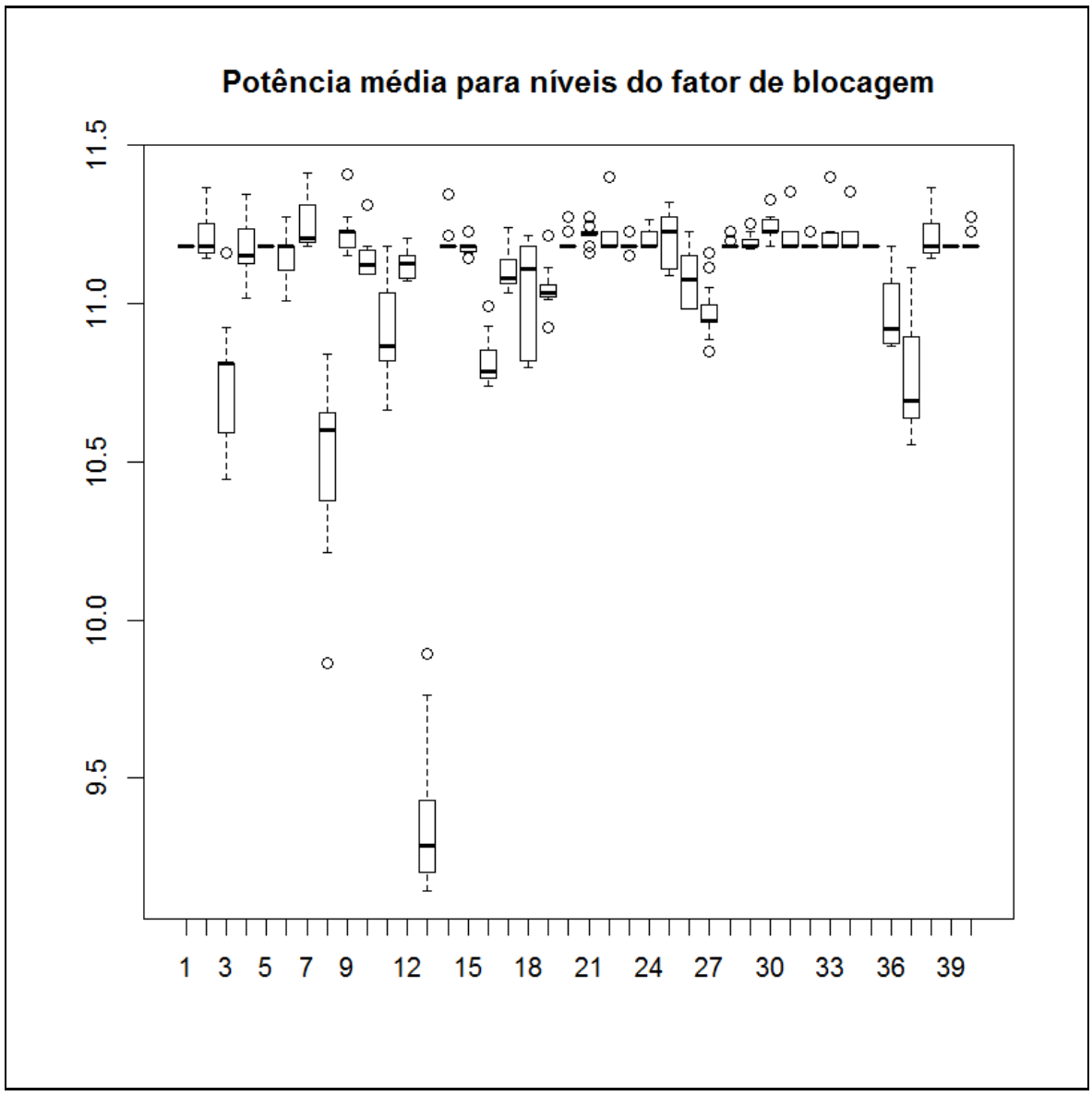

Figura 5.8: Boxplots da potência média para cada bloco.

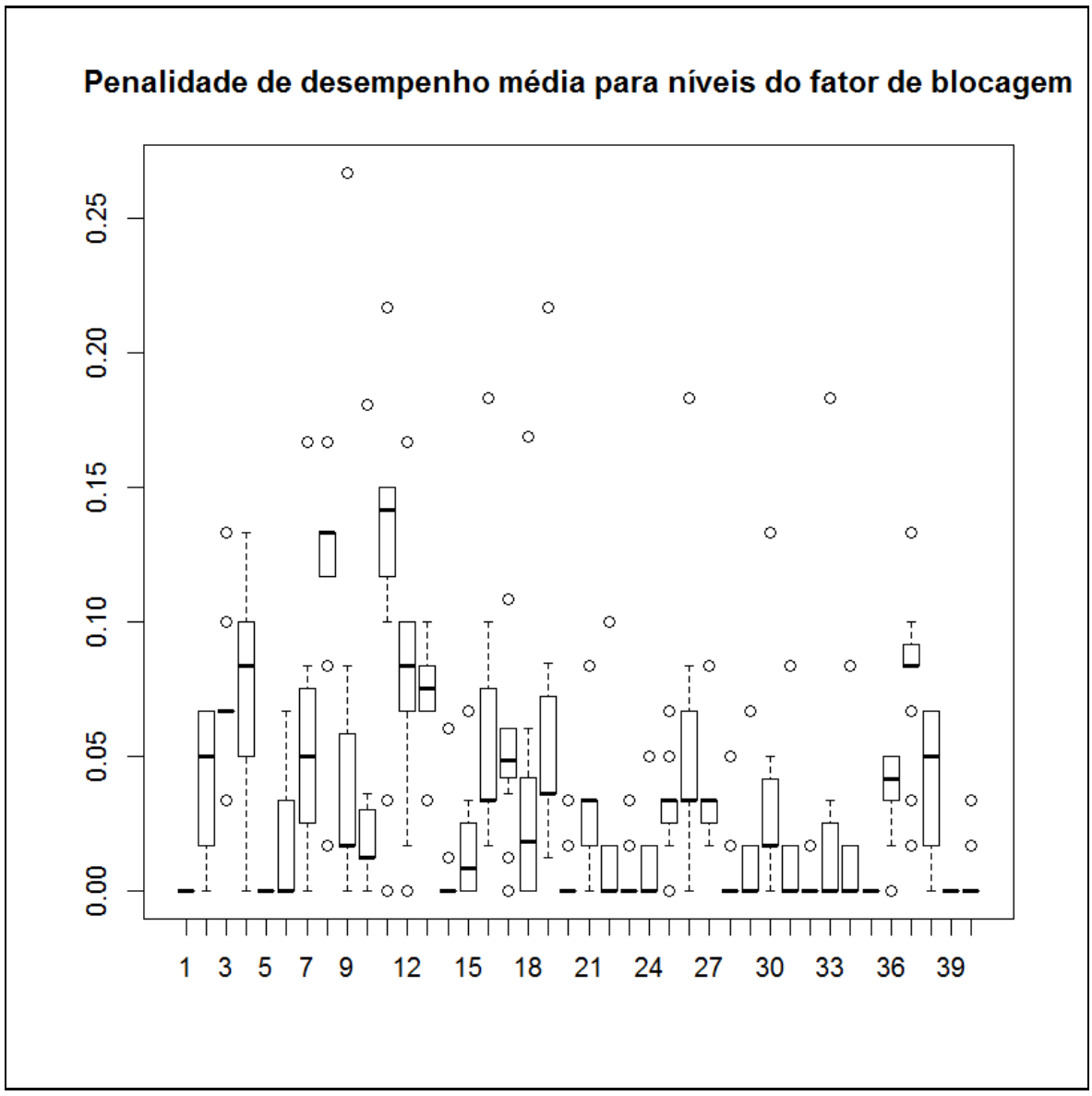

Figura 5.9: Boxplots da penalidade de desempenho média para cada bloco.

5.6 Pareamento de Amostras

Verificando a tabela de dados do experimento[2] , vemos que em cada bloco temos, para todos os níveis do fator L, dois valores, sendo cada um de um nível do fator Algoritmo. Podemos considerar estes valores como pares, já que eles se apresentam como a repetição de um experimento para os mesmos níveis de carga de trabalho de utilização da Internet e de restrição de penalidade de desempenho. Então, analizando o experimento do ponto de vista do fator Algoritmo, consideramos que o mesmo possui amostras pareadas Y' da variável de resposta Y. Seja $Y = y_{ijk}$, então Y' é:

$$Y' = (TD, TF) = (y_{1jk}, y_{2jk}), \tag{5.1}$$

onde:

- Y é a variável de resposta;
- Y' é a variável de resposta com pareamento das amostras de acordo com o fator algoritmo;
- i é o nível do fator algoritmo;
- j é o nível do fator restrição de penalidade de desempenho;
- k é o bloco.

Para testes descritos a seguir, são utilizadas amostras das diferenças entre os valores pareados. A fim de analisarmos as amostras das diferenças, foram feitos alguns gráficos desses valores que seguem abaixo:

Pelo histograma da Figura 5.10 e pelo seu valor de *skewness* computado, identificamos um certo nível de simetria para a variável de resposta potência média, mas não para a penalidade de desempenho média. Em contrapartida, o gráfico *qqplot*, o valor *kurtosis* calculado e os testes de Shapiro-Wild e Anderson-Darling indicam que não há normalidade dos dados.

[2]https://www.dropbox.com/s/gvg12p9a872kcgy/Tabela%20-%20Experimento.xlsx

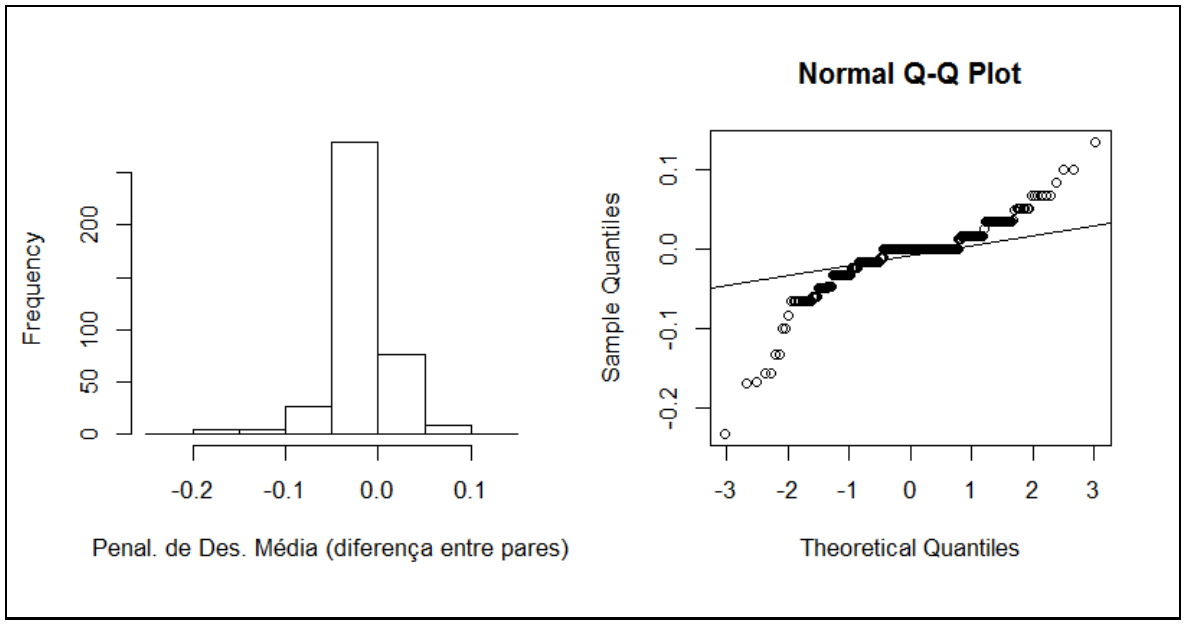

Figura 5.10: Verificação da normalidade das diferenças entre os dados pareados (necessário para futuros testes).

Teste	Valor (potência média)	Valor (penal. de desemp. média)
skewness	0.4178545	-1.5437
kurtosis	7.331499	11.5881
zskewness	3.411768	-12.60425
zkurtosis	29.93072	47.30824
Shapiro-Wilk	p-value < 2.2e-16	p-value < 2.2e-16
Anderson-Darling	p-value < 2.2e-16	p-value < 2.2e-16

Tabela 5.2: Testes de normalidade das variáveis de resposta.

5.7 Testes Paramétricos/Não Paramétricos/Intervalos de Confiança

Com as amostras pareadas, é excluída a opção de fazer análise de variância (ANOVA), já que a mesma requer amostras não pareadas. Além disso, não foi encontrada normalidade dos dados, tanto nos dados brutos quanto na diferença entre os pareamentos, eliminando a possibilidade de utilização de testes paramétricos.

Com as amostras pareadas é possível fazer o "teste de média zero"com intervalos de confiança pelos seguintes motivos:

- O teste avalia a diferença entre duas amostras, dadas as diferenças entre cada par de valores pareados;
- Os blocos influenciam ambos os valores de cada par da variável de resposta, e a diferença entre eles reduz o efeito do bloco sobre estas variáveis, considerando que não haja efeito de interação entre o fator Bloco e os demais fatores;
- O tamanho das amostras será suficiente para que as médias tenham uma distribuição aproximadamente normal, de acordo com o Teorema do Limite Central. São 40 pares de valores para as análises por nível do fator L e 400 pares de valores para as análises independentes do nível de L.

Nas figuras 5.11 e 5.12 estão os gráficos de barras dos valores médios de cada variável de resposta para os níveis do fator L, e também dos respectivos "testes de média zero".

As barras indicam que o valor médio da diferença entre os pareamentos é menor que zero para as duas variáveis de resposta. Ou seja, há a possibilidade de que o algoritmo TD tenha melhor desempenho e menor potência média, dado que a diferença entre os pareamentos é feita por respectivamente $penalidadededesempenho_{TD} - penalidadededesempenho_{TF}$ e $potencia_{TD} - potencia_{TF}$. Para se ter um resultado com significância estatística, foi feito o teste de média zero com alfa = 10%.

Com os testes de média zero, vemos que em geral (independente dos outros fatores), o algoritmo TD obteve melhor desempenho para as duas variáveis de resposta.

Uma opção de teste não-paramétrico para confrontar com o teste de média zero seria o teste de Wilcoxon Signed Rank para amostras pareadas. O problema é que o teste de

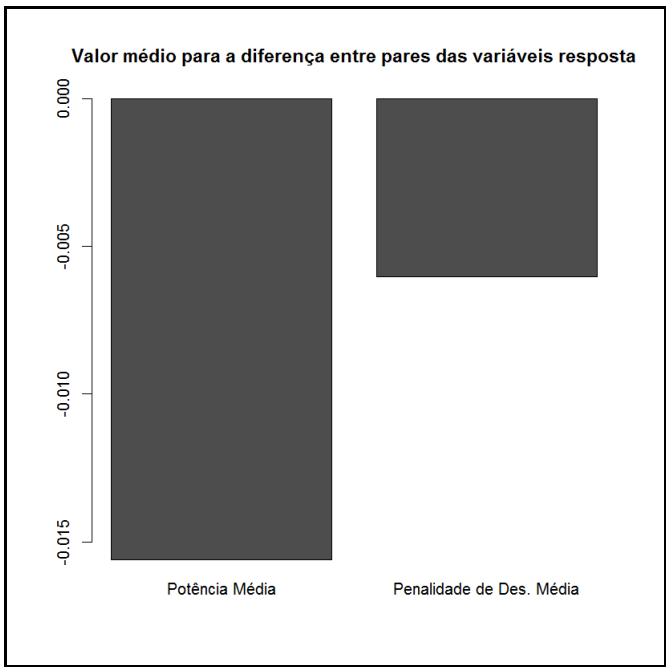

Figura 5.11: Gráfico de barras com a média amostral das diferenças entre os pareamentos, para cada variável de resposta.

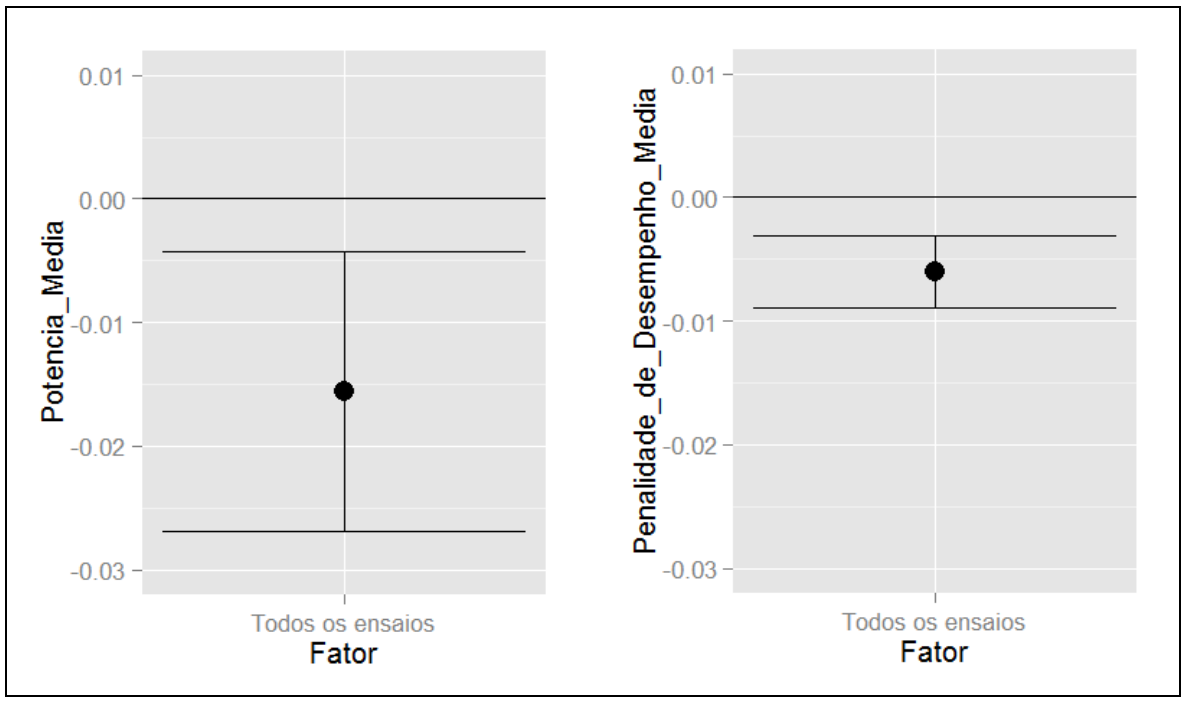

Figura 5.12: Testes de média zero para as variáveis de resposta.

Wilcoxon requer amostras com distribuição das diferenças entre os pares simétrica em torno da média, mas de acordo com a Figura 5.10, a amostra da penalidade de desempenho média não possui esta característica. Por esta razão o teste foi feito apenas para a variável de resposta Potência Média.

Amostra Pareada	Estatística V	p-value	Rejeita H0 (alpha=0,1)
Potência Média	12942	0,0003329	TRUE

Tabela 5.3: Teste de Wilcoxon Signed Hank para a variável de resposta Potência Média.

Ambos os testes de Wilcoxon e os testes de média zero indicam, com um nível de significância de 0.1, que o algoritmo TD possui valor médio menor que o algoritmo TF para as variáveis de resposta Potência Média e Penalidade de Desempenho Média, rejeitando as hipóteses nulas H1-0 e H1.2-0 do experimento.

A fim de verificar se o fator Restrição de Penalidade de Desempenho (L) influencia as variáveis de resposta, também foi feito o teste de média zero para os níveis dos fatores de L. Como as amostras para cada nível de L possuem 40 valores, mesmo os dados não possuindo distribuição normal, há valores suficientes para que o teste seja confiável. A seguir estão os gráficos de barras com valores médios de cada variável de resposta para os níveis de L.

As barras indicam que o valor médio da diferença entre os pareamentos é menor que zero em metade dos casos. Ou seja, há a possibilidade de que o algoritmo TD tenha melhor desempenho (menor potência média e menor penalidade de desempenho média), dado que a diferença entre os pareamentos é feita por TD-TF. Para se ter um resultado com significância estatística, foram feitos testes de média zero com alpha = 10%.

De acordo com os intervalos de confiança para um nível de significância de 10%, verifica-se que o algoritmo TD possui menor potência para os níveis 0.1 e 0.2 de L, enquanto que nos demais níveis, não há diferença entre os dois algoritmos. Com relação à Penalidade de Desempenho Média, o algoritmo TD possui melhor desempenho para os níveis 0,6, 0,7, 0,8, 0,9 e 1.

Para confrontamento com os testes de média zero, foi utilizado o teste não paramétrico de Friedman, que é adequado para dados pareados, com níveis de fator maiores que 2 e com replicações do experimento em blocos distintos. Para adaptar o nosso caso aos requisitos

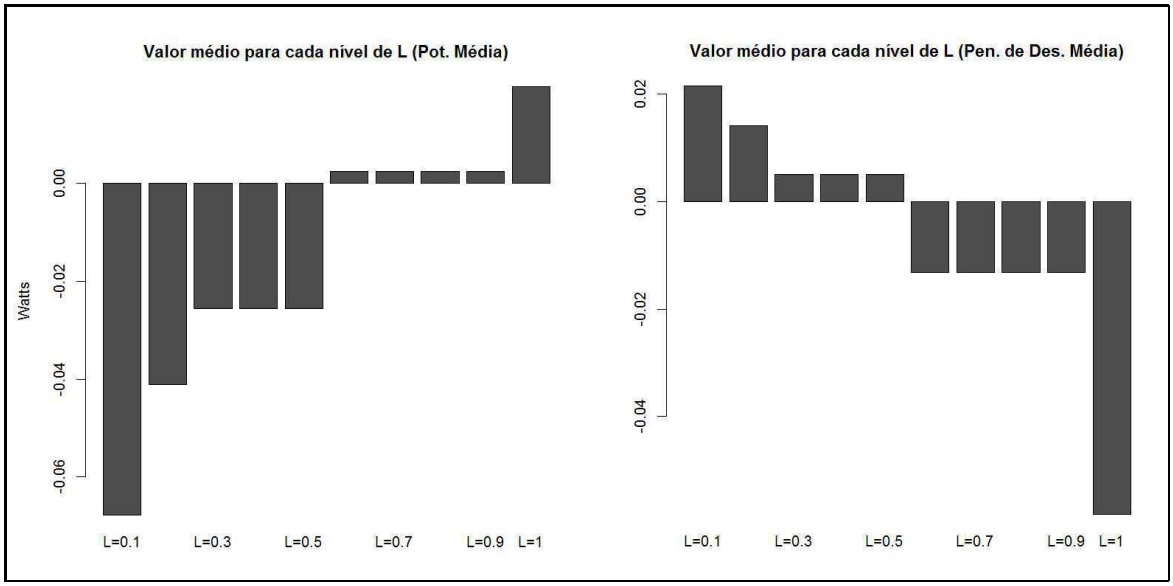

Figura 5.13: Gráficos de barras com a média amostral das diferenças entre os pareamentos, para cada variável de resposta e para cada nível do fator L.

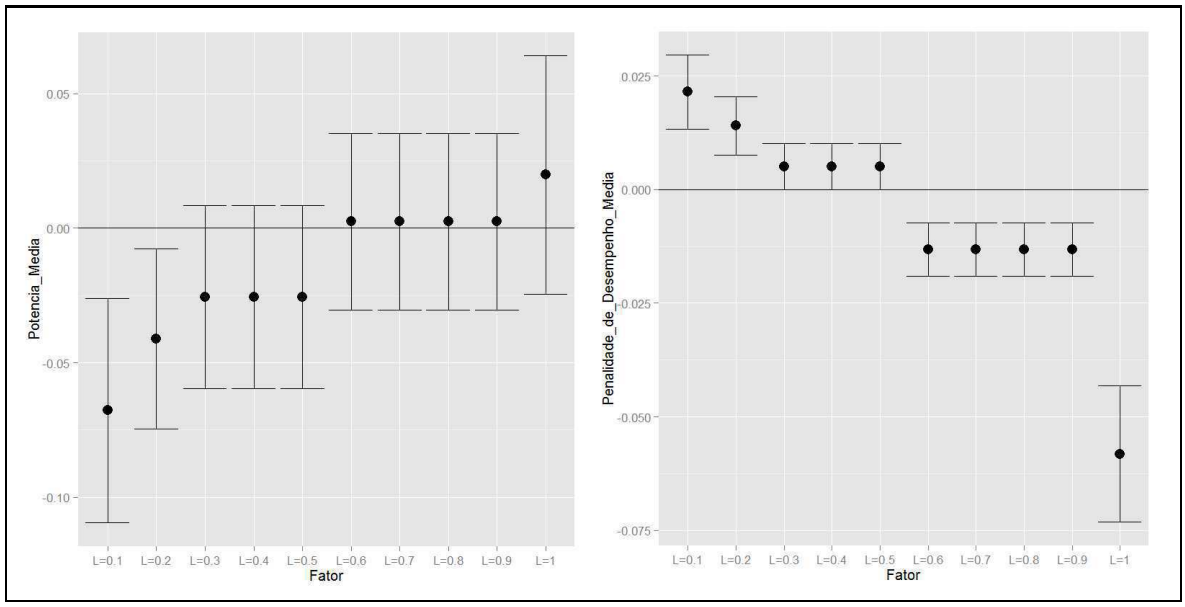

Figura 5.14: Testes de média zero para as variáveis de resposta Potência Média (à esquerda) e Penalidade de Desempenho Média (à direita), com amostras agrupadas por níveis de L.

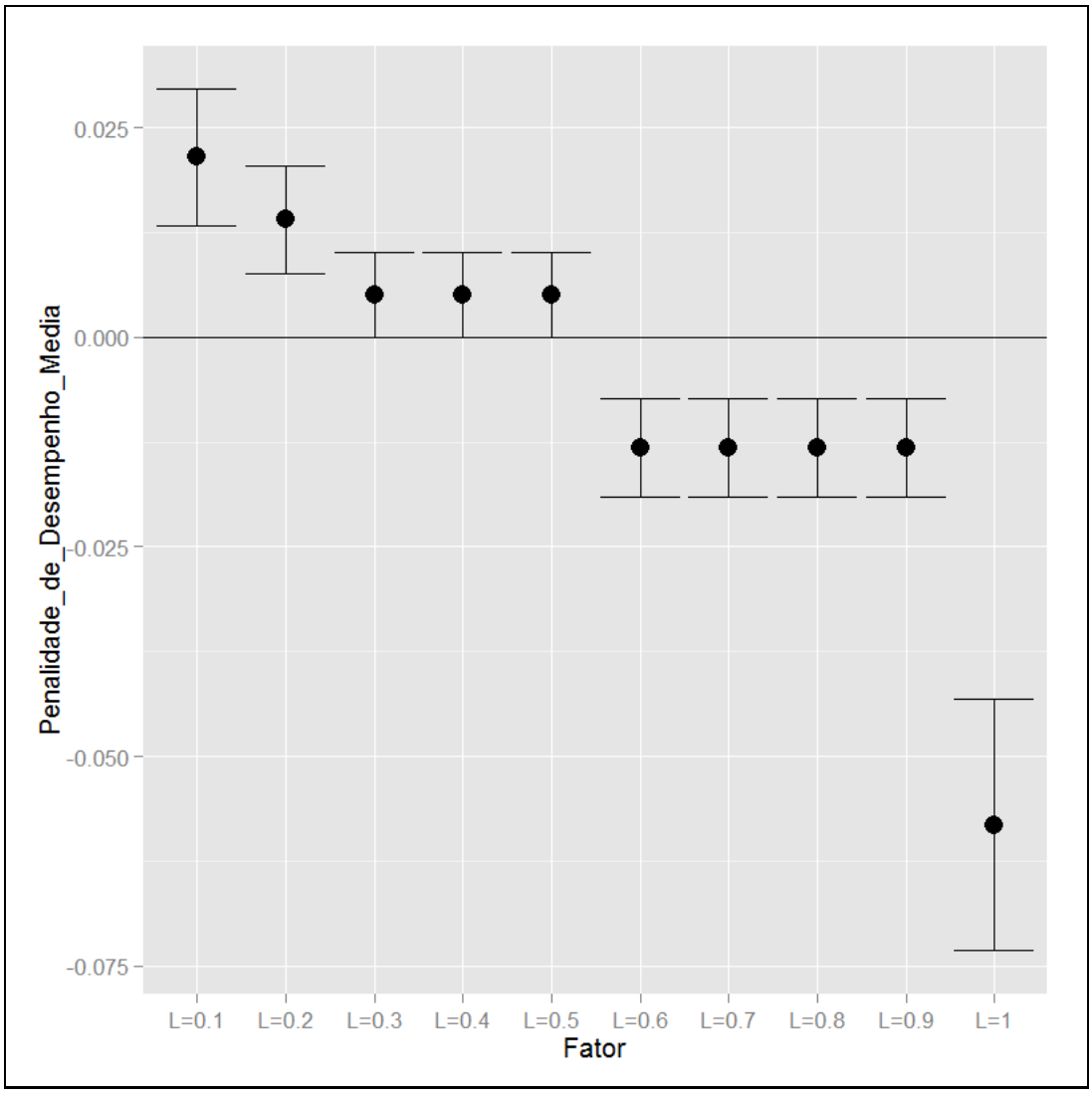

Figura 5.15: Testes de média zero para a variável de resposta Penalidade de Desempenho Média, com amostras agrupadas por níveis de L.

do teste de Friedman, as amostras de cada algoritmo foram separadas e o pareamento considerado foi entre os níveis do fator L, para cada algoritmo. A Tabela 5.4 abaixo exibe os resultados dos testes, dado que a hipótese nula é: ignorando o efeito dos blocos, os valores da variável de resposta são os mesmos em cada grupo.

Variável de Resposta	Algoritmo	Friedman Chi-Quadrado	p-value	Rejeita H0 (alpha=0.1)
Potência Média	TD	29.8691	0.0004618	TRUE
Potência Média	TF	20.2736	0.0163	TRUE
Penalidade de Des. Média	TD	185.7433	< 2.2e-16	TRUE
Penalidade de Des. Média	TF	268.6474	< 2.2e-16	TRUE

Tabela 5.4: Teste de Friedman realizado nas variáveis de resposta.

De acordo com os testes de Friedman realizados, verificamos para todos os casos que a hipótese nula é rejeitada, indicando que as amostras em cada nível de L não provém da mesma população, estando de acordo com os testes de média zero. Além disso, a característica "visual"dos testes de média zero nos dá mais informações do que os testes de Friedman, para o caso em estudo.

Uma idéia de comparar os algoritmos era a de efetuar algum teste para amostras pareadas, utilizando as amostras de cada bloco. Assumindo a hipótese nula que o algoritmo TF é igual ou melhor, identificaríamos a proporção de blocos em que o algoritmo TD se sairia melhor, e através de um teste para proporções (*sign test*, de Wilcoxon), verificaríamos se TD é melhor que TF com significância de 10%, mas a quantidade de amostras pareadas por bloco é de 5 pares, o que prejudicaria muito o poder do teste, além de que seria utilizado um teste não paramétrico, que em geral possui poder menor que um teste paramétrico. Devido a estas condições pouco favoráveis à confiança nos resultados do possível teste, o mesmo não foi realizado.

5.8 Conclusões Acerca do Experimento

Com base nos testes realizados neste documento, podemos afirmar com um nível de significância de 10% que:

- O algoritmo TD possui melhor desempenho do que o algoritmo TF para a variável de resposta Potência Média, nos níveis 0.1 e 0.2 do fator Restrição de Penalidade de Desempenho, mas para estes mesmos níveis de L o desempenho de TD é inferior a TF para a variável de resposta Restrição de Penalidade de Desempenho;

- O algoritmo TD possui melhor desempenho do que o algoritmo TF para a variável de resposta Penalidade de Desempenho Média, nos níveis 0.6, 0.7, 0.8, 0.9 e 1 do fator Restrição de Penalidade de Desempenho;

- O algoritmo TD possui melhor desempenho do que o algoritmo TF para a variável de resposta Potência Média, analisando os dados como um todo (independente do valor de L), para as duas variáveis de resposta.

Mesmo sendo inferior ou igual em alguns níveis de L quando observamos a variável de resposta Penalidade de Desempenho Média, o algoritmo TD respeita os níveis impostos pela própria restrição do fator L. Então os valores apresentados são aceitáveis. Desta forma, este experimento contribui como um indicador favorável para a utilização da política de TD em dispositivos móveis.

Por conta deste experimento ser uma simulação, há ausência de informações sobre o aumento do consumo de energia devido à utilização da política de TD, e se este fato poderia alterar o resultado real de uma comparação entre as políticas de TD e TF. Porém, os resultados obtidos abriram caminho para que novos experimentos fossem realizados com medição real de consumo de energia, os quais são descritos no próximo capítulo.

Capítulo 6

Viabilidade da Comparação Entre Modelos por Meio de Medição de Potência

Neste capítulo está descrito o experimento piloto realizado para identificação dos fatores que serão utilizados na comparação entre os algoritmos de TF e TD com medição real de potência. Devido à possibilidade de um número alto de repetições de cada ensaio para obter o nível de significância requerido, é provável que seja inviável a utilização dos mesmos fatores do experimento do Capítulo 4. Tanto a quantidade de fatores e de níveis de cada fator, quanto a quantidade de repetições em cada ensaio, para comparação entre as políticas de GDE, são identificados neste capítulo.

6.1 Objetivos do Experimento no *Template* GQM

Analisar dois algoritmos de gerenciamento dinâmico de energia para interfaces de rede baseados em timeout

com a intenção de compará-los

com respeito à sua eficácia

do ponto de vista dos usuários de aparelhos eletrônicos de consumo que dependem de bateria

no contexto de utilização de notebooks por usuários que acessam conteúdos diversos na

web e *offline*.

6.2 Definição de Hipóteses

Neste experimento, assim como no experimento do capítulo anterior, as métricas utilizadas para medição da eficácia de um algoritmo são dadas em Watts, para a potência média, e sem unidade, para a penalidade de desempenho média. A questão de pesquisa é:

P1: Os algoritmos com *timeout* dinâmico (TD) e com *timeout* fixo (TF) apresentam diferenças significativas de desempenho quando são testados em um dispositivo móvel?

Como temos duas variáveis de resposta que identificam o desempenho do algoritmo, duas hipóteses nulas foram formuladas:

- Hipótese Nula 1 ($H1_0$): o consumo de energia do notebook não sofre alteração com a mudança de políticas de GDE;
- Hipótese Nula 2 ($H2_0$): a penalidade de desempenho média dos algoritmos é igual. Ou seja, $\bar{l}(TF) = \bar{l}(TD)$.

Da mesma forma, as hipóteses alternativas também foram formuladas:

- Hipótese Alternativa 1 ($H1_1$): o consumo de energia dos algoritmos é diferente;
- Hipótese Alternativa 2 ($H2_1$): $\bar{l}(TD) < \bar{l}(TF)$.

Caso as hipóteses nulas sejam rejeitadas, pretende-se identificar o algoritmo com melhor desempenho com o auxílio do projeto de experimento descrito a seguir.

6.2.1 Obtenção dos Parâmetros para o Cálculo de Potência Média e Penalidade de Desempenho Média

Na implementação das políticas de *timeout* fixo e dinâmico, alguns parâmetros característicos do dispositivo móvel utilizado são necessários. De acordo com a Equação (3.6), o valor esperado da potência média, dado um *timeout* δ, depende das constantes P_{on}, P_{off}, E_s e T_s, e para encontrar os seus valores, foi medida a potência consumida durante a variação de estados da interface de rede.

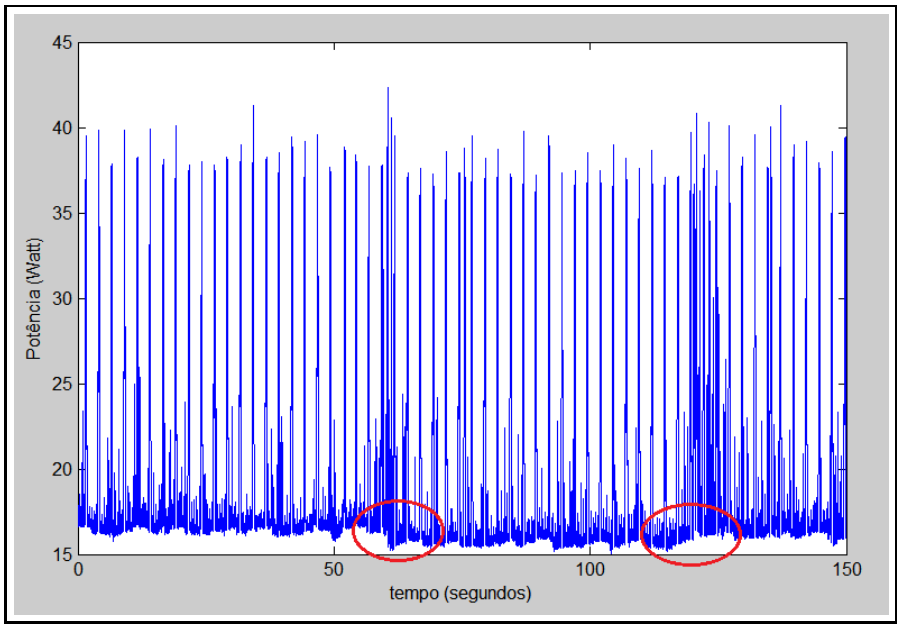

Figura 6.1: Observação da potência consumida pelo dispositivo móvel durante as alterações do estado da interface de rede.

A Figura 6.1 exibe a potência consumida num intervalo de dois minutos e meio com a interface de rede tendo o seu estado alternado de acordo com a seguinte rotina:

- Início: interface de rede ligada e ociosa;
- 1º minuto: interface de rede colocada em modo de baixo consumo (primeiro círculo vermelho da figura);
- 2º minuto: interface de rede reativada, e mantida ociosa (segundo círculo vermelho da figura);

Os dados obtidos nesta medição apresentaram os seguintes valores de parâmetros para o dispositivo móvel utilizado:

- $P_{on} = 18,6192 Watt$;
- $P_{off} = 18,0182 Watt$;
- $E_s = 215,0882 Joule$;
- $T_s = 10,5 segundos$.

O dispositivo utilizado nos experimentos é um *notebook* Positivo Premium Select modelo 8010, com processador Intel Core i7-620M, 4GB de memória RAM, 640GB de HD e sistema operacional Microsoft Windows 7 Home Premium.

6.2.2 Repetições

Pretende-se analizar os resultados deste experimento com um nível de significância $\alpha = 10\%$, seguindo o mesmo padrão do experimento realizado no Capítulo 4. Para garantir este valor de α, foi utilizado o método descrito em Jain et al. [8] onde encontra-se o número mínimo n de repetições necessárias com a equação (6.1). Com este número de repetições, espera-se que o experimento atenda ao requisito desejado, que no neste caso se trata de um nível de confiança $\rho = 90\%$.

$$n = (\frac{z_{1-\alpha/2} \times s}{\alpha \times \mu})^2 \tag{6.1}$$

onde,

- $z_{1-\alpha/2} = 1,65$ é o valor que determina 90% de nível de confiança;
- $\mu = 18,6192$ é a potência média para as n repetições, com a interface rede ligada e ociosa;
- $s = 5.6775$ é o desvio padrão das médias para as n repetições.

Aplicando os valores na equação (6.1), obtém-se aproximadamente um número mínimo de repetições $n = 25$. Ou seja, cada ensaio delineado no planejamento do experimento deve ser repetido pelo menos 25 vezes para que os dados obtidos tenham um nível de confiança de 90%.

6.3 Planejamento do Experimento

O experimento consiste na medição do consumo de um *notebook* com uma política instalada. Durante o experimento, é executado um *script* que representa a utilização do *notebook* com algumas atividades cotidianas, como edição de arquivos de texto, navegação na web, reprodução de vídeos *online* e *offline*, dentre outras que serão descritas posteriormente.

Aproveitando os resultados do experimento realizado no Capítulo 4, verifica-se que a política de TD obteve menor consumo de energia para os níveis 0,1 e 0,2 do fator restrição de penalidade de desempenho. Os valores de penalidade de desempenho média foram mais elevados do que os da política de TF, mas ainda estão obedecendo o limite L. Portanto, neste experimento é levado em conta um nível do fator L devido ao tempo necessário para a realização do mesmo, dado que cada ensaio é repetido 25 vezes para atingir um nível de confiança de 90%. A Tabela 6.1 mostra os ensaios do experimento levando em consideração os níveis de cada fator, mas ignorando a quantidade de repetições para cada combinação de fatores.

Nº do Tratamento	Algoritmo	Restrição de Penalidade de Desempenho
01	TF	0,1
02	TD	0,1
03	Nenhuma política de GDE	-

Tabela 6.1: Tratamentos realizados no segundo experimento.

6.3.1 Fatores e variáveis de resposta

Os fatores controlados durante o experimento são:

- Algoritmo (A)
  - 3 níveis: Algoritmo de Timeout Dinâmico (TD), Algoritmo de Timeout Fixo (TF), nenhum algoritmo;
- Restrição de penalidade de desempenho (L): limiar que indica o máximo de penalidades de desempenho que a política deve alcançar;
  - 1 nível: 0,1.

As variáveis de resposta observadas:

- Potência média (Watt);
- Penalidade de desempenho média.

6.3.2 Forma de coleta dos dados e análise

Coleta dos dados

O equipamento utilizado para a coleta dos dados é chamado DAQ, o qual obtém do *notebook*, cujo consumo é avaliado, a sua potência, corrente e tensão durante todo o experimento. Os dados coletados são enviados para um computador que os recebe, exibe em tempo de execução e os armazena para análise posterior. A plataforma experimental é montada conforme o layout da Figura 6.2.

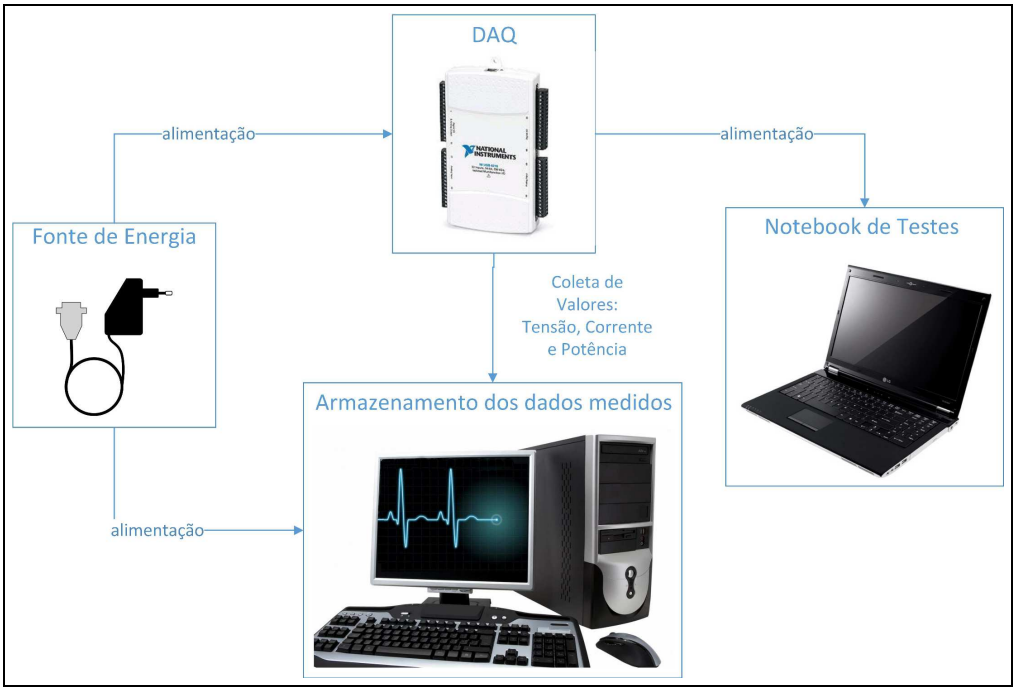

Figura 6.2: Plataforma experimental.

Análise dos dados

A análise dos dados leva em conta praticamente os mesmos passos da análise do Capítulo 4. Para comparar as duas políticas de GDE, é feito:

- Estimação de erros;
- Estimação de intervalos de confiança provenientes da eficácia de cada algoritmo;

- Verificação da normalidade dos dados para identificar que prova estatística será mais adequada;
- Determinação se a variação encontrada entre os valores de potência média e penalidade de desempenho média para cada uma das técnicas é significativa ou não.

6.3.3 Análise de ameaças à validade

Neste experimento são consideradas as seguintes ameaças à validade:

- Má implementação do algoritmo otimizado (utilização de *loops* e/ou chamadas recursivas, por exemplo, que podem acarretar maior consumo dependendo do contexto);
- Aplicação de terceiros também acessando a internet, afetando a aleatoriedade do fator estorvo;
- Aplicação de terceiros em execução em *background* (como o *software* Antivirus, no caso do Sistema Operacional Windows, por exemplo) pode aumentar o consumo de energia, influenciando diretamente a variável de resposta;
- Realização de menos repetições que o esperado devido ao custo das repetições, diminuindo o *power* do experimento e podendo implicar em afirmações inválidas.

6.4 Análise dos Dados

Da mesma forma do experimento do capítulo anterior, deve ser realizado o pareamento das amostras para comparação. Além disso, já se sabe que os dados não possuem distribuição bem definida, então são realizados testes não paramétricos e análises de intervalos de confiança.

6.5 Conclusões Acerca do Experimento

Segundo o experimento piloto usado para verificação da viabilidade da medição do consumo das políticas de GDE, foi encontrado um valor n relativamente elevado ($n = 25$) para a

quantidade de repetições de cada ensaio, de forma que seja obtido um nível de confiança de 90% na comparação das políticas. Para minimizar o custo do tempo de execução na realização do experimento comparativo, foi descartada a possibilidade de variação dos níveis do fator Restrição de Penalidade de Desempenho (L), mantendo-o com um valor fixo 0,1.

Capítulo 7

Considerações Finais

As interfaces de rede estão presentes em uma variedade de sistemas computacionais alimentados à bateria, tais como *notebooks*, *ultrabooks*, *tablets* e *smartphones*. Considerando-se que a vida útil da bateria é uma limitação, há oportunidades para o gerenciamento de energia reduzindo-se o consumo de energia de interfaces de rede ociosas.

Este trabalho apresentou uma nova abordagem para redução do consumo de energia em dispositivos móveis visando a utilização de interfaces de rede sem fio de forma mais eficiente. A política apresentada é um modelo adaptativo de outra política proposta anteriormente, e foi elaborada levando em consideração técnicas que minimizam a utilização de recursos computacionais.

Os resultados obtidos no primeiro experimento, descrito no Capítulo 4, demonstraram que a política proposta é viável, o que levou ao planejamento da execução do segundo experimento, descrito no Capítulo 5, o qual não foi concluído até a apresentação deste documento. Comparando os resultados das duas políticas analizadas, verifica-se que a política de *timeout* dinâmico apresenta redução de consumo para valores específicos de restrição de penalidade de desempenho e se mostra uma ferramenta útil para obtenção de minutos a mais de utilização do dispositivo móvel.

Para trabalhos futuros, pretende-se concluir o experimento especificado no Capítulo 5 para comprovação da eficácia da política de *timeout* dinâmico. Além disso há a possibilidades de refinamento da referida política de GDE no que diz respeito a: melhoria do módulo classificador (ver Seção 3 do Capítulo 3) por meio de técnicas de *machine learning*, tais como redes neurais e *support vector machines* (SVM); e estudo de viabilidade da estimação

do momento adequado para religamento da interface de rede, visando a minimização das penalidades de desempenho.

Além do refinamento da política proposta, há relevância na exploração da sua eficácia em outros tipos de dispositivos móveis. Sendo assim, testes em *tablets*, *smartphones* e outros eletrônicos de consumo que se encaixam no perfil onde a política se torna aplicável são uma possibilidade válida e de curto/médio prazo.

Bibliografia

[1] Victor R. Basili. Software modeling and measurement: the Goal/Question/Metric paradigm. Technical report, Techreport UMIACS TR-92-96, University of Maryland at College Park, College Park, MD, USA, 1992.

[2] Luca Benini, Alessandro Bogliolo, and Giovanni De Micheli. A survey of design techniques for system-level dynamic power management. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(3):299–316, 2000.

[3] Luca Benini, Alessandro Bogliolo, A Paleologo, and Giovanni De Micheli. Policy optimization for dynamic power management. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(6):813–833, 1999.

[4] Le Cai and Yung-Hsiang Lu. Energy management using buffer memory for streaming data. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(2):141–152, 2005.

[5] Vinay Devadas and Hakan Aydin. On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications. *Computers, IEEE Transactions on*, 61(1):31–44, 2012.

[6] Gaurav Dhiman and Tajana Simunic Rosing. Dynamic power management using machine learning. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 747–754. ACM, 2006.

[7] Shao-Min Huang, Chen Hu, and Long-Ning Qi. Predictive dynamic power management policy with consideration of outlying data. In *Solid-State and Integrated Circuit Technology, 2006. ICSICT '06. 8th International Conference on*, pages 2064–2066. IEEE, 2006.

[8] R. K. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1 edition, 1991.

[9] Q Jiang, H-S Xi, and B-Q Yin. Adaptive optimisation of timeout policy for dynamic power management based on semi-markov control processes. *IET control theory & applications*, 4(10):1945–1958, 2010.

[10] Branislav Kveton, Prashant Gandhi, Georgios Theocharous, Shie Mannor, Barbara Rosario, and Nilesh Shah. Adaptive timeout policies for fast fine-grained power management. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 1795. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[11] Yung-Hsiang Lu and Giovanni De Micheli. Comparing system level power management policies. *Design & Test of Computers, IEEE*, 18(2):10–19, 2001.

[12] Saulo O D Luiz, Angelo Perkusich, Bruna M J Cruz, Breno H M Neves, and G M da S Araujo. Optimization of timeout-based power management policies for network interfaces. *Consumer Electronics, IEEE Transactions on*, 59(1):101–106, 2013.

[13] S. O. Park, J. K. Lee, J. H. Park, and S. J. Kim. Adaptive power management system for mobile multimedia device. *IET Communications*, 6(11):1407, 2012.

[14] Nathaniel Pettis, Le Cai, and Yung-Hsiang Lu. Statistically optimal dynamic power management for streaming data. *Computers, IEEE Transactions on*, 55(7):800–814, 2006.

[15] Yan Wang and Xiangheng Shen. Adaptive dynamic power management policy: Two value alternate basis. *Advanced Science Letters*, 11(1):783–786, 2012.

[16] Wang Yue, Zhao Xia, and Chen Xiangqun. A task-specific approach to dynamic device power management for embedded system. In *Embedded Software and Systems, 2005. Second International Conference on*, pages 7–pp. IEEE, 2005.

[17] Murtaza Zafer and Eytan Modiano. Minimum energy transmission over a wireless channel with deadline and power constraints. *Automatic Control, IEEE Transactions on*, 54(12):2841–2852, 2009.