CAMILA GONZAGA DE ARAUJO

MODELOS PARA AVALIAÇÃO DE DISPONIBILIDADE EM AMBIENTE DE INTERNET DAS COISAS: UM ESTUDO APLICADO EM SERVIÇOS *MHEALTH* UTILIZANDO DISPOSITIVO *WEARABLE*

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE

2017

Camila Gonzaga de Araujo

Modelos para Avaliação de Disponibilidade em Ambiente de Internet das Coisas: um estudo aplicado em serviços *mHealth* utilizando dispositivo *Wearable*

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Dr. Paulo Romero Martins Maciel*

RECIFE

2017

Camila Gonzaga de Araujo

Modelos para Avaliação de Disponibilidade em Ambiente de Internet das Coisas: um estudo aplicado em serviços *mHealth* utilizando dispositivo *Wearable*

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 06/03/2017

BANCA EXAMINADORA

Prof. Dr. Ricardo Massa Ferreira de Lima
Centro de Informática/UFPE

Prof. Dr. Paulo Antonio Leal Rego
Departamento de Sistemas de Informação/UFC

Prof. Dr. Paulo Romero Martins Maciel
Centro de Informática/UFPE
(Orientador)

Resumo

A convergência tecnológica entre a Internet das Coisas (IoT), dispositivos móveis e vestíveis (*wearable*) resultou, nos últimos anos, no considerável aumento pelo interesse em serviços e aplicações de saúde móvel (*Mobile Health - mHealth*). Estes dispositivos ainda apresentam recursos limitados e uma das soluções adotadas para superar estas limitações tecnológicas é recorrer ao poder da *Mobile Cloud Computing (MCC)*. A MCC é um paradigma que tem como característica prover recursos computacionais da nuvem para estender a capacidade dos dispositivos móveis. Neste contexto, é preciso destacar que a variedade de componentes (dispositivos *wearable*, dispositivos móveis, nuvem e interfaces de rede) e a interação entre estes trazem desafios para assegurar níveis desejados de disponibilidade.

Dessa forma, esse trabalho tem como proposta realizar a avaliação de disponibilidade de um ambiente IoT, utilizando a tecnologia baseada em MCC para o provimento do serviço *mHealth*, com a finalidade de auxiliar o planejamento de infraestruturas *mHealth*.Para isto, o trabalho propõe modelos hierárquicos para avaliar a disponibilidade desse serviço. Inicialmente, definiu-se a Arquitetura Base, sem mecanismos de redundância, a qual foi modelada a partir de um modelo hierárquico composto de *Reliability Block Diagram (RBD)* e de *Continuous Time Markov Chain (CTMC)*, e validado através de um *testbed* de injeção de falhas e reparos. A Arquitetura Base é composta por um *smartwatch*, um *smartphone* e infraestrutura de nuvem. Com base no modelo da Arquitetura Base, aplicamos a técnica de análise de sensibilidade para identificar gargalos de disponibilidade. A partir dos resultados obtidos, por intermédio da análise de sensibilidade, apresentamos duas adaptações da Arquitetura Base: uma arquitetura com múltiplas interfaces de rede e uma arquitetura baseada em *Cloudlet*. As duas adaptações são avaliadas em termos de disponibilidade e *downtime* anual. Por fim, avaliamos a confiabilidade e disponibilidade dos dispositivos móveis, bem como investigamos os efeitos do uso de diferentes interfaces de conexão sobre a autonomia das baterias e sua relação com a disponibilidade do serviço.

Os referidos resultados expressam que a Arquitetura *Cloudlet* proporcionou aumento da disponibilidade do serviço, em comparação com a Arquitetura Base. Também podemos inferir que, dentre as variações de diferentes interfaces de conexões, quando o usuário possui conexão *Bluetooth* e conectividade com Banda Larga e Rede Móvel ativas, a disponibilidade do serviço apresenta melhores resultados. A principal contribuição desta pesquisa é a proposição de modelos para a avaliação de disponibilidade de um ambiente IoT, os quais poderão ser utilizados como subsídio para o planejamento de infraestruturas de serviços *mHealth* na nuvem.

Palavras-chave: Modelos Hierárquicos. Análise de Disponibilidade. Computação Móvel em Nuvem. Dispositivos Vestíveis. Internet das Coisas.

Abstract

The technological convergence between the Internet of Things (IoT), mobile devices and wearable devices has, last few years, resulted in the considerable increase by the interest in services mobile health care applications (mHealth). The mobile devices are still quite resource-constrained and Mobile Cloud Computing (MCC) intends to mitigate mobile computing challenges. MCC is a paradigm that offers cloud computing for extending the capacity of mobile devices. The architecture of MCC includes a variety of components, such as: wearable devices, mobile devices, cloud and network interfaces. The interaction between these components brings challenges to guarantee desired levels of availability.

In this way, this research proposes perform the availability evaluation of an IoT environments, using MCC-based technology, for the provision of the a mHealth service, with the goal of assist in mHealth planning. For that aim, this work proposes hierarchical models to evaluate the availability of this service. Initially, a Basic Architecture is specified with no redundancy. The Basic Architecture was modeled with a hierarchical model composed by Reliability Diagram Models (RBD) and Continuous Time Markov Chains (CTMC). Additionally, the Basic Architecture was validated by a fault injection approach. The Basic Architecture is composed of a smartwatch, a smartphone and a cloud infrastructure. Based on the Basic Architecture model, a sensitivity analysis was performed to find availability bottlenecks. Based on the sensitivity analysis we proposed two extended versions of the Basic Architecture: an architecture with multiple network interfaces and a Cloudlet-based Architecture. These two extensions were evaluated in terms of availability and annual downtime. Finally, we evaluated the reliability and availability of mobile devices, as well as investigating the effects of using different connection interfaces on battery autonomy and its relation to service availability.

These results have indicated that Cloudlet-based Architecture increased the service availability compared to the Basic Architecture. By varying the interfaces the experiments have indicated one configuration with superior availability. The configuration was composed by a Bluetooth connection, active Broadband and a Mobile Network connectivity. The main contribution of this work is the proposed model for evaluating availability of IoT environments, used as input for the planning of infrastructure mHealth systems.

Keywords: Hierarchical Models. Availability Analysis. Mobile Cloud Computing. Wearable Device. Internet of Things.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

Sumário

1

Introdução

A área da saúde está em constante evolução e, frequentemente, as chamadas tecnologias de ponta são desenvolvidas com o propósito de serem aplicadas em suas práticas cotidianas. Neste contexto, um sistema de saúde computacional ou eletrônico é classificado como *eHealth* (*Electronic Health*) (ADIBI, 2015). A *eHealth* tem como premissa a utilização de sistemas computacionais como ferramenta para a elaboração de informações ligadas à saúde do paciente (ADIBI, 2015).

A intersecção, entre o *eHealth* e a popularização dos dispositivos móveis, propiciou o advento do conceito de *Mobile Health* (mHealth)(ADIBI, 2015). Os sistemas *mHealth* modificaram o paradigma dos sistemas de saúde, impulsionados por sua variedade de características, como a facilidade de conectividade, a portabilidade e a utilização de sensores capazes de auxiliar no processo de quantificação dos dados físicos dos pacientes (ADIBI, 2015; SALVI, 2015).

Os dispositivos móveis tais como: *smartphones* e *tablets*, não possuem recursos suficientes para esse processo de quantificação dos dados, como por exemplo, o monitor de frequência cardíaca. Em contrapartida, os dispositivos vestíveis (*wearable*), como *smartwatch*, possuem uma diversidade maior de sensores, por este motivo estão sendo incorporados em aplicações referente à saúde, tais como: monitoramento de pacientes acamados (HUANG; HSU, 2005), cuidados com idosos (MOHAMED; CHOI; IRAQI, 2014), acompanhamento de gestantes (PENDERS et al., 2015), entre outros (SU; GURURAJAN, 2010; GATZOULIS; IAKOVIDIS, 2007; LYMBERIS; DITTMAR, 2007).

O *smartwatch* é projetado para trabalhar de forma complementar ao *smartphone*, conectado por meio de uma rede sem fio e podendo, eventualmente, fornecer dados diretamente no visor do relógio de modo independente. O *smartwatch* possui sistema operacional e aplicativos próprios, além de funcionalidades tais como: realizar chamadas; enviar mensagens de texto; acessar as informações meteorológicas; fornecer dados de atividades físicas, geolocalização, entre outros (STROUD, 2015).

Neste cenário, os dados dos pacientes são monitorados utilizando os sensores de seus próprios dispositivos pessoais de saúde (que são conectados à internet) e compartilhando-os com seus médicos remotamente. Ao conectar dispositivos com diferentes recursos a uma rede,

potencializa-se o surgimento de novas aplicações. Os sensores inteligentes e as aplicações pervasiva permitem que a vida cotidiana seja totalmente conectada, em que uma variedade de coisas ou objetos ao nosso redor podem interagir para criar um ambiente onipresente (SILVA et al., 2013). Neste sentido, este cenário pode ser caracterizado como Internet das Coisas (*Internet of things* (IoT)) (ATZORI; IERA; MORABITO, 2010).

A ideia básica do conceito de IoT é a existência difundida de uma variedade de coisas ou objetivos, todos integrados, interagindo um com os outros, de forma a atingir objetivos comuns, criando um ambiente mais onipresente (ATZORI; IERA; MORABITO, 2010). Com isto, emerge uma nova gama de aplicações, tais como: coleta de dados e monitoramento de pacientes, sensoriamento de ambientes de difícil acesso e inóspitos, entre outras.

Vale salientar que tanto os dispositivos móveis quanto os dispositivos vestíveis possuem recursos computacionais limitados, principalmente quanto à capacidade de processamento e armazenamento (SATYANARAYANAN et al., 2009). Portanto, a computação em nuvem surge como uma alternativa, tendo em vista que uma das suas principais características é a capacidade de fornecer recursos computacionais em abundância para seus clientes (ADIBI, 2015; ARMBRUST et al., 2010). Sob esta premissa, surge o paradigma *Mobile Cloud Computing* (MCC), que possui a capacidade de prover recursos computacionais da nuvem para estender a capacidade de dispositivos móveis (KUMAR; LU, 2010; SATYANARAYANAN et al., 2009).

Contudo, apesar da alta heterogeneidade e a larga distribuição dos dispositivos com recursos que a IoT e a computação em nuvem podem oferecer, é preciso destacar que a variedade de componentes (como o *smartwatch*, o *smartphone* e a própria *nuvem*) existente nesse cenário pode produzir desafios, principalmente, sob a perspectiva da dependabilidade do serviço. Isto é, deve ser assegurado que cada componente que faz parte da arquitetura do serviço tenha seu funcionamento adequado, mitigando o comprometimento do serviço como um todo.

A dependabilidade é a capacidade de um sistema computacional fornecer um serviço de forma confiável (LAPRIE, 1992). De acordo com AVIZIENIS; LAPRIE; RANDELL (2001), a dependabilidade de um sistema é a habilidade de evitar falhas de serviços que são mais frequentes ou mais severas do que o aceitável. A dependabilidade de um sistema está relacionada a uma série de requisitos não funcionais que o serviço deverá garantir (AVIZIENIS; LAPRIE; RANDELL, 2001), estes requisitos são: confiabilidade, disponibilidade, segurança, confidencialidade, integridade e manutenabilidade (MACIEL et al., 2011). Neste trabalho, são abordados os requisitos de confiabilidade e disponibilidade.

Atingir níveis satisfatórios de dependabilidade em ambientes IoT com serviços em nuvem não é uma tarefa trivial. Entretanto, para avaliar estes níveis, o uso de modelos é considerado eficaz, uma vez que prevê o comportamento dos componentes envolvidos. Visto que tal avaliação pode contribuir com melhorias na qualidade do serviço provido e no planejamento de infraestruturas, este trabalho concentra esforços na proposição de modelos, com o intuito de avaliar a disponibilidade de um serviço em nuvem para um cenário *mHealth* em ambientes IoT,

composto por três camadas, sendo duas camadas de dispositivos[1] (vestível e móvel) e a outra camada a infraestrutura em nuvem.

Para tanto, adotamos uma modelagem hierárquica heterogênea utilizando modelos compostos por *Reliability Block Diagram* (RBD) e *Continuous Time Markov Chain* (CTMC). O modelo RBD principal tem a finalidade de avaliar a disponibilidade do serviço de forma geral e o modelo *Continuous Time Markov Chain* (CTMC) de estimar a autonomia da bateria dos dispositivos móveis. A validação foi realizada tanto para os modelos CTMCs da bateria dos dispositivos quanto para o modelo RBD. Além disso, uma análise de sensibilidade foi utilizada para identificar gargalos de disponibilidade do serviço.

Por fim, avaliamos a confiabilidade e disponibilidade dos dispositivos móveis, bem como investigamos os efeitos do uso de diferentes interfaces de conexão sobre a autonomia das baterias e sua relação com a disponibilidade do serviço.

1.1 Motivação e Justificativa

Um sistema *mHealth* tem como suas principais características otimizar os cuidados com a saúde do paciente de forma onipresente, não restringindo-os ao consultório médico, e reduzir custos médicos, evitado deslocamentos em alguns casos. Entretanto, existem muitos desafios no planejamento de infraestruturas que dão suporte aos serviços *mHealth*, como a disponibilidade dos dados dos pacientes (ZOU et al., 2007).

A utilização de dispositivos *wearable* para medição de dados de saúde proporciona uma gama de benefícios, principalmente, quando são utilizados sensores não invasivos, vez que tendem a ser melhor aceitos entre os usuários. Neste caso, o *smartwatch* torna-se conveniente, pois possui funcionalidades essenciais para o acompanhamento dos usuários. Com o uso do *smartwatch* além de ter acesso aos sensores, é possível ter um *feedback* das condições físicas em um tempo de resposta hábil.

Os diferentes modelos de negócio existentes na área da saúde possibilitam o desenvolvimento dos mais variados tipos de estratégias de monitoramento, preocupados em oferecer cada vez mais, versatilidade e funcionalidade aos produtos e serviços (PINOCHET; LOPES; SILVA, 2014). No que se refere à utilização do método MCC, a preocupação maior é com as interrupções dos serviços, haja vista, que mesmo ocorrendo de forma esporádica, tais falhas podem causar prejuízos, como o não monitoramento do usuário por determinado período de tempo (DOUKAS; PLIAKAS; MAGLOGIANNIS, 2010). Assim, sistemas *mHealth* em um ambiente MCC, precisam ser capazes de prover serviços com alta disponibilidade, ou seja,

[1] O conceito de dispositivos móveis e dispositivos vestíveis são diferentes. Sendo assim, nem todo dispositivo móvel é vestível e nem todo dispositivo vestível é móvel. Neste trabalho, utilizaremos um *smartwatch*, como dispositivo vestível (que também pode ser considerado como móvel, devido às suas características) e um *smartphone*, como dispositivo móvel. Logo, em futuras referências, podemos utilizar o termo "dispositivos" ou "dispositivos móveis" quando nos referirmos a ambos, simultaneamente. Quando no singular, utilizaremos o termo "dispositivo vestível" para o *smartwatch* e "dispositivo móvel" para o *smartphone*.

precisam manter seus serviços disponíveis por um máximo de tempo possível.

Com o intuito de representar computacionalmente os principais aspectos dos sistemas *mHealth*, a modelagem pode fornecer subsídios para o planejamento e avaliação da dependabilidade, abstraindo complexidades técnicas (JAIN, 1990). A avaliação dos atributos da dependabilidade, através de modelos, é um meio viável para propor melhorias aos sistemas de computação em nuvem. Portanto, a modelagem é um meio que possibilita a avaliação de diversos tipos de sistemas, inclusive, os que demandem uma alta disponibilidade dos serviços (KIM; MACHIDA; TRIVEDI, 2009).

A utilização de modelos que tenham o objetivo de auxiliar no planejamento de infraestruturas de serviço *mHealth* na nuvem em um ambiente IoT contribui para o alcance de um alto índice de disponibilidade e evita a perda de informações, entre outras consequências indesejadas que só tendem a ocorrer quando há a indisponibilidade do serviço (KIM; MACHIDA; TRIVEDI, 2009). Questões como confiabilidade e disponibilidade tornaram-se fatores cruciais para a garantia da continuidade desse tipo de serviço (SILVA et al., 2013).

Neste cenário, é importante construir soluções capazes de suprir a demanda de informações que são geradas, ainda mais, por se tratar de informações vitais para a saúde do usuário. De tal modo, é indispensável que sejam analisadas previamente todas as características que possam afetar a disponibilidade do sistema. Assim, a principal motivação deste trabalho é definir um modelo formal para representar e avaliar a disponibilidade de um serviço *mHealth* na nuvem, em um ambiente IoT, onde a fonte de informação advém de sensores contidos em um *smartwatch*.

1.2 Objetivos

O objetivo principal desta pesquisa é a proposição de modelos para a avaliação de disponibilidade para prover suporte ao planejamento de infraestruturas de serviços *mHealth* na nuvem, em um ambiente IoT, sendo um *smartwatch* o principal mecanismo de coleta de dados do usuário.

De forma mais específica este trabalho se propõe a:

- Propor e avaliar modelos para a avaliação de disponibilidade em um ambiente IoT para prover serviço *mHealth* na nuvem;
- Validar o modelo da Arquitetura Base alcançada através do experimento de injeção de falhas;
- Realizar análise de sensibilidade na Arquitetura Base, com intuito de identificar os componentes críticos;
- Propor e avaliar extensões da Arquitetura Base, com o objetivo de identificar alternativas que podem melhorar a disponibilidade do serviço;

- Elaborar modelos para a avaliação da autonomia da bateria dos dispositivos móveis e para a avaliação da disponibilidade e confiabilidade dos mesmos.

1.3 Trabalhos Relacionados

Esta seção apresenta uma análise sobre um conjunto de trabalhos coletados, através de um levantamento bibliográfico, no qual são realçados os pontos que foram abordados e as técnicas utilizadas por estes. Sendo os temas destes trabalhos relacionados a: "modelos de avaliação de disponibilidade e desempenho para dispositivos *wearable*, especificamente o dispositivo *smartwatch*" e "sistemas móveis com ênfase em *mHealth* ".

As etapas do processo realizadas no presente estudo são descritas na Figura 1.1. A

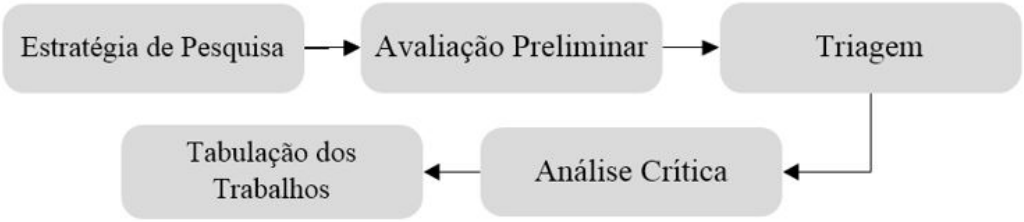


Figura 1.1 Etapas do processo dos trabalhos relacionados.

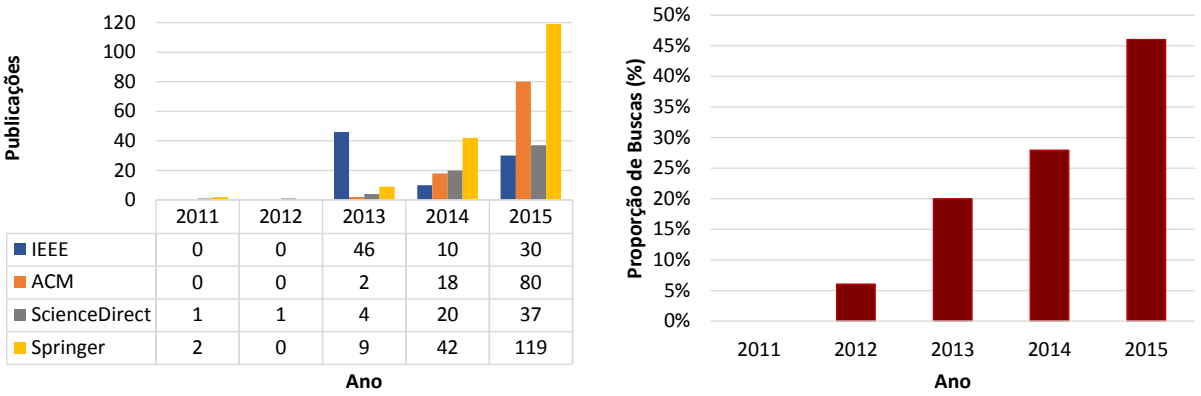
estratégia de pesquisa é a etapa de definição de busca, usada para coletar trabalhos similares, através de alguns critérios definidos para selecionar os estudos mais relevantes. Para tanto, nossa estratégia foi aplicada no período de setembro a dezembro de 2015, filtrando as publicações em quatro bases de dados eletrônicas indexadas: *IEEEXplore, ACM, Springer e ScienceDirect*. Assim como também foram feitas buscas eletrônicas no *Scholar Google*. Para tal, aplicamos um filtro de busca definido através de termos e palavras chaves. A *string* de busca resultante é representada pelo Código 1.1:

Código 1.1 String de busca.

```
"Analytical Modeling" AND
("mHealth" OR "Mobile Cloud Computing") AND
("Dependability" OR "Availability" OR "Reliability") AND
("Smartwatch" OR "Wearable")
```

Após a extração dos artigos, foi realizada uma **avaliação preliminar** onde foi classificado cada resultado de pesquisa, através de seu contexto e relevância em relação à pesquisa. Sendo assim, este levantamento recolheu aproximadamente 50 documentos científicos que possuem relação direta com o tema pesquisado. Dessa forma, foi possível identificar a existência crescente de pesquisas em *mHealth* utilizando dispositivos *wearables* e computação móvel, embora boa parte destes tenham como foco principal estudos ligados a interface do usuário, a tolerância de falhas, a aplicações voltadas à saúde de pacientes, à segurança da informação, entre outros.

A Figura 1.2 (a) apresenta a quantidade de publicações relacionadas ao tema desta pesquisa nas principais bases científicas, no período de 2011 à 2015, e a Figura 1.2 (b) ilustra o interesse pelo tema nas buscas realizadas na ferramenta Google, no mesmo período. Logo, é possível observar um interesse crescente na área.



(a) Bases de dados eletrônicas

(b) Buscas *Google Trends*

Figura 1.2 Buscas nas bases de dados

Depois de recolher os estudos, foi realizado uma **triagem**, a partir da aplicação de critérios de exclusão, a fim de restringir a amostra a um grupo de trabalhos mais relevantes. Em geral, os critérios de exclusão incluem: (i) artigos publicados anteriores ao ano de 2011, (ii) trabalhos não revisados pela comunidade acadêmica; (iii) trabalhos não escritos no idioma inglês; (iv) propostas não originais; (v) versões duplicadas; (vi) pesquisas não relacionadas a técnicas de dependabilidade para dispositivos *smartwatch* e MCC; e (vii) pesquisas não relacionadas a sistemas móveis com ênfase em *mHealth* apoiado pelo paradigma MCC. Assim, dentre todos os documentos analisados, apenas 10 deles foram considerados relevantes e, que estão diretamente relacionados com a presente pesquisa ou podem contribuir em algum nível de relevância com nosso estudo.

Por fim, através do conjunto de trabalhos selecionados, foi realizada uma **análise crítica** de seus conteúdos finalizando-os com uma **tabulação dos trabalhos** onde apresentamos os parâmetros adotados em cada trabalho em comparação com o presente estudo.

1.3.1 Resultados

Trabalhos que abordam a avaliação de disponibilidade em ambientes *mobile cloud* com aplicações *wearable* para *mHealth* são facilmente encontrados na literatura atual, contudo, são escassos os trabalhos que fazem uso de modelagem e técnicas de dependabilidade neste cenário. Com a crescente popularidade dos dispositivos *wearables*, alguns trabalhos científicos abordaram o tema sob outras perspectivas, entre estes se destacam as pesquisas relacionadas com as áreas da saúde e atividades físicas.

Na obra de MAGLOGIANNIS et al. (2014), por exemplo, há o objetivo de apresentar um

sistema de lembrete eletrônico, com informações de saúde, para dispositivos *smartwatch*, que se comunicam via *Bluetooth* com o *smartphone*. Tão logo, a principal funcionalidade do sistema envolve a criação de lembretes usando um computador ou o próprio dispositivo *Android*. Os lembretes são armazenados em uma infraestrutura de nuvem que também possibilita outros tipos de usuários com o papel de assistente de saúde, tais como o médico e o farmacêutico, que ficam aptos a utilizar o sistema criando ou atualizando os lembretes. Para tal autor, essa assistência é voltada principalmente para os grupos idosos ou de pessoas com deficiências sensoriais significativas. Para a implementação do projeto foi adotado um protótipo de *smartwatch* na plataforma *Pebble* que recebe as notificações em forma de áudio, alertas visuais ou vibrações.

Já a pesquisa desenvolvida por DOUKAS; PLIAKAS; MAGLOGIANNIS (2010) implementa um aplicativo móvel, denominado de *@HealthCloud*, e serviços de gerenciamento de informação relativos à saúde baseado em MCC. A pesquisa desenvolvida avalia o desempenho das aplicações *mHealth* em sistemas *Android*, com e sem a utilização de recursos de computação em nuvem. A medição de desempenho foi realizada através do tempo de comunicação entre o dispositivo móvel e o servidor de aplicação, considerando as tecnologias de comunicação Banda Larga e Rede Móvel.

No trabalho de CHEN et al. (2014) foi apresentado o monitoramento e o desempenho, em tempo real, de um treino de maratona, utilizando dispositivo *wearable*, para monitorar a velocidade do ritmo cardíaco, analisando o estado fisiológico do corredor e sugerindo adaptações apropriadas na intensidade do treino.

A alta disponibilidade do serviço em nuvem é uma caraterística muito importante para um sistema *mHealth* e a nuvem (OLIVEIRA et al., 2013). Além disso, a eficiência energética é uma propriedade fundamental de qualquer dispositivo móvel, portanto, algumas pesquisas visam avaliar quais as estratégias que devem ser aplicadas para melhorar este aspecto nos dispositivos móveis em ambientes MCC (OLIVEIRA et al., 2013; ARAUJO et al., 2014; SILVA et al., 2014).

O trabalho desenvolvido por OLIVEIRA et al. (2013) propõe uma metodologia para a avaliação de disponibilidade e consumo energético para ambientes MCC. Avaliou-se o impacto da comunicação sem fio e do consumo de energia sobre a disponibilidade de sistemas, através de modelagem hierárquica heterogênea, combinando *Stochastic Petri Net* (SPN) e modelos *Reliability Block Diagram* (RBD). O estudo foi realizado em diferentes cenários, considerando as tecnologias de comunicação sem fio: 3G e *WiFi*. Desta forma, foi possível verificar em qual momento a conexão 3G ou *WiFi* é mais adequada. Os resultados demonstram que o 3G provoca uma menor disponibilidade e também é o principal responsável pela descarga da bateria, porém, quando combinado com o protocolo *WiFi* em uma redundância de comunicação, mostra melhores resultados de disponibilidade e de tempo de operação de bateria. Entretanto, este trabalho não foca em nenhuma categoria de serviço específico.

Do mesmo modo, é abordado no trabalho de ARAUJO et al. (2014) a utilização da modelagem hierárquica para a avaliação de disponibilidade em ambiente MCC. Os autores propõem um modelo de alto nível que caracteriza o comportamento de um sistema *mHealth*,

sendo possível representar o caminho desde a origem, ou seja, a mensagem ter sido recolhida a partir do paciente, até chegar ao destino, o dispositivo móvel. Para tanto, foi necessário a proposição de modelos que representem todo o domínio de uma nuvem móvel, incluindo: infraestrutura de nuvem, comunicação sem fio e o dispositivo móvel. O modelo de nuvem foi dividido em vários submodelos RBDs. Já os modelos de comunicação foram concebidos utilizando modelos SPN, representando a comunicação de dois mecanismos sem fio: *WiFi* e 3G. Neste estudo foram considerados vários cenários em diferentes possibilidades de comunicação sem fio, taxas de descarga de bateria e diferentes tempos de espera. Os resultados mostraram que o tempo de espera é a métrica com maior impacto na disponibilidade do sistema *mHealth* e que baterias mais eficientes ajudariam a melhorar a disponibilidade do sistema. No entanto, este trabalho não aborda uma metodologia de validação dos modelos propostos e não adota dispositivos *wearables*.

Em MATOS et al. (2015) utilizam modelagem hierárquica e quatro diferentes técnicas de análise de sensibilidade, a fim de determinar os parâmetros que causam o maior impacto sobre a disponibilidade de uma nuvem móvel. Para esse fim, adotou-se uma arquitetura similar a de OLIVEIRA et al. (2013), a qual foi modelada a partir de um modelo hierárquico, composto por modelos RBD e CTMC. Os resultados obtidos demostraram que as abordagens distintas proporcionaram resultados semelhantes quanto à classificação de sensibilidade, entretanto, o trabalho não foca em nenhuma categoria de serviço específico, porém os parâmetros adotados neste estudo serão úteis para a presente pesquisa.

O trabalho desenvolvido por SILVA et al. (2014) investiga o impacto no consumo de energia de dispositivos móveis (*smartphone*) e a sua disponibilidade sob a operação em diferentes protocolos de aplicação e de conexão de rede de dados. Um gerador de carga de trabalho foi implantado, para simular um ambiente de tráfego real e testar os protocolos *Short Polling, Comet, Websocket e XMPP (Extensible Messaging and Presence Protocol)*. Os autores avaliaram o consumo de energia desses protocolos nas conexões *WiFi* e 3G. A disponibilidade de cada cenário foi estimada por meio de modelos hierárquicos heterogêneos, combinando modelos CTMC, SPN e RBD. Os resultados indicaram que o protocolo *Websocket* provou ser o protocolo mais eficaz quando utilizado em uma conexão 3G. Contudo, protocolo XMPP consumiu menos energia do que o *Websocket* em uma conexão *WiFi*. Portanto, os autores defendem que o protocolo XMPP é o que possui maior eficiência energética quando as conexões de dados são combinadas, ou seja, gasta menos energia que os demais protocolos estudados.

Os dispositivos *wearable*, mesmo possuindo limitações em relação ao consumo energético, têm se tornado uma importante ferramenta de estudo para resolução de problemas ligados a programas da área de saúde e atividades físicas. Isto se deve, principalmente, pela possibilidade de explorar seus sensores sem fio embutidos, que são utilizados por pesquisas e sistemas comerciais para controlar o nível de atividade do usuário de acordo com seu movimento diário.

MORTAZAVI et al. (2014) explora os sensores de acelerômetro e giroscópio de um *smartwatch*, para serem utilizados em atividades físicas. Seu trabalho analisou a diferença de

desempenho dos sensores de um *smartwatch* através de um treino de cinco exercícios distintos, a fim de contar repetições e classificar movimentos da rotina de exercícios, sendo cada um deles analisados sob a perspectiva de três cenários. Os resultados mostraram que a combinação dos sensores é capaz de fornecer os melhores resultados de classificação.

A criticidade em sistemas similares aos discutidos nesse contexto, justifica as pesquisas que utilizam de técnicas para avaliação de disponibilidade para garantir um bom funcionamento, como é apresentado no estudo de ABDALI-MOHAMMADI; BAJALAN; FATHI (2015) propõe uma arquitetura de um sistema *wearable* para aplicações de saúde dividida em três camadas: *wearable*, *Mobile Computing* e *Cloud Computing*. Esta arquitetura visa criar sistemas portáteis para aplicações de saúde em que a tolerância a falhas mantinha o funcionamento regular do sistema. A confiabilidade foi avaliada utilizando as métricas de confiabilidade e tempo médio de falha. O trabalho resultou em uma confiabilidade de 56% superior quando há redundância na camada *Mobile Computing* em comparação à mesma arquitetura sem a referida redundância.

O estudo apresentado por CASILARI; OVIEDO-JIMÉNEZ (2015) analisou os sensores de acelerômetro e giroscópio de um *smartwatch* integrado a um *smartphone* em um cenário onde foi possível monitorar, de forma automática, a queda do usuário. O sistema foi instalado nos respectivos dispositivos, que se comunicam via *Bluetooth*. A aplicação monitorou o movimento do usuário com base nos dados recebidos do acelerômetro e do giroscópio que estavam embutidos nos mesmos. Desta forma, quando ocorria a queda do usuário esta era logo detectada pela aplicação do *smartwatch*, que por meio de uma mensagem de alerta informava a ocorrência para o *smartphone* via *Bluetooth*. No entanto, a queda só era levada em consideração caso o aplicativo em execução no *smartphone* também detectasse o evento e dentro de um curto intervalo em segundos, antes ou depois de receber a referida mensagem. Os resultados obtidos evidenciaram que a utilização conjunta dos dois dispositivos de detecção aumentava nitidamente a capacidade do sistema de evitar falsos alarmes ou falsos positivos.

Da mesma forma que a pesquisa atual propõe uma arquitetura em três camadas, no trabalho elaborado por ABDALI-MOHAMMADI; BAJALAN; FATHI (2015) também é apresentada uma arquitetura em três camadas, no entanto, não foram propostos modelos para que melhor pudesse avaliar a confiabilidade do serviço. Outro trabalho discutido anteriormente e que também possui características relevantes é o elaborado por CASILARI; OVIEDO-JIMÉNEZ (2015), indicando que a junção do *smartwatch* com o *smartphone* aumenta a capacidade do sistema.

Sendo assim, de acordo com os trabalhos supracitados, é possível identificar várias pesquisas que lidam com o monitoramento de pacientes utilizando sensores sem fio. Estes estudos têm como contribuição a esta pesquisa apresentar a importância da adoção de sistemas de saúde móvel utilizando *wearables* e dispositivos móveis. Além disso, estas obras reforçam a eficácia e importância do uso de sensores sem fio no monitoramento de saúde e avaliação de rendimento de atividades.

Contudo, não foram encontradas publicações que abordassem o nosso tema de pesquisa

especificamente. Não identificamos proposição de modelos de desempenho e de disponibilidade para ambientes *mHealth*, usando *smartwatch* em um ambiente MCC, talvez devido ao fato de que o *smartwatch*, no seu formato atual de uso, seja uma tecnologia relativamente nova.

A Tabela 1.1 apresenta um comparativo entre os trabalhos relacionados e a pesquisa desenvolvida. Já a Tabela 1.2 são apresentados os trabalhos relacionados, delineando suas limitações e melhorias. O principal objetivo é destacar os pontos fortes e principais limitações da presente pesquisa em relação às demais. Os trabalhos selecionados foram classificados e analisados de acordo com os seguintes tópicos: Contexto *mHealth*, Dispositivo Móvel, MCC, Interfaces de Rede, Análise de Sensibilidade e Modelos. As pesquisas apresentadas por ARAUJO et al. (2014); SILVA et al. (2014); OLIVEIRA et al. (2013); MATOS et al. (2015) foram os únicos trabalhos estudados que têm como objetivo a proposição de modelos, onde os parâmetros adotados por estes irão contribuir com esta pesquisa. Os demais trabalhos relacionados não apresentaram um modelo capaz de analisar, em diferentes cenários, a avaliação da sua disponibilidade, porém, demonstram a viabilidade desta pesquisa.

Com relação às limitações neste trabalho, vale ressaltar que alguns pontos, como os custos e desempenho, não foram abordados durante o desenvolvimento da pesquisa, visto que isso poderia causar um distanciamento do propósito principal. Contudo, é possível observar a importância dessa pesquisa em comparação aos trabalhos relacionados, com a proposição de modelos para a avaliação do serviço analisado, bem como a sua validação, além de uma análise de sensibilidade, a fim de verificar gargalos de disponibilidade.

Tabela 1.1 Parâmetros envolvidos nos trabalhos relacionados.

Autores	Contexto mHealth	Dispositivos Móveis	MCC	Interfaces de Rede	Análise de Sensibilidade	Modelos
MAGLOGIANNIS et al. (2014)	✓	*smartwatch e smartphone*	✓	*Bluetooth*	-	-
DOUKAS; PLIAKAS; MAGLOGIANNIS (2010)	✓	*smartphone*	✓	3G e WiFi	-	-
CHEN et al. (2014)	✓	*smartphone*	-	-	-	-
OLIVEIRA et al. (2013)	-	*smartphone*	✓	3G e WiFi	-	RBD, SPN e CTMC
ARAUJO et al. (2014)	✓	*smartphone*	✓	3G e WiFi	-	RBD e SPN
SILVA et al. (2014)	✓	*smartphone*	✓	2G, 3G e WiFi	-	RBD, SPN e CTMC
MORTAZAVI et al. (2014)	✓	*smartwatch*	-	-	-	-
ABDALI-MOHAMMADI; BAJALAN; FATHI (2015)	✓	Sensores Corporais *e smartphone*	✓	*Bluetooth*	-	-
CASILARI; OVIEDO-JIMÉNEZ (2015)	-	*smartwatch*	-	*Bluetooth*	-	-
MATOS et al. (2015)	-	*smartphone*	✓	3G e WiFi	✓	RBD e CTMC
Esta Pesquisa	✓	*smartwatch e smartphone*	✓	*Bluetooth*, 3G e WiFi	✓	RBD e CTMC

Tabela 1.2 Trabalhos relacionados.

Autores	Limitações	Melhorias
MAGLOGIANNIS et al. (2014)	O sistema necessita de conexão com a internet, uma vez que os lembretes não são armazenados localmente.	A adoção de modelos pode ser utilizado para avaliar, em diferentes cenários, a disponibilidade do sistema.
DOUKAS; PLIAKAS; MAGLOGIANNIS (2010)	Não foi apresentado nenhum modelo capaz de avaliar diferentes composições de cenários para o dispositivo móvel e não foi realizado testes ou simulações em um ambiente real.	Com proposição de modelos onde fosse possível avaliar diferentes cenários para o dispositivo móvel, avaliando a sua dependabilidade.
CHEN et al. (2014)	Mais experimentos são necessários para demonstrar o funcionamento do sistema, além disso, o sistema é apenas para uso interno.	Avaliar a disponibilidade e confiabilidade do sistema em ambiente externo através da proposição de modelos.
OLIVEIRA et al. (2013)	O trabalho limitou-se a três cenários distintos, porém nenhum com foco em tecnologia *wearable*.	Adoção da conexão *Bluetooth* para avaliação do consumo energético e a inclusão de um dispositivo *wearable*.
ARAUJO et al. (2014)	Este trabalho não aborda uma metodologia de validação dos modelos propostos.	A inclusão de dispositivos *wearable*, aumentaria as possibilidades de estudo para diferentes cenários em ambiente *mHealth*.
SILVA et al. (2014)	O estudo limitou-se a protocolos de conexões WiFi e 3G.	Como melhoria, poderia incluir uso de *wearable* e comunicação sem fio *Bluetooth*.
MORTAZAVI et al. (2014)	Não realizou teste de comunicação sem fio, limitando somente a dois sensores do *smartwatch*.	Adição de sensores para frequência cardíaca e a proposição de modelos para avaliar a confiabilidade do sistema.
ABDALI-MOHAMMADI; BAJALAN; FATHI (2015)	A arquitetura proposta só é possível para ambientes que suportem estrutura de nuvem.	Proposição de Modelos para avaliar a disponibilidade arquitetura proposta.
CASILARI; OVIEDO-JIMÉNEZ (2015)	Não houve comunicação de redundância de conexão entre o *smartphone* e o *smartwatch*.	A adoção de modelos para avaliar a disponibilidade e confiabilidade em diferentes cenários combinando o uso do *smartphone* e o *smartwatch*.
MATOS et al. (2015)	Não foi levado em consideração a autonomia da bateria do dispositivo móvel.	Adoção da conexão *Bluetooth* e inclusão de dispositivos *wearable*, aumentaria as possibilidades de estudo em MCC.

1.4 Estrutura da Dissertação

Esta dissertação está estruturada em 7 **capítulos**. Sendo o presente **Capítulo 1**: introdutório, os demais apresentando-se como segue:

Capítulo 2: aborda fundamentos teóricos relevantes aos temas abordados, apresentando temas como: *Mobile Cloud Computing*; Dependabilidade; Modelagem; Análise de Sensibilidade; Injeção de Falhas e Intervalo de Confiança da Disponibilidade.

Capítulo 3: apresenta a metodologia aplicada, como também destaca a Arquitetura Base.

Capítulo 4: detalha os modelos propostos. Os formalismos adotados foram *Reliability Block Diagram* (RBD) e *Continuous Time Markov Chain* (CTMC).

Capítulo 5: descrevemos o experimento realizado para a coleta de parâmetros do modelo e o processo de validação do modelo da Arquitetura Base.

Capítulo 6: apresenta os estudos de caso desta pesquisa, baseados nos modelos propostos.

Capítulo 7: apresenta as conclusões e trabalhos futuros.

2

Fundamentação Teórica

Este capítulo apresenta um background conceitual sobre os temas abordados para uma melhor compreensão deste trabalho. Inicialmente, abordamos os conceitos relacionados a Internet of things (IoT) e Mobile Cloud Computing (MCC). Em seguida, apresentamos os conceitos básicos de dependabilidade e os atributos de interesse deste trabalho. Posteriormente, introduzimos técnicas de modelagens como RBD e CTMC. Por fim, apresentamos uma explanação sobre análise de sensibilidade e técnica de injeção de falhas para validação do modelo.

2.1 Internet das Coisas (*Internet of Things* - IoT)

A Internet das Coisas (*Internet of things* (IoT)) é um novo paradigma que está rapidamente ganhando espaço no cenário de telecomunicações sem fio. O conceito básico de IoT é a presença de um conjunto de variedade de coisas ou objetivos, como: celulares, relógios, tvs, sensores, atuadores e etc, todos integrados, interagindo um com os outros, de forma a atingir objetivos comuns, criando um ambiente mais onipresente (ATZORI; IERA; MORABITO, 2010).

Diversas definições de IoT podem ser identificadas na comunidade cientifica, o que nos evidenciam o forte interesse na questão e na vivacidade dos debates sobre o assunto. No geral, a IoT pode ser definida como uma rede de objetos interconectados (visão de computação ubíqua), onde estes objetos podem ser qualquer coisa física, que podem ainda possuir seus próprios endereços (Internet Protocol (IP)), podendo trocar informações através de uma rede de sensores, gerando oportunidades tanto no âmbito acadêmico quanto no industrial.

Recentemente observou-se que alguns esforços estão sendo direcionados a aplicações em diferentes cenários, principalmente sobre alguns aspectos do cotidiano e comportamento das pessoas, tais como: cidades inteligentes (iluminação inteligente e qualidade do ar), medição inteligente (fluxo de água e vazamentos de água), situações críticas (níveis de radiação, detecção de incêndio), varejo (pagamento inteligente e localização item), na indústria (monitoramento de processos e segurança), na agricultura (casas verdes e a irrigação inteligente), automação residencial (uso de energia e água), e saúde ou *mHealth* (vigilância paciente e controle de medicamentos de uso contínuo) (SILVA et al., 2013).

No setor de *mHealth*, que é o foco da nossa pesquisa, a tecnologia IoT pode fornecer uma série de aplicações, como também pode ser usada para melhorar as atuais soluções de monitoramento de saúde.

Neste cenário, os pacientes irão transportar sensores médicos para monitorar as informações como temperatura corporal, pressão arterial, atividade respiratória. Outros sensores, como dos *wearables* (por exemplo, acelerômetros, giroscópios) ou fixos (sensor de proximidade) serão usados para coletar dados para monitorar as atividades do paciente em seus ambientes de vida. As informações serão agregadas localmente e transmitidas a centros médicos remotos, que poderão realizar monitoramento remoto avançado e serão capazes de agir de forma rápida quando necessário. Com o uso de sensores *wearables*, juntamente com as aplicações adequadas executadas em dispositivos pessoal móvel, permitem que os usuários acompanhem suas atividades diárias (passos andados, calorias queimadas, exercícios realizados, monitoramento cardíaco, etc.), oferecendo sugestões para melhorar seu estilo de vida e prevenindo o aparecimento de problemas de saúde.

Em relação aos protocolos de comunicação padrão, a IoT tem a capacidade de lidar com diferentes tecnologias, tais como Protocolos de Internet, Redes de Comunicações Móveis, *Wireless Sensor Networks* (WSN), *Radio-Frequency IDentification* (RFID), e outros protocolos de sensoriamento (MA, 2011).

Na Tabela 2.1 apresentamos os seis principais elementos necessários para fornecer as funcionalidades de IoT, adaptada de AL-FUQAHA et al. (2015):

Tabela 2.1 Elementos de Construção e Tecnologias IoT

Elementos IoT	**Exemplos**
Identificação	EPC, uCode, IPv4, IPv6
Sensor	Sensores Inteligentes, Atuadores ou *Wearable*
Comunicação	RFID, NFC, BLE, Bluetooth, Z-Wave, WiFi
Computação	Arduino, Smartphones, Android, *Cloud (Hadoop, Nimbits, etc.)*
Serviço	*Smart Home, Smart City, mHealth*
Semântica	*Resource Description Framework (RDF), Web Ontology Language (OWL) e Efficient XML Interchange (EXI)*

Identificação: considerado um dos elementos mais importantes, visto que é primordial a identificação dos objetos para então conectá-los à internet. Muitos métodos de identificação estão disponíveis para o IoT, como códigos de produtos eletrônicos (*electronic product codes* - EPC), códigos ubíquos (uCode) e os métodos de endereçamento de objetos IoT incluem IPv6, IPv4 e 6LoWPAN.

Sensores/Atuadores: são fundamentais no contexto de IoT. Os sensores de IoT coletam dados ou informações de objetos relacionados na rede, em seguida, estes podem armazenar os dados em um banco de dados ou encaminhar para a nuvem. Os sensores IoT podem ser: sensores inteligentes, atuadores ou dispositivos *wearable*. Os sensores são dispositivos capazes de fornecer informações acerca da situação de uma entidade de interesse ou ambiente em que a

mesma está inserida. Enquanto os atuadores são dispositivos capazes de atuar em entidades de interesse ou ambientes.

Comunicação: também considerado um dos elementos fundamentais, pois através dela é possível conectar os objetos inteligentes. Além disso, este elemento desempenha um papel importante no consumo de energia dos objetos, sendo, portanto, um fator crítico, visto que grande quantidade de energia (recurso limitado) é empregada para realizar a comunicação. Exemplos de tecnologias utilizadas nesse item são: WiFi, *Bluetooth*, IEEE 802.15.4, RFID.

Computação: é relacionada a unidade de processamento, por exemplo, micro controladores, processadores, FPGAs. Em outras palavras, é tudo que possibilita aos objetos inteligentes a capacidade de computar. Por exemplo: os componentes de *hardware* que permitem a comunicação e interação com as entidades físicas e os componentes de *software* que abstraem a comunicação com os dispositivos.

Serviços: são componentes de *software* que permitem o acesso de usuários aos recursos. No geral a IoT pode prover diversas classes de serviços, dentre elas, destacam-se os *Serviços de Identificação*, o quais são os mais básicos e importantes serviços, pois as aplicações os utilizam para mapear as *Entidades Físicas* (EF) (de interesse do usuário) em *Entidades Virtuais* (EV), em outras palavras, estes serviços facilitam mapear objetos do mundo real e os respectivos objetos do mundo virtual. Os *Serviços de Agregação de Informações*, os quais coletam e sumariza dados brutos obtidos dos objetos inteligentes que precisam ser processados e enviados para as aplicações. Os *Serviços de Colaboração e Inteligência* são aqueles que agem sobre os serviços de agregação de informações usando os dados obtidos e processados para tomar decisões e reagir de modo adequado e os *Serviços de Ubiquidade* que visam prover *Serviços de colaboração e Inteligência* em qualquer momento e qualquer lugar em que eles sejam necessários.

Semântica: refere-se à habilidade de extração de conhecimento da diversidade de objetos na IoT, ou seja, a semântica trata da descoberta e uso inteligente dos recursos, para possibilitar o provimento de determinado serviço. Além disso, deve efetuar o reconhecimento e análise dos dados para realizar corretamente a tomada de decisões. Como exemplo, podem ser usadas diversas tecnologias tais como: *Resource Description Framework (RDF), Web Ontology Language (OWL) e Efficient XML Interchange (EXI).*

2.2 *Mobile Cloud Computing*

A *Mobile Cloud Computing* (MCC) surgiu com a interseção da computação móvel com a computação em nuvem. A computação em nuvem (*Cloud Computing*) é um modelo computacional cujos recursos como poder de processamento, rede, armazenamento e *software* são entregues como serviço através da Internet e podem ser acessados remotamente (ARMBRUST et al., 2010). De acordo com Instituto Nacional de Padrões e Tecnologia dos Estados Unidos da América (*NIST - National Institute of Standards and Technology*) a computação em nuvem é um modelo utilizado para habilitar o acesso ubíquo, conveniente e sob demanda a um conjunto

de recursos computacionais compartilhados (rede, servidores, armazenamento, aplicações e serviços), que pode ser rapidamente provido e liberado com o mínimo de esforço gerencial ou interação com o provedor de serviços (STANDART, 2015).

Existem várias definições da MCC na literatura e diferentes pesquisas aludem diferentes conceitos da computação móvel. O paradigma da MCC, como já supracitado, surgiu como a intersecção da computação móvel e a computação em nuvem.

O objetivo principal da MCC é usar a capacidade de processamento e armazenamento de uma nuvem computacional para estender as capacidades de dispositivos móveis mais limitados, de forma a melhorar o desempenho e aliviar o consumo energético de aplicações móveis mais pesadas (SATYANARAYANAN et al., 2009).

Este paradigma poderá permitir o provisionamento de aplicações móveis mais inteligentes, que estendam as capacidades cognitivas de usuários, tais como: reconhecimento de voz, processamento de linguagem natural, visão computacional, aprendizado de máquina, realidade aumentada, planejamento e tomada de decisão (SATYANARAYANAN et al., 2009).

Uma arquitetura da MCC pode ser apresentada em diferentes formas, dependendo do contexto. Na Figura 2.1 apresentamos um exemplo de arquitetura básica para MCC adaptada de DINH et al. (2013).

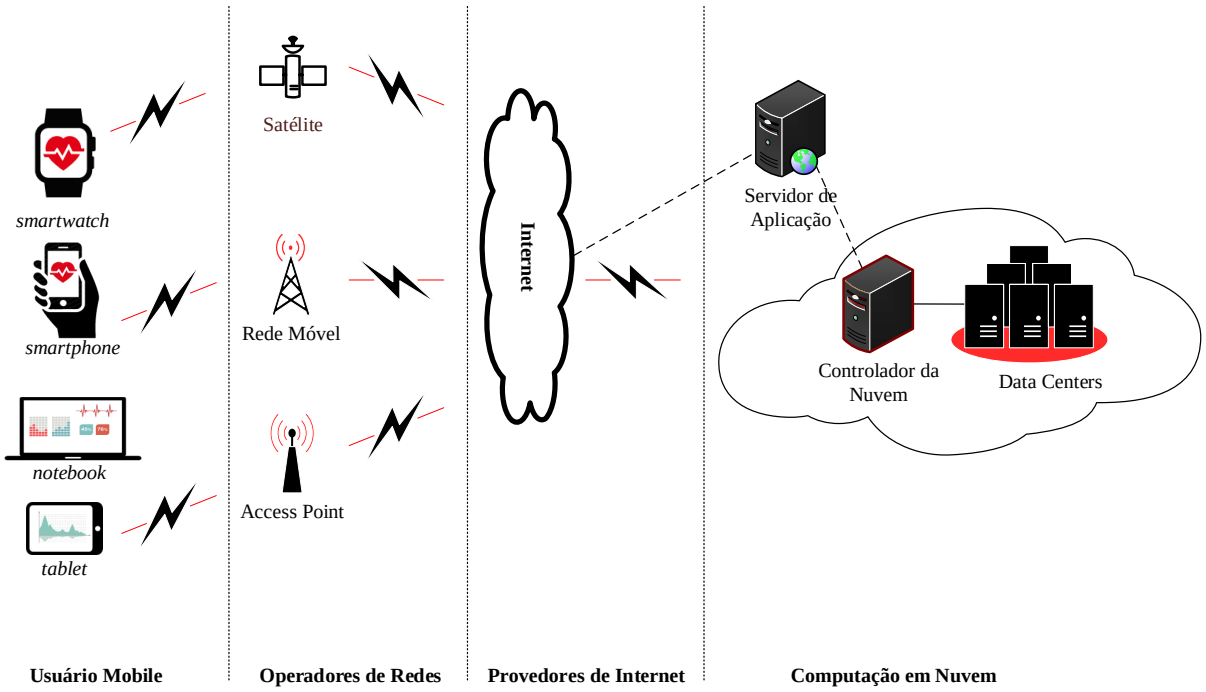


Figura 2.1 Arquitetura MCC.

Na Figura 2.1, os dispositivos móveis são conectados a redes móveis através de operadores de redes (satélite, *access point* ou estação de transmissão de sinal). Quando informações são solicitadas pelos usuários, as requisições são transmitidas para uma central de processamento, que é conectada aos servidores que proveem os serviços de autenticação, autorização e gerenciamento de usuários. Depois de processadas as informações chegam à nuvem através da

Internet. Na nuvem, os controladores processam a requisição para prover o serviço requisitado aos usuários.

É possível observar que a nuvem móvel é composta basicamente pelos mesmos componentes e categorias de serviços de uma computação em nuvem. Em adição aos modelos de serviço tradicionais da computação em nuvens (*Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) ou *Software as a Service* (SaaS)), a *Mobile Cloud Computing* trouxe outro modelo de serviço chamado *Mobile Backend-as-a-Service* (MBaaS). Uma plataforma de MBaaS provê funcionalidade básica para aplicações móveis em nuvem, como por exemplo, gerenciamento de usuários, notificações *push* e integração com redes sociais (SAREEN, 2013).

Nesta arquitetura, uma vez que a computação é migrada para a nuvem, os requisitos sobre a capacidade dos dispositivos móveis são diminuídos, solucionando em partes o problema da limitação dos dispositivos móveis, onde uma gama maior de clientes pode ser atingida, como *smartphones* com maiores recursos, *smartphones* mais limitados, *tablets* e até mesmo alguns *feature phones* (QI; GANI, 2012). Com a nuvem sendo utilizada como plataforma de processamento das informações, tanto as redes sem fio, como a infraestrutura de nuvem passam a ser itens críticos na disponibilidade das aplicações. Dessa forma, prover uma infraestrutura que atenda a níveis aceitáveis de disponibilidade passa a ser um desafio para a MCC.

2.3 Dependabilidade

O conceito de dependabilidade (*dependability*) apareceu pela primeira vez na década de 1820, quando Charles Babbage propôs uma máquina para automatizar cálculos de matemática com o objetivo de melhor desempenho e evitar o máximo de erros nas operações (MACIEL et al., 2012; ESARY; PROSCHAN, 1970). A medida que a ciência e os sistemas foram evoluindo, uma série de caraterísticas e várias especificidades foram sendo citadas, assim foram surgindo outras definições e conceitos com relação a dependabilidade, como formulado por LAPRIE (1985) definindo a dependabilidade como a capacidade de um sistema computacional fornecer um serviço de forma confiável.

De acordo com AVIZIENIS; LAPRIE; RANDELL (2001) a dependabilidade de um sistema é a habilidade de evitar falhas de serviços que são mais frequentes ou mais severas do que o aceitável. A dependabilidade de um sistema está relacionada a uma série de requisitos não funcionais que o serviço deverá garantir para obter satisfação e fidelidade dos usuários (AVIZIENIS; LAPRIE; RANDELL, 2001). A Figura 2.2[1] apresenta a taxonomia dos sistemas de funcionamento confiável, que compreende em: Atributos (*Attributes*), Meios (*Means*) e Ameaças (*Threats*):

Os **atributos** possibilitam que medidas quantitativas, que são cruciais, sejam obtidas para a análise dos serviços ofertados. Os **meios** são os caminhos pelos quais a dependabilidade é alcançada. As **ameaças** referem-se às falhas, aos erros e aos defeitos (MACIEL et al., 2012).

[1]Fonte: AVIZIENIS; LAPRIE; RANDELL (2001). Adaptado pelo autor.

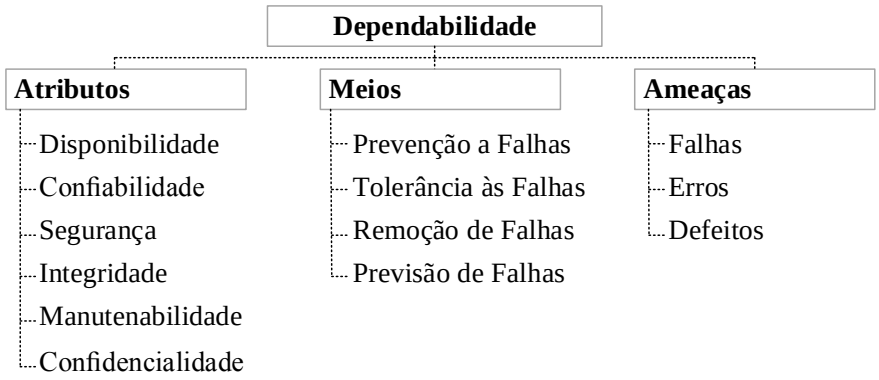


Figura 2.2 Árvore de dependabilidade

De acordo com a Figura 2.2, os **atributos** consistem em:

- Disponibilidade (*Availability*): a capacidade do sistema estar de prontidão para prover o serviço corretamente;
- Confiabilidade (*Reliability*): continuidade da prestação do serviço, é a probabilidade que o sistema irá prover o serviço continuamente, sem erros, até certo instante t;
- Segurança (*Safety*): ausência de consequências catastróficas;
- Confidencialidade (*Confidentiality*): preservação da informação, ausência de divulgação desautorizada de informação;
- Integridade (*Integrity*): ausência de alterações impróprias no estado do sistema e
- Manutenabilidade (*Maintainability*): a habilidade de sofrer reparos e modificações.

Os **meios** são agrupados em:

- Prevenção (*Fault Prevention*): prevenir a ocorrência ou introdução de falhas;
- Tolerância (*Fault Tolerance*): entregar o serviço correto mesmo na presença de falhas;
- Remoção (*Fault Removal*): reduzir o número ou a severidade das falhas, e
- Previsão (*Fault Forecasting*): estimar o número presente, a incidência futura e as consequências das falhas.

Técnicas de prevenção, previsão, remoção e tolerância a falhas, por sua vez, contribuem para manter níveis desejados de dependabilidade, sobretudo em sistemas críticos (AVIZIENIS; LAPRIE; RANDELL, 2001).

Por fim, as **ameaças** são agrupadas em:

- Falha (*Fault*): é definida como a falha de um componente, subsistema ou sistema que interage com o sistema em questão.
- Erro (*Errors*): é definido como um estado que pode levar à ocorrência de uma falha.
- Defeito (*Failure*): representa o desvio do funcionamento correto de um sistema.

Dentre os atributos de dependabilidade, a disponibilidade e confiabilidade, foram adotadas no presente estudo, a seguir, a representação formal.

Disponibilidade

A disponibilidade é a probabilidade que o sistema esteja operacional durante um determinado período de tempo (MACIEL et al., 2012). Esta probabilidade pode ser dada através da razão entre o tempo de funcionamento esperado do sistema e a soma entre o tempo de funcionamento e o tempo de falha esperado (MACIEL et al., 2011). A disponibilidade é expressa pela Equação 2.1.

$$A = \frac{E[Uptime]}{E[Uptime] + E[downtime]}, \tag{2.1}$$

onde, o $E[Uptime]$ é o tempo de funcionamento esperado do sistema e o $E[downtime]$ é o tempo de falha esperado do sistema.

Para cálculo das métricas de dependabilidade, é necessário conhecer os tempos de falha *Time to Failure* (TTF) e o tempo do reparo *Time to Repair* (TTR) dos componentes presentes nos modelos. Entretanto, quando os tempos de *uptime* e *downtime* não estão disponíveis, geralmente são usados os valores médios entre os eventos de falha e reparo do sistema. A equação para o cálculo da disponibilidade pode ser escrita conforme apresentado na Equação 2.2.

$$A = \frac{MTTF}{(MTTF + MTTR)}, \tag{2.2}$$

onde, o *Mean Time to Failure* (MTTF) é tempo médio para que ocorram falhas no sistema e o *Mean Time to Repair* (MTTR) é o tempo médio para levar o sistema novamente ao estado de funcionamento, após a ocorrência de uma falha.

Considerando que os tempos de falha e reparo são exponencialmente distribuídos, a disponibilidade pode ser obtida através de suas respectivas taxas, onde λ corresponde à taxa de falha (1/MTTF) e μ indica a taxa de reparo (1/MTTR). Assim, a disponibilidade pode ser expressa pela Equação 2.3.

$$A = \frac{\lambda}{(\lambda + \mu)} \tag{2.3}$$

Outra forma de representar a disponibilidade é através do número de noves (9's). O número de noves corresponde a contagem dos algarismos consecutivos iguais a 9, após a vír-

gula (MARWAH et al., 2010). Por exemplo, se um serviço apresenta $99,55\%$ de disponibilidade, também pode ser interpretado como um sistema com $2,34$ noves de disponibilidade. O número de noves são calculados através da Equação 2.4.

$$N = -log_{10}(1-A) \tag{2.4}$$

Ao contrário da disponibilidade, a indisponibilidade representa a probabilidade do sistema não estar disponível, pode ser calculada a partir das Equação 2.5.

$$UA = 1-A \tag{2.5}$$

Com base no valor da indisponibilidade é possível obter o *downtime* anual do sistema. O *downtime* anual representa o número esperado de horas que o sistema estará indisponível no intervalo de um ano, é calculado pela fórmula da Equação 2.6.

$$Downtime_{anual} = UA \times 8760h \tag{2.6}$$

Confiabilidade

Embora os conceitos de disponibilidade e confiabilidade estejam relacionados, a confiabilidade pode ser definida como a probabilidade de um sistema não falhar até um determinado tempo. A confiabilidade também pode ser definida como a probabilidade de que um sistema irá executar a sua função pretendida durante um período de tempo de funcionamento sem qualquer falha (KUO; ZUO, 2003). A função de confiabilidade pode ser expressa na Equação 2.7.

$$R(t) = P(T > t), t \geq 0 \tag{2.7}$$

O $R(t)$ representa função da confiabilidade, onde t corresponde ao tempo que se deseja obter a confiabilidade e T é a variável aleatória que representa o tempo para falha, e que está dentro do intervalo [0; t] (KUO; ZUO, 2003).

As métricas de dependabilidade com modelos podem ser computadas para um determinado intervalo de tempo (análise transiente), ou quando o sistema entra em equilíbrio (análise estacionária), podendo ser obtidas através de análise ou simulação (BOLCH et al., 2006).

A análise transiente analisa o comportamento do modelo a partir de uma marcação inicial no tempo zero até um instante especificado. A análise estacionária nos dá a distribuição de probabilidades estacionárias do sistema, ou seja, a fração do tempo que o sistema permanece em cada um dos seus estados quando ele atinge o equilíbrio.

A simulação possibilita extrair informações de um modelo, onde algumas vezes é o método mais adequado. Vale ressaltar, que isto ocorre, por exemplo, quando o espaço de estados de um sistema é muito grande para ser estudado analiticamente.

Nesta dissertação, foi adotada a análise estacionária para computar a disponibilidade da

infraestrutura do serviço analisado. No entanto, para a análise de confiabilidade dos dispositivos móveis utilizados, foi adotada a análise transiente. Os modelos adotados para as avaliações serão apresentados na próxima seção.

2.4 Modelos para Avaliação

A modelagem vem sendo bastante utilizada para avaliar os mais variados tipos de sistemas, sejam estes computacionais ou não (KIM; MACHIDA; TRIVEDI, 2009). A avaliação de dependabilidade por meio de modelos e de simulação é um meio viável para propor melhorias aos sistemas de computação em nuvem. Modelos para avaliação de dependabilidade podem ser classificados em duas categorias: modelos combinatoriais e modelos baseados em espaço de estados (MALHOTRA; TRIVEDI, 1994; MACIEL et al., 2011). Modelos combinatoriais capturam as condições que levam um sistema a falhar ou permanecer em funcionamento em termos de relacionamentos estruturais entre seus componentes (MACIEL et al., 2012). Exemplos de modelos combinatoriais são: Diagrama de Blocos de Confiabilidade (*Reliability Block Diagram (RBD)*) e Árvores de Falhas (*Fault Trees*) (MACIEL et al., 2012). Modelos baseados em espaços de estados representam o comportamento do sistema em através de estados e ocorrências de eventos, expressas como as transições de estado (MACIEL et al., 2012). Cadeias de Markov (BOLCH et al., 2006) e Redes de Petri Estocásticas (*Stochastic Petri Net* (SPN)), (MOLLOY, 1982) são dois tipos principais de modelos baseado em estados. Esses tipos de modelo diferem de um para outro, não só na facilidade de utilização para uma aplicação em particular, mas em termos de poder de modelagem (MALHOTRA; TRIVEDI, 1994). Por exemplo, as métricas de dependabilidade, como disponibilidade, podem ser calculadas através de modelos baseados em espaço de estados, como SPN, ou de modelos combinatórias, como o RBD. O RBD apresenta uma vantagem em relação ao fornecimento dos resultados, pois apresenta cálculos mais rápidos por meio de suas fórmulas do que as simulações e as análises numéricas dos modelos SPNs. Entretanto, modelos SPNs têm um maior poder de detalhamento da representação (MARSAN et al., 1994; BALBO, 2001).

Modelos da mesma categoria ou de categorias distintas, podem ser combinados em diferentes níveis de compreensão, levando a modelos maiores, dispostos de maneira hierárquica (MALHOTRA; TRIVEDI, 1994). A modelagem hierárquica heterogênea (MILLER; FISHWICK, 1993) descreve qualquer metodologia de modelagem que suporte várias representações conceitualmente distintas. Para essa modelagem o modelo é dividido numa hierarquia de submodelos, onde os submodelos são expressos em nível superior ou nível mais baixo da hierarquia (SAHNER; TRIVEDI; PULIAFITO, 2012). O modelo de nível superior representa o sistema baseado no relacionamento entre seus componentes(subsistema), enquanto que modelos de nível mais baixo descrevem o comportamento detalhado de cada componente.

Formalismos de modelagem heterogêneos podem ser usados hierarquicamente para formar um único modelo de sistema. Esta modelagem têm sido usada para lidar com a complexidade

de vários tipos de sistemas, como por exemplo: redes de sensores (KIM; MACHIDA; TRIVEDI, 2009), redes de telecomunicação (TRIVEDE et al., 2006) e nuvens privadas (DANTAS et al., 2012). Nesta seção apresentamos os conceitos básicos dos modelos utilizados neste estudo.

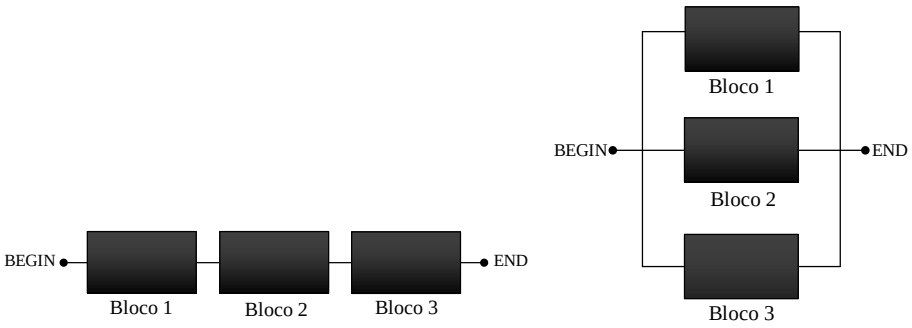
2.4.1 Diagramas de Bloco de Confiabilidade - RBD

Diagrama de Blocos de Confiabilidade (*Reability Block Diagram*-RBD) é um tipo de modelo combinatório proposto inicialmente para analisar a confiabilidade de sistemas baseados nas relações de seus componentes (MACIEL et al., 2011). Posteriormente, este formalismo foi estendido para a análise de disponibilidade e manutenibilidade.

O modelo RBD é uma descrição gráfica da conexão dos componentes do sistema que podem ser usados para determinar o estado geral do sistema a partir do estado dos seus componentes (SAHNER; TRIVEDI; PULIAFITO, 2012). Este modelo é formado pelos seguintes componentes: vértice de origem, vértice de destino, componentes do sistema (representados por blocos) e arcos conectando os blocos e os vértices.

A estrutura de um modelo RBD é representada por subsistemas ou componentes conectados de acordo com a sua função no relacionamento de confiabilidade (SAHNER; TRIVEDI; PULIAFITO, 2012). Este modelo pode ser organizado em série, em paralelo ou em estruturas *k-out-of-n*, ou ainda em combinações entre essas organizações (KUO; ZUO, 2003). Uma conexão paralela é utilizada para mostrar redundância de componentes e criar múltiplos caminhos. Quando os componentes estão conectados sem redundância, ou de forma sequencial, eles podem ser representados por uma conexão em série.

A Figura 2.3 apresenta exemplos de modelos RBD quando os componentes estão em série (Figura 2.3(a)) ou em paralelo (Figura 2.3(b)).



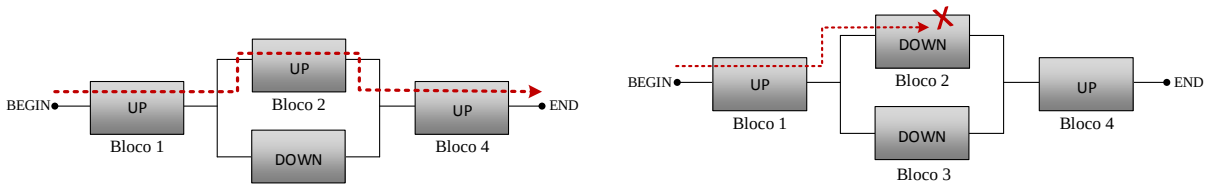
(a) Componentes em Série (b) Componentes em Paralelo

Figura 2.3 Exemplos de estruturas de um RBD.

A estrutura *k-out-of-n*, que é formada por *n* componentes, e necessita do funcionamento de pelo menos *k* componentes para estar operacional (KUO; ZUO, 2003; MACIEL et al., 2011). Por exemplo, uma estrutura em que haja 5 componentes e necessita-se de 3 funcionando para prover o serviço esperado, temos uma estrutura *3-out-of-5* (ou 3 de 5).

A estrutura do RBD apresenta a organização necessária para o sistema funcionar, ou seja, o diagrama representa o modo operacional do ambiente que está sendo modelado (MELO et al., 2013). O modo operacional apresenta quais componentes do sistema devem estar em funcionamento para que o sistema deva responder corretamente.

O modelo RBD está disponível caso haja um caminho contínuo do vértice de origem ao vértice de destino. A Figura 2.4(a) representa o sistema disponível, mesmo com a falha de um dos componentes, uma vez que é possível através do vértice de origem atingir o vértice de destino. Já na representação do sistema indisponível, ilustrado na Figura 2.4(b), as falhas dos componentes impedem que através do vértice de origem, seja alcançado o vértice de destino.



(a) Sistema Disponível (b) Sistema Indisponível

Figura 2.4 Disponibilidade em RBD.

Quando os componentes estão organizados em série, é necessário apenas que um componente falhe para o sistema falhar. A disponibilidade de modelos RBD com blocos em série é obtida através do produto das disponibilidades de cada um dos n blocos componentes. Um sistema com n componentes em série, como ilustra a Figura 2.3(a), a disponibilidade pode ser calculada a partir da Equação 2.8, onde A_s corresponde ao valor da disponibilidade do sistema e A_i a disponibilidade do componente i. Já a sua confiabilidade é calculada a partir da Equação 2.9, onde R_s corresponde à confiabilidade do sistema no instante de tempo t.

$$A_s = \prod_{i=1}^{n} A_i \tag{2.8}$$

$$R_s(t) = \prod_{i=1}^{n} R_i(t) \tag{2.9}$$

Quando os blocos estão organizados em paralelo, o sistema só irá falhar quando todos os elementos falharem. Desta maneira, um sistema com n componentes, tal como mostra a Figura 2.3(b), possui a disponibilidade calculada pela Equação 2.10, onde A_s é a disponibilidade do sistema, A_i a disponibilidade do componente i e $(1-A_i)$ corresponde a indisponibilidade. Já a confiabilidade é calculada pela Equação 2.11, onde, $R_s(t)$ é a confiabilidade do sistema no instante de tempo t e $R_i(t)$ corresponde à confiabilidade do componente i no instante de tempo t.

$$A_s = 1 - \prod_{i=1}^{n} (1 - A_i) \tag{2.10}$$

$$R_s(t) = 1\prod_{i=1}^{n}(1 - R_i(t)) \tag{2.11}$$

Os modelos RBDs são utilizados em sistemas que contêm módulos independentes, onde cada um pode ser representado por um bloco de confiabilidade. A organização em blocos permite a modelagem de situações onde os comportamentos de falha e reparo de cada componente não afetam os demais (MACIEL et al., 2011). Isso facilita a modelagem de sistemas que não possuem dependência entre os componentes. Dessa forma, caso seja necessário a modelagem de sistema mais complexos, onde exista uma dependência entre os componentes, o modelo RBD não será suficiente para representar, assim, o projetista deve recorrer a uma abordagem de modelagem hierárquica ou a um modelo de espaço de estados, na tentativa de obter resultados mais expressivos.

2.4.2 Cadeias de Markov

Modelos de *Markov* são blocos de construção fundamentais sobre os quais muitas técnicas quantitativas de análise de desempenho são construídas (KOLMOGOROFF, 1931; TRIVEDI, 2002). Tais modelos podem ser utilizado para representar as interações entre os vários componentes do sistema, tanto para fins descritivos quanto preditivos (MENASCE; DOWDY; ALMEIDA, 2004). Na ciência da computação, em particular, este formalismo é bastante conveniente para descrever e analisar propriedades dinâmicas de sistemas computacionais (BOLCH et al., 2006).

As cadeias de *Markov* podem ser divididas em duas classes, de acordo com o parâmetro de tempo: *Discrete Time Markov Chain* (DTMC), para tempo discreto e CTMC, para tempo contínuo. A principal diferença entre essas duas classes é que nas CTMCs, as transições entre os estados podem ocorrer em qualquer instante do tempo, enquanto que nas DTMCs, as transições só podem ser feitas em pontos discretos no tempo.

A representação gráfica de um modelo através do formalismo de cadeia de *Markov*, pode ser representada usando digramas de transições entre estados, onde os vértices representam os estados e os arcos do diagrama representam as possíveis transições entre os estados (CASSANDRAS; LAFORTUNE, 2009). As transições entre estados representam a ocorrência de eventos (MACIEL et al., 2011).

A Figura 2.5 apresenta um exemplo de modelo CTMC de disponibilidade com dois estados e duas transições. Onde, o estado A representa que o sistema está funcionando, a partir deste, a falha pode ocorrer (λ), levando ao estado B. Do estado B, apenas o reparo (μ) é possível.

Cadeias de *Markov* também possuem uma representação em forma de matriz, denominada matriz de taxa de transição Q. A matriz Q possui as informações sobre as transições dos estados na cadeia de *Markov*. Esta é utilizada para a resolução de cadeias de *Markov*. Cada elemento localizado fora da diagonal principal representa a taxa de ocorrência dos eventos que efetivam a transição dos estados do sistema. Os elementos contidos na diagonal principal são os valores

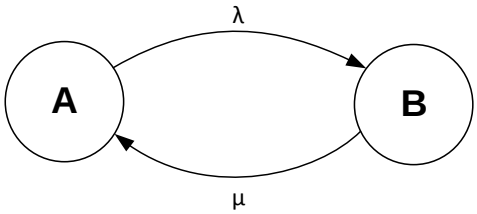


Figura 2.5 Exemplo de Modelo Básico de Disponibilidade CTMC

necessários para que a soma dos elementos de cada linha seja igual a zero. Por exemplo, utilizando a CTMC da Figura 2.5, teremos a matriz da Equação 2.12. As probabilidades de transição dos estados podem ser calculadas através da Equação 2.13.

$$Q = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix} \tag{2.12}$$

$$p_{i,j} = (s,t) = P(X(t) = j | X(s) = i) \tag{2.13}$$

A análise estacionária de uma cadeia de *Markov* consiste em encontrar a probabilidade de o sistema estar em um determinado estado, considerando um longo tempo de execução. Estas probabilidades são independentes do estado inicial do sistema e são representadas pelo vetor $\pi = [\pi_1, \pi_2, ..., \pi_n]$, onde π_1 é a probabilidade estacionária para o estado *i.* Uma vez que a cadeia de *Markov* considerada é uma cadeia ergódica, podemos encontrar a probabilidade estacionária através do sistema linear formado pelas Equação 2.14 e Equação 2.15 (BOLCH et al., 2006).

$$\pi Q = 0 \tag{2.14}$$

$$\sum_{i=1}^{n} \pi_i = 1 \tag{2.15}$$

Na Equação 2.14, Q é a matriz de estados e π (vetor de probabilidade) é o autovetor correspondente ao autovalor unitário da matriz de transição. A Equação 2.15 é a condição de normalização, adicionada para assegurar que a solução obtida é um único vetor de probabilidade.

Uma cadeia de *Markov* é dita ergódica, se é possível alcançar qualquer estado a partir de qualquer outro estado, em um número *n* finito de passos, em outras palavras, é a possibilidade não nula de qualquer estado poder ser alcançado através de uma ou mais transições a partir de qualquer outro estado. É importante ressaltar que a soma dos elementos do vetor de probabilidade π deve ser igual a 1, ou seja, $\|\pi\| = 1$ (MACIEL et al., 2011). Desta forma, a utilização de CTMCs permite avaliações tanto de desempenho, quanto de disponibilidade. Suas aplicações vão desde a quantificação da vazão de uma linha de produção até a determinação de tempos de falha e reparo em sistemas críticos (BAIER et al., 2003).

Para uma avaliação de disponibilidade é importante dar preferência a modelos focados

no objetivo a ser avaliado. O enfoque deve estar no propósito do modelo, que deve ser o mais simples para atender ao propósito de responder o objetivo desejado (SILVA et al., 2014). Desta forma, adotamos o modelo CTMC para a modelagem e análise do processo de descarga da bateria. Os modelos CTMCs concebidos serão apresentados na Seção 4.2.

Outros formalismos também podem representar o comportamento deste componente (Bateria), como por exemplo, SPN e Redes de Fila (*Queueing Networks, QN*) (BOLCH et al., 2006). Estes formalismos possuem uma correspondência com as cadeias de *Markov*, permitindo o uso de ferramentas automatizadas para a geração e solução dos modelos.

A técnica teoria das Filas, são representadas, genericamente, por um processo de chegada de cliente para serem atendidos por servidores a um sistema de atendimento. O sistema recebe um ou mais serviços, executados por certa quantidade de servidores. Nesse sentido, as formações de filas ocorrem, porque a procura pelo serviço, é maior do que a capacidade do sistema de atender a demanda. Os processos ocorrem em sua ordem de chegada, consideram o atendimento em primeiro lugar de quem chegou primeiro na fila (*First In, First Out (FIFO)*) ou o atendimento em primeiro lugar de quem chegou por último na fila (*Last In,First Out (LIFO)*) (BOLCH et al., 2006). Modelos baseados em filas são úteis para a análise de sistemas nos quais conflitos ocorrem quando diversas entidades tentam acessar simultaneamente o mesmo recurso (JAIN, 1990).

Rede de Petri estocástica (SPN) é uma das extensões de rede de Petri (PN) utilizada para a modelagem de desempenho e dependabilidade (BALBO, 2001). Uma rede de Petri estocástica adiciona tempo ao formalismo de redes de Petri, com a diferença de que os tempos associados às transições temporizadas são distribuídas exponencialmente, enquanto o tempo associado às transições imediatas é zero. As transições temporizadas modelam atividades através dos tempos associados, de modo que o período de habilitação da transição temporizada corresponde ao período de execução da atividade, e o disparo da transição temporizada corresponde ao término da atividade (MARSAN et al., 1994; BALBO, 2001). Um problema que ocorre com modelos baseados em espaço de estados é conhecido como explosão do espaço de estados (VALMARI, 1998), quando o tamanho do espaço de estados cresce muito e os recursos de memória não são suficientes para resolver o modelo, inviabilizando, muitas vezes, sua construção ou solução.

Devido às características supracitadas, o modelo CTMC é escolhido devido à sua característica de ausência de memória, o que é relevante para representar o comportamento de descarga da bateria, pois não há dependência do histórico para que a descarga de energia ocorra. A bateria irá descarregar e mudar seu estado de carga atual independente de seu estado anterior e dependendo apenas do parâmetro associado para que ocorra a mudança de estado. A ausência de memória garante que os valores do estado atual em uma cadeia de *Markov* sejam totalmente independentes dos valores dos estados anteriores (GARG et al., 1995).

2.5 Análise de Sensibilidade Diferencial

A análise de sensibilidade é um método que tem como objetivo determinar os fatores de maior influência em um sistema (FRANK; ESLAMI, 1980). Esse método é importante para avaliação dos efeitos dos parâmetros de entrada nas métricas de interesse, e tem como objetivo a identificação de gargalos de disponibilidade e desempenho do sistema (MATOS et al., 2012). Essa abordagem pode ser utilizada para encontrar gargalos de desempenho ou confiabilidade no sistema, guiando assim a otimização dos processos (BLAKE; REIBMAN; TRIVEDI, 1988).

Existem vários métodos disponíveis para a realização de análise de sensibilidade, entre estes podemos citar o projeto de experimento fatorial, análise de correlação, análise de regressão, análise de perturbação e análise diferencial, que também é conhecido como análise de sensibilidade paramétrica ou método direto (HAMBY, 1994). A análise de sensibilidade paramétrica foi adotada para este estudo, pois pode ser realizada de forma eficiente em modelos e é normalmente utilizada em estudos de disponibilidade e desempenho.

Uma outra técnica de análise de sensibilidade, que também foi adotada neste estudo, é um dos métodos mais simples que consiste em variar um parâmetro por vez repetidamente, mantendo os outros fixos. Ao aplicar este método, o *ranking* de sensibilidade é obtido observando as mudanças correspondentes na saída do modelo (HAMBY, 1994). Esta técnica possibilita a identificação de parâmetros que podem ser removidos sem efeito significativo para os resultados. Segundo MATOS et al. (2012), modelos grandes, com dezenas de taxas, podem ser drasticamente reduzidos usando esta abordagem.

A análise de sensibilidade paramétrica é realizada através do cálculo das derivadas parciais para as métricas de interesse de cada parâmetro. Por exemplo, considerando a métrica Y, que depende de um parâmetro λ, a sensibilidade de Y com relação a λ é calculado com a Equação 2.16 e Equação 2.17. Onde S_λ e S^*_λ são também referidos como coeficientes de sensibilidade, cujos valores ordenados produzem o *ranking* usado para comparar o grau de influência entre todos os parâmetros (HAMBY, 1994).

$$S_\lambda(Y) = \frac{\partial_Y}{\partial\lambda} \tag{2.16}$$

$$S^*_\lambda(Y) = \frac{\partial_Y}{\partial\lambda}\left(\frac{\lambda}{Y}\right) \tag{2.17}$$

2.6 Importância dos componentes para a Disponibilidade (*Availability Importance*)

A importância dos componentes é um parâmetro que indica o impacto que um componente em particular pode exerce na disponibilidade global do sistema. A importância de componentes pode ser analisada em termos de confiabilidade (*Reliability Importance*), e de

disponibilidade (*Availability Importante*). Este estudo restringirá a importância dos componentes em relação a disponibilidade (*Availability Importante*) que foi utilizada para identificar os componentes mais críticos que integram os dispositivos móveis.

O índice de *Availability Importante* $\left(I_i^A\right)$ considera o tempo médio para falha dos componentes e o tempo médio de recuperação dos componentes. O cálculo utilizado é apresentado na Equação 2.18, em seguida, os resultados apresentados são normalizados pelo maior valor, conforme Equação 2.19.

$$I_i^A = A_s(l_i, p^i) - A_s(0_i, p^i), \tag{2.18}$$

onde p^i representa o vetor de disponibilidades dos componentes com o *i*ésimo componente removido; 0_i representa a condição quando o componente i é falho e l_i a condição quando o componente i está no modo operacional.

$$In_i = \frac{I_i}{I_x}, \tag{2.19}$$

onde In_i é o índice normalizado para o componente i; I_i é o valor do índice não normalizado para o componente i e, I_x é o valor do maior índice não normalizado entre os componentes.

Essa análise foi utilizada no Estudo de Caso IV, apresentado na Seção 6.4, com o objetivo identificar a importância dos componentes que integram a infraestrutura dos dispositivos móveis em relação a disponibilidade.

2.7 Injeção de Falha

A injeção de falhas é uma abordagem amplamente utilizada em experimentos para a avaliação de atributos de dependabilidade no sistema (ARLAT et al., 1993). Esta técnica é baseada em falhas ou erros predeterminadas introduzidos em um sistema (através de experimentos controlados), com o objetivo de observar o comportamento do sistema mediante a presença de tais falhas inserida, fornecendo *feedback* em tempo hábil (HSUEH; TSAI; IYER, 1997).

Em geral, uma estratégia baseada em injeção de falhas e monitoramento engloba um gerador de carga de trabalho, um injetor de falha, um monitor do sistema e o sistema de destino. O injetor de falha é responsável por injetar eventos de falhas/reparos no sistema de destino. O gerador de carga de trabalho gera comandos (ações falha e reparo) e controla o injetor de falha, enquanto o monitor do sistema observa o status e o comportamento do sistema de destino, coleta dados e fornece medidas e estatísticas (HSUEH; TSAI; IYER, 1997), (ARLAT et al., 1993).

As técnicas de injeção de falhas, segundo ZIADE et al. (2004), são classificadas em cinco categorias: injeção de falha baseada em *hardware*, injeção de falha baseada em *software*, injeção de falha baseada em simulação, injeção de falha baseada em emulação e injeção de falha híbrida.

Injeção de falha baseada em *hardware* requer um hardware adicional para inserir

falhas no sistema alvo. As falhas de *hardware* podem ser de dois tipos: com contato e sem contato (HSUEH; TSAI; IYER, 1997). Na injeção de falha em *hardware* com contato, o injetor tem contato físico direto com o sistema alvo e pode produzir variações de corrente ou tensão no sistema testado. No tipo sem contato, o injetor produz algum evento externo que acarreta a falha do componente desejado.

Injeção de falha baseada em *software* envolve a modificação do estado de funcionamento do software em execução no sistema em teste. Esta técnica é dívida em duas categorias: tempo de compilação ou tempo de execução (HSUEH; TSAI; IYER, 1997; ZIADE et al., 2004). Injetores de falha em tempo de compilação fazem alterações do código fonte do sistema original, a fim de emular o efeito de falhas de hardware e software no sistema. Injetores de falhas de tempo de execução, por sua vez, não exigem a modificação do código fonte do sistema. Em vez disso, este tipo de injetor utiliza algum mecanismo que serve de gatilho para a injeção de falhas. Existem três técnicas para implementar o gatilho que irá disparar as falhas: ***Time-out***: o mais simples dos métodos. Este método utiliza um temporizador (*hardware* ou *software*) para controlar a injeção de falhas. Quando termina o período de tempo predeterminado a falha é injetada; ***Exception/trap***: neste caso, exceções de *hardware* ou *software* transferem o controle para o injetor de falhas, onde as falhas são inseridas sempre que um determinado evento ou condição ocorrer, e ***Code insertion***: instruções são adicionadas ao *software* alvo para permitir que a inserção de falhas ocorra antes de determinados comandos. Ao contrário da categoria de inserção de falhas durante a compilação, este método apenas adiciona instruções ao sistema sem modificar o código já existente.

Injeção de falha baseada em simulação implica na construção de um modelo simulado do sistema em análise. O modelo simulado é modelado tornando o mais realista possível ao comportamento do sistema em análise (ZIADE et al., 2004).

Injeção de falha baseada em emulação foi introduzido como uma solução para melhor reduzir o tempo de execução em comparação com o método de injeção de falha baseada e simulação (KOOLI; NATALE, 2014).

Injeção de falha híbrida é uma abordagem que envolve a combinação das técnicas de injeção de falha baseada em hardware e software (KOOLI; NATALE, 2014).

As técnicas de injeção falha tem sido aceita a bastante tempo para a validação de dependabilidade dos sistemas por meio da análise do comportamento dos dispositivos quando ocorre uma falha (ZIADE et al., 2004). A escolha entre os métodos para injeção de falhas depende do tipo de falhas que se pretende injetar e o tempo para cria-la. Desta forma, a presente pesquisa utilizou métodos de falhas injetadas em nível de software com a técnica *Time-out* que consiste na injeção de falhas ao final de um tempo determinado.

2.8 Intervalo de Confiança da Disponibilidade

A validação refere-se assegurar que as premissas utilizadas no desenvolvimento dos modelos são aceitáveis na medida em que, se implementado corretamente, os modelos irão produzir resultados compatíveis ao observado em sistemas reais (JAIN, 1990).

Para que o modelo seja considerado válido, é necessário que este proporcione o cálculo de métricas com erros de exatidão dentro de níveis considerados aceitáveis, denominado intervalo de confiança da disponibilidade. Deste modo, uma vez que satisfaça as exigências, pode-se afirmar que o modelo corresponde ao comportamento do sistema real, de outro modo, ajustes nos modelos devem ser realizados de modo a representar corretamente o comportamento do sistema real.

Em vista disso, a Arquitetura Base (ver Figura 4.1) foi submetida a um processo de validação (ver Seção 5.2). E para obter o intervalo de confiança, este estudo empregou o método proposto por KEESEE (1965) e para validar o modelo da autonomia da bateria, empregamos o método *Bootstrap* (EFRON; TIBSHIRANI, 1994). Deste modo, é possível atribuir mais confiança aos modelos propostos. As equações aplicadas dos dois métodos são apresentadas a seguir.

Método Keesee

A validação é realizada comparando os resultados obtidos dos modelos com os resultados coletados diretamente do *testbed* (KEESEE, 1965). Com posse dos resultados obtidos, o primeiro passo é adaptar a equação da disponibilidade (Equação 2.2), simplificando-a conforme é apresentado na Equação 2.20.

$$A = \frac{\mu}{\lambda + \mu} = \frac{1}{1 + \frac{\lambda}{\mu}} = \frac{1}{1 + \rho}, \tag{2.20}$$

onde ρ é relação entre $\frac{\lambda}{\mu}$. Considera-se para o experimento o n sendo os eventos de falha e reparo, o tempo total de falha sendo S_n e o tempo total de reparo é Y_n. Para estimar o valor de λ e μ, é utilizada a Equação 2.21, respectivamente.

$$\hat{\Lambda} = \frac{n}{S_n} \;\; e \;\; \hat{M} = \frac{n}{Y_n} \tag{2.21}$$

Um intervalo de confiança 100x(1-α) para λ e μ é dado por uma distribuição Qui-quadrada com os parâmetros definidos na Equação 2.22 e Equação 2.23, respectivamente.

$$\left(\frac{\chi^2_{2n;1-\frac{\alpha}{2}}}{2S_n}, \frac{\chi^2_{2n;\frac{\alpha}{2}}}{2S_n} \right) \tag{2.22}$$

$$\left(\frac{\chi^2_{2n;1-\frac{\alpha}{2}}}{2Y_n}, \frac{\chi^2_{2n;\frac{\alpha}{2}}}{2Y_n} \right) \tag{2.23}$$

Deste modo, o estimador de probabilidade máxima para a relação $\frac{\lambda}{\mu}$ é $\hat{\rho}$, definido na Equação 2.24.

$$\hat{\rho} = \frac{\hat{\Lambda}}{\hat{M}} = \frac{\frac{n}{S_n}}{\frac{n}{Y_n}} = \frac{Y_n}{S_n} \tag{2.24}$$

Um intervalo de confiança para ρ é dado entre ρU (superior) e ρL (inferior), descrita por uma função de probabilidade da Distribuição F, vista na Equação 2.25.

$$\rho_U = \frac{\hat{\rho}}{F_{2n;2n;1-\frac{\alpha}{2}}} \; e \; \rho_L = \frac{\hat{\rho}}{F_{2n;2n;\frac{\alpha}{2}}} \tag{2.25}$$

Por fim, torna-se possível calcular o intervalo de confiança para a disponibilidade do sistema real, onde o estimador de probabilidade máximo para a disponibilidade é $\hat{A} = \frac{1}{1+\hat{\rho}}$. Uma vez que a disponibilidade A é uma função monotonicamente decrescente de ρ, o intervalo de confiança 100x(1-α) para A, ou seja, A_U e A_L, é encontrado através da Equação 2.26.

$$\left(\frac{1}{1+\rho_L}, \frac{1}{1+\rho_U} \right) \tag{2.26}$$

Com o intervalo de confiança definido, a validação do modelo proposto pode ser realizada, e pode, por conseguinte, ser determinado se a disponibilidade calculada através do modelo está dentro do intervalo de confiança obtido para o sistema real.

Método Bootstrap

O método *Bootstrap*, proposto por EFRON; TIBSHIRANI (1994), consiste em aplicar um esquema de reamostragem de uma amostra original x, a fim de realizar inferências sobre uma estatística através de sua distribuição aproximada obtida a partir das reamostras x^*.

A reamostragem *Bootstrap* visa simular diversos cenários a partir de uma pequena amostra, desta forma o método constrói um espaço amostral e realiza inferências sobre os parâmetros de interesse para o modelo.

Essa técnica permite aproximar a distribuição de uma variável aleatória pela distribuição empírica baseada em uma amostra de tamanho finito desta variável. A amostragem é feita com reposição, da distribuição da qual os dados são obtidos, se esta é conhecida (*bootstrap paramétrico*) ou da amostra original (*bootstrap não paramétrico*) (EFRON; TIBSHIRANI, 1994).

Em uma amostra *Bootstrap*, a estimativa se aproxima do valor real quando o número de reamostragem tende a infinito, ou seja, quando a reamostragem é realizada várias vezes, sendo que o tamanho da reamostragem é escolhido de acordo com a finalidade. De acordo

com (CHERNICK, 2011; EFRON; TIBSHIRANI, 1994) para obter boas estimativas dos limites de confiança seria necessário no mínimo 1000 replicações.

Para obter um intervalo de confiança 95%, ordena-se as estimativas de interesse obtidas em ordem crescente delimitando os quantis de ordem 0.025 e 0.975. Em nosso estudo está sendo considerado a média, em cada reamostra que foi gerada da amostra original. Na distribuição *Bootstrap*, o intervalo entre esses dois valores (0.025 e 0.975) é frequentemente utilizado como um intervalo de confiança por si só, sendo conhecido como um intervalo de confiança percentil do *Bootstrap* (EFRON; TIBSHIRANI, 1994).

Para auxiliar no processo do método *Bootstrap*, foi desenvolvido um *script* (Apêndice A) através da ferramenta *Mathematica*[2]. O processo realizado também pode ser representado pelo Pseudocódigo 1, onde os parâmetros de entradas são: o número de reamostras que se pretende gerar, o nível de significância e amostra original; representados por B, α e $data$, respectivamente.

Pseudocódigo 1 Pseudo algoritmo *Bootstrap*

Data: B, α, $data$;
Result: *Intervalo de confiança baseado nos percentis Bootstrap*

```
1:
2:  n ← size ( data );
3:
4:  for i = 1; i <= B; i++ do
5:      for j = 1; j <= n; j++ do
6:          sample[j] ← rand ( data );
7:      end
8:      means[i] ← mean ( sample );
9:  end
10:
11: confidenceInterval ← Quantile [ means, (α/2,(1-α/2)) ];
12: return confidenceInterval
```

As principais etapas do pseudocódigo são explicadas em linguagem natural da seguinte forma:

- O pseudocódigo inicia com uma amostra original, de tamanho n, para aplicar o método de reamostragem B vezes e para obter as estimativas dos limites de confiança (α);
- Na linha 6, cria uma amostra com reposição (*sample*), baseada na amostra original, de tamanho n, repetindo B vezes;
- Na linha 8, obtém a estimativa de interesse, neste caso, está sendo considerado a média em cada reamostra que foi gerada da amostra original;
- Na linha 11, determina o intervalo de confiança percentil.

[2] *Mathematica: http://www.wolfram.com/mathematica/?source=nav*

Uma ilustração com cada passo desse método é apresentado na Figura 2.6.

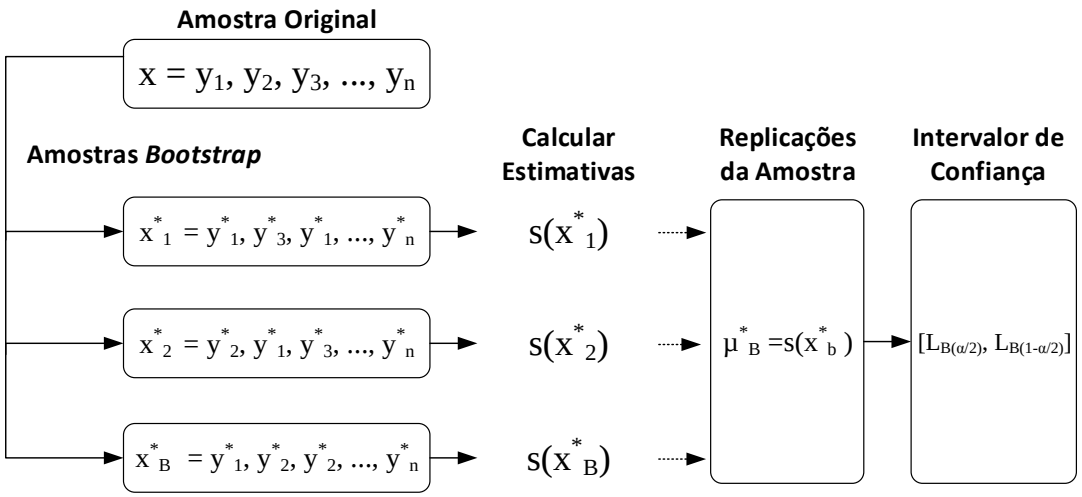


Figura 2.6 Ilustração do método *Bootstrap*

Inicialmente, obtêm-se uma amostra original x=(y_1, y_2, y_3, ...,y_n), de tamanho n. Uma amostra *Bootstrap* (x_1^*, x_2^*, ..., x_B^*) é obtida por amostragem aleatória B vezes, com reposição, a partir da amostra original. A notação ($*$) indica que x^* é um conjunto de dados reorganizado da amostra original, onde alguns elementos podem não aparecer e outros aparecer com mais frequência. Por exemplo, assumimos uma amostra original x=(y_1, y_2, y_3, y_4, y_5, y_6) com n=6, uma amostra *Bootstrap* gerada pode ser representada por x^*=(y_6^*, y_5^*, y_1^*, y_2^*, y_2^*, y_1^*). Cada elemento em y tem a mesma probabilidade de ser selecionado.

Em seguida, para cada amostra *Bootstrap* estima-se a estatística de interesse ($s(x_b^*)$), resultando nas replicações da amostra, $\mu_b^* = s(x_b^*)$, b=1,2,3,...B, onde B é o número de amostras *Bootstrap* geradas. Desta forma, a cada amostra *Bootstrap* tem-se replicações, que é o valor da estatística avaliada.

No intervalo de confiança *Bootstrap* percentil utiliza-se o percentil $(\alpha/2)$ (limite inferior) e $(1-\alpha/2)$ (limite superior) das estatísticas amostrais no *Bootstrap*. Assim, com o coeficiente de confiança (100 x (1-α))% é obtido pelos (α/2) -ésimo e (1-α/2) -ésimo percentis da replicação da amostra. Os valores obtidos do *Bootstrap* são atribuídos ao modelo para estimar a disponibilidade dentro do intervalo de confiança.

3

Metodologia de Avaliação e Arquitetura Base

Este capítulo apresenta a metodologia utilizada para a modelagem e avaliação de disponibilidade de um ambiente IoT para o provimento de serviço mHealth na nuvem. Em seguida, apresentamos o cenário base, que consiste em determinar uma arquitetura com os requisitos mínimos para execução do serviço analisado.

3.1 Metodologia: Visão Geral

Os procedimentos adotados na avaliação desta pesquisa foram norteados por uma metodologia que abrange desde a compreensão das características dos serviços *mHealth* até a análise dos resultados inerentes à própria pesquisa. A metodologia utilizada é representada por um fluxograma, conforme ilustrado na Figura 3.1. O fluxo de atividades contidas na metodologia é descrito em pormenor nesta seção.

- **Entendimento do sistema -** Esta etapa aborda o sistema a ser analisado. Compreende a identificação das características de um serviço *mHealth* na nuvem, utilizando o paradigma IoT, bem como seus componentes, comportamentos e interações. Ainda nesta etapa, são definidas quais métricas de dependabilidade que serão avaliadas no serviço em questão. No presente trabalho foram definidas as métricas de disponibilidade e indisponibilidade do serviço;

- **Definição da Arquitetura Base -** Esta etapa consiste na elaboração de uma arquitetura, definida como Arquitetura Base, contendo os requisitos mínimos para o provisionamento do serviço. O intuito é verificar os principais componentes sob o ponto de vista da disponibilidade do serviço. A partir da definição desta arquitetura, temos a base para a geração de modelos. Esta atividade é melhor detalhada na Seção 3.2;

- **Concepção do modelo -** Esta etapa concerne na geração do modelo que represente o sistema a ser avaliado. A geração do modelo é definida com base nos componentes delineados nas etapas anteriores. Desta forma, a partir da definição da Arquitetura

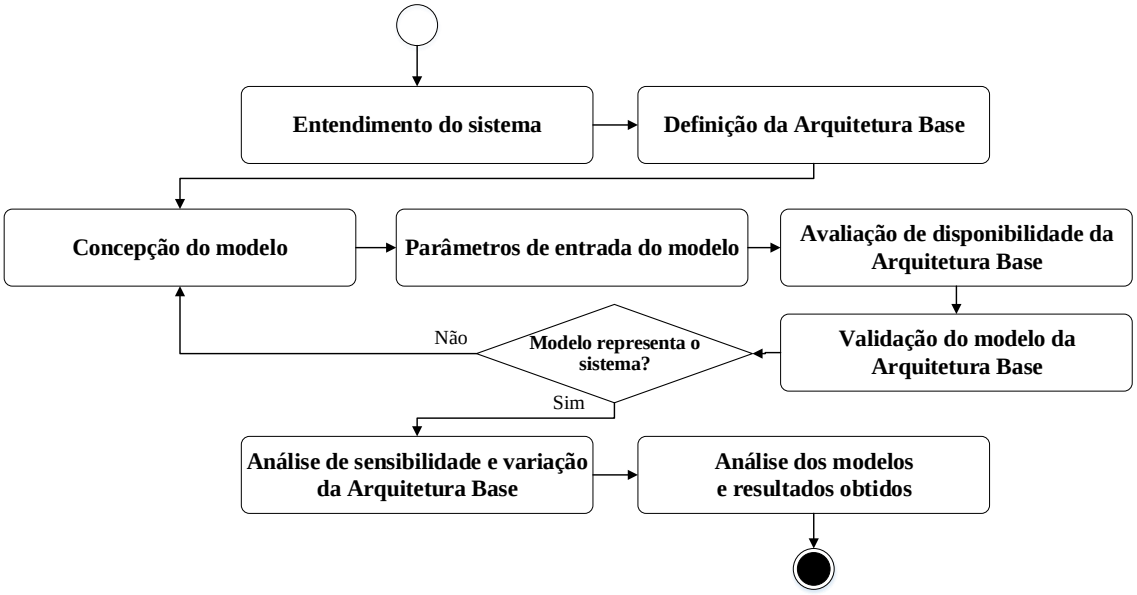


Figura 3.1 Fluxograma da metodologia aplicada

Base, foi aplicada uma abordagem utilizando modelagem hierárquica heterogênea para representar a arquitetura e avaliar a disponibilidade do serviço. Para representar o relacionamento entre os componentes e subcomponentes que formam a Arquitetura Base, utilizou-se o diagrama de blocos de confiabilidade (RBD) (Seção 4.1). Para o processo de descarga da bateria, em ambos os dispositivos, utilizamos o modelo baseado em estado CTMC (Seção 4.2). Além disso, utilizando os modelos RBD e CTMC, foi possível avaliar disponibilidade e confiabilidade de cada dispositivo móvel, como também, os efeitos da autonomia da bateria sobre a disponibilidade do serviço. Para compor os modelos são utilizadas as ferramentas de modelagem: *Mercury*[1] e *Sharpe* [2]. A obtenção das equações de fórmulas fechadas foram dadas através da ferramenta *Mathematica*[3], cujo processo para concepção das equações está disponível em MODCS (2015a). Ainda nesta fase, quando necessário, ajustes são realizados de modo a representar corretamente o comportamento do sistema.

- **Parâmetros de entrada do modelo -** Para que o modelo seja refinado e avaliado, é necessário que se obtenha ou que se colete dados acerca do sistema. Portanto, nesta etapa, é verificado se existem valores de falha e reparo (MTTF e MTTR) para cada componente dos modelos concebidos na etapa anterior (Concepção dos Modelos). Em posse desses valores, é necessário que haja a parametrização dos modelos. Em

[1] *Mercury* (SILVA et al., 2012). Disponível em: http://www.modcs.org/?page_id=1397
[2] *Sharpe* (TRIVEDI, 2015). Disponível em: http://sharpe.pratt.duke.edu
[3] *Mathematica* (WOLFRAM RESEARCH, 2015). Disponível em: http://www.wolfram.com/mathematica

alguns casos, esses valores podem ser obtidos através de dados dos fabricantes ou na literatura. Entretanto, caso algum parâmetro essencial seja desconhecido, faz-se necessário realizar a descoberta através de experimentos para obtenção e validação de tais informações.

Neste estudo, dentre os parâmetros necessários para a exploração dos modelos, apenas os valores relacionados à autonomia da bateria dos dispositivos móveis não foram identificados na literatura e nem publicados pelos fabricantes. Portanto, foi necessário realizar um experimento para autonomia da bateria dos dispositivos móveis, com intuito de estimar o tempo de descarga das baterias. O experimento e a validação da autonomia da bateria é detalhado na Seção 5.1.

- **Avaliação de disponibilidade da Arquitetura Base -** Após constatada a existência de todos os valores necessários para parametrização dos modelos (RBD e CTMC), nesta etapa, é possível realizar uma análise de dependabilidade da Arquitetura Base, verificando métricas de disponibilidade e indisponibilidade anual (*downtime*). A parametrização dos modelos foi efetuada a partir dos valores de falha e reparo dos componentes que formam a Arquitetura Base (esta avaliação é apresentada na Seção 6.1).

- **Validação do modelo da Arquitetura Base -** A etapa de validação é responsável por verificar se o modelo representa o mesmo comportamento que o sistema real. Para tal, foi criado um ambiente de teste controlado, onde é instalado e configurado o sistema, conforme a Arquitetura Base, para a realização do experimento de injeção de falhas. A injeção de falhas ocorreu através de algoritmos, desenvolvidos em *Shell Script*, para inserir falhas no serviço do sistema com o intuito de simular o comportamento do ambiente real, e para monitorar o serviço com a finalidade de verificar se o sistema está operacional ou não. O ambiente de teste para a injeção de falhas é detalhado na Seção 5.2.1. Em posse dos resultados gerados pelos modelos propostos, a partir da etapa anterior, com os resultados obtidos no experimento de injeção de falhas, foi realizada a validação do modelo da Arquitetura Base (Seção 5.2.2).

- **Modelo representa o sistema -** Para que o modelo seja considerado válido, é necessário que este esteja dentro do intervalo de confiança da disponibilidade do serviço. Uma vez que satisfaça a condição, o passo seguinte é a análise de sensibilidade e variação da Arquitetura Base. Caso a condição não seja satisfatória, ajustes nos modelos devem ser realizados. O processo de validação do modelo da Arquitetura Base será melhor descrito na Seção 5.2.2.

- **Análise de sensibilidade e variação da Arquitetura Base:** Nesta etapa para identificar os parâmetros dos componentes que apresentam um maior impacto na disponibilidade da Arquitetura Base, uma análise de sensibilidade é aplicada. Foi aplicada uma

análise detalhada, onde foi possível observar os efeitos causados na disponibilidade ao variar os valores dos parâmetros.

Considerando os resultados fornecidos pela Análise de Sensibilidade, é possível criar extensões da Arquitetura Base. Neste estudo, dois cenários são modelados ampliando o modelo da Arquitetura Base, visando aumentar a disponibilidade do sistema. O primeiro investiga múltiplas interfaces de rede, enquanto o segundo estuda a possibilidade de incluir *Cloudlet*. Uma análise detalhada foi realizada sobre ambas as arquiteturas, com o objetivo de obter os resultados de disponibilidade e indisponibilidade anual. Ainda nessa fase, foi necessário reajustar os modelos CTMCs, que representam a autonomia da bateria dos dispositivos móveis. Visto que, os novos cenários gerados introduzem diferentes interfaces de conexão.

- **Análise dos modelos e resultados obtidos:** Por fim, nesta última etapa, é possível realizar uma análise dos modelos projetados e apresentar os resultados obtidos através de uma comparação entre a Arquitetura Base e suas extensões. Também foi possível avaliar cada componente que integra a infraestrutura dos dispositivos móveis, representado pelo RBD. Através dessa avaliação foi possível observar qual possui maior impacto na disponibilidade dos dispositivos, para assim aprofundar estudos e elaborar técnicas que visem ampliar a disponibilidade do elemento. Métricas desta natureza indicam como um componente em particular é relevante para o sistema. Ainda nesta etapa, é apresentado o tempo médio no qual o dispositivo móvel irá funcionar sem que apresente falhas em seus componentes e investigar o impacto do uso de diferentes interfaces de conexão sobre a autonomia da bateria em relação a disponibilidade do serviço.

3.2 Arquitetura Base

Um cenário base é inicialmente utilizado para o estudo, denominado de Arquitetura Base, é uma arquitetura sem redundância, possuindo os requisitos mínimos para o provimento do serviço, com intuito de verificar os principais componentes do sistema sob o ponto de vista da disponibilidade do serviço.

A Figura 3.2 representa a Arquitetura Base proposta, com uma visão ampla do funcionamento do sistema proposto, nomeado como *MoDCS mHealth*, que pode ser dívida em três camadas: *wearable*; dispositivos móveis, como *smartphones* e *tablets*; e Nuvem. A Arquitetura Base explorada por este estudo é uma extensão da arquitetura analisada por OLIVEIRA et al. (2013) com a adição do dispositivo *smartwatch* e *interfaces de rede*. Todos esses componentes são importantes do ponto de vista da disponibilidade, visto que a falha em um deles pode acarretar na indisponibilidade do serviço.

Essa arquitetura representa um ambiente IoT para um serviço *mHealth* na nuvem, uti-

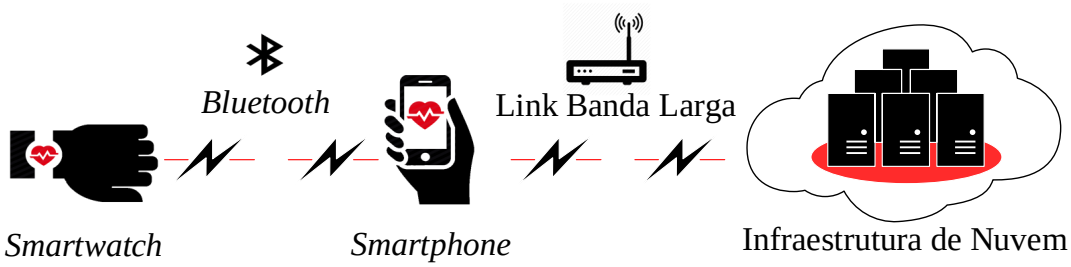


Figura 3.2 Arquitetura Base proposta.

lizando um dispositivo *smartwatch* que fornece informações relacionadas à saúde do usuário (frequência cardíaca), e os envia para o *smartphone* (*gateway*) através de uma conexão *Bluetooth*. O *smartphone*, por sua vez, recebe as informações do *smartwatch* e as encaminha para a nuvem. Os dados são encaminhados para o armazenamento na Nuvem através da conexão Banda Larga. Com as informações armazenadas na Nuvem, outros dispositivos móveis poderão ser utilizados para realizar o acompanhamento quando necessário.

O dispositivo *smartwatch*[4] adotado para este estudo utiliza o sistema operacional *Android Wear 1.4*, possuindo suporte a conexão *Bluetooth* e Banda Larga (podendo conectar-se à dispositivos *Android*, com versão igual ou superior à 4.3); bateria com capacidade de 410 mAh e tecnologia *Lithium Ion*; opera em uma frequência 1200Mhz; processador *Qualcomm*, *Quad core* com 1.2GHz.

Já o dispositivo *smartphone*[5] utiliza o sistema operacional *Android 4.4.2*, modelo GT-i9192; processador *Qualcomm Snapdragon* 400 MSM8930 / Krait 300; bateria com capacidade de 1900 mAh; opera em frequência de 850/900/1900/2100 para tecnologia 3G e GSM 850/900/1800/1900.

A infraestrutura da nuvem, ilustrada na Figura 3.3, é formada pelo Controlador de Nuvem, pelo Gerenciador de Armazenamento e nós de *clusters*. O Controlador de Nuvem corresponde ao nó físico onde executam os softwares de gerenciamento da infraestrutura de nuvem. Suas principais atribuições são monitorar e alocar recursos de um ou mais *clusters* e provisionar estes recursos para os clientes da nuvem (OLIVEIRA et al., 2013). O Gerenciador de Armazenamento provê mecanismos de armazenamento para a nuvem, com a finalidade de armazenar imagens de máquinas virtuais, *snapshots* de instâncias de VMs, dados de aplicações da nuvem, etc (OLIVEIRA et al., 2013). Os componentes da nossa arquitetura recebem nomes genéricos para enfatizar que o modelo não está ligado a nenhum *framework* específico.

Quanto à infraestrutura da Nuvem, concentramos os esforços em dois pontos: Servidor Web e Servidor de Banco de Dados, conforme ilustrado na Figura 3.3. Estes dois servidores são implantados em máquinas virtuais que executam na nuvem. Este enfoque se dá pelo fato destes estarem intrinsecamente relacionados ao provimento direto do serviço. Consideramos que

[4] Especificações: http://smartwatches.specout.com/l/83/LG-Watch-Urbane e http://lg.com/br/wearables/lg-W150

[5] Especificações: http://www.samsung.com/br/support/model/GT-I9192ZKPZTO e http://smartphones.specout.com/l/539/Samsung-Galaxy-S4-Mini

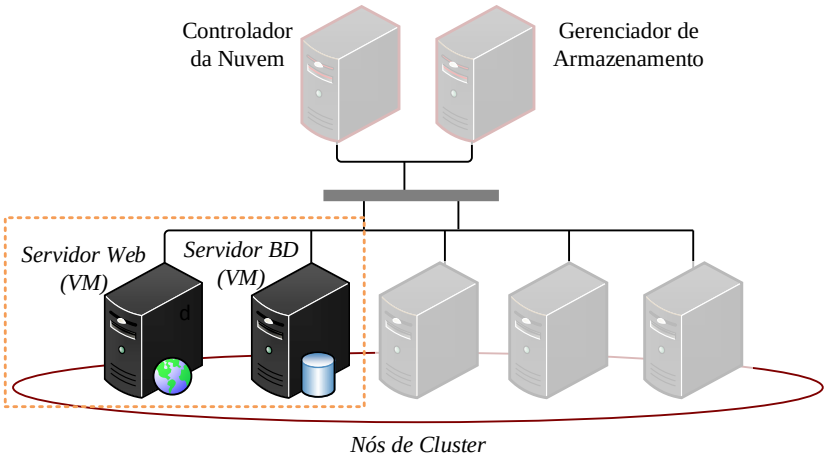


Figura 3.3 Infraestrutura da Nuvem.

a infraestrutura da Nuvem está funcionando de maneira adequada, bem como o gerenciamento dos nós está sendo realizado de modo eficiente, não provocando impacto na disponibilidade do ponto de vista do cliente móvel.

Cada máquina virtual executa em um nó de *cluster* diferente da infraestrutura. Os nós de *cluster* são os componentes da nuvem responsáveis por oferecer seus recursos computacionais para os clientes da nuvem através de virtualização. Uma infraestrutura em nuvem é formada por mais componentes além dos dois nós de *cluster* que executam as máquinas virtuais que fornecem o serviço aos clientes móveis.

Contudo, neste estudo iremos considerar que a falha nos demais componentes da nuvem: nós de *cluster* adicionais, gerenciador da infraestrutura e gerenciador de armazenamento, não provocam impacto na disponibilidade do ponto de vista do cliente móvel.

Desta forma, a falha nos demais componentes poderiam afetar apenas a habilidade de gerenciar novas máquinas virtuais ou executar tarefas administrativas nos nós de *cluster*. Enquanto os nós sob os quais as máquinas virtuais que proveem o serviço para os clientes estiverem funcionando corretamente, os clientes estarão aptos a utilizar o serviço.

Na Seção 4.1 é apresentado o modelo de disponibilidade para a Arquitetura Base, bem como cada componente que influencia na disponibilidade do sistema.

3.3 Considerações Finais

Este capítulo inicialmente apresentou a metodologia utilizada para avaliação de disponibilidade, através de uma sequência de passos que parte do entendimento do sistema até as análises dos modelos e resultados obtidos. Em seguida, apresentamos a Arquitetura Base, sem redundância e que possui os requisitos mínimos para o provimento do serviço *mHealth* em uma nuvem móvel. Desta forma, com a definição da arquitetura e seus componentes, temos a base para a geração de modelos.

4

Modelos

Neste capítulo, apresentamos os modelos criados para a avaliação da disponibilidade da Arquitetura Base. Inicialmente é apresentada uma modelagem hierárquica heterogênea que representa a Arquitetura Base e os seus componentes. Posteriormente, é apresentado um modelo para estimar a autonomia da bateria dos dispositivos móveis. Por meio dos modelos gerados foi possível obter as fórmulas importantes para calcular as métricas de disponibilidade.

4.1 Modelo de Disponibilidade da Arquitetura Base

Esta seção apresenta os modelos proposto para a Arquitetura Base. Uma abordagem de modelagem hierárquica heterogênea foi adotada para representar a arquitetura e avaliar a disponibilidade do serviço. A representação ocorreu por meio de modelos RBD e CTMC.

Modelos hierárquicos têm sido amplamente utilizados para o planejamento e avaliação de dependabilidade de vários tipos de sistemas, algumas aplicações de modelagem utilizadas são: computação em nuvem (SOUSA et al., 2014; DANTAS et al., 2012), computação em nuvem móvel (MATOS et al., 2015; OLIVEIRA et al., 2013), consumo energético de dispositivos móveis (SILVA et al., 2014), programação de rejuvenescimento de software em ambientes de nuvem (MELO et al., 2013), plataforma MBaaS (COSTA et al., 2015), *Vod Streaming* na nuvem (BEZERRA et al., 2014; DE MELO et al., 2014) e entre outros.

O modelo RBD foi adotado por representar em alto nível o relacionamento entre os componentes que compõem a arquitetura proposta. O RBD indica como o funcionamento dos componentes de um sistema afeta a operação do sistema como um todo. Além disso, o RBD possibilita o cálculo de métricas de dependabilidade, como a disponibilidade (MACIEL et al., 2012), que é o objetivo desse estudo.

No entanto, o modelo RBD não é suficiente para a modelagem dos subcomponentes Bateria dos dispositivos móveis, pois não aborda a modelagem de transição de estados que ocorre nestes subcomponentes, desta forma foi adotado um modelo baseado em estados, o CTMC (MACIEL et al., 2011). O modelo CTMC pode ser utilizado em análise de disponibilidade, além de tornar possível a obtenção de equações de fórmula fechada para obtenção de disponibilidade

do serviço modelado.

Sendo assim, a Figura 4.1 representa o modelo hierárquico heterogêneo para a Arquitetura Base descrita na Seção 3.2. Os blocos do modelo que apresentam sombreamento indicam que o componente é representado por outro submodelo de forma hierárquica.

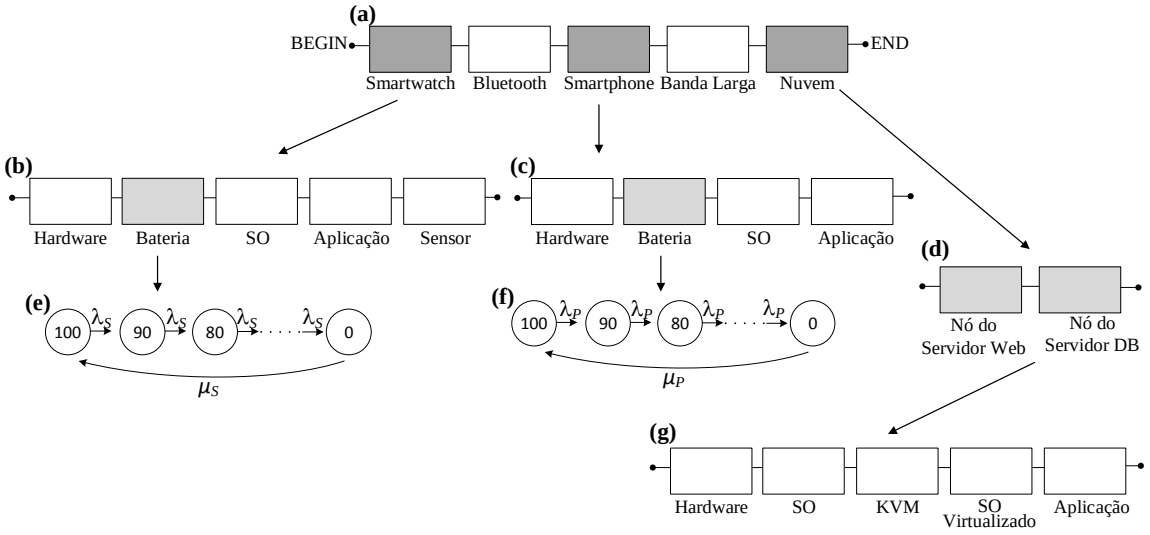


Figura 4.1 Modelo hierárquico para a Arquitetura Base.

O nível superior, descrito na Figura 4.1 (a), representa o modelo de alto nível RBD da arquitetura proposta. Este modelo é caracterizado pela composição de cinco blocos em série dos seguintes subsistemas: *Smartwatch* (SW), *Bluetooth* (BT), *Smartphone* (SP), Banda Larga (BL) e Nuvem (MC). Todos estes subsistemas são importantes do ponto de vista da disponibilidade, desta forma, o sistema estará disponível somente se todos os subsistemas estiverem funcionando corretamente, consequentemente, o modo operacional é representado pela Expressão 4.1 (MACIEL et al., 2012):

$$OM_{Sys} = SW \wedge BT \wedge SP \wedge BL \wedge MC, \tag{4.1}$$

onde SW, BT, SP, BL, e MC são funções *booleanas* que expressam o estado de funcionamento dos respectivos componentes do sistema. A multiplicação da disponibilidade de cada subsistema, resulta na disponibilidade total da Arquitetura Base, que é representada pela Equação 4.2.

$$A_{Sys} = A_{SW} \times A_{BT} \times A_{SP} \times A_{BL} \times A_{MC} \tag{4.2}$$

Onde:

A_{Sys} representa a disponibilidade da Arquitetura Base;

A_{SW} representa a disponibilidade do *Smartwatch*;

A_{BT} representa a disponibilidade do *Bluetooth*;

A_{SP} representa a disponibilidade do *Smartphone*;
A_{BL} representa a disponibilidade do Banda Larga;
A_{MC} representa a disponibilidade da Nuvem.

Já a indisponibilidade do sistema pode ser calculada com a Equação 4.3 e o *downtime* anual em horas pode ser representado através da Equação 4.4.

$$UA = 1 - A_{Sys} \tag{4.3}$$

$$Dty_{Sys} = UA \times 8760h \tag{4.4}$$

Os blocos que representam o *Smartwatch*, *Smartphone* e a Nuvem têm suas disponibilidades calculadas através de submodelos, enquanto os blocos *Bluetooth* e Banda Larga não são expandidos. Desta forma, a disponibilidade dos blocos do *Bluetooth* e da Banda Larga é estimada pela Equação 2.2, utilizando os valores médios de falha e reparo para cada componente. A falha da conectividade com a Banda Larga é constituída pela falha do ponto de acesso ou pelo bloqueio do sinal (CHEN et al., 2003).

A Figura 4.1 (b), representa o modelo RBD referente ao subsistema do dispositivo *smartwatch*, que é composto por cinco blocos em série: *Hardware* (HW), Bateria (BR), Sistema Operacional (SO), Aplicação (AP) e Sensor (SR). A disponibilidade do *Hardware* (A_{HW}), Sistema Operacional (A_{OS}), Aplicação (A_{AP}) e Sensor (A_{SR}), é calculada utilizando as respectivas taxas de reparo e falhas, aplicados à Equação 2.2. A disponibilidade da Bateria (A_{BR}) é calculada através do modelo CTMC, apresentado na Seção 4.2. Sendo assim, a disponibilidade do subsistema *Smartwatch* é dada pela Equação 4.5.

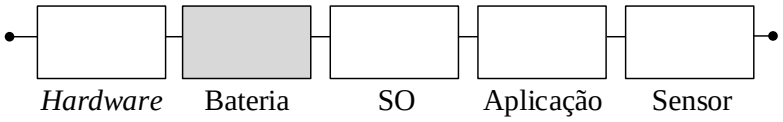


Figura 4.1 (b) RBD do subsistema *Smartwatch*

$$A_{Sw} = A_{HW} \times A_{BR} \times A_{SO} \times A_{AP} \times A_{SR} \tag{4.5}$$

O modelo RBD do *Smartphone* (Figura 4.1 (c)), por sua vez, é representado por quatro blocos em série: *Hardware* (HW), Bateria (BR), Sistema Operacional (SO) e Aplicação (AP). A disponibilidade do *Hardware* (A_{HW}), Sistema Operacional (A_{OS}) e Aplicação (A_{AP}), é calculada utilizando os respectivos valores de reparo e falhas, aplicados à Equação 2.2. A disponibilidade da Bateria (A_{BR}) é calculada através do modelo CTMC, apresentado na Seção 4.2. A disponibilidade do *Smartphone* é dada pela Equação 4.6.

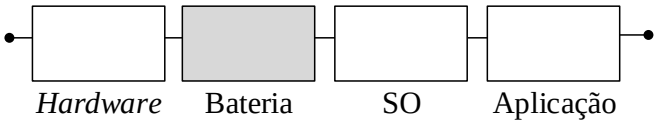


Figura 4.1 (c) RBD do subsistema *Smartphone*

$$A_{Sp} = A_{HW} \times A_{BR} \times A_{SO} \times A_{AP} \tag{4.6}$$

A disponibilidade da Nuvem (Figura 4.1 (d)) é representada por dois blocos em série correspondentes aos nós que fornecem o serviço para o usuário móvel: o Servidor Web (WB) e o Servidor de Banco de Dados (DB). Ambos os nós são representados por um mesmo modelo RBD (Figura 4.1 (g)) com cinco blocos em série: *Hardware* (HW), Sistema Operacional (SO), *Software* de Virtualização KVM (KVM), Sistema Operacional Virtualizado (VM) e a Aplicação (AP). A Equação 4.7 calcula a disponibilidade dos nós, enquanto Equação 4.8 calcula a disponibilidade da Nuvem.

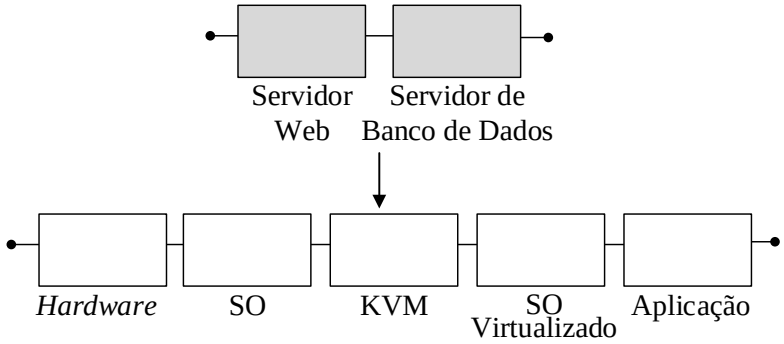


Figura 4.1 (d) RBD do subsistema Nuvem

$$A_{Nd} = A_{HW} \times A_{SO} \times A_{KVM} \times A_{VM} \times A_{AP} \tag{4.7}$$

$$A_{MC} = A_{WB} \times A_{DB} \tag{4.8}$$

4.2 Modelo de Autonomia da Bateria

Nesta seção, são apresentados os modelos CTMCs para o processo de descarga da bateria dos dispositivos móveis da Arquitetura Base. O modelo CTMC foi adotado devido a sua característica de processo probabilístico e composto por um conjunto de estados finitos, que possibilita a transição de estados apenas com base na situação atual do estado sem considerar

seu histórico, como de fato é o comportamento das baterias. Esta abordagem também pode ser encontrada em trabalhos anteriores como MATOS et al. (2015); SILVA et al. (2014); OLIVEIRA et al. (2013).

Para representar o processo de descarga da bateria do *smartwatch*, elaboramos um modelo CTMC (apresentado na Figura 4.1 (e)) seguindo uma distribuição *Erlang* (TRIVEDI, 2008) de 10 estágios, onde todos os estágios possuem a mesma taxa exponencial. Esta taxa é o inverso do tempo médio para a descarga de 10% da capacidade da bateria. O processo de descarga é baseado no percentual do nível da bateria, iniciando em 100% e finalizando em 0%, onde cada estado representa um nível da bateria. O estado 0% representa a bateria totalmente descarregada, ou seja, indisponível, enquanto que todos os outros são estados onde a bateria está disponível, o estado 100% representa a bateria totalmente carregada. Neste modelo, é considerada a conexão *Bluetooth* para comunicação entre o *smartwatch* e o *smartphone*. A descarga é modelada em passos de 10%, que ocorre sob a taxa λ_S ao usar conexão *Bluetooth*.

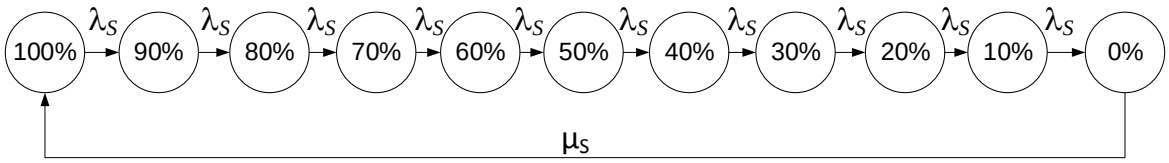


Figura 4.1(e) CTMC para descarga da bateria do *smartwatch*

Esse estudo pressupõe a existência de um *smartwatch* de reposição, que é utilizado para substituir o dispositivo totalmente descarregado. Este evento é representado pela taxa de substituição do estado 0% ao estado 100%, através do parâmetro μ_S.

Desta forma, considerando o modelo descrito na Figura 4.1 (e), a disponibilidade da bateria do *smartwatch* é calculada pela fórmula fechada apresentada na Equação 4.9, em que, λ_S representa a taxa de descarga e a taxa de substituição é denotada por μ_S. O n representa a quantidade de transições que o modelo faz até chegar à descarga completa, neste caso, $n = 10$.

$$A_{BR_{SW}} = \frac{n \times \mu_S}{(\lambda_S + n \times \mu_S)} \tag{4.9}$$

Similar ao modelo da bateria do *smartwatch*, a autonomia da bateria do *smartphone* e a sua substituição é representada pelo CTMC descrito na Figura 4.1 (f), onde λ_P representa a taxa de descarga da bateria, μ_P representa a taxa de substituição e n a taxa de transição para cada estado. Este modelo assume a existência de uma bateria reserva, que é usada para substituir a bateria em uso quando esta é completamente descarregada.

No cenário proposto para estimar a autonomia da bateria do *smartphone*, consideramos como parâmetros tanto o *Bluetooth* quanto a Banda Larga, para a conexão entre o *Smartphone* e os componentes das extremidades da arquitetura (*Smartwatch* e Nuvem). A fórmula fechada

para o cálculo de disponibilidade do modelo proposto é determinado pela Equação 4.10:

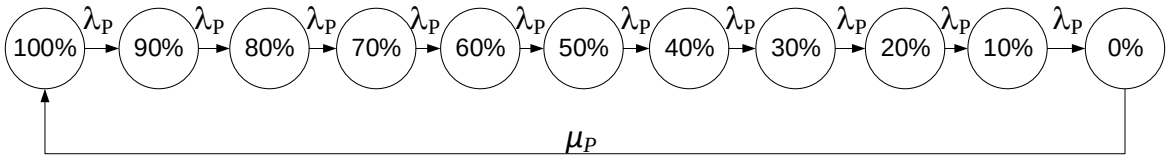


Figura 4.1(f) CTMC para descarga da bateria do *smartphone*

$$A_{BR_{SP}} = \frac{n \times \mu_P}{(\lambda_P + n \times \mu_P)} \tag{4.10}$$

A disponibilidade da bateria dos dispositivos é obtida de acordo com os parâmetros de tempo associados em cada modelo. Os parâmetros associados, definidos por λ_S e λ_P, têm seus valores coletados através de amostras de experimentos realizados no ambiente de testes, descritos na Seção 5.1.2. Dessa forma, através da avaliação dos modelos CTMCs foram obtidos os valores de disponibilidade da bateria dos dispositivos.

4.3 Considerações Finais

Este Capítulo apresentou técnicas de modelagem hierárquica heterogênea para representar uma plataforma *mHealth* na plataforma MCC no ambiente IoT, onde modelos foram combinados sendo eles: *Reliability Block Diagrams* (RBD) e o modelo de espaço de estados *Continuos Time Markov Chain* (CTMC). O modelo hierárquico RBD e CTMC foi elaborado para avaliar a disponibilidade da Arquitetura Base, e o CTMC para avaliar a descarga da bateria dos dispositivos móveis. Em nosso estudo, consideramos que o modo de falha da bateria é causado por sua descarga completa. Deste modo, para estimar a disponibilidade dos componentes Bateria foram utilizados os tempos de autonomia das baterias dos dispositivos.

5

Estimativas da Autonomia da Bateria e Injeção de Falhas e Reparos

Este capítulo apresenta o ambiente de experimentação, utilizado para estimar a autonomia da bateria dos dispositivos móveis, com o intuito de realizar o levantamento de dados da descarga da bateria, sendo estes coletados e utilizados como parâmetros para os modelos CTMCs. Posteriormente, é apresentada a validação dos modelos CTMCs para verificar se os mesmos possuem todas as características necessárias para representação adequada da autonomia da bateria dos dispositivos móveis. Por fim, detalhamos o experimento do testbed de injeção de falhas e reparos. Com a execução deste experimento é obtido métricas de disponibilidade do serviço, que serão utilizados no processo de validação do modelo da Arquitetura Base.

5.1 Autonomia da Bateria: Experimento e Validação do Modelo

Os parâmetros referentes aos tempos de falha e reparo da bateria dos dispositivos (*smartwatch* e *smartphone*), foram obtidos através de experimento em um ambiente controlado. Tal ambiente foi idealizado para representar características do cenário real de aplicação. Portanto, esta seção apresenta os mecanismos utilizados para a coleta dos dados da autonomia energética, tanto do *smartphone* quanto do *smartwatch*.

5.1.1 Ambiente do Experimento para Autonomia da Bateria

Por este experimento possuir como objetivo averiguar o tempo de autonomia energética dos dispositivos móveis, nós instanciamos uma máquina virtualizada na nuvem, provendo assim um serviço de envio e recebimento de dados para os dispositivos.

O ambiente elaborado é composto por um *smartwatch* (*LG Urbane w150*), um *smartphone* (*Samsung Galaxy S4 Mini Duos GT-i9192*) e por uma instância virtualizada, utilizando o conceito de MCC.

Quanto aos demais recursos, foram desenvolvidos dois aplicativos para otimizar e automatizar o processo de coleta de dados. Um aplicativo para o *smartphone* e outro para o *smartwatch*, sendo implementados nas plataformas *Android* e *Android Wear*, respectivamente. Os aplicativos possuem uma estrutura lógica similar, porém foram ajustados para atender as particularidades das respectivas plataformas. Em ambos os casos, os aplicativos fornecem informações energéticas dos dispositivos, contudo, o aplicativo da plataforma *Android Wear*, fornece ainda informações relacionadas à frequência cardíaca do usuário, através das interfaces de rede. A Figura 5.1, ilustra a topologia do experimento.

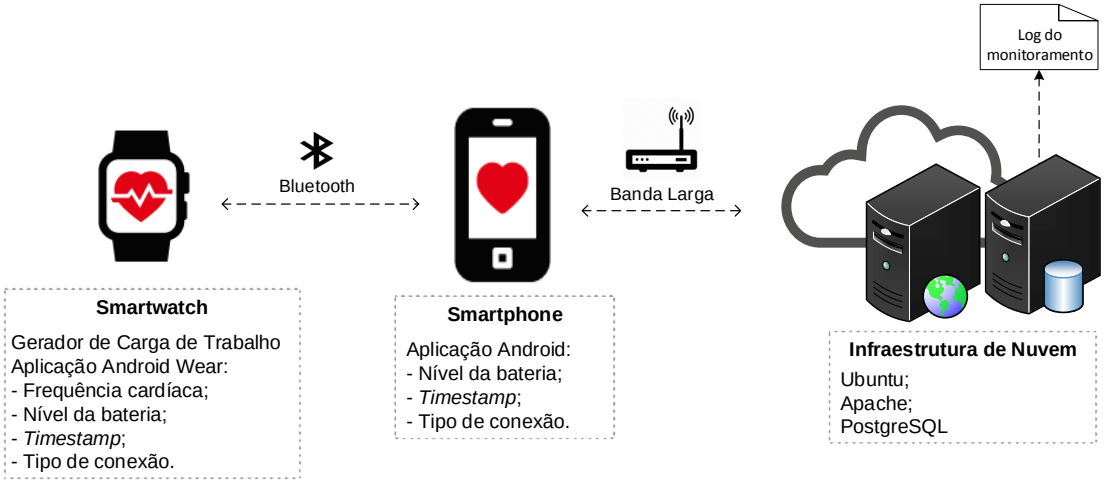


Figura 5.1 Topologia do ambiente para estimar a autonomia da bateria dos dispositivos móveis

Conforme ilustrado na Figura 5.1, o cenário do experimento abrange:

- ***Smartwatch***: é responsável por medir a frequência cardíaca do usuário através do sensor de batimento cardíaco. Para tanto, desenvolvemos uma aplicação (Android Wear) capaz de acessar as informações do sensor e enviá-las para a nuvem, através da conexão (*Bluetooth*) com o *smartphone*, gerando assim uma carga de trabalho que representa o comportamento real do cenário. Esta mesma aplicação é utilizada para monitorar o comportamento da bateria do dispositivo, registrando a cada mudança de nível (de 100 à 0, em porcentagem) da bateria, bem como o *timestamp* e a conexão utilizada, gerando um *log* do experimento para análise posterior.

- ***Smartphone***: exerce a função de ponte de comunicação entre os componentes *smartwatch* (através do *Bluetooth*) e nuvem (através da Banda Larga). O *smartphone* recebe as informações do *smartwatch* e as encaminha para a nuvem, recebendo de volta uma confirmação de recebimento, o *smartphone* encaminha para o *smartwatch* a resposta da nuvem. Enquanto este processo está sendo executado, sem interferência da aplicação, a aplicação registra cada mudança de nível (de 100 à 0, em porcentagem) da bateria, bem como o *timestamp* e a conexões utilizadas, gerando um *log* do experimento para análise posterior.

- ***Nuvem***: contém como principais componentes, um servidor de aplicação (*Apache*) e servidor de banco de dados (*PostgreSQL*). Desenvolvemos uma aplicação em linguagem Java que recebe os dados oriundos dos dispositivos e realiza a persistência no banco de dados (PostgreSQL), bem como disponibiliza uma interface para visualização dos dados. Sendo assim, todos os dados recebidos podem ser acessados posteriormente.

Para a realização do experimento, algumas aplicações foram desativadas em ambos os dispositivos móveis, ficando apenas aplicações necessárias para seu funcionamento, como os serviços do sistema operacional que não podem ser desativados. Por se tratar de uma representação de alta fidelidade do cenário de aplicação, o monitoramento da autonomia da bateria considerou uma configuração de uso onde o *smartwatch* deveria:

1. Consultar constantemente os sensores que provêm informações sobre a saúde do usuário;
2. Garantir que a interface de conexão com o *smartphone* fosse a *Bluetooth*, em detrimento às demais;
3. Verificar as informações relacionadas a bateria, tais como: carregador plugado, fonte de carregamento (*Alternating Current* (AC) ou via *Universal Serial Bus* (USB)), carregamento completo, estado atual da autonomia (em porcentagem);
4. Renderizar as informações no *display* do dispositivo para acompanhamento;
5. Enviar as informações coletadas (sensores, bateria e interface de conexão) para armazenamento na nuvem. O envio foi realizado sempre que algum estado do dispositivo sofria alteração.

É importante destacar duas configurações particulares do *smartwatch*: Tela sempre ligada e Modo de economia de bateria. O desligamento automático da tela do *smartwatch* tem como objetivo principal economizar a bateria, no entanto, o desligamento automático da tela desativa o sensor responsável por coletar os dados de batimento cardíaco. Desta forma, para que o sensor continuasse sempre ativo, sem interrupções, a opção de Tela sempre ligada permaneceu ativada. Já o Modo de economia da bateria, tem como objetivo prolongar a vida útil sempre que a carga está baixa. Ao ativar este recurso, o aparelho diminui o desempenho para economizar bateria, ocorrendo muitas vezes a perca de conexão com a rede. Desta forma, para o experimento esta configuração teve que permanecer desativada.

Quanto ao *smartphone*, as funcionalidades são similares, entretanto, a principal diferença é que as interfaces de redes ativas no dispositivo foram *Bluetooth* e Banda Larga, provendo, assim, a conexão com o *smartwatch* e a Nuvem, respectivamente. Outro aspecto que vale ressaltar é

que no *smartphone*, não ocorre a captura de dados dos sensores, visto que as informações de interesse são oriundas do *smartwatch*.

Apesar dos dispositivos móveis estarem contidos na mesma arquitetura, analisamos os dispositivos em momentos distintos, evitando que a falha energética de um pudesse interferir no funcionamento do outro. Deste modo, inicialmente, coletamos os tempos de descarga da bateria do *smartwatch*, ficando o *smartphone* conectado ao carregador. No segundo momento, invertemos o foco, ficando o *smartwatch* conectado a fonte de alimentação e o *smartphone* sendo observado. Neste último caso, o *smartwatch* continua enviando dados dos sensores, contudo, apenas o *smartphone* envia informações da autonomia da bateria.

Os aplicativos desenvolvidos possuem o mesmo objetivo. O Pseudocódigo 2 representa a estrutura lógica do aplicativo desenvolvido para o *smartwatch* e o Pseudocódigo 3 representa o núcleo lógico do aplicativo desenvolvido para o *smartphone*, ambos sob a perspectiva do monitoramento da autonomia energética.

Pseudocódigo 2 Coleta de informações da autonomia da bateria do *smartwatch*

```
Data: connection, heartRate;
 1: SETCONNECTION( connection );
 2:
 3: // Thread:  leitura e envio do BPM
 4: function WORKLOAD( ).start():
 5:     bpmCheck ← GETSENSORVALUE( heartRate );
 6:     while true do
 7:         bpm ← GETSENSORVALUE( heartRate );
 8:         if bpm != bpmCheck then
 9:             SENDTOCLOUD( bpm );
10:             bpmCheck ← bpm;
11:         end
12:     end
13: end
14:
15: // Monitor da bateria
16: levelCheck ← GETBATTERYLEVEL( );
17: while TRUE do
18:     level ← GETBATTERYLEVEL( );
19:     if level < levelCheck then
20:         SENDTOCLOUD( level, timestamp, connection );
21:         levelCheck ← level;
22:     end
23: end
```

Analisando o Pseudocódigo 2, observamos que, por se tratar da representação do *smartwatch*, na linha 1 é estabelecida que a conexão *Bluetooth* é única ativa, isso ocorre pois o intuito é monitorar o comportamento da autonomia da bateria no cenário real. Na linha 4 é criada e inicializada uma *thread* que será executada em paralelo ao processo principal do

aplicativo. Esta *thread* representa a carga de trabalho do cenário avaliado e envia à nuvem o valor da frequência cardíaca (linha 9), toda vez que for observado uma mudança deste valor. Enquanto o aplicativo estiver em funcionamento, esta *thread* estará sendo executada.

O funcionamento do monitor do nível da bateria é representado nas linhas 16 à 13, do Pseudocódigo 2. A linha 17 indica que, enquanto o aplicativo estiver em funcionamento, as instruções das linhas 18 e 19 serão executadas de forma repetida. A linha 18 representa que o valor do nível da bateria é coletado, sendo verificado na linha 19 se este valor atual é inferior ao último valor enviado à nuvem, e em caso positivo, o valor do nível da bateria é enviado à nuvem (linha 20) e o último valor enviado é atualizado (linha 21).

Pseudocódigo 3 Coleta de informações da autonomia da bateria do *smartphone*

```
Data: connectionSmartwatch, connectionCloud;
 1: SETCONNECTION( connectionSmartwatch );
 2: SETCONNECTION( connectionCloud );
 3:
 4: // Monitor da bateria
 5: levelCheck ← GETBATTERYLEVEL( );
 6: while TRUE do
 7:     level ← GETBATTERYLEVEL( );
 8:     if level < levelCheck then
 9:         SENDTOCLOUD( level, timestamp, connectionSmartwatch, connectionCloud );
10:         levelCheck ← level;
11:     end
12: end
```

Analisando o Pseudocódigo 3, observamos que o funcionamento do processo de coleta de informações da bateria (linhas 5 à 12) é o mesmo apresentado no Pseudocódigo 2. Isto é, ao ser identificado uma decremento no nível da bateria as informações de data, hora, conexões ativas e o próprio nível da bateria são enviadas à nuvem. Contudo, as linhas 1 e 2 representam que as conexões *Bluetooth* e banda larga permanecem ativas durante o experimento, visto que para representar o cenário real é necessário que o *smartphone* esteja conectado com o *smartwatch*, através da conexão *Bluetooth* e esteja conectado com a nuvem, através da conexão banda larga, conforme apresentado anteriormente na Figura 5.1.

Destacando que, no caso do *smartphone*, o gerenciamento da carga de trabalho é realizado pelo sistema operacional do próprio dispositivo, isto é, o *smartphone* recebe a carga de trabalho do *smartwatch* e apenas encaminha para seu destino (a nuvem) e o aplicativo contido no *smartphone* tem a função apenas de registrar as informações necessárias para o estudo.

Vale salientar que os aplicativos desenvolvidos são parametrizáveis, o que os tornam genéricos. Sendo assim, é possível utilizá-los em diversas fases da pesquisa, bem como em outros cenários, onde sejam utilizados os mesmos dispositivos. Um exemplo de reuso desta aplicação está no Capítulo 6 (Seção 6.3) onde estendemos a Arquitetura Base a fim de investigar alternativas.

5.1.2 Validação do Modelo de Autonomia da Bateria.

Após a recarga completa da bateria, iniciou-se o processo de descarga, observando os estados de decremento, iniciando em 100% e finalizando em 0%. Quanto aos dados coletados, após o processo de descarga ser repetido *n* vezes, para cada dispositivo, obtivemos um *log* de cada procedimento, onde cada *log* é composto por cada estado da bateria (de 100% à 0%), além da data, hora e interface de conexão ativa. Considerando estas amostras, obtivemos as médias do tempo de descarga de ambos os dispositivos.

Os valores obtidos através do experimento são associados às transições de estados dos modelos CTMCs representados na Figura 4.1 (e) e Figura 4.1 (f). O experimento foi realizado no ambiente de teste utilizando duas possíveis redes de comunicação o *Bluetooth* e Banda Larga.

O experimento foi dividido em duas etapas. Na primeira etapa, foi realizado o experimento do dispositivo *smartwatch* e na segunda etapa, foi realizado o experimento do dispositivo *smartphone*. Cada experimento foi repetido 10 vezes para calcular a média do tempo de descarga, devido a variação do tempo de descarga ter apresentado uma variação mínima, entre cada intervalo do experimento. Observamos que em 10 repetições a média de tempo de descarga apresentou pouca variação comparado ao número menor de amostras.

A Tabela 5.1 apresenta a média dos valores resultantes do experimento, para o *smartwatch* e o *smartphone*, respectivamente. O tempo de descarga da energia da bateria não é linear, isto significa que os tempos para atingir cada um dos estados são diferentes.

Tabela 5.1 Média amostral do experimento para autonomia da bateria dos dispositivos móveis.

Amostras do Smartwatch		**Amostras do Smartphone**	
Descarga	**Conexão Bluetooth**	**Descarga**	**Conexão Bluetooth e Banda Larga**
100% - 90%	01:04:32	**100% - 90%**	07:48:02
90% - 80%	01:04:31	**90% - 80%**	06:16:46
80% - 70%	01:03:32	**80% - 70%**	05:46:16
70% - 60%	01:09:52	**70% - 60%**	04:10:32
60% - 50%	00:55:27	**60% - 50%**	03:23:51
50% - 40%	01:02:33	**50% - 40%**	03:41:19
40% - 30%	01:01:52	**40% - 30%**	03:33:06
30% - 20%	01:01:46	**30% - 20%**	03:17:51
20% - 10%	01:00:36	**20% - 10%**	02:33:38
10% - 0%	00:56:17	**10% - 0%**	02:14:23
Total	10:20:59	**Total**	42:45:44

Além da descarga, também foi observado o tempo para a recarga de ambos os dispositivos. O tempo de recarga da bateria representa o tempo entre o estado de descarga total e o estado de carga total. Desta forma, foi observado uma média de tempo de recarga total de $1h32min$ para o dispositivo *smartwatch* e $1h53min$ para o dispositivo *smartphone*.

Como supracitado, nosso estudo assume a existência de dois componentes extras, um *smartwatch* e uma bateria extra para o *smartphone*, que são usados para substituição, quando

os mesmos estiverem descarregados. Desta forma, foi observado durante o experimento, o tempo médio de 5 (cinco) minutos para a substituição dos dispositivos, incluindo o tempo de inicialização do sistema operacional, sensores e aplicação.

A Tabela 5.2 apresenta os parâmetros de entrada para o modelo da bateria do *smartwatch*, utilizando a conexão *Bluetooth*. Já a Tabela 5.3 apresenta os parâmetros de entrada para o modelo da bateria do *smartphone*, utilizando a conexão *Bluetooth* e Banda Larga. Considerando o tempo de recarga, o tempo médio de 5 (cinco) minutos para a substituição da bateria dos dispositivos.

Estes parâmetros são as médias dos valores resultante do experimento realizado sobre cada um dos dispositivos, apresentados na Tabela 5.1. Estes valores são mensurados com base no tempo que foi dispendido para atingir o estado de descarga total da bateria e foram particionados em conformidade com o número de transições existentes no modelo, que compreende o estado de início da descarga energética até o estado final. Estes valores são convertidos em taxas para serem atribuídos aos parâmetros de cada modelo.

Tabela 5.2 Parâmetros de entrada Figura 4.1 (e).

Parâmetros	**Tempo(h)**	**Taxa(1/h)**
Descarga (λ_S)	1.03500	0.96618
Recarga (μ_S)	0.0833	12

Tabela 5.3 Parâmetros de entrada Figura 4.1 (f).

Parâmetros	**Tempo(h)**	**Taxa(1/h)**
Descarga (λ_P)	4.27611	0.23385
Recarga (μ_P)	0.0833	12

Em posse dos valores apresentados nas tabelas Tabela 5.2 e Tabela 5.3, o modelo CTMC apresentado na Figura 4.1 (e) pôde ser avaliado através da fórmula fechada, na Equação 4.9, resultando em uma disponibilidade de 99.201277%. Já o modelo CTMC apresentado na Figura 4.1 (f) teve sua avaliação através da Equação 4.10, resultando em uma disponibilidade de 99.805498%.

As validações dos modelos CTMCs são efetuadas com base nos tempos de descarga obtidos dos experimentos. A partir dos dados extraídos do experimento não é possível determinar o tipo de distribuição devido ao tamanho das amostras. Em vista disso, é adotado o método de inferência não paramétrica para obter aproximações de distribuições amostrais. Desta forma, com base nas amostras dos experimentos realizados para cada dispositivo móvel é aplicado o método *Bootstrap* (EFRON; TIBSHIRANI, 1994) para encontrar o intervalo de confiança de cada parâmetro, considerando um índice de 95% com erro máximo de 5%. Para obter um intervalo de confiança, consultamos os $0,025$ e $0,975$ quantis dos dados obtidos das 1000 reamostras geradas pelo *Bootstrap*, pois segundo CHERNICK (2011), o número de reamostragens necessárias para se obter boas estimativas seria, de pelo menos, 1000 repetições *Bootstrap*.

A partir das estimativas da amostra original, foi gerada uma reamostragem, em seguida essas estimativas são ordenadas, assim foi possível definir os limites inferior e superior do intervalo de confiança *Bootstrap*. A reamostragem do *Bootstrap* ocorreu através da elaboração de um *script* na ferramenta *Mathematica* (Apêndice A).

Para o experimento do *smartwatch*, foi encontrado o limite inferior de $0.932463h$ e limite superior de $1.00676h$. Para o experimento do *smartphone*, foi encontrado o limite inferior

de 0.205713h e limite superior de 0.32677h. Estes novos valores são atribuídos aos modelos para avaliação da disponibilidade da bateria. Os intervalos e resultados estão representados na Tabela 5.4 e na Tabela 5.5.

Tabela 5.4 Validação do modelo CTMC do *smartwatch* (Figura 4.1 (e))

Método	Disponibilidade
Experimento	95% IC (99,1680134%; 99,2289390%)
Modelo	99,201277%

Tabela 5.5 Validação do modelo CTMC do *smartphone* (Figura 4.1 (f))

Método	Disponibilidade
Experimento	95% IC (99,72843%;99,82886%)
Modelo	99,8054987%

A Tabela 5.4 apresenta o intervalo de confiança do modelo da bateria do *smartwatch*. O intervalo de confiança obtido através do método *bootstrap*, compreende valores de disponibilidade entre $99,1680134\%$ e $99,2289390\%$, e a disponibilidade obtida através da avaliação do modelo da bateria (Figura 4.1 (e)) é de $99,201277\%$.

A Tabela 5.5 apresenta o intervalo de confiança do modelo da bateria do *smartphone*. O intervalo de confiança obtido através do método *bootstrap*, compreende valores de disponibilidade entre $99,72843\%$ e $99,82886\%$, e a disponibilidade obtida através da avaliação do modelo da bateria (Figura 4.1 (f)) é de $99,8054987\%$.

De acordo com os resultados, apresentados nas Tabelas 5.4 e 5.5, os valores obtidos pelos modelos permanecem dentro dos intervalos de confiança. Desta forma, é possível observar que o comportamento do modelo é adequado para representar o componente em questão, isto significa que o modelo é condizente com o funcionamento do componente no ambiente real.

5.2 Injeção de Falhas: Experimento e Validação do Modelo

Com o intuito de verificar se o comportamento do modelo da Arquitetura Base (Figura 4.1) é semelhante ao ambiente real, um experimento de injeção de falhas e reparos foi realizado em um ambiente de testes controlado, para validação do modelo. Para tanto, foi aplicada a técnica de injeção de falhas baseada em *software* aplicando a técnica de tempo de execução com gatilho baseado em *time-out*. Nesta seção, inicialmente é apresentado o ambiente de teste para injeção de falhas, posteriormente, apresentamos o processo de validação do modelos da Arquitetura Base; esse processo é dividido em duas etapas: injeção de falhas (análise dos dados coletados) e validação.

5.2.1 Ambiente de Teste para Injeção de Falhas

Para o experimento, o processo de injeção de falhas e reparos ocorreu no componente Nuvem, visto que este possui os principais recursos para o provimento do serviço como um todo. Desta forma, foi possível analisar se o serviço proposto continua operando mesmo na presença de falhas, e principalmente, verificar se o modelo representa o funcionamento do sistema real.

O ambiente de teste é composto por um *smartwatch* (*LG Urbane w150*), um *smartphone* (*Samsung Galaxy S4 Mini Duos GT-i9192*) e por dois computadores, sendo um responsável por injetar falhas e reparos no serviço, além de visualizar o monitoramento dos status do serviço (*UP*/*DOWN*). O computador injetor possui um Processador *Intel Core i7*, 6 GB de RAM, HD 1T SATA e Sistema Operacional *Ubuntu 14.04*. Já o outro computador é representado por uma instância virtualizada na nuvem, fornecendo assim o serviço de Nuvem Privada.

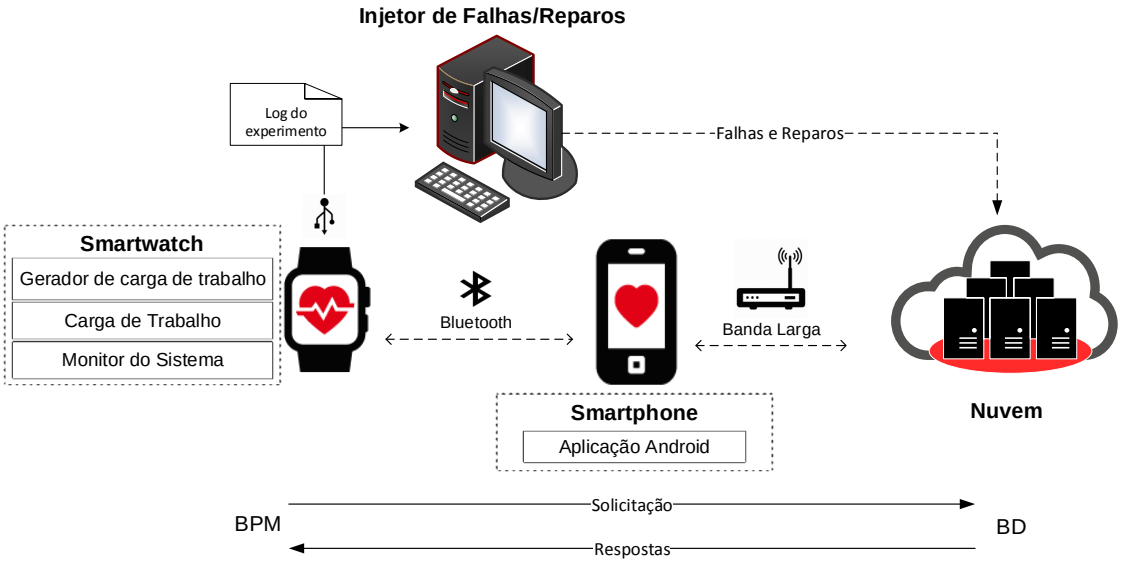


Figura 5.2 Componentes do ambiente teste

Conforme a Figura 5.2, o ambiente de teste abrange:

- ***Gerador de Carga de Trabalho***: envia informações da frequência cardíaca para a Nuvem. As informações são coletadas pelo sensor do *smartwatch* e enviadas com o auxílio do *smartphone*, através das conexões *Bluetooth* e Banda Larga. As informações são enviadas a cada mudança de leitura de batimento, durante todo o experimento.

- ***Monitor do Sistema***: exerce a sua função em conjunto com o Gerador de Carga de Trabalho, ou seja, também é representado pelo *smartwatch*. O computador monitor observa o status do sistema, constatando se o serviço está disponível, enquanto o experimento está sendo executado. Para cada envio de informação realizada, foi registrado o status do retorno (se houve sucesso ou falha na requisição). Deste

modo, o *log* dos eventos foi armazenado no Monitor do Sistema, sendo possível ser consultado pelo Computador Injetor, através da conexão ADB (*Android Device Brigde*), utilizando uma comunicação serial.

- ***Smartphone***: exerce a função de ponte de comunicação entre os componentes das extremidades da arquitetura. O *smartphone* recebe as informações do *smartwatch* (através de uma conexão *Bluetooth*) e as encaminha para a Nuvem (através da Banda Larga), após a confirmação de recebimento, o *smartphone* encaminha para o *smartwatch* a resposta da Nuvem.

- ***Nuvem***: composta por uma instância virtual, contendo como principais componentes um servidor de aplicação (*Apache*) e um servidor de banco de dados (*PostgreSQL*). Nesta instância, uma aplicação recebe os dados oriundos do *smartwatch* e realiza a persistência no banco de dados.

- ***Computador Injetor***: executa, de forma controlada, os algoritmos de injeção de falhas e reparos, escritos com a linguagem *Shell Script*. O *script* executa comandos de falhas e reparo sob o serviço. O processo de injeção de falhas subsidiou o monitoramento do comportamento do sistema. O comportamento do injetor é representado pelo Pseudocódigo 4 e o *script* completo de injeção de falhas e reparos é apresentado no Apêndice B.

Pseudocódigo 4 Algoritmo do *script* para injeção de falhas e reparos

```
Data: ipAddress, port;
 1: while true do
 2:     status ← VERIFYSERVICE( ipAddress, port );
 3:
 4:     if status == up then
 5:         SLEEP ( random_Exponential_Time_MTTF );
 6:         STOP_SERVICE ( ipAddress, port );
 7:     end
 8:
 9:     if status == down then
10:         SLEEP ( random_Exponential_Time_MTTR );
11:         START_SERVICE ( ipAddress, port );
12:     end
13: end
```

Em relação ao processo de injeção de falhas e reparos representado pelo Pseudocódigo 4, é possível observar (na linha 1) que, enquanto não for suspenso, o processo se repete. De modo resumido, o processo é composto por duas ações básicas, injetar falha (linha 6) e injetar reparo (linha 10) no serviço em questão. Para tanto, é observado previamente o estado do serviço (se *UP* ou *DOWN*), representado pela linha 2. De acordo com o estado do serviço, é aguardado um

tempo exponencialmente distribuído, baseado no MTTF (linha 5) ou no MTTR (linha 9), antes de injetar de fato uma falha ou reparo, respectivamente.

A Listagem 5.1, apresenta o trecho essencial do *script* injetor de falhas e reparos. A Listagem 4 e a Listagem 5.1 estão organizados de forma que a identificação da linha de uma possui representação respectiva na outra, ou seja, a linha 1 da Listagem 4 está relacionada com a linha 1 da Listagem 5.1 e assim por diante.

Código 5.1 Script para injeção de falha e reparo

```
while [ true ] ; do
        status=$(nmap -p 8080 192.168.10.10 | grep 8080/tcp | awk '{
           print $2}')

    if [ "$status" = "open" ] ; then
      sleep $(Rscript expFail.R | awk '{print $2}')
      sudo service tomcat7 stop
    fi

    if [ "$status" = "closed" ] ; then
      sleep $(Rscript expRepair.R | awk '{print $2}')
      sudo service tomcat7 start
    fi
done
```

Código 5.2 Saída da ferramenta Nmap

```
Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-10 18:02 BRT
Nmap scan report for 192.168.10.10
Host is up (0.000094s latency).
PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
```

Analisando a Listagem 5.1, observamos que a linha 1 indica que enquanto o *script* injetor estiver em funcionamento, o ciclo de injeção de falhas e reparos no serviço se repete até ser interrompido ou finalizado. Em cada ciclo é realizada uma verificação do *status* do sistema através da ferramenta *nmap*[1], a Listagem 5.2 apresenta a saída da execução da ferramenta *nmap*, onde a linha 5 possui três parâmetros, sendo o segundo o *status* do serviço observado. Logo, a linha 2 da Listagem 5.1 atribui à variável *status* o estado atual do serviço executado na porta 8080 do *host* 192.168.10.10, este estado pode ser *"open"* ou *"closed"*.

[1] *nmap*: utilitário que dispõe de recursos para realizar escaneamento de portas, descoberta de serviços em rede, execução e monitoramento de *scripts* em *hosts* remotos (LYON; FYODOR, 2009)

A linha 4 da Listagem 5.1 verifica se o conteúdo da variável *status* é *"open"*, caso seja, entende-se que o serviço observado está funcionando e, portanto, será injetada uma falha. Para tanto, para manter a essência do experimento, o algoritmo gera um tempo exponencial distribuído para a injeção de falha com base no MTTF do serviço, com ajuda da *Ferramenta R* (PROJECT, 2015). Após a obtenção do tempo gerado é executado um comando de espera (*sleep*) de acordo com o este tempo, fazendo com que o injetor aguarde um tempo antes de injetar de fato uma falha no serviço. Na linha 6 é executado o comando para parar o serviço, a estrutura do comando é *"sudo service <nome_do_serviço> <ação>"*. Com a falha injetada o ciclo reinicia e é verificado novamente se o sistema está disponível. Considerando que o serviço não está disponível, é obtido um tempo exponencialmente distribuído através da *Ferramenta R*, com base no MTTR. Após aguardar o tempo ser alcançado é realizado um reparo automático, fazendo com que o serviço volte a funcionar. A condição de parada é estipulada pelo avaliador, através de um comando de escape.

Por fim, os dados levantados através do experimento são tabulados e analisado com o intuito de verificar se representam de fato o cenário estudado, conforme é apresentado na Seção 5.2.2. A Figura 5.3 exemplifica através de um diagrama o processo básico para injeção de falhas e reparos.

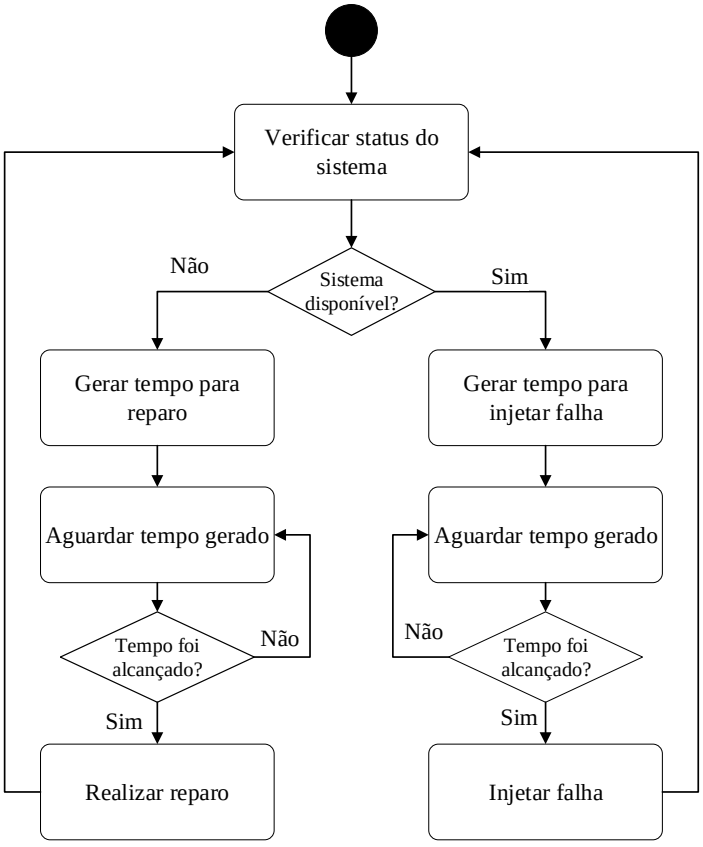


Figura 5.3 Diagrama de atividade da injeção de falhas e reparos

5.2.2 Validação do Modelo da Arquitetura Base

A etapa de validação é responsável por verificar se o modelo representa o mesmo comportamento que o sistema real. Para que o modelo seja considerado válido, é necessário que este esteja dentro do intervalo de confiança da disponibilidade. Desta forma, realizamos a validação do modelo através dos cálculos estatísticos propostos por KEESEE (1965), que consiste em calcular o intervalo de confiança da disponibilidade.

Esta abordagem também pode ser encontrada em trabalhos anteriores para análise de disponibilidade em serviços como: plataforma MBaaS (COSTA et al., 2015), *Vod Streaming* na nuvem (BEZERRA et al., 2014), serviços *Video Surveillance as a Service* (VSaaS) (MAGNO, 2015), nuvens privadas (MELO et al., 2013), entre outros. A Figura 5.4 apresenta o processo de validação, representado por um fluxo de trabalho proposto por COSTA et al. (2015), onde se destacam duas etapas: Injeção de Falhas e Validação.

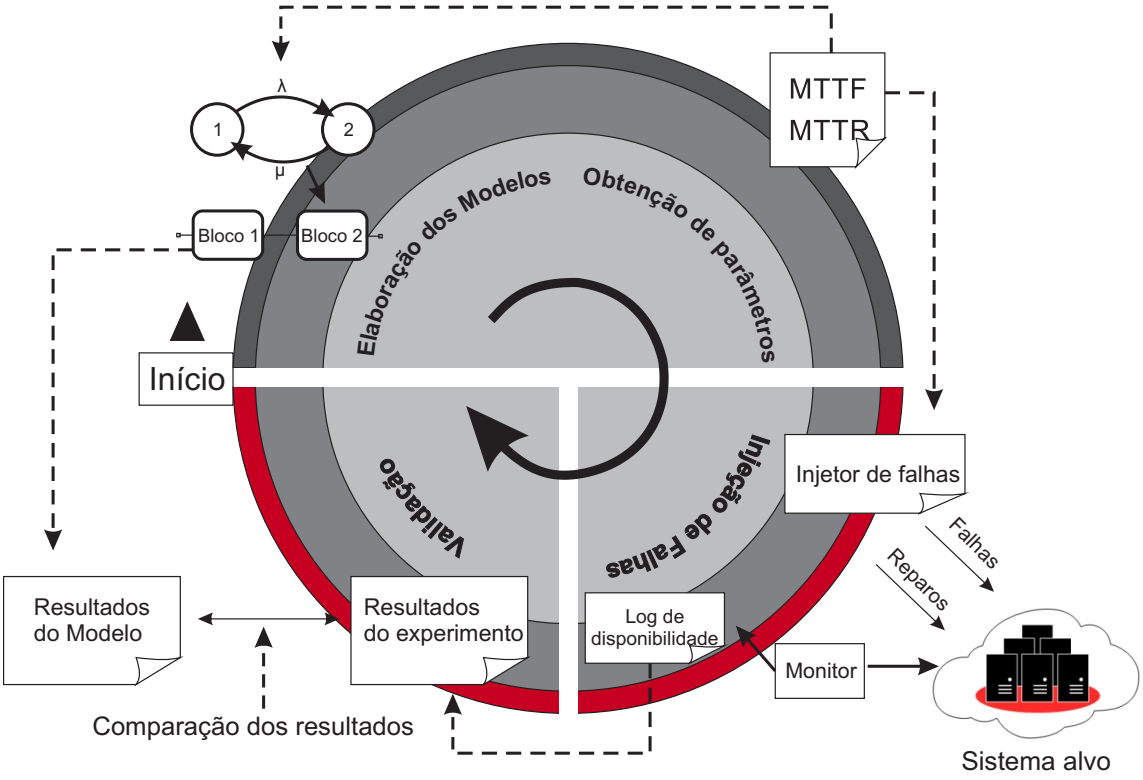


Figura 5.4 Fluxo de Trabalho para Validação do Modelo da Arquitetura Base

Injeção de Falhas: Análise dos Dados Coletados

O injetor de falhas utilizado para os testes foi desenvolvido com a linguagem *Shell Script*, seu pseudocódigo foi apresentado na Seção 5.2.1 no Pseudocódigo 4. Para iniciar a utilização de injeção de falhas para validação do modelo, é necessário conhecer os tempos de falha e reparo da Arquitetura Base. Em virtude do tempo necessário para a ocorrência de falhas serem muito longos, aplicamos um fator de redução (SOUZA et al., 2013). A aplicação do fator de

redução permitiu acelerar o experimento, dado que se os valores reais obtidos fossem adotados o experimento levaria meses até a ocorrência de uma falha e horas para que as falhas fossem corrigidas, demandando assim muito tempo para a execução do experimento. Para o estudo em questão, foi adotado um fator de redução de 600 para os eventos de falha e de 100 para eventos reparo da Arquitetura Base. Deste modo, a *Ferramenta R* (PROJECT, 2015) foi adotada para gerar os tempos de injeção de falha e reparo de acordo com uma distribuição exponencial, com base nos MTTF e MTTR da disponibilidade do serviço da Arquitetura Base. A Tabela 5.6 apresenta os valores utilizados para iniciar a injeção de falhas. A obtenção desses tempos será detalhada no Capítulo 6, na Seção 6.1.

Tabela 5.6 Taxas utilizadas na injeção de falhas.

Componente	Tempo Real(h)	Após Fator de Redução
MTTF	3.341850	0.005570
MTTR	0.092420	0.0009242

Após o término do experimento, é gerado um *log*, contendo os dados obtidos do injetor e do monitor. O experimento foi iniciado às 00:16:38 do dia 15 de julho de 2016 e foi finalizado às 10:48:13 do dia 20 de julho de 2016. Logo, o ambiente foi supervisionado durante um período de 130.5264 horas (5.4 dias). Durante o período de monitoramento, o serviço acumulou 12999.6749 horas de tempo total de falha e 317.7906 horas de tempo total de reparo.

Desta forma, os resultados gerados são usados para o cálculo da disponibilidade do experimento e do intervalo de confiança da disponibilidade.

Validação: Método Keesee

Para iniciar o cálculo do intervalo de confiança da disponibilidade, proposto por KEESEE (1965), é necessário ter os dados extraídos do experimento real, que irão servir como parâmetro de entrada para as equações, que são: tempo total de falhas, tempo total de reparos e quantidade de eventos aplicados durante o experimento de injeção de falhas.

A partir do *log* gerado pelo experimento, encontramos o tempo total de falhas e o tempo total de reparos. Deste modo, podemos encontrar o valor de ρ, conforme a Equação 5.1, no qual, o tempo total de falhas S_n que foi igual a $12999.6749h$, e o tempo total de reparos Y_n que foi igual a $317.7906h$.

$$\rho = \frac{Y_n}{S_n} = \frac{317,7906}{12999,6749} = 0,024446 \tag{5.1}$$

A equação da disponibilidade estacionária do experimento, utilizando o método KEESEE (1965), é dada pela Equação 5.2.

$$A = \frac{1}{1+\rho} = \frac{1}{1+0,024446} = 0,976137 \tag{5.2}$$

Em seguida, analisamos a quantidade total de eventos injetados no ambiente, que corresponde a 446 eventos de falhas e reparos, este valor é utilizado como o grau de liberdade para o cálculo da Distribuição-F (*F-distribution*). Foram calculados os valores ρ_U (superior) e ρ_L (inferior) da Distribuição-F com 95% de confiança, através da ferramenta *Minitab* [2]. O alcance desses valores também pode ser representado pela Equação 5.3 e Equação 5.4. Onde α representa o nível de confiança.

$$\rho_U = \frac{\rho}{f446;446;1-\frac{\alpha}{2}} = \frac{0,024446}{0,8304} = 0,029438 \tag{5.3}$$

$$\rho_L = \frac{\rho}{f446;446;\frac{\alpha}{2}} = \frac{0,024446}{1,204} = 0,020304 \tag{5.4}$$

Assim, torna-se possível calcular o intervalo de confiança para a disponibilidade do sistema real, A_L e A_U. Para isso utilizamos a Equação 5.5.

$$\left(\frac{1}{1+\rho_U}, \frac{1}{1+\rho_L}\right) = \left(\frac{1}{1+0,029438}, \frac{1}{1+0,020304}\right) = (0,97140299 : 0,98010003) \tag{5.5}$$

A partir do intervalo de confiança definido, a validação do modelo proposto da Arquitetura Base foi realizada. O modelo só será dado como válido se a análise dos resultados obtidos estiver dentro do intervalo de confiança (de 95%) da disponibilidade. Utilizando a Equação 4.2 do modelo RBD da Figura 4.1 (a), é possível encontrar a disponibilidade de 97,30887%, a qual encontra-se dentro do intervalo de confiança que vai de A_L e A_U, definido na Tabela 5.7. Portanto, o modelo proposto não tem evidências para ser refutado como modelo que representa o comportamento do ambiente real. Desta forma, o modelo demonstra ser confiável para ser utilizado na avaliação de disponibilidade do ambiente, bem como a proposição de novas arquiteturas.

Tabela 5.7 Método Keesee - Intervalo de confiança para A e ρ

Intervalo de Confiança 95%		
ρ	ρ_U	0,029438
	ρ_L	0,020304
A	A_U	98,0100%
	A_L	97,1403%
Disponibilidade do Experimento	97,6137%	
Disponibilidade do Modelo	97,3088%	

[2]Minitab. Disponível em: http://www.minitab.com/pt-br/.

5.3 Considerações Finais

Este capítulo apresentou o ambiente desenvolvido para estimar a autonomia da bateria dos dispositivos *smartwatch* e *smartphone*. Para automatizar o experimento, desenvolvemos uma aplicação na plataforma *Android Wear* (para o *smartwatch*) e uma aplicação na plataforma *Android* (para o *smartphone*). Com a execução do experimento, obtivemos os parâmetros de tempos de falha e reparo, tanto do *smartwatch* quanto do *smartphone*. Vale salientar que os aplicativos desenvolvidos são parametrizáveis, o que os tornam genéricos. Sendo assim, é possível utilizá-los em diversas fases da pesquisa, bem como em outros cenários, onde sejam utilizados os mesmos dispositivos. Por fim, construímos um *testbed* que teve como objetivo principal expor o serviço analisado a uma quantidade razoável de falhas e reparos, e assim, coletar informações que foram utilizadas como entrada para o cálculo do intervalo de confiança, utilizados no processo de validação do modelo da Arquitetura Base. Isto posto, os nossos ambientes desenvolvidos para os experimentos, podem ser utilizados para trabalhos futuros, que utilizem dos mesmos recursos computacionais.

6

Estudo de Caso

Este capítulo tem como objetivo avaliar a disponibilidade do serviço do cenário base, empregando modelos RBD e CTMC, para tal apresentamos quatro estudos de caso. O Estudo de Caso I (Seção 6.1) apresenta uma análise de disponibilidade do modelo da Arquitetura Base. O Estudo de Caso II (Seção 6.1) apresenta uma análise de sensibilidade. Com base nos resultados da análise de sensibilidade, apresentamos no Estudo de Caso III (Seção 6.1) a avaliação do impacto de mecanismos de redundância na Arquitetura Base. Por fim, no Estudo de Caso IV (Seção 6.1) foram analisados os dispositivos móveis, sob três aspectos: criticidade dos componentes em relação a disponibilidade, confiabilidade dos dispositivos e a autonomia da bateria em relação a disponibilidade do serviço utilizando diferentes interfaces de conexão.

6.1 Estudo de Caso I: Avaliação de Disponibilidade do Modelo da Arquitetura Base

Este primeiro estudo de caso tem como objetivo principal avaliar a disponibilidade estacionária da Arquitetura Base proposta, desta forma, a análise de disponibilidade realizada pode servir como referência para projetos futuros, com foco no aumento da disponibilidade do sistema.

Inicialmente, para realizar a avaliação são apresentados os parâmetros de entrada que foram inseridos nos modelos da Arquitetura Base do serviço *mHealth* ilustrada na Figura 4.1, apresentada na Seção 4.1.

A Tabela 6.1 apresenta os valores da taxa de descarga da bateria dos dispositivos móveis. O tempo de transição observado é convertido em taxa para ser utilizado como parâmetro do modelo, definidas como λ_S e λ_P, representadas na Figura 4.1 (e) e Figura 4.1 (f). Estes valores foram obtidos por meio de experimento controlado, descrito na Seção 5.1. As taxas de substituição, definidos como μ_S e μ_P, são calculadas com base no tempo médio necessário para substituir a bateria, a partir da ocorrência de falha.

A Tabela 6.2 apresenta os parâmetros de entrada para o modelo do *smartwatch*, representado na Figura 4.1 (b). Os parâmetros do componente *Hardware* foram adaptados a

Tabela 6.1 Parâmetros de entrada do CTMC da bateria do *smartwatch* e *smartphone*.

Parâmetro	λs	μs	λp	μp
Valor (h^{-1})	0.966184	12	0.233857	12

partir de OLIVEIRA et al. (2013). Os parâmetros dos componentes Sistema Operacional e Aplicação foram adaptados a partir de KIM; MACHIDA; TRIVEDI (2009). Os parâmetros do componente Sensor foram obtidos a partir do estudo de confiabilidade de *Wireless Body Area Networks* (WBAN) realizado por PEIRAVI (2010).

Tabela 6.2 Parâmetros de entrada para o componente *Smartwatch*

Componentes	**MTTF (h)**	**MTTR (h)**
Hardware	22461.5	1.667
Bateria	10.35000	0.083333
SO	1440.9	0.033
Aplicação	336.7	0.0167
Sensor	40341.67	0.1667

A Tabela 6.3 apresenta os parâmetros dos componentes do *smartphone*, representado na Figura 4.1 (c). Os parâmetros referentes à *Hardware*, Sistema Operacional e Aplicação são baseados em valores apresentados em OLIVEIRA et al. (2013) e MATOS et al. (2015). Já o parâmetro da Bateria foi coletado através de experimento próprio, conforme apresentado na Seção 5.1.2.

Tabela 6.3 Parâmetros de entrada para o componente *Smartphone*

Componentes	**MTTF (h)**	**MTTR (h)**
Hardware	22461.5	1.667
Bateria	42.76111	0.083333
SO	1440.9	0.033
Aplicação	336.7	0.0167

Os parâmetros de entrada para as interfaces de conexão (Figura 4.1 (a)) *Bluetooth* e Banda Larga foram provenientes de CARROZZA et al. (2008) e D-LINK (2012), respectivamente. A Tabela 6.4 apresenta estes parâmetros.

Tabela 6.4 Parâmetros de entrada para os componentes *Bluetooth* e Banda Larga

Componentes	**MTTF (h)**	**MTTR (h)**
Bluetooth	4881.605	0.00953
Banda Larga	5.99640	0.07896

Quanto ao MTTF e MTTR dos componentes da Nuvem, do modelo da Arquitetura Base (Figura 4.1 (g)), foram extraídos a partir DANTAS et al. (2012) e são apresentados na Tabela 6.5.

Tabela 6.5 Parâmetros de entrada para o componente Nuvem

Componentes	MTTF (h)	MTTR (h)
Hardware	8760	1.667
SO	2893	0.25
KVM	2990	1
VM	2893	0.25
Aplicação	788.4	1

Para a análise dos modelos hierárquicos baseados em RBD utilizou-se a ferramenta *Mercury* (MODCS, 2015b) e para a avaliação dos CTMCs, além da ferramenta *Mercury*, utilizou-se a ferramenta *Sharpe* (SAHNER; TRIVEDI; PULIAFITO, 2012).

Após inserir os parâmetros de entrada, foi possível realizar a análise de dependabilidade da Arquitetura Base. Os resultados da análise do Estudo de Caso I, em termos de disponibilidade e *downtime* anual, são apresentados na Tabela 6.6.

Tabela 6.6 Análise de dependabilidade da Arquitetura Base

Disponibilidade	Disponibilidade (9's)	*Downtime* (h/a)
0.9730887	1.570065	235.743h

Neste estudo de caso, propomos avaliar a disponibilidade da Arquitetura Base. Para isto, modelos hierárquicos RBD e CTMC foram usados para analisar a arquitetura. O modelo que representa a funcionalidade do serviço avaliado foi validado, apresentado na seção 5.2.2, garantindo assim que o modelo da Arquitetura Base apresenta o comportamento próximo ao de um sistema real. Como resultado da avaliação, a Arquitetura Base alcança uma disponibilidade de 97.30%, o que corresponde a um *downtime* anual de 235.74 horas ou 9.8 dias por ano.

Podemos considerar que a análise de disponibilidade, realizada nesta seção, apresentou resultados aceitáveis em relação à disponibilidade do serviço. Ainda assim, nas próximas seções, iremos avaliar outros aspectos desta arquitetura.

6.2 Estudo de Caso II: Análise de Sensibilidade do Modelo da Arquitetura Base

Neste segundo estudo de caso, fornecemos uma análise de sensibilidade paramétrica para identificar os componentes mais relevantes do modelo da Arquitetura Base. Pois, segundo MATOS et al. (2012), os diferentes componentes que constituem um sistema computacional não necessariamente contribuem igualmente para a disponibilidade e desempenho do sistema. Por isso, adotamos a análise de sensibilidade para investigar os parâmetros dos componentes que causam impacto significativo na disponibilidade do serviço.

Para realizar a análise de sensibilidade paramétrica dessa arquitetura as etapas seguidas foram:

Etapa 1. Definir modelo que será analisado: O modelo RBD, representado pela Figura 4.1 (a), foi usado para calcular os valores de sensibilidade da Arquitetura Base.

Etapa 2. Calcular índice de sensibilidade: A ferramenta Mercury (MODCS, 2015b) foi utilizada para calcular o índice de sensibilidade do modelo RBD. O cálculo pode ser realizado pela Equação 6.1, que utiliza a derivada parcial e os valores de disponibilidade de cada componente.

$$\begin{aligned} S_\theta(A) = {} & \frac{\partial A_{SW}}{\partial \theta} \times A_{BT} \times A_{SP} \times A_{BL} \times A_{MC} + \\ & A_{SW} \times \frac{\partial A_{BT}}{\partial \theta} \times A_{SP} \times A_{BL} \times A_{MC} + \\ & A_{SW} \times A_{BT} \times \frac{\partial A_{SP}}{\partial \theta} \times A_{BL} \times A_{MC} + \\ & A_{SW} \times A_{BT} \times A_{SP} \times \frac{\partial A_{BL}}{\partial \theta} \times A_{MC} + \\ & A_{SW} \times A_{BT} \times A_{SP} \times A_{BL} \times \frac{\partial A_{MC}}{\partial \theta} \end{aligned} \tag{6.1}$$

Onde:

A_{SW} representa a disponibilidade do *Smartwatch*;
A_{BT} representa a disponibilidade do *Bluetooth*;
A_{SP} representa a disponibilidade do *Smartphone*;
A_{BL} representa a disponibilidade da Banda Larga;
A_{MC} representa a disponibilidade da Nuvem.

Para alcançar a derivada parcial do SW, BT, SP, BL ou MC é necessário aplicar as taxas de falha (λ) e reparo (μ) de cada componente (um por vez) nas Equação 6.2 (com relação ao λ) e 6.3 (com relação ao μ), substituindo o x pelo componente de interesse:

$$\frac{\partial Ax}{\partial \lambda_x} = -\frac{\mu_x}{(\mu_x + \lambda_x)^2} \tag{6.2}$$

$$\frac{\partial Ax}{\partial \mu_x} = -\frac{\mu_x}{(\lambda_x + \mu_x)^2} + \frac{1}{\lambda_x + \mu_x} \tag{6.3}$$

A Tabela 6.7 apresenta os parâmetros de entrada de cada componente das Equações 6.2 e 6.3. Após aplicamos estes parâmetros de entrada podemos alcançar os índices de sensibilidade, cujos valores estão listados na Tabela 6.8.

A Tabela 6.8 encontra-se o *ranking* de sensibilidade para os parâmetros considerados nesta análise. Os resultados são ordenados de acordo com os valores absolutos dos índices de

Tabela 6.7 Parâmetros de entrada para o *ranking* de sensibilidade.

Parâmetro	**Descrição**	**Valor (h^{-1})**
λ_{SW}	Taxa de Falha do SW	1/9.964950
μ_{SW}	Taxa de Reparo do SW	1/0.081748
λ_{BT}	Taxa de Falha do BT	1/4881.605
μ_{BT}	Taxa de Reparo do BT	1/0.00953
λ_{SP}	Taxa de Falha do SP	1/36.908221
μ_{SP}	Taxa de Reparo do SP	1/0.077352
λ_{BL}	Taxa de Falha da BL	1/5.996402
μ_{BL}	Taxa de Reparo da BL	1/0.07896
λ_{MC}	Taxa de Falha do MC	1/207.55820
μ_{MC}	Taxa de Reparo do MC	1/0.81761

sensibilidade. O valor absoluto reflete o quão fortemente um parâmetro pode influenciar na disponibilidade. Os valores negativos indicam que existe uma relação inversa entre os parâmetros e a disponibilidade do sistema.

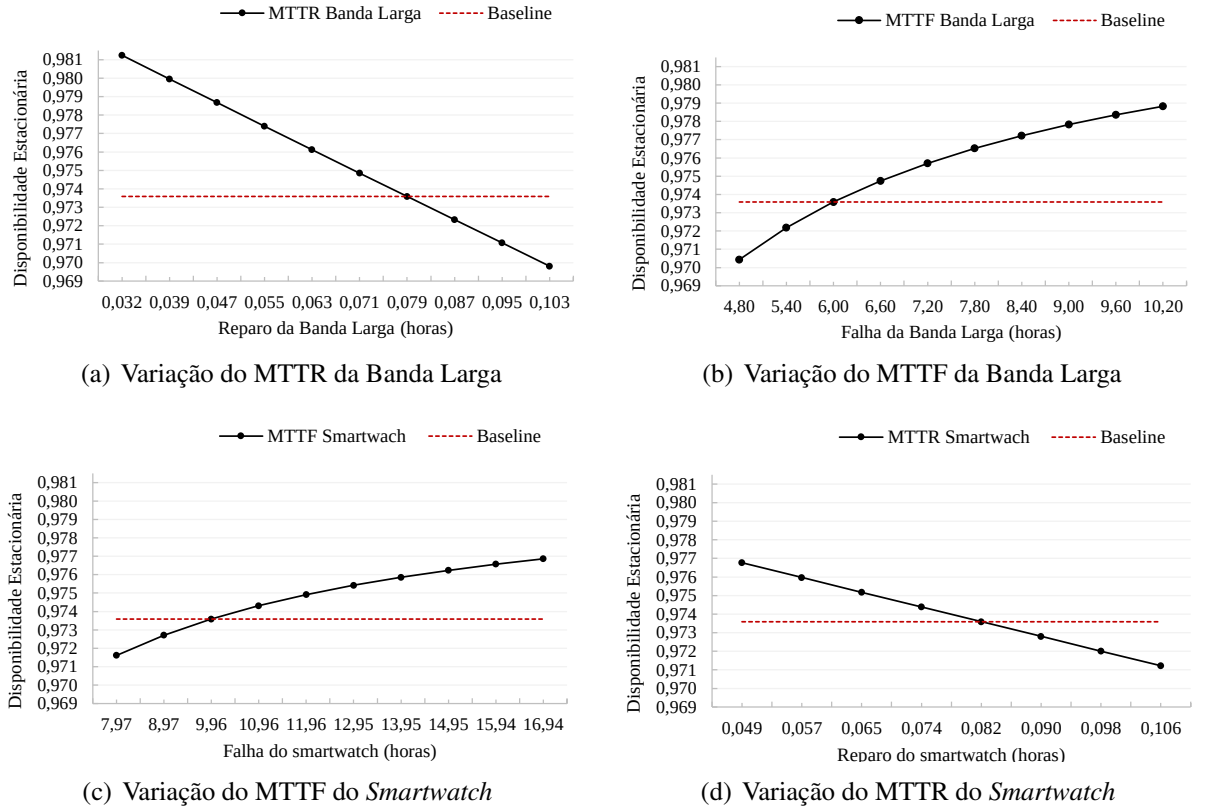
Tabela 6.8 *Ranking* das sensibilidades dos índices da Arquitetura Base.

Parâmetros	$S_\theta(A)$
MTTR Banda Larga	-1.299770×10^{-2}
MTTF Banda Larga	1.299770×10^{-2}
MTTF *Smartwatch*	8.136817×10^{-3}
MTTR *Smartwatch*	-8.136817×10^{-3}
MTTF Nuvem	3.923732×10^{-3}
MTTR Nuvem	-3.923732×10^{-3}
MTTF *Smartphone*	2.091429×10^{-3}
MTTR *Smartphone*	-2.091429×10^{-3}
MTTR *Bluetooth*	$-1.9522229 \times 10^{-6}$
MTTF *Bluetooth*	1.9522229×10^{-6}

Etapa 3. Análise dos resultados obtidos: Após realizados os cálculos é possível fazer uma análise dos resultados obtidos. Desta forma, podemos verificar que os parâmetros de MTTR Banda Larga e MTTF Banda Larga possuem os valores mais altos de sensibilidade, por isso, pode-se afirmar que o componente Banda Larga tem o maior impacto sobre a disponibilidade de estado estacionário do sistema completo e que qualquer alteração em seus parâmetros terá um impacto positivo ou negativo sobre a disponibilidade do sistema.

Após finalizar estas três etapas, realizamos uma análise mais detalhada. Para tanto, variamos um parâmetro de cada vez, mantendo os demais fixos, assim foi possível observar, o efeito sobre a disponibilidade. Para essa abordagem, uma porcentagem determinada do valor médio do parâmetro pode ser usada como variação (HAMBY, 1994). Sendo assim, os valores dos parâmetros foram variados em passos médios de 10%.

A Figura 6.1 e Figura 6.2 representam graficamente os resultados quando variamos um parâmetro por vez. Os gráficos são organizados de acordo com a ordem do *ranking* da Tabela 6.8.



(a) Variação do MTTR da Banda Larga

(b) Variação do MTTF da Banda Larga

(c) Variação do MTTF do *Smartwatch*

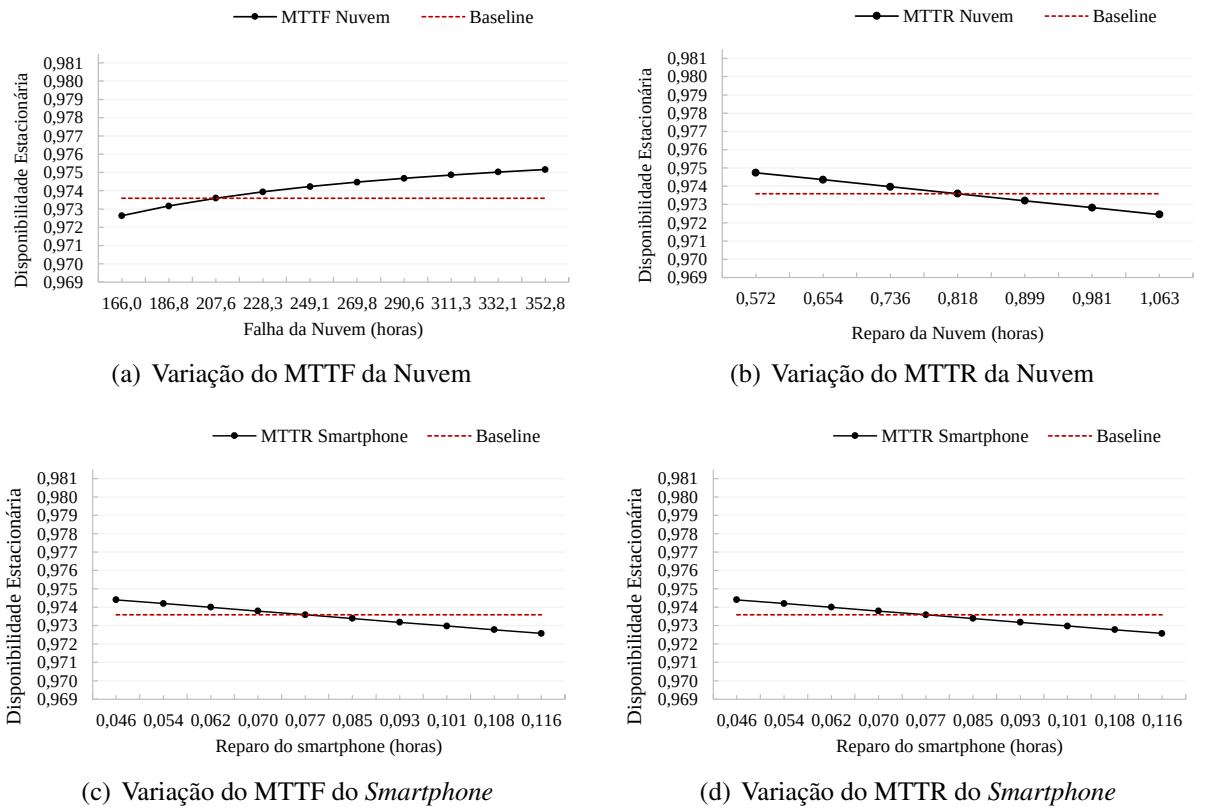
(d) Variação do MTTR do *Smartwatch*

Figura 6.1 Variação da análise de sensibilidade dos componentes Banda Larga e Smartwatch.

A disponibilidade estacionária é representada em cada gráfico por uma linha tracejada na horizontal e a linha contínua representa o valor que a disponibilidade poderia assumir, caso os tempos e taxas fossem modificadas. Todavia, esta análise é melhor aplicada aos componentes mais relevantes que foram encontrados na análise de sensibilidade. Desta forma, como o componente *Bluetooth* foi identificado como o ultimo do *ranking* da análise de sensibilidade e sua variação de parâmetros não afetou a disponibilidade do serviço, a sua representação gráfica não será apresentada.

A Figura 6.1(b) e a Figura 6.1(a) apresentam a variação dos parâmetros do componente de maior impacto, a Banda Larga. Para o parâmetro MTTR Banda Larga, variamos em intervalos de 0.031566-0.102666 horas. É observado que com a redução desse parâmetro existe uma melhora considerável da disponibilidade, onde a disponibilidade iniciou em $0,969307$ com o maior tempo de reparo e finalizou em $0,980740$ com o menor tempo de reparação, o que equivale a uma redução de $28,43\%$ no *downtime*. Quanto ao MTTF Banda Larga, foi feita uma variação em intervalos de 4.796402-10.196402 horas. A disponibilidade inicia em $0,969935$ e finaliza em $0,978327$, ou seja, uma redução de $19,46\%$ no *downtime*.

No parâmetro do MTTF *Smartwatch*, o tempo de falha teve um intervalo de variação de 7.971964 a 16.940424 horas. A disponibilidade é inicialmente $0,971113$ e termina em $0,976360$, uma redução de $12,16\%$ no *downtime*. O parâmetro MTTR *Smartwatch*, o tempo de reparo teve uma variação de 0.049049 a 0.122622 horas. A disponibilidade é inicialmente $0,976267$ e



(a) Variação do MTTF da Nuvem

(b) Variação do MTTR da Nuvem

(c) Variação do MTTF do *Smartphone*

(d) Variação do MTTR do *Smartphone*

Figura 6.2 Variação da análise de sensibilidade dos componentes Nuvem e Smartphone.

termina com $0,969150$, uma redução de $11,81\%$ no *downtime*. Podemos considerar o resultado relevante, uma vez que este componente é o segundo com maior impacto na disponibilidade. Esta análise é representada por Figura 6.1(d) e 6.1(c).

A Figura 6.2(a) e a Figura 6.2(b) representa os parâmetros do componente da Nuvem. Na Tabela 6.8 podemos observar que este é o terceiro componente de maior impacto sobre a disponibilidade. Os valores do parâmetro MTTF foram variados em 166.046564 a 352.848948 horas e o MTTR em 0.5723277 a 1.0628943 horas. No parâmetro MTTF a disponibilidade inicia em $0,972135$ e finaliza em $0,974663$, assim é possível verificar uma redução de $5,85\%$ no *downtime*. Já o MTTR, a disponibilidade inicia em $0,971944$ e finaliza em $0,974235$, ou seja, uma redução de $4,26\%$ no *downtime*.

A Figura 6.2(c) e Figura 6.2(d) representa os parâmetros de falhas e reparos do *smartphone*. No parâmetro MTTF *Smartphone* foi variado em um intervalo de 29.526539-62.743896 horas, podemos observar com essa variação um pequeno impacto na disponibilidade, pois a disponibilidade inicia em $0,972580$ e finaliza em $0,973927$, ou seja, uma redução de $3,12\%$ no *downtime*. O parâmetro MTTR *Smartphone* onde a variação foi de 0.046412-0.116029 horas. É possível verificar um pequeno aumento na disponibilidade em relação a *baseline*, onde a disponibilidade inicia em $0,972072$ e finaliza em $0,973903$, com uma redução de $3,03\%$ no *downtime*. Comparado aos componentes *Banda Larga, Smartwatch e a Nuvem*, os parâmetros do *Smartphone* tem o menor impacto na disponibilidade do sistema.

Por fim, os parâmetros do *Bluetooth*, onde de acordo com a Tabela 6.8, é o componente com menor impacto na disponibilidade. No parâmetro MTTF, a variação foi no intervalo de 3905.284 a 8298.728 horas. A disponibilidade é inicialmente $0,9730882$ e termina em $0,97308959$, uma redução de $0,00291\%$ no *downtime*, o que podemos considerar como uma redução irrelevante.

É possível observar nos gráficos dos parâmetros do *smartphone* a sua curvatura muito próxima da linha tracejada do *baseline* da disponibilidade. No entanto, apesar do *Bluetooth* não ter impacto significativo sobre a disponibilidade do sistema, ratificando os resultados da Tabela 6.8, que apresenta esse parâmetro como o de menor impacto na disponibilidade, esse componente, poderá ser usado em conjunto com o componente da Banda Larga, como redundância na interface de rede. Portanto, pode-se concluir, conforme a curvatura dos gráficos apresentados, que quanto mais eficiente for o processo de reparação, melhor será a disponibilidade apresentada.

Com a identificação do componente da Banda Larga como sendo o maior índice de sensibilidade da Arquitetura Base, o resultado desse estudo de caso, sugere que ao estender a Arquitetura Base para melhorar a disponibilidade, o componente Banda Larga é um dos pontos focais a serem considerados. No entanto, os outros componentes da arquitetura não devem ser desconsiderados. Dessa forma, extensões da arquitetura serão propostas e investigadas com o intuito de otimizar os resultados de disponibilidade, através da adoção de redundância dos componentes, conforme apresentado na Seção 6.3.

6.3 Estudo de Caso III: Avaliação do Impacto de Mecanismos de Redundância na Arquitetura Base

No Estudo de Caso III avaliamos duas extensões, elaboradas a partir do modelo da Arquitetura Base. O primeiro modelo da arquitetura estendida concentra-se em adicionar redundância na interface de rede, enquanto a segunda, adiciona redundância na infraestrutura de nuvem, utilizando *Cloudlet*. As extensões foram baseadas na classificação da análise de sensibilidade paramétrica apresentada na Tabela 6.8. A avaliação dos modelos das arquiteturas teve como base as métricas de disponibilidade e *downtime* anual. Por fim, no final desse estudo de caso, apresentamos um comparativo entre a disponibilidade e o *downtime* anual da Arquitetura Base com as arquiteturas estendidas.

6.3.1 Modelo da Arquitetura com Múltiplas Interfaces de Rede

Considerando que a análise de sensibilidade, apresentada na Seção 6.2, indica que o componente Banda Larga é o mais crítico, em termos de disponibilidade, ou seja, tem maior criticidade no serviço. Uma alternativa que pode aumentar a disponibilidade do serviço é a utilização de uma segunda interface de rede para a comunicação dos dispositivos móveis. Assim,

uma nova arquitetura é proposta (Figura 6.3), a partir da adição de novas interfaces de rede para aumentar a disponibilidade, mitigando possíveis falhas que possam ocorrer tendo uma única interface de conexão.

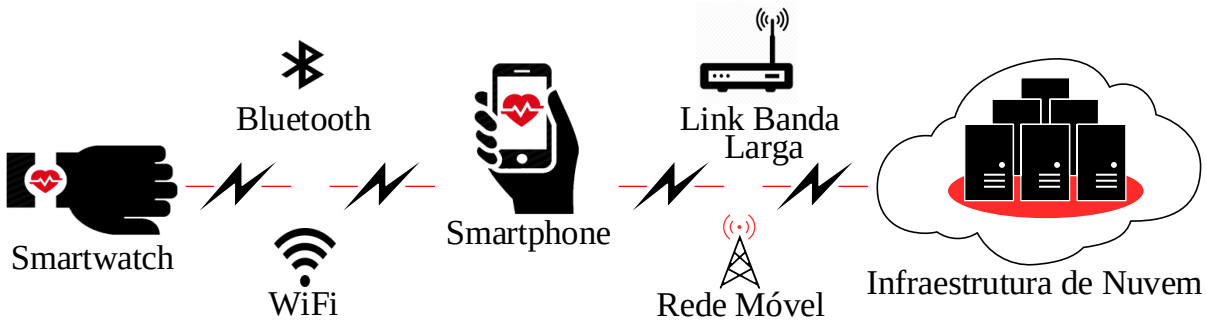


Figura 6.3 Arquitetura com Múltiplas Conexões de Rede.

A arquitetura representada na Figura 6.3, a redundância é aplicada em dois pontos: entre o *Smartwatch* e o *Smartphone*; e entre o *Smartphone* e a Nuvem. Primeiramente, adicionamos uma conexão *WiFi*, entre o *Smartwatch* e o *Smartphone*, mantendo a conexão *Bluetooth*. No segundo ponto, adicionamos uma conexão de Rede Móvel, entre o *Smartphone* e a Nuvem, mantendo a conexão Banda Larga. Neste cenário, a conexão *WiFi* entre os dispositivos, refere-se a uma rede local sem fios (*Wireless Local Area Network* (WLAN)).

Deste modo, caso ocorra indisponibilidade da conexão *Bluetooth*, a conexão entre o *Smartwatch* e o *Smartphone* será estabelecida através da conexão *WiFi*, e vice-versa. Logo, a indisponibilidade do serviço causada por falha na comunicação entre *Smartwatch* e *Smartphone* só será possível quando tanto o *Bluetooth* quanto a *WiFi* estiverem indisponíveis simultaneamente.

Processo similar ocorre na comunicação entre o *Smartphone* e a Nuvem. Caso ocorra indisponibilidade da conexão Banda Larga, a conexão entre o *Smartphone* e a Nuvem será estabelecida através da conexão de Rede Móvel, e vice-versa. Logo, a indisponibilidade do serviço causada por falha na comunicação entre *Smartphone* e Nuvem só será possível quando tanto o Banda Larga quanto a Rede Móvel estiverem indisponíveis simultaneamente.

Vale ressaltar que a desconexão ou falha de uma ou mais interface de rede não afeta a disponibilidade dos demais. Como supracitado, poderá ocorrer indisponibilidade no serviço, se duas interfaces de rede que possuem a mesma função (conexão entre *Smartwatch* e *Smartphone* ou conexão entre *Smartphone* e Nuvem) não estiverem disponíveis no mesmo instante.

A Figura 6.4 apresenta o modelo RBD da Arquitetura com Múltiplas Interfaces de Rede. As interfaces de rede redundantes são representadas em blocos paralelos, assim, é necessário que pelo menos um dos componentes em paralelo esteja funcionando. Os blocos restantes, *Smartwatch*, *Smartphone* e Nuvem, são representados através de submodelos da mesma forma que na Arquitetura Base (Figura 4.1 (a)), exceto pela Bateria dos dispositivos móveis, pois a autonomia da bateria de um dispositivo móvel depende fortemente da interface de rede utilizada para comunicação (OLIVEIRA et al. (2013)). Sendo assim, os submodelos da bateria do

Smartwatch e do *Smartphone*, são representados pelo CTMC descrito na Figura 6.5 e Figura 6.6, respectivamente.

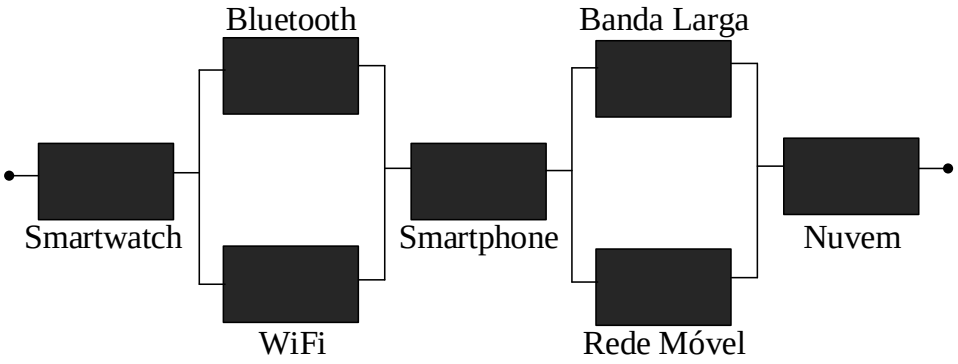


Figura 6.4 Modelo RBD da Arquitetura com Múltiplas Interfaces de Rede.

O modo operacional do modelo RBD representado na Figura 6.4, é representado pela Expressão 6.4 e a disponibilidade é calculada através de Equação 6.5.

$$OM = SW \wedge (BT \vee WF) \wedge SP \wedge (RM \vee BL) \wedge MC \tag{6.4}$$

$$A = A_{SW} \times (1-(1-A_{BT}) \times (1-A_{WF})) \times A_{SP} \times (1-(1-A_{RM}) \times (1-A_{BL})) \times A_{MC} \tag{6.5}$$

Onde:
A representa a disponibilidade da Arquitetura;
A_{SW} representa a disponibilidade do *Smartwatch*;
A_{BT} representa a disponibilidade do *Bluetooth*;
A_{WF} representa a disponibilidade do *WiFi*;
A_{SP} representa a disponibilidade do *Smartphone*;
A_{RM} representa a disponibilidade do Rede Móvel;
A_{BL} representa a disponibilidade do Banda Larga;
A_{MC} representa a disponibilidade da Nuvem.

Já a indisponibilidade da arquitetura pode ser calculada com a Equação 6.6 e o *downtime* anual em horas pode ser representado através da Equação 6.7.

$$UA = 1 - A \tag{6.6}$$

$$Dty_A = UA \times 8760h \tag{6.7}$$

O modelo CTMC, representado na Figura 6.5, apresenta os processos de descarga da bateria baseado no percentual do nível da bateria, iniciando em 100% e finalizando em 0%. A

análise do modelo fornece a disponibilidade do componente Bateria, considerando a conexão *Bluetooth* e *WiFi*.

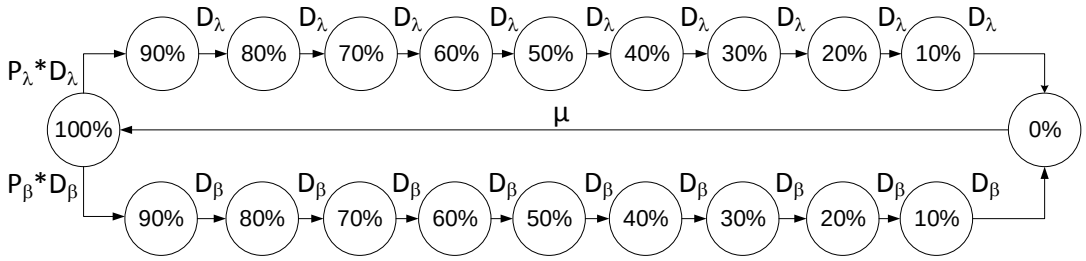


Figura 6.5 Modelo de descarga da bateria do *smartwatch*.

No primeiro estado, o estado 100%, a bateria está completamente carregada, enquanto o estado 0% representa o estado da bateria completamente descarregada. Este CTMC denota um tempo de descarga após uma distribuição de *Erlang* de 10 estágios (TRIVEDI, 2008). As taxas de descarga são representadas em intervalos de 10%, denotado por D_λ para *Bluetooth* e D_β para *WiFi*. A probabilidade do dispositivo está conectado através do *Bluetooth* durante o processo de descarga do dispositivo é representado por P_λ e utilizando a conexão *WiFi* é representado por P_β. A transição entre *Bluetooth* e *WiFi* não é representada no modelo, por motivo de simplicidade. Assumimos que esta implicação não produz impacto significante nos resultados do modelo completo. Quando o modelo atinge o estágio final considera-se que a bateria está descarregada e para retornar a sua condição inicial é necessário que ocorra a substituição do dispositivo; este evento é representado pela taxa de substituição do estado 0% ao estado 100%, através do parâmetro μ.

Considerando este modelo da Figura 6.5, a disponibilidade da bateria é computada através da Equação 6.8. O n representa a quantidade de transições que o modelo faz até chegar a descarga completa, neste caso, $n = 09$.

$$\mathrm{A}_{BR_{SW}} = \frac{(\mu \times (1 + n \times P_\beta + n \times P_\lambda))}{(\mu + D_\beta \times P_\beta + n \times \mu \times P_\beta + D_\lambda \times P_\lambda + n \times \mu \times P_\lambda)} \tag{6.8}$$

Onde:

$A_{BR_{SW}}$ representa a disponibilidade da bateria do *Smartwatch*;
μ representa o tempo do dispositivo ser substituído;
P_β representa a probabilidade de estar conectado à *WiFi*;
P_λ representa a probabilidade de estar conectado ao *Bluetooth*;
D_β representa a taxa de descarga da bateria utilizando *WiFi*;
D_λ representa a taxa de descarga da bateria utilizando *Bluetooth*.

Similar ao modelo de descarga do *smartwatch*, o modelo CTMC ilustrado na Figura 6.6 representa o processo de descarga da bateria do *smartphone*, considerando quatro parâmetros

possíveis para a comunicação de dados: *Bluetooth*, *WiFi*, Banda Larga e Rede Móvel.

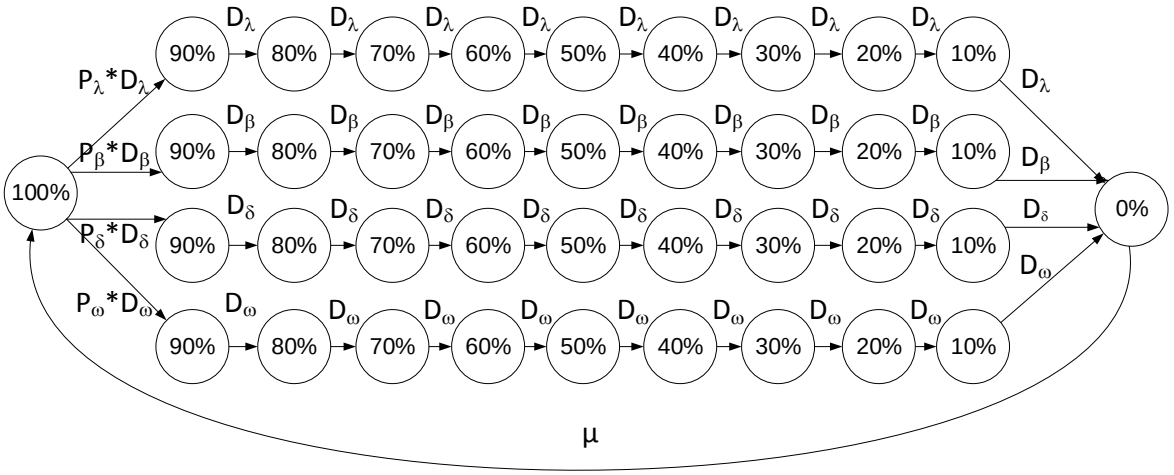


Figura 6.6 Modelo de descarga da bateria do *smartphone*.

O parâmetro P_λ, representa a probabilidade do *smartphone* de estar conectado ao *Bluetooth* e Banda Larga. O *Bluetooth* é o meio de comunicação entre os dispositivos (*smartwatch* e *smartphone*) e a conexão Banda Larga é o meio de comunicação entre o *smartphone* e a nuvem, como apresentado na Figura 6.3. A taxa de descarga ao utilizar *Bluetooth* e a conexão Banda Larga é representado por D_λ. Da mesma forma, os parâmetros P_β, P_δ e P_ω, referem-se a probabilidade de estar conectado simultaneamente ao *Bluetooth* e Rede Móvel; *WiFi* e Rede Móvel; e *WiFi* e Banda Larga. Os parâmetros de descarga são representados por D_β, D_δ e D_ω, respectivamente. A taxa de substituição da bateria do dispositivo é denotado pelo parâmetro μ. O n representa a quantidade de transições que o modelo faz até chegar a descarga completa, neste caso, $n = 09$.

Considerando modelo representado na Figura 6.6, a disponibilidade da bateria é dada pela Equação 6.9.

$$A_{BR_{SP}} = \frac{\mu \times (1 + n \times P_\beta + n \times P_\delta + n \times P_\lambda + n \times P_\omega)}{D_\beta \times P_\beta + D_\delta \times P_\delta + D_\lambda \times P_\lambda + D_\omega \times P_\omega + \mu \times (1 + n \times P_\beta + n \times P_\delta + n \times P_\lambda + n \times P_\omega)} \quad (6.9)$$

Onde:

$A_{BR_{SP}}$ representa a disponibilidade da bateria do *Smartphone*;

μ representa o tempo da bateria ser substituída;

P_λ representa a probabilidade de estar conectado ao *Bluetooth* e Banda Larga;

P_β representa a probabilidade de estar conectado ao *Bluetooth* e Rede Móvel;

P_δ representa a probabilidade de estar conectado à *WiFi* e Rede Móvel;

P_ω representa a probabilidade de estar conectado à *WiFi* e Banda Larga;

D_λ representa a taxa de descarga do *Bluetooth* e Banda Larga;

D_β representa a taxa de descarga do *Bluetooth* e Rede Móvel;

D_δ representa a taxa de descarga da *WiFi* e Rede Móvel;

D_ω representa a taxa de descarga *WiFi* e Banda Larga.

Avaliação da Disponibilidade do Modelo da Arquitetura com Múltiplas Interfaces.

Os parâmetros utilizados no modelo representado na Figura 6.10 são os mesmos apresentados na Seção 6.1, com exceção da bateria dos dispositivos e as conexões: *Bluetooth*, *WiFi* e Rede Móvel.

Para coletar as taxas do modelo da bateria dos dispositivos (*smartwatch* e *smartphone*) desse cenário, foi repetido o experimento descrito na Seção 5.1, observando o tempo de descarga, considerando as interfaces de rede distintas: *WiFi* e Rede Móvel.

As Tabelas 6.9 e 6.10 listam os parâmetros de entrada da bateria dos dispositivos.

Tabela 6.9 Parâmetros de entrada para bateria do *smartwatch* (Figura 6.5).

Parâmetros	D_λ	P_λ	D_β	P_β	μ
Valor	0.966183	0.7	1.113172	0.3	12

Tabela 6.10 Parâmetros de entrada para bateria do *smartphone* (Figura 6.6).

Parâmetros	P_λ	D_λ	P_β	D_β	P_δ	D_δ	P_ω	D_ω	μ
Valor	0.6	0.233857	0.2	0.298433	0.05	0.269179	0.15	0.215698	12

A Tabela 6.11 apresenta os parâmetros de entrada das interfaces de conexão. Os valores de MTTF e MTTR para a Rede Móvel foram extraídos de COOPER; FARRELL (2007), enquanto as taxas do *Bluetooth* são obtidas de CARROZZA et al. (2008). O MTTF e o MTTR para o componente *Bluetooth* diferem dos valores do modelo de Arquitetura Base porque a conexão *WiFi* interfere na conexão *Bluetooth* (CARROZZA et al., 2008).

Tabela 6.11 Parâmetros de entrada para *Bluetooth*, *WiFi* e Rede Móvel.

Componentes	**MTTF (h)**	**MTTR (h)**
Bluetooth	458.7085	0.00751
WiFi	10000	1.667
Rede Móvel	3.999807	0.078361

A Tabela 6.12 apresenta os resultados da análise do modelo ilustrado na Figura 6.4, em termos de disponibilidade e *downtime* anual. Os resultados demonstram que a Arquitetura com Múltiplas Interfaces causa um impacto positivo na disponibilidade em relação a Arquitetura Base. É possível observar uma redução de 45% no *downtime* anual em relação a Arquitetura Base com a inclusão de Múltiplas Interfaces. Estes valores reforçam o resultado da análise de

sensibilidade apresentada no Estudo de Caso II (Seção 6.2), no qual é ratificado que a Banda Larga é o componente com o maior índice de sensibilidade da Arquitetura Base.

Tabela 6.12 Análise de dependabilidade da Arquitetura com Múltiplas Interfaces.

Disponibilidade	**Disponibilidade (9's)**	***Downtime* (h)**
0,9852017	1.829	129.63

6.3.2 Modelo da Arquitetura Cloudlet

Nesta seção, uma segunda arquitetura estendida é proposta a partir da inclusão de uma *Cloudlet*, com objetivo de investigar se o uso deste tipo de arquitetura poderá apresentar melhorias no serviço.

A partir dos resultados da análise de sensibilidade da Arquitetura Base (Seção 6.2), foi identificado que o componente *Smartwatch* é o segundo mais crítico do modelo, no entanto, como mencionado na Seção 6.1, assumimos como estratégia de redundância a existência de um dispositivo extra para substituição, de forma a reduzir o tempo de reparo. Sendo assim, o enfoque foi dado ao componente Nuvem, visto que este foi o terceiro componente de maior criticidade no serviço. Desta forma, adotamos a Arquitetura *Cloudlet* como mecanismo de redundância à Nuvem para continuar provendo o serviço para o cliente, mesmo quando a Nuvem estiver indisponível.

O conceito de *Cloudlet* foi introduzido por SATYANARAYANAN et al. (2009) como uma infraestrutura computacional que está na mesma rede local dos usuários, possuindo os mínimos requisitos necessários para o provimento do serviço principal. Essa nova arquitetura tem como objetivo prover recursos computacionais para clientes móveis com um tempo de resposta mais baixo, uma vez que tais recursos estarão próximos dos usuários, na mesma WLAN. Desta forma, os clientes podem obter os benefícios da computação em nuvem, sem sofrer dos problemas que surgem ao acessar uma nuvem remota (OLIVEIRA et al., 2013). A Figura 6.7 representa a Arquitetura *Cloudlet*.

O usuário terá acesso aos serviços da *Cloudlet* enquanto permanecer na área de cobertura da WLAN, onde a *Cloudlet* está localizada. Entretanto, caso o usuário encontre-se fora do alcance da *Cloudlet*, ou seja, fora da área de cobertura da WLAN, é possível utilizar o serviço da nuvem através da rede móvel, como ilustrado na Figura 6.7. No caso em que a *Cloudlet* esteja indisponível, o cliente pode acessar o serviço da nuvem principal via rede WLAN ou conexão Rede Móvel. Por outro lado, se a nuvem principal estiver indisponível, o usuário pode utilizar o serviço da *Cloudlet* apenas se estiver na área de cobertura da WLAN.

A Figura 6.8 apresenta o RBD para a arquitetura estendida, com a inclusão da *Cloudlet*. O modelo da Figura 6.8 apresenta o componente *Cloudlet* em paralelo com o componente Nuvem, podendo ser acessado através da conexão banda larga. O componente Nuvem pode ser acessado tanto através da conexão banda larga quanto através da conexão de rede móvel, desta

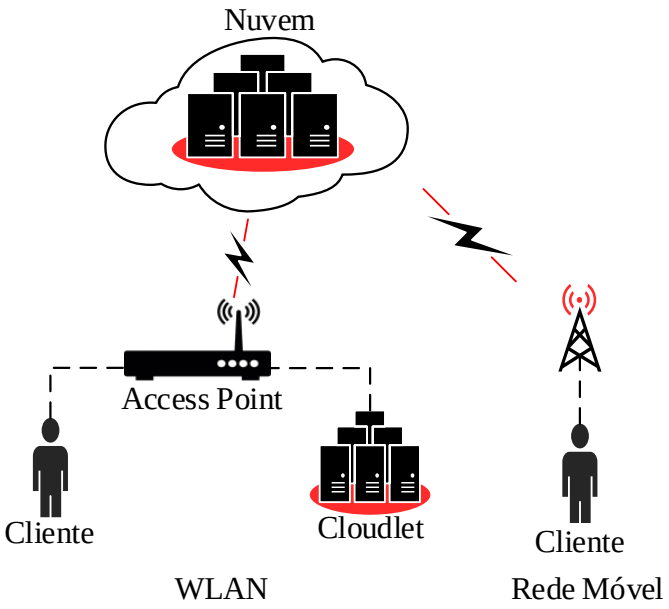


Figura 6.7 Arquitetura Cloudlet.

forma, a sua representação por meio de modelo RBD exige a repetição de componentes. O bloco Nuvem ocorre em dois locais, no entanto, não se trata de duas nuvens diferentes, mas de duas ocorrências do mesmo componente.

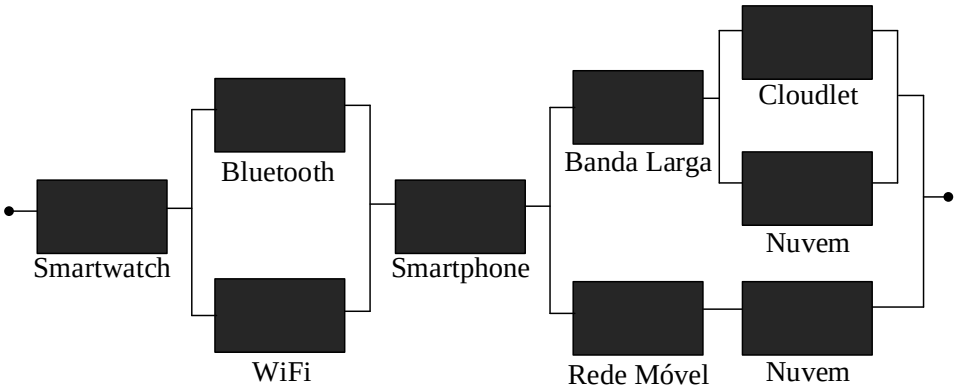
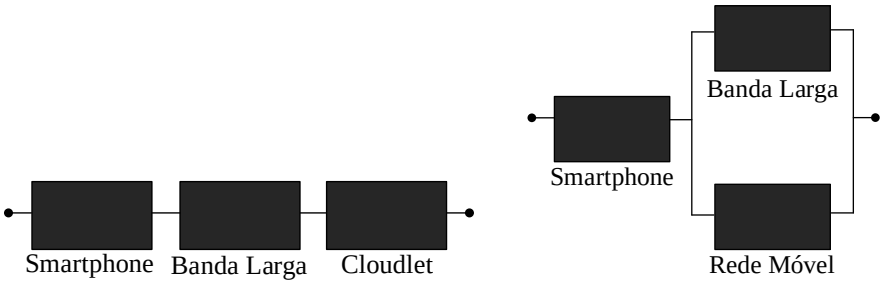


Figura 6.8 Modelo RBD para Arquitetura *Cloudlet*.

Para a avaliação desse modelo é necessário fatorar o componente repetido (MACIEL et al., 2012), gerando dois submodelos, representados em Figura 6.9(a) e Figura 6.9(b). O primeiro submodelo (Figura 6.9(a)) representa o componente Nuvem no modo de indisponibilidade, a disponibilidade é calculada pela Equação 6.10. O segundo submodelo (Figura 6.9(b)) representa o componente Nuvem no modo de disponibilidade, o cálculo da disponibilidade é dado pela Equação 6.11. A disponibilidade do componente Nuvem para estes dois casos é dada pela Equação 6.12.

$$A_a = A_{SP} \times A_{BL} \times A_{CL} \tag{6.10}$$



(a) Componente Nuvem indisponível (b) Componente Nuvem disponível

Figura 6.9 Fatoração do modelo em RBD da Arquitetura *Cloudlet*.

$$A_b = A_{SP} \times (1 - (1 - A_{RM}) \times (1 - A_{BL})) \tag{6.11}$$

$$A_{f_{MC}} = A_{MC} \times A_a + (1 - A_{MC}) \times A_b \tag{6.12}$$

O modo operacional do modelo da Arquitetura *Cloudlet* (Figura 6.8) é apresentado pela Expressão 6.13, enquanto a disponibilidade desta arquitetura é calculada pela Equação 6.14.

$$OM = SW \wedge (BT \vee WF) \wedge SP \wedge (BL \wedge (CL \vee MC) \vee (RM \wedge MC)) \tag{6.13}$$

$$\begin{aligned} A_{Cloudlet} = A_{SW} \times (1 - (1 - A_{BT}) \times (1 - A_{WF})) \\ \times A_{SP} \times (A_{CL} \times A_{BL} + A_{f_{MC}} \times (A_{RM} - (-1 + A_{CL} + A_{RM}) \times A_{BL})) \end{aligned} \tag{6.14}$$

Onde:

$A_{Cloudlet}$ representa a disponibilidade da Arquitetura *Cloulet*;
A_{SW} representa a disponibilidade do *Smartwatch*;
A_{BT} representa a disponibilidade do *Bluetooth*;
A_{WF} representa a disponibilidade da *WiFi*;
A_{SP} representa a disponibilidade do *Smartphone*;
A_{BL} representa a disponibilidade da Banda Larga;
A_{RM} representa a disponibilidade da Rede Móvel;
A_{CL} representa a disponibilidade da *Cloudlet*;
$A_{f_{MC}}$ representa a disponibilidade da Nuvem refatorada;
A_{MC} representa a disponibilidade da Nuvem.

Já a indisponibilidade da Arquitetura *Cloudlet* pode ser calculada com a Equação 6.15 e o *downtime* anual em horas, pode ser representado através da Equação 6.16.

$$UA = 1 - A_{Cloudlet} \tag{6.15}$$

$$Dty_{Cloudlet} = UA \times 8760h \tag{6.16}$$

Avaliação da Disponibilidade do Modelo da Arquitetura *Cloudlet*

Os parâmetros utilizados para a bateria dos dispositivos móveis são os mesmos apresentados nas Tabelas 6.9 e 6.10. Os demais componentes são apresentados na Seção 6.1 e Tabela 6.11. Os resultados dessa análise, em termos de disponibilidade e *downtime* anual, são apresentados na Tabela 6.13.

Tabela 6.13 Análise de dependabilidade da Arquitetura *Cloudlet*.

Disponibilidade	Disponibilidade (9's)	Downtime (h)
0,989041	1.960	96.00h

Com base na análise de dependabilidade dessa proposta, o resultado apresentou uma disponibilidade de 98,90%, o que representa um tempo de *downtime* anual de 96.0 horas, que corresponde a quatro dias por ano. Esse resultado apresenta um melhor ganho de disponibilidade quando comparado à Arquitetura Base. Na próxima seção será apresentado um comparativo entre as três arquiteturas.

6.3.3 Avaliação e Comparação entre as Arquiteturas

Nesta seção, comparamos a Arquitetura Base com as suas duas versões estendidas, avaliadas nas Seções 6.3.1 e 6.3.2. A Figura 6.10 apresenta a disponibilidade e o *downtime* anual em horas para as três arquiteturas analisadas.

Podemos observar que a disponibilidade da Arquitetura Base é de 97.30% e o *downtime* de 235.74h. Ao adicionar múltiplas conexões de rede, há um aumento de 1,21% na disponibilidade e o *downtime* diminuiu para 129.63h, ou seja, o tempo de inatividade ao longo de um ano é 45% menor, em relação à Arquitetura Base.

Os resultados são ainda mais promissores quando se estende a Arquitetura Base, adicionando redundância na Nuvem. Neste estudo houve um aumento de 1.60% na disponibilidade e o *downtime* diminui de 235.74h para 96.0h (139.74h de diferença), o que representa que o tempo de inatividade ao longo de um ano é 59% menor, quando comparado aos resultados da Arquitetura Base.

Ao analisar os gráficos, podemos inferir que ao aplicar redundância nos componentes de maior criticidade da arquitetura obtivemos os melhores resultados. A Arquitetura *Cloudlet* mostrou ser a melhor opção para aumentar a disponibilidade do serviço proposto. Considerando que esta arquitetura dispõe dos mesmos recursos que a arquitetura com múltiplas interfaces de

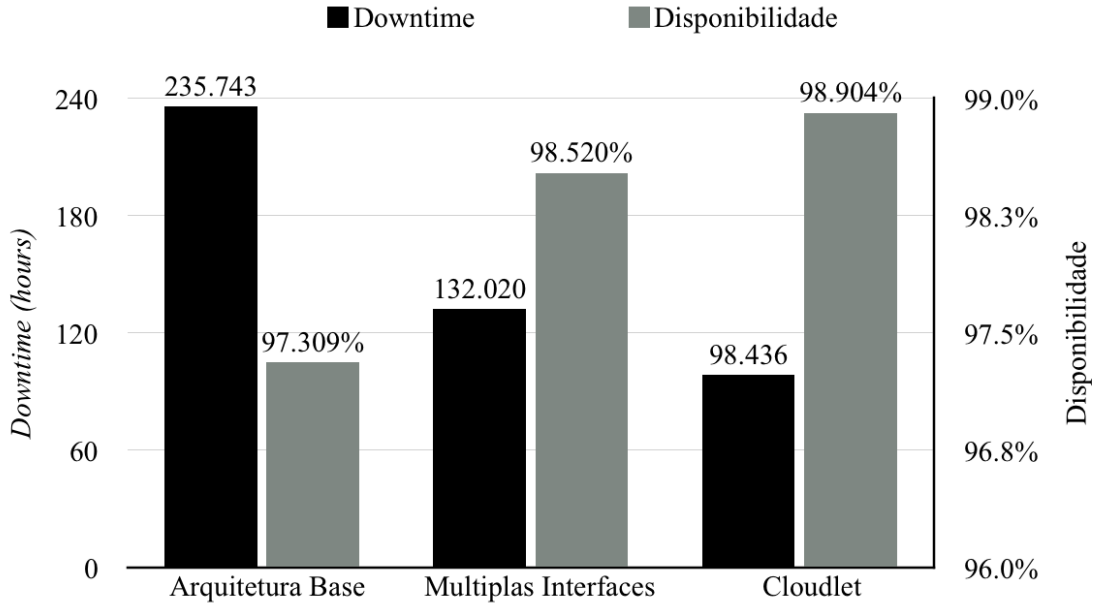


Figura 6.10 Comparação da disponibilidade e *downtime* entre as três arquiteturas

conexão, diferenciando-se pela utilização de uma *Cloudlet*, não podemos desprezar a arquitetura com múltiplas interfaces de conexão, pois apresentou resultados de disponibilidade superiores a Arquitetura Base e possui um custo inferior à Arquitetura *Cloudlet*.

Vale destacar que no escopo desta pesquisa, a análise de custo dos cenários não foi abordada. Contudo, considera-se implícito o custo adicional de manutenabilidade do componente *Cloudlet* em comparação com a arquitetura com múltiplas interfaces de conexão.

No entanto, ao introduzir redundância de conexão no sistema deve ser considerado que pode haver diferença da autonomia da bateria dos dispositivos. Isso significa que as baterias podem descarregar mais rapidamente, o que diretamente pode afetar a disponibilidade do serviço. Logo, devido a necessidade de investigação desta hipótese, na Seção 6.4, analisamos o aspecto da disponibilidade dos dispositivos móveis em diferentes cenários.

6.4 Estudo de Caso IV: Avaliação de Disponibilidade dos Dispositivos Móveis

Nesta seção, avaliamos o modelo RBD do *smartwatch* ilustrado na Figura 4.1 (b) e do *smartphone* ilustrado na Figura 4.1 (c), estes foram detalhados na Seção 4.1. Os modelos RBDs que representam os dispositivos são formados por outros componentes que podem apresentar falhas em algum determinado tempo e impactar na disponibilidade e confiabilidade dos dispositivos. Além disso, com a inclusão de diferentes conexões pode causar impacto na autonomia da bateria, e consequentemente, sobre disponibilidade dos dispositivos e do serviço.

Isto posto, este Estudo de Caso teve três finalidades: (i) identificar a importância dos componentes em relação a disponibilidade; (ii) conhecer o tempo médio no qual o dispositivo

móvel irá funcionar sem que apresente falhas em seus componentes; (iii) investigar os efeitos do uso de diferentes interfaces de conexão sobre a autonomia das baterias e sua relação com a disponibilidade do serviço.

A Figura 4.1 (b) representa o modo de falha do dispositivo *smartwatch*, que é composto por cinco blocos em série: *Hardware* (HW), Bateria (BR), Sistema Operacional (SO), Aplicação (AP) e Sensor (SR). O modelo RBD do *smartphone* (Figura 4.1 (c)), por sua vez, é representado por quatro blocos em série: *Hardware* (HW), Bateria (BR), Sistema Operacional (SO) e Aplicação (AP).

Os componentes, supracitados, integram a infraestrutura dos dispositivos móveis e através dos parâmetros de entrada, apresentados na Seção 6.1 e na Seção 6.3.1, obtivemos os valores de MTTF e o MTTR dos dispositivos, para a análise de disponibilidade e confiabilidade dos mesmos. A Tabela 6.14 apresenta o MTTF e o MTTR do *smartwatch* e do *smartphone*, obtidos através dos modelos RBDs apresentados na Figura 4.1 (b) e Figura 4.1 (c).

Tabela 6.14 MTTF e MTTR dos dispositivos.

Componentes	**MTTF (h)**	**MTTR (h)**
Smartwatch	9.545505	8.181543×10^{-2}
Smartphone	35.348173	7.760621×10^{-2}

Quando disponível, o *smartphone* leva mais tempo para falhar e menos tempo para ser reparado em comparação com o *smartwatch*. Entre os dispositivos, a diferença absoluta do MTTF é a cerca de $25.802h$, enquanto o MTTR apenas $0.0042092h$ (0.25 minutos). Podemos inferir que o *smartwatch* desempenha um papel de maior criticidade no sistema, uma vez que seu tempo médio entre falhas é menor do que o do *smartphone*, enquanto a diferença irrelevante do MTTR indica que o tempo de reparo pode ser desconsiderado na análise, pois não representa um aspecto crítico.

A análise de importância de componentes é um método útil para os projetistas, fabricantes e especialistas em manutenção, para descobrir como a falha de um componente poderá afetar o funcionamento de todo o sistema e, assim, identificar os pontos fracos do sistema (SI; LIU; CAI, 2009), guiando assim, os pontos fracos que requerem melhorias.

Desta forma, para estimar a importância dos componentes dos dispositivos móveis, utilizamos o método de Importância para a Disponibilidade (*Availability Importante*, descrito na Seção 2.6). Através da avaliação de cada componente, que integram a infraestrutura dos dispositivos, é possível destacar quais componentes têm maior impacto na disponibilidade do sistema, ou quais componentes que, aumentando sua disponibilidade, aumentarão significativamente a disponibilidade do serviço todo.

Diferente da análise de sensibilidade paramétrica realizada no Estudo de Caso da Seção 6.2, esta abordagem considera apenas os componentes dos modelos RBDs dos dispositivos *smartwatch* e *smartphone*. Assim, os componentes foram analisados a partir da perspectiva da disponibilidade. Com base no MTTF e MTTR de cada componente foi possível obter, através da

ferramenta *Mercury* (SILVA et al., 2012), a análise de importância dos componentes em relação a disponibilidade.

Na Figura 6.11 são apresentados índices de importância de componentes dos dispositivos, considerando os tempos de falha e reparo de cada componente analisados nos modelos RBDs (Figura 4.1 (b) e Figura 4.1 (c)).

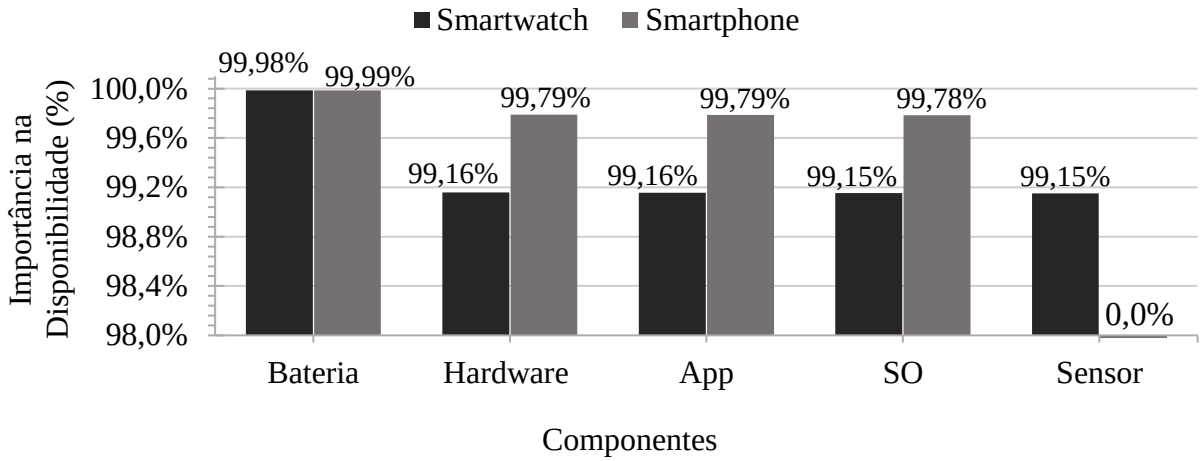


Figura 6.11 Importância dos componentes na disponibilidade.

Como podemos observar, o *Hardware*, o Sistema Operacional, a Aplicação e o Sensor apresentam valores de disponibilidade bastante semelhantes. Por outro lado, a Bateria tem um impacto de disponibilidade mais significativo (maior que 99,9%). Deste modo, o componente Bateria de ambos os dispositivos possui o maior índice de importância no aspecto disponibilidade. Isso significa que o consumo de energético é um fator importante e deve ser investigado, em trabalhos futuros, no intuito de fornecer mecanismos e técnicas mais eficientes de utilização deste recurso e consequentemente otimizar a disponibilidade do serviço, em relação a este aspecto.

Com os diferentes componentes que integram a infraestrutura dos dispositivo móveis e com diferentes tempos de falhas e reparos para cada componente, a confiabilidade dos dispositivos vai sendo degradada com o tempo, isto é, haverá um instante no qual os dispositivos móveis irão falhar.

Assim, uma análise de confiabilidade foi realizada para verificar a operação estável dos dispositivos ao longo do tempo. O tempo de observação total corresponde a quarenta e oito horas ($48h$), com intervalos de duas horas ($2h$). A Figura 6.12 mostra como a confiabilidade dos dispositivos diminui ao longo do tempo. A probabilidade dos dispositivos fornecerem o serviço ao qual foram designados, sem que ocorra interrupção do serviço até um determinado instante é de 43.25% para o *smartwatch*, com tempo aproximado do seu MTTF, e 36.12% para o *smartphone*, com o tempo aproximado do seu MTTF.

Por fim, investigamos os efeitos do uso de diferentes interfaces de rede sobre a disponibilidade do serviço e da autonomia da bateria dos dispositivos móveis. Para a análise, variamos os parâmetros de probabilidade de conexão do modelo apresentado na Figura 6.5 e Figura 6.6.

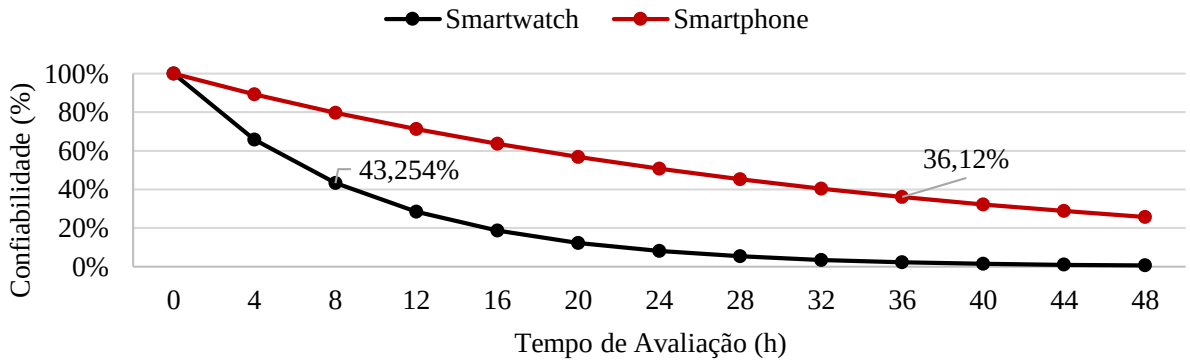


Figura 6.12 Importância da confiabilidade do *smartwatch* e do *smartphone*.

Dessa forma, é possível avaliar a disponibilidade da bateria nas diferentes composições de cenários, investigando assim o impacto causado na autonomia da bateria.

Para o estudos de caso, sete cenários distintos são propostos para avaliação. Cada cenário considera distintas probabilidades de conexão. A Figura 6.13 apresenta os cenários utilizados para este estudo de caso.

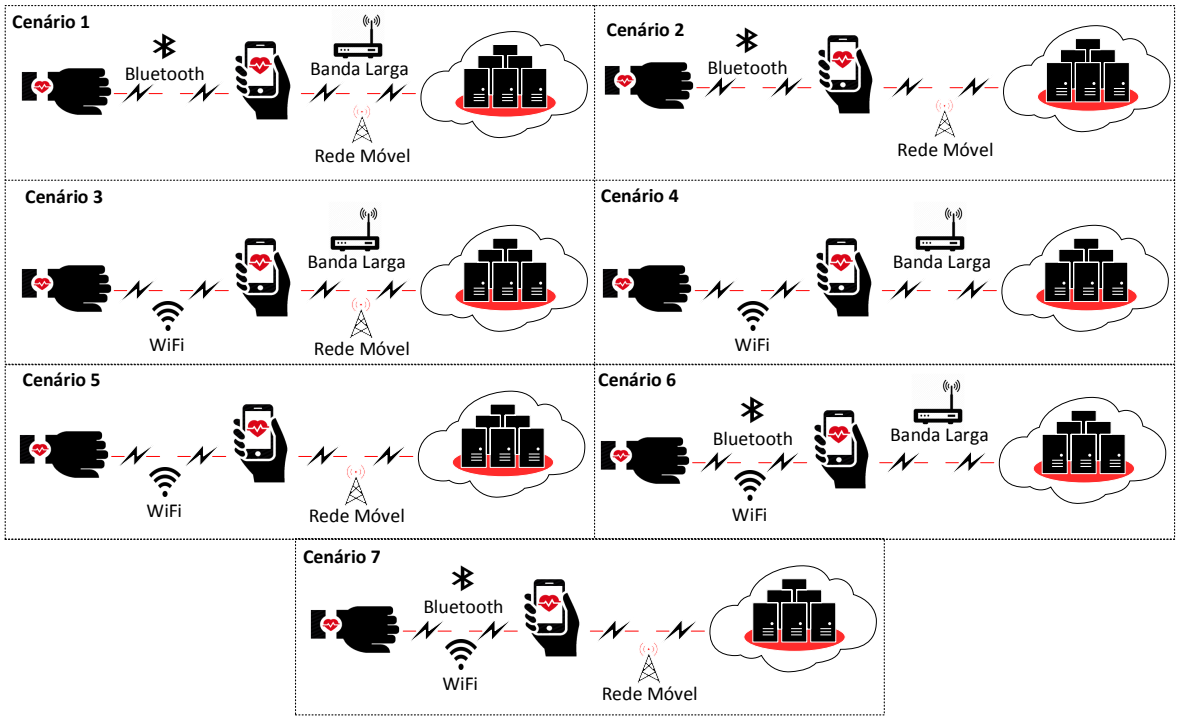


Figura 6.13 Cenários propostos para estudos de caso.

- **Cenário 1** - O usuário possui conexão *Bluetooth* e conectividade com Banda Larga e Rede Móvel disponíveis;
- **Cenário 2** - O usuário possui conexão *Bluetooth* e conectividade com Rede Móvel disponível;

- **Cenário 3** - O usuário possui conexão *WiFi* e conectividade com Banda Larga e Rede Móvel disponíveis;
- **Cenário 4** - O usuário possui conexão *WiFi* e conectividade com Banda Larga disponível;
- **Cenário 5** - O usuário possui conexão *WiFi* e conectividade com Rede Móvel disponível;
- **Cenário 6** - O usuário possui as conexões *Bluetooth* e *WiFi* e conectividade com Banda Larga disponível;
- **Cenário 7** - O usuário possui as conexões *Bluetooth* e *WiFi* e conectividade com Rede Móveis disponível;

A Tabela 6.15 apresenta, de forma estruturada, os cenários propostos e os parâmetros utilizados para os modelos CTMCs (Figura 6.5 e Figura 6.6). Os parâmetros associados nas probabilidades de conexões foram obtidos com base no experimento descrito na Seção 5.1.2 e em estudos correlacionado ao tema, a citar MATOS et al. (2015); SILVA et al. (2014); OLIVEIRA et al. (2013).

Tabela 6.15 Cenários propostos e parâmetros de entrada para os CTMCs.

Cenário	Conexões Ativas	***Smartwatch***		***Smartphone***			
		P_λ	P_β	P_λ	P_δ	P_β	P_ω
1	*Bluetooth*, Banda Larga e Rede Móvel	1	0	0.7	0	0.3	0
2	*Bluetooth* e Rede Móvel	1	0	0	0	1	0
3	*WiFi*, Banda Larga e Rede Móvel	0	1	0	0.3	0	0.7
4	*WiFi* e Banda Larga	0	1	0	0	0	1
5	*WiFi* e Rede Móvel	0	1	0	1	0	0
6	*Bluetooth*, *WiFi* e Banda Larga	0.7	0.3	0.7	0	0	0.3
7	*Bluetooth*, *WiFi* e Rede Móveis	0.7	0.3	0	0.3	0.7	0

Com base nos tempos de descarga obtidos, estes novos valores são atribuídos a fórmula fechada gerada pelo modelo da Arquitetura *Cloudlet* apresentado na Figura 6.8. Assim, a disponibilidade do serviço, do ponto de vista do cliente, é calculada, sendo possível investigar o impacto na disponibilidade do serviço. Os cenários 1, 2, 3, 4, 5, 6 e 7, têm a sua disponibilidade calculada através das adaptações da Equação 6.14, representadas pelas Equações 6.17, 6.18, 6.19, 6.20, 6.21, 6.22 e 6.23, respectivamente.

$$A = A_{SW} \times A_{BT} \times A_{SP} \times (A_{CL} \times A_{BL} + A_{f_{MC}} \times (A_{RM} - (-1 + A_{CL} + A_{RM}) \times A_{BL})) \quad (6.17)$$

$$A = A_{SW} \times A_{BT} \times A_{SP} \times A_{RM} \times A_{MC} \quad (6.18)$$

$$A = A_{SW} \times A_{WF} \times A_{SP} \times (A_{CL} \times A_{BL} + A_{f_{MC}} \times (A_{RM} - (-1 + A_{CL} + A_{RM}) \times A_{BL})) \quad (6.19)$$

$$A = A_{SW} \times A_{WF} \times A_{SP} \times A_{BL} \times (1 - (1 - A_{CL}) \times (1 - A_{MC})) \quad (6.20)$$

$$A = A_{SW} \times A_{WF} \times A_{SP} \times A_{RM} \times A_{MC} \quad (6.21)$$

$$A = A_{SW} \times (1 - (1 - A_{BT}) \times (1 - A_{WF})) \times A_{SP} \times A_{BL} \times (1 - (1 - A_{CL}) \times (1 - A_{MC})) \quad (6.22)$$

$$A = A_{SW} \times (1 - (1 - A_{BT}) \times (1 - A_{WF})) \times A_{SP} \times A_{RM} \times A_{MC} \quad (6.23)$$

Onde:
A_{SW} representa a disponibilidade do *Smartwatch*;
A_{BT} representa a disponibilidade do *Bluetooh*;
A_{WF} representa a disponibilidade da *WiFi*;
A_{SP} representa a disponibilidade do *Smartphone*;
A_{BL} representa a disponibilidade da Banda Larga;
A_{RM} representa a disponibilidade da Rede Móvel;
A_{CL} representa a disponibilidade da *Cloudlet*;
$A_{f_{MC}}$ representa a disponibilidade da Nuvem refatorada;
A_{MC} representa a disponibilidade da Nuvem.

A Tabela 6.16 apresenta os resultados da análise de disponibilidade obtidos de todos os cenários.

Tabela 6.16 Resultado da disponibilidade de todos cenários.

Cenário	Conexões Ativas	Disponibilidade (%)	Downtime(h)
1	*Bluetooth*, Banda Larga e Rede Móvel	98,932%	93.53
3	*WiFi*, Banda Larga e Rede Móvel	98,815%	103.77
6	*Bluetooth*, *WiFi* e Banda Larga	97,660%	205.01
4	*WiFi* e Banda Larga	97,571%	212.80
2	*Bluetooth* e Rede Móvel	96,645%	293.89
7	*Bluetooth*, *WiFi* e Rede Móveis	96,616%	296.44
5	*WiFi* e Rede Móvel	96,534%	303.60

Nesta análise, o Cenário 5, com apenas *WiFi* e Rede Móvel, apresentou o pior resultado, com 96,53% de disponibilidade, totalizando 12.6 dias de inatividade anual. Esse resultado

pode ser explicado pela ausência de redundância de conexão e da *Cloudlet*. No entanto, a disponibilidade aumenta significativamente no Cenário 1, quando as conexões *Bluetooth*, Banda Larga e Rede Móvel estão disponíveis. Obtivemos uma disponibilidade de $98,93\%$ e apenas 3.8 dias de inatividade anual. Desta forma, o *downtime* diminui de 303.6h para 93.5h, ou seja, $69,19\%$ menos tempo de indisponibilidade anual, quando comparado ao Cenário 5.

Contudo, vale destacar que o Cenário 3 apresentou resultados promissores, possuindo uma diferença relativamente pequena, $0,11\%$ a menos, comparado com a disponibilidade do melhor cenário. Deste modo, o Cenário 3 deve ser considerado como uma alternativa, principalmente pelo fato da rede *WiFi* (Cenário 3) possuir uma área de cobertura maior que o *Bluetooth* (Cenário 1).

O gráfico ilustrado na Figura 6.14 apresenta o tempo médio de descarga da bateria, estimado através do modelo CTMC (Figura 6.5 e Figura 6.6), considerando os sete cenários com probabilidades de conexão distintas.

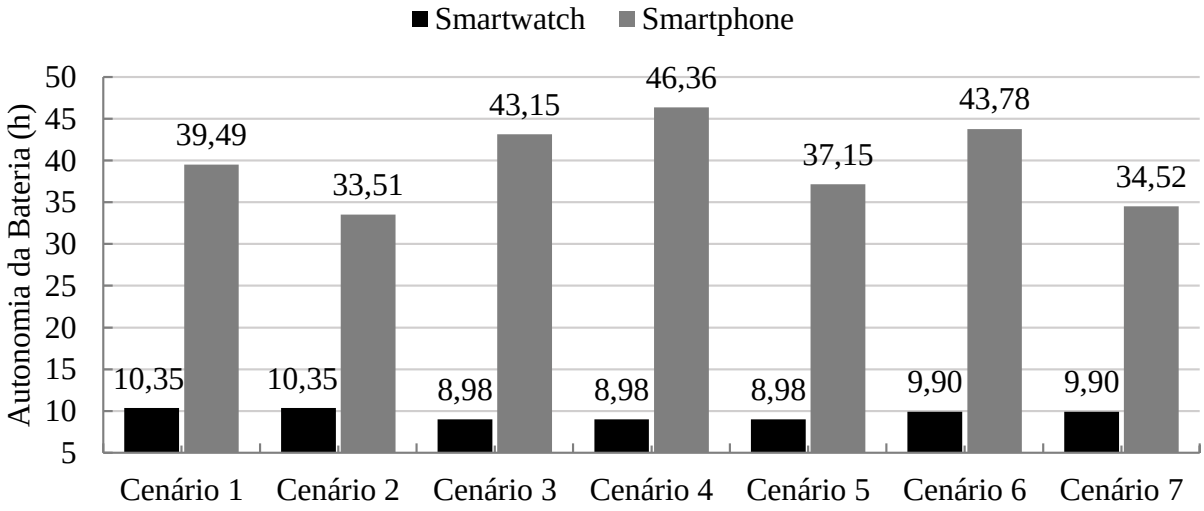


Figura 6.14 Tempo médio da autonomia da bateria.

Através da variação dos parâmetros do modelo CTMC do *smartwatch* (Figura 6.5), foi possível observar um impacto significativo na autonomia da bateria. Quando ambas as interfaces de conexão estão ativas (*Bluetooth* e *WiFi*), o tempo médio de descarga é de 9.9h (Cenários 6 e 7). A autonomia da bateria apresenta o melhor tempo ao utilizar a conexão *Bluetooth* (Cenários 1 e 2), com um tempo médio de descarga de 10.35h, uma diferença de 1.36h a mais ao utilizar a conexão *WiFi* (Cenários 3, 4 e 5).

O *smartphone*, teve seu tempo estimado através do modelo CTMC (Figura 6.6), considerando também os mesmos cenários. Os resultados mostram que, com as conexões *Bluetooth* e Rede Móvel disponíveis (Cenário 2), possui um tempo médio de descarga de 33.5h, o menor tempo de vida da bateria. Quando as conexões *WiFi* e Banda Larga estão disponíveis (Cenário 4), o tempo médio de vida da bateria é maior, com uma diferença de 12.8h a mais, se comparado ao Cenário 2.

Já o Cenário 3, com *WiFi*, Banda Larga e Rede Móvel, apresentou o segundo melhor tempo de vida da bateria, com apenas 3.2h em relação ao cenário que apresentou o melhor tempo de autonomia (Cenário 4).

Portanto, neste estudo de caso, avaliamos a autonomia da bateria dos dispositivos móveis, levando em consideração o impacto que a descarga da bateria pode causar sobre a disponibilidade e confiabilidade dos dispositivos móveis. O estudo proporcionou resultados de 9.545505 horas para MTTF e 8.181543×10^{-2} horas para MTTR do dispositivo *smartwatch*; e 35.348173 horas para MTTF e 7.760621×10^{-2} horas para MTTR do dispositivo *smartphone*, envolvendo todos os componentes abordados no modelo RBD e as diferentes interfaces de conexão.

Identificamos o componente Bateria, como o componente que possui a maior importância no aspecto da disponibilidade com 99,9% em ambos os dispositivos. Além disso, foi possível investigar os efeitos causados na confiabilidade dos dispositivos no período de 48h, onde o tempo de operação é impactado, resultando em valores cada vez mais baixos no decorrer do tempo.

Por fim, investigamos o impacto do uso de diferentes interfaces de rede sobre a disponibilidade do serviço e da autonomia da bateria, onde o cenário com a Rede Móvel, Banda Larga e *Bluetooth* disponíveis, apresentou melhor disponibilidade.

Levando em consideração os aspectos da autonomia da bateria, o cenário com a probabilidade de não haver conexão com *WiFi* apresentou melhor autonomia no *smartwatch*, já o *smartphone*, teve sua melhor autonomia no cenário com apenas a conexão *WiFi* e Banda Larga disponíveis.

Os resultados fornecem indícios de que a utilização de diferentes interfaces de rede promove um aumento da disponibilidade, em comparação com a Arquitetura Base, que não apresenta redundância de conexão. Contudo, a fim de mitigar o impacto causado pelas interfaces de conexão sob o tempo de vida da bateria dos dispositivos, pode-se inferir que o usuário pode priorizar o uso das conexões *Bluetooth* e Banda Larga, nos dispositivos *smartwatch* e *smartphone*, respectivamente, pois estes causaram um impacto menor sob a autonomia da bateria. Dessa forma, nossos resultados podem auxiliar nos estudos e elaborar técnicas que visem ampliar a disponibilidade de *wearables* e dispositivos móveis, dentro do contexto IoT e computação em nuvem.

6.5 Considerações Finais

Neste capítulo foram apresentados quatro estudos de casos que abordaram aspectos importantes relacionados a disponibilidade de ambientes IoT e computação em nuvem, que utiliza dispositivos *wearables* e dispositivos móveis. Os resultados obtidos fornecem um arcabouço para o planejamento de ambientes MCC, que utilizem os mesmos recursos da arquitetura proposta, relacionando os principais fatores que afetam sua disponibilidade. Além de proporcionar uma maior compreensão dos fatores que provocam impactos negativos e positivos sobre a disponibilidade de tais ambientes.

O primeiro estudo de caso desse capítulo refere-se à análise de disponibilidade da Arquitetura Base. Modelos hierárquicos, como RBD e CTMC foram utilizados para o cálculo das métricas de dependabilidade (disponibilidade e indisponibilidade anual). Os resultados sugerem que a disponibilidade pode ser aumentada com a adoção de mecanismos de redundância, visto que essa Arquitetura Base não apresenta redundâncias.

Para identificar gargalos de disponibilidade e conduzir a otimização dos processos, aumentando a disponibilidade do serviço, aplicamos a técnica de análise de sensibilidade paramétrica. Através dessa análise, as taxas de falha e reparo do componente Banda Larga, foram identificadas como o de maior criticidade, sendo os componentes *smartwatch* e Nuvem, como o segundo e terceiro, respectivamente.

Com base na classificação da análise de sensibilidade, foram criadas duas arquiteturas candidatas a Arquitetura Base, como solução para aumentar e otimizar a disponibilidade do serviço. A primeira arquitetura estendida se concentra em adicionar redundância na interface de conexão, enquanto a segunda, adiciona redundância na Nuvem, incluindo a *Cloudlet*. Assim, foi possível obter melhores resultados aplicando a redundância nos componentes de maior criticidade. Estes resultados são apresentados na Seção 6.3.3.

O último estudo de caso, possui três finalidades. Inicialmente apresentamos uma análise através dos modelos RBDs de cada dispositivo, onde foi possível analisar o índice de importância de componentes sobre disponibilidade dos dispositivos móveis, sendo seu componente mais relevante a Bateria. Em seguida, uma análise de confiabilidade foi realizada para verificar o tempo de operação estável dos dispositivos no período de $48h$. Por fim, investigamos o impacto do uso de diferentes interfaces de conexão sobre a autonomia da bateria e a disponibilidade do serviço em sete cenários, com distintas ofertas de conectividade de rede. Desta forma, os resultados dessa análise podem auxiliar nos estudos e elaborar técnicas que visem ampliar a disponibilidade dos componentes.

7

Conclusão

Por intermédio dos avanços tecnológicos dos últimos anos juntamente com a popularização de dispositivos móveis e *wearables*, a sociedade está cada vez mais conectada, abrindo caminho para a explosão da IoT. Esse conceito permite o compartilhamento de dados entre máquinas computacionais e objetos do cotidiano. Uma das áreas inseridas no âmbito de internet das coisas é a *mHealth*.

Contudo, as restrições deste paradigma computacional, como a instabilidade das conexões sem fio e limitações da bateria dos dispositivos, podem levar a indisponibilidade do serviço fornecido. Projetistas e administradores de nuvem, precisam projetar e gerenciar seus sistemas, de forma a manter o serviço altamente disponível.

Tendo em vista o desafio para atingir um nível satisfatório de disponibilidade para um serviço *mHealth* na nuvem, em um ambiente IoT, este estudo apresentou que a modelagem hierárquica é importante e eficaz, para lidar com tarefas não triviais - principalmente na área da saúde móvel. Desta forma, esse trabalho realizou um estudo de avaliação de disponibilidade de um ambiente IoT para o provimento do serviço *mHealth* na nuvem, com o objetivo de alcançar melhorias na disponibilidade, bem como, auxiliar o planejamento de infraestruturas dentro do contexto estudado. Investigamos três diferentes arquiteturas, a fim de verificar a viabilidade de cada uma, em relação à disponibilidade. Para isso, a representação dessas arquiteturas foi realizada através de modelagem hierárquica com RBD e CTMC, que foi validada através de um experimento de injeção de falhas.

Inicialmente, investigamos a primeira arquitetura definida como Arquitetura Base, sem redundância que teve como principal objetivo modelar e validar o serviço proposto. Os resultados da avaliação da Arquitetura Base apresentaram uma disponibilidade de 97,30% e um *downtime* anual de 235.74h. É importante ressaltar que os parâmetros dos modelos CTMCs, foram coletados com base no experimento para autonomia da bateria, com intuito de analisar o tempo de descarga da bateria de ambos dispositivos.

Através da Arquitetura Base, fez-se uma análise de sensibilidade paramétrica, na qual foi possível verificar os gargalos de disponibilidade e propor variações da arquitetura, com o objetivo de melhorar a disponibilidade do serviço. O resultado da análise de sensibilidade da

Arquitetura Base identificou que o componente Banda Larga é o componente de maior impacto sobre a disponibilidade, isso sugere que a implementação de redundância nesse componente pode melhorar a disponibilidade. Desta forma, duas extensões da Arquitetura Base foram criadas, adicionando componentes de redundância com base no *ranking* da análise de sensibilidade. A primeira arquitetura estendida explorou a redundância nas conexões de rede, enquanto a segunda aplicou o uso de *Cloudlet*, utilizando também a redundância de conexão de rede. A Arquitetura *Cloudlet* apresentou os melhores resultados em termos de disponibilidade e tempo de inatividade anual, atingindo 98,90% de disponibilidade e 96.0h de inatividade.

Em seguida, aprofundamos a investigação focando os dispositivos móveis. Identificamos que o componente Bateria, é o componente que possui a maior importância no aspecto da disponibilidade, com 99,9% em ambos os dispositivos. Nossos resultados também demostraram o efeito causado na confiabilidade dos dispositivos, onde o tempo de operação é impactado, resultando em valores cada vez mais baixos no decorrer do tempo. Por fim, investigamos os efeitos do uso de diferentes interfaces de conexão sobre a autonomia das baterias e sua relação com a disponibilidade do serviço. Considerando os distintos cenários de utilização, os resultados apresentaram que a disponibilidade atinge níveis maiores quando as conexões Rede Móvel, Banda Larga e *Bluetooth* estão disponíveis (Seção 6.4, Cenário 1). Detectamos que o tempo médio de operação da bateria do dispositivo *smartwatch* apresentou melhor resultado utilizando a conexão *Bluetooth*, em comparação com *WiFi*, já o *smartphone* teve sua melhor autonomia no cenário com apenas as conexões WiFi e Banda Larga disponíveis para realizar a comunicação com o *smartwatch* e a nuvem, respectivamente.

Portanto, a abordagem utilizada e os resultados obtidos nessa pesquisa contribuem com um arcabouço para o planejamento de futuras soluções *mHealth* na nuvem em ambientes IoT que utilizem recursos semelhantes aos analisados neste trabalho.

7.1 Contribuições

As principais contribuições deste estudo são:

- Proposição de modelos, através de uma abordagem utilizando modelagem hierárquica RBD e CTMC. Estes modelos representam uma infraestrutura de serviços *mHealth* em um ambiente MCC, utilizando um *smartwatch* como principal mecanismo de coleta de dados de saúde do usuário. Através destes modelos, foi possível analisar métricas de dependabilidade deste serviço como disponibilidade e *donwtime* anual.

- Desenvolvemos um experimento para avaliar a autonomia da bateria dos dispositivos *smartwatch* e *smartphone*. No experimento realizado foi observada a descarga da bateria e a relevância das interfaces de conexão sobre a autonomia da bateria dos dispositivos, além de proporcionar os valores necessários para a parametrização do

modelo CTMC. Os recursos desenvolvidos para automatizar este experimento podem ser reutilizados em trabalhos futuros.

- O modelo proposto, ajuda a analisar como a disponibilidade do sistema pode ser melhorada, por meio da análise de sensibilidade paramétrica. Sendo possível verificar os gargalos de disponibilidade e propor variações da Arquitetura Base com o objetivo de melhorar a disponibilidade do serviço. Além disso, o resultado da análise de sensibilidade pode ser utilizado para guiar outros trabalhos que analisem a mesma infraestrutura.

- Foram propostas duas extensões da Arquitetura Base como solução para aumentar e otimizar a disponibilidade do serviço. Foi possível obter melhores resultados aplicando a redundância nos componentes de maior criticidade. Como também, foi possível avaliar o impacto dessa redundância no aumento da disponibilidade do sistema, onde a Arquitetura *Cloudlet* proporcionou um melhor nível de disponibilidade. É notório que a avaliação das arquiteturas estendidas, sem a utilização dos modelos, seria uma tarefa complexa e dispendiosa. As arquiteturas e seus respectivos modelos que as representam, podem ser utilizados em extensões de arquiteturas, dentro do mesmo ambiente analisado, em trabalhos futuros.

- Uma avaliação onde foi possível observar o índice de importância dos componentes dos dispositivos móveis, através da avaliação do modelo RBD que representa cada dispositivo móvel. Sendo possível destacar quais os componentes que apresentaram maior impacto na disponibilidade. A avaliação também proporcionou uma análise de confiabilidade dos dispositivos móveis, onde foi possível observar que, ao decorrer do tempo de utilização dos dispositivos, a sua probabilidade de prover o serviço é degradada, resultando em valores cada vez mais baixos. Por fim, nossos resultados apresentaram os efeitos do uso de diferentes interfaces de conexão sobre a autonomia da bateria e sua relação com a disponibilidade do serviço. Além disso, os resultados obtidos nessa avaliação podem nortear novas pesquisas na elaboração de técnicas que visem ampliar a disponibilidade dos componentes.

Desta forma, os objetivos descritos na Seção 1.2 foram alcançados e além das contribuições mencionadas acima, foi publicado um artigo com base nos resultados dessa pesquisa:

- C.Araujo, F.Silva, I.Costa, F.Vaz, S.Kosta e P. Maciel. ***Supporting availability evaluation in MCC-based mHealth planning***. Journal IET Electronics Letters. 2016. (**Qualis A1**).

Outros artigos ainda estão em fase de submissão, contemplando outros aspectos e resultados desta pesquisa.

7.2 Trabalhos Futuros

Nesta seção, listamos algumas questões que podem ser investigadas em trabalhos futuros:

- Analisar outros atributos de dependabilidade no mesmo ambiente proposto, tais como: Segurança, Confidencialidade, Integridade e Manutenabilidade;
- Realizar estudos de desempenho e performance nas arquiteturas propostas. A análise de desempenho, como por exemplo, poderá avaliar em diferentes cenários aquele que melhor apresenta a relação entre o consumo energético, a disponibilidade e o desempenho;
- Avaliar a disponibilidade das interfaces de rede dos dispositivos móveis, levando em consideração fatores que podem intervir na conectividade, como o tempo para conexão, a distância geográfica, os fatores climáticos, a mobilidade, entre outros. Dado a complexidade de estados que as interfaces de rede podem assumir, para esta avaliação, a proposição de modelos baseados em estado, como o *Stochastic Petri Net* (SPN), pode ser adotado para representar o comportamento de conectividade dos dispositivos entre as possíveis redes de comunicação de dados neste ambiente;
- Investigar mecanismos que possam melhorar a disponibilidade do componente *smartwatch*. Bem como, avaliar alternativas que possam otimizar a eficiência energética dos dispositivo;
- Aplicar análise de sensibilidade nas variações das arquiteturas alcançadas, para identificar gargalos de disponibilidade do sistema, além disso, utilizar modelos para quantificação dos custos e benefícios de cada arquitetura, considerando aspectos de desempenho e dependabilidade.

Referências

ABDALI-MOHAMMADI, F.; BAJALAN, V.; FATHI, A. Toward a Fault Tolerant Architecture for Vital Medical-Based Wearable Computing. **Journal of medical systems**, [S.l.], v.39, n.12, p.1–12, 2015.

ADIBI, S. **Mobile health**: a technology road map. [S.l.]: Springer, 2015. v.5.

AL-FUQAHA, A. et al. Internet of things: a survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys & Tutorials**, [S.l.], v.17, n.4, p.2347–2376, 2015.

ARAUJO, J. et al. Dependability evaluation of a mhealth system using a mobile cloud infrastructure. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), 2014. **Anais...** [S.l.: s.n.], 2014. p.1348–1353.

ARLAT, J. et al. Fault injection and dependability evaluation of fault-tolerant systems. **IEEE Transactions on Computers**, [S.l.], v.42, n.8, Aug 1993.

ARMBRUST, M. et al. A view of cloud computing. **Communications of the ACM**, [S.l.], v.53, n.4, p.50–58, 2010.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: a survey. **Computer Networks**, [S.l.], v.54, n.15, p.2787 – 2805, 2010.

AVIZIENIS, A.; LAPRIE, J.; RANDELL, B. **Fundamental concepts of dependability**. [S.l.]: Univ. of Newcastle upon Tyne, 2001.

BAIER, C. et al. Model-checking algorithms for continuous-time Markov chains. **IEEE Transactions on software engineering**, [S.l.], v.29, n.6, p.524–541, 2003.

BALBO, G. Introduction to stochastic Petri nets. In: **Lectures on Formal Methods and PerformanceAnalysis**. [S.l.]: Springer, 2001. p.84–155.

BEZERRA, M. C. et al. Availability modeling and analysis of a vod service for eucalyptus platform. In: SYSTEMS, MAN AND CYBERNETICS (SMC), 2014 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.3779–3784.

BLAKE, J. T.; REIBMAN, A. L.; TRIVEDI, K. S. Sensitivity analysis of reliability and performability measures for multiprocessor systems. In: ACM SIGMETRICS PERFORMANCE EVALUATION REVIEW. **Anais...** [S.l.: s.n.], 1988. v.16, n.1, p.177–186.

BOLCH, G. et al. **Queueing networks and Markov chains**: modeling and performance evaluation with computer science applications. [S.l.]: John Wiley & Sons, 2006.

CARROZZA, G. et al. Dependability Evaluation and Modeling of the Bluetooth Data Communication Channel. In: PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING. **Anais...** [S.l.: s.n.], 2008. p.245–252.

CASILARI, E.; OVIEDO-JIMÉNEZ, M. A. Automatic fall detection system based on the combined use of a smartphone and a smartwatch. **PloS one**, [S.l.], v.10, n.11, p.e0140929, 2015.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to discrete event systems**. [S.l.]: Springer Science & Business Media, 2009.

CHEN, D. et al. Dependability Enhancement for IEEE 802.11 Wireless LAN with Redundancy Techniques. In: DSN. **Anais...** [S.l.: s.n.], 2003. p.521–528.

CHEN, J.-J. et al. A wearable virtual coach for Marathon beginners. In: PARALLEL AND DISTRIBUTED SYSTEMS (ICPADS), 2014 20TH IEEE INT. CONF. ON. **Anais...** [S.l.: s.n.], 2014. p.915–920.

CHERNICK, M. R. **Bootstrap methods**: a guide for practitioners and researchers. [S.l.]: John Wiley & Sons, 2011. v.619.

COOPER, T.; FARRELL, R. Value-Chain Engineering of a Tower-Top Cellular Base Station System. In: VEHICULAR TECHNOLOGY CONFERENCE (IEEE VTC). **Anais...** [S.l.: s.n.], 2007. p.3184–3188.

COSTA, I. et al. Availability Evaluation and Sensitivity Analysis of a Mobile Backend-as-a-service Platform. **Quality and Reliability Engineering International**, [S.l.], 2015.

D-LINK. **D-link Wireless n150 Router**. <http://www.dlink.com/us/en/home-solutions/connect/routers/dir-601-wireless-n-150-home-router>.

DANTAS, J. et al. An availability model for eucalyptus platform: an analysis of warm-standy replication mechanism. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), 2012. **Anais...** [S.l.: s.n.], 2012. p.1664–1669.

DE MELO, R. M. et al. Redundant VoD streaming service in a private cloud: availability modeling and sensitivity analysis. **Mathematical Problems in Engineering**, [S.l.], v.2014, 2014.

DINH, H. T. et al. A survey of mobile cloud computing: architecture, applications, and approaches. **Wireless communications and mobile computing**, [S.l.], v.13, n.18, p.1587–1611, 2013.

DOUKAS, C.; PLIAKAS, T.; MAGLOGIANNIS, I. Mobile healthcare information management utilizing Cloud Computing and Android OS. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY, 2010. **Anais...** [S.l.: s.n.], 2010. p.1037–1040.

EFRON, B.; TIBSHIRANI, R. J. **An introduction to the bootstrap**. [S.l.]: CRC press, 1994.

ESARY, J.; PROSCHAN, F. A reliability bound for systems of maintained, interdependent components. **Journal of the American Statistical Association**, [S.l.], v.65, n.329, p.329–338, 1970.

FRANK, P.; ESLAMI, M. Introduction to system sensitivity theory. **IEEE Transactions on Systems, Man, and Cybernetics**, [S.l.], v.10, n.6, p.337–338, 1980.

GARG, S. et al. Analysis of software rejuvenation using Markov regenerative stochastic Petri net. In: SOFTWARE RELIABILITY ENGINEERING, 1995. PROCEEDINGS., SIXTH INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 1995. p.180–187.

GATZOULIS, L.; IAKOVIDIS, I. Wearable and portable eHealth systems. **IEEE Engineering in Medicine and Biology Magazine**, [S.l.], v.26, n.5, p.51–56, 2007.

HAMBY, D. A review of techniques for parameter sensitivity analysis of environmental models. **Environmental monitoring and assessment**, [S.l.], v.32, n.2, p.135–154, 1994.

HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. **Computer**, [S.l.], v.30, n.4, p.75–82, Apr 1997.

HUANG, H.-P.; HSU, L.-P. Development of a wearable biomedical health-care system. In: INTELLIGENT ROBOTS AND SYSTEMS. **Anais...** [S.l.: s.n.], 2005.

JAIN, R. **The art of computer systems performance analysis**: techniques for experimental design, measurement, simulation, and modeling. [S.l.]: John Wiley & Sons, 1990.

KEESEE, W. **A Method of Determining a Confidence Interval for Availability**. [S.l.]: DTIC Document, 1965.

KIM, D. S.; MACHIDA, F.; TRIVEDI, K. S. Availability modeling and analysis of a virtualized system. In: DEPENDABLE COMPUTING, 2009. PRDC'09. 15TH IEEE PACIFIC RIM INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2009. p.365–371.

KOLMOGOROFF, A. Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung. **Mathematische Annalen**, [S.l.], v.104, p.415–458, 1931.

KOOLI, M.; NATALE, G. D. A survey on simulation-based fault injection tools for complex systems. In: DESIGN TECHNOLOGY OF INTEGRATED SYSTEMS IN NANOSCALE ERA (DTIS), 2014 9TH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.1–6.

KUMAR, K.; LU, Y.-H. Cloud Computing for Mobile Users: can offloading computation save energy? **Computer**, [S.l.], p.51–56, 2010.

KUO, W.; ZUO, M. J. **Optimal reliability modeling**: principles and applications. [S.l.]: John Wiley & Sons, 2003.

LAPRIE, J.-C. Dependable computing and fault-tolerance. **Digest of Papers FTCS-15**, [S.l.], p.2–11, 1985.

LAPRIE, J.-C. Dependability: basic concepts and terminology. In: **Dependability**: basic concepts and terminology. [S.l.]: Springer, 1992. p.3–245.

LYMBERIS, A.; DITTMAR, A. Advanced Wearable Health Systems and Applications - Research and Development Efforts in the European Union. **Engineering in Medicine and Biology Magazine, IEEE**, [S.l.], 2007.

LYON; FYODOR, G. **Nmap Network Scanning**: the official nmap project guide to network discovery and security scanning. [S.l.]: USA: Insecure, 2009.

MA, H.-D. Internet of Things: objectives and scientific challenges. **Journal of Computer Science and Technology**, [S.l.], v.26, n.6, p.919–924, 2011.

MACIEL, P. et al. Performance and dependability in service computing: concepts, techniques and research directions, ser. **Premier Reference Source. Igi Global**, [S.l.], 2011.

MACIEL, P. R. et al. Dependability modeling. **Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions**, [S.l.], p.53–97, 2012.

MAGLOGIANNIS, I. et al. Mobile reminder system for furthering patient adherence utilizing commodity smartwatch and Android devices. In: WIRELESS MOBILE COMMUNICATION AND HEALTHCARE (MOBIHEALTH), 2014 EAI 4TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.124–127.

MAGNO, C. **Avaliação da disponibilidade de video surveillance as service (VSAAS)**. 2015. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco - UFPE.

MALHOTRA, M.; TRIVEDI, K. S. Power-hierarchy of dependability-model types. **IEEE Transactions on Reliability**, [S.l.], v.43, n.3, p.493–502, 1994.

MARSAN, M. A. et al. **Modelling with generalized stochastic Petri nets**. [S.l.]: John Wiley & Sons, Inc., 1994.

MARWAH, M. et al. Quantifying the Sustainability Impact of Data Center Availability. **SIGMETRICS Perform. Eval. Rev.**, [S.l.], v.37, n.4, p.64–68, Mar. 2010.

MATOS, R. d. S. et al. Sensitivity analysis of server virtualized system availability. **IEEE Transactions on Reliability**, [S.l.], v.61, n.4, p.994–1006, 2012.

MATOS, R. et al. Sensitivity analysis of a hierarchical model of mobile cloud computing. **Simulation Modelling Practice and Theory**, [S.l.], v.50, p.151–164, 2015.

MELO, M. et al. Availability study on cloud computing environments: live migration as a rejuvenation mechanism. In: ANNUAL IEEE/IFIP INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS (DSN), 2013. **Anais...** [S.l.: s.n.], 2013. p.1–6.

MENASCE, D. A.; DOWDY, L. W.; ALMEIDA, V. A. F. **Performance by Design**: computer capacity planning by example. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

MILLER, V. T.; FISHWICK, P. A. Graphic modeling using heterogeneous hierarchical models. In: WINTER SIMULATION, 25. **Proceedings...** [S.l.: s.n.], 1993. p.612–617.

MODCS. **Obtenção da Fórmula Fechada**. http://www.modcs.org/wp-content/uploads/2015/01/Tutorial-Mathematica-F

MODCS. **Mercury Tool**. http://www.cin.ufpe.br/ bs/MercuryTool/mercury.html.

MOHAMED, O.; CHOI, H.-J.; IRAQI, Y. Fall detection systems for elderly care: a survey. In: NEW TECHNOLOGIES, MOBILITY AND SECURITY (IEEE NTMS 2014). **Anais...** [S.l.: s.n.], 2014. p.1–4.

MOLLOY, M. K. Performance analysis using stochastic Petri nets. **IEEE Transactions on computers**, [S.l.], v.100, n.9, p.913–917, 1982.

MORTAZAVI, B. J. et al. Determining the single best axis for exercise repetition recognition and counting on smartwatches. In: INTERNATIONAL CONFERENCE ON WEARABLE AND IMPLANTABLE BODY SENSOR NETWORKS, 2014. **Anais...** [S.l.: s.n.], 2014. p.33–38.

OLIVEIRA, D. et al. Availability and energy consumption analysis of mobile cloud environments. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 2013. **Anais...** [S.l.: s.n.], 2013. p.4086–4091.

PEIRAVI, A. Connectance and reliability computation of wireless body area networks using signal flow graphs. **Life Science**, [S.l.], p.52–56, 2010.

PENDERS, J. et al. Wearable Sensors for Healthier Pregnancies. **Proc. of the IEEE**, [S.l.], p.179–191, 2015.

PINOCHET, L.; LOPES, A.; SILVA, J. Inovações e Tendências Aplicadas nas Tecnologias de Informação e Comunicação na Gestão da Saúde. **Revista de Gestão em Sistemas de Saúde**, [S.l.], v.3, n.2, p.11–29, 2014.

PROJECT, R. **The R Project for Statistical Computing**. https://www.r-project.org/.

QI, H.; GANI, A. Research on mobile cloud computing: review, trend and perspectives. In: DIGITAL INFORMATION AND COMMUNICATION TECHNOLOGY AND IT'S APPLICATIONS (DICTAP), 2012 SECOND INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2012. p.195–202.

SAHNER, R. A.; TRIVEDI, K.; PULIAFITO, A. **Performance and reliability analysis of computer systems**: an example-based approach using the sharpe software package. [S.l.]: Springer Science & Business Media, 2012.

SALVI, P. **Telemedicina, e-Health e m-Health**: o que elas nos reservam? http://saudebusiness.com/noticias/telemedicina-saida-para-reducao-de-custos-e-melhoria-noatendimento/.

SAREEN, P. Cloud computing: types, architecture, applications, concerns, virtualization and role of it governance in cloud. **International Journal of Advanced Research in Computer Science and Software Engineering**, [S.l.], v.3, n.3, 2013.

SATYANARAYANAN, M. et al. The case for vm-based cloudlets in mobile computing. **IEEE pervasive Computing**, [S.l.], v.8, n.4, p.14–23, 2009.

SI, S.; LIU, F.; CAI, Z. Failure importance analysis models based on Bayesian network. In: INDUSTRIAL ENGINEERING AND ENGINEERING MANAGEMENT, 2009. IE&EM'09. 16TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2009. p.151–154.

SILVA, B. et al. Astro: an integrated environment for dependability and sustainability evaluation. **Sustainable Computing Informatics and Systems**, [S.l.], v.3, n.1, 2012.

SILVA, I. et al. A dependability evaluation tool for the Internet of Things. **Computers and Electrical Engineering**, [S.l.], v.39, n.7, p.2005 – 2018, 2013.

SILVA, V. C. O. da et al. Energy Consumption in Mobile Devices Considering Communication Protocols. **Advances in Information Sciences and Service Sciences**, [S.l.], v.6, n.5, p.1, 2014.

SOUSA, E. et al. Dependability evaluation of cloud infrastructures. In: SYSTEMS, MAN AND CYBERNETICS (SMC), 2014 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.1282–1287.

SOUZA, D. et al. A tool for automatic dependability test in eucalyptus cloud computing infrastructures. **Computer and Information Science**, [S.l.], v.6, n.3, p.57, 2013.

STANDART, N. I. o. T. **Cloud Computing**. http://www.nist.gov.

STROUD, F. **Smartwatch**. http://www.webopedia.com/TERM/S/ smartwatch.html.

SU, Y.; GURURAJAN, R. The determinants for adoption of wearable computer systems in traditional Chinese hospital. In: WEARABLE COMPUTING SYSTEMS (APWCS), 2010 ASIA-PACIFIC CONFERENCE ON. **Anais...** [S.l.: s.n.], 2010. p.375–378.

TRIVEDE, K. S. et al. Modeling High Availability. In: PACIFIC RIM INTERNATIONAL SYMPOSIUM ON DEPENDABLE COMPUTING (PRDC'06), 2006. **Anais...** [S.l.: s.n.], 2006. p.154–164.

TRIVEDI, K. S. **Probability and Statistics with Reliability, Queuing and Computer Science Applications**. 2nd edition.ed. Chichester, UK: John Wiley and Sons Ltd., 2002.

TRIVEDI, K. S. **Probability e statistics with reliability, queuing and computer science applications**. [S.l.]: John Wiley e Sons, 2008.

TRIVEDI, K. S. **Sharpe portal**. https://sharpe.pratt.duke.edu/.

VALMARI, A. The state explosion problem. In: **Lectures on Petri nets I**: basic models. [S.l.]: Springer, 1998. p.429–528.

WOLFRAM RESEARCH, I. **Wolfram Mathematica 10.0**. https://www.wolfram.com/mathematica/.

ZIADE, H. et al. A survey on fault injection techniques. **Int. Arab J. Inf. Technol.**, [S.l.], v.1, n.2, p.171–186, 2004.

ZOU, X. et al. Dependability and security in medical information system. In: INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION. **Anais...** [S.l.: s.n.], 2007. p.549–558.

Apêndice

A

Script Bootstrap

```
data = {$x_1$, $x_2$, $x_3$, ...,$x_n$}
D= EmpiricalDistribution[data];
estimated$D$ = FindDistribution[data];
alpha = 0.05;
numberofresamples = 1000;

MeanData = Table[Mean[RandomChoice[data, Length[data]]],{
   numberofresamples}];
Print["MeanData=", MeanData]
Print["MeanQuantile= ", Quantile[MeanData, {alpha/2, (1-alpha/2)}]]
```

B

Script Injetor

```
#!/bin/bash
#=== Parameters ==============
ipService=192.168.10.10
portService=8080
serviceName='tomcat7'

scriptR_Repair='expRepair.R'
scriptR_Fail='expFail.R'

UP='open'
DOWN='closed'

#=== Functions ===============
saveLog() {
    echo  $1
    date "+%d/%m/%Y - %T  : $1" >> injetor.log
}

getStatus() {
    status=$(nmap -p $portService $ipService | grep $portService/tcp
        | awk '{print $2}')  # open | closed
}

#=== Main ====================
saveLog "======================================"
saveLog "Projeto MoDCS mHealth Wearable - Camila"
saveLog "Iniciando o injetor"': '$serviceName

getStatus

while [ true ]
do
    if [ $status = $UP ]
    then
```

```
    TimeR=$(Rscript $scriptR_Fail | awk '{print $2}')
    saveLog "Aguardando tempo para injetar a falha: "''$TimeR
    sleep $TimeR

    saveLog "Iniciando falha..."
    sudo service $serviceName stop
    while [ $status = $UP ]
    do
      getStatus
    done

    # Falha finalizada (status = DOWN)
    saveLog $serviceName': '$status
  fi

  if [ $status = $DOWN ]
  then
    TimeR=$(Rscript $scriptR_Repair | awk '{print $2}')
    saveLog "Aguardando tempo para injetar o reparo: "''$TimeR
    sleep $TimeR

    saveLog "Iniciando reparo..."
    sudo service $serviceName start
    while [ $status = $DOWN ]
    do
      getStatus
    done

    # Reparo finalizado (status = UP)
    saveLog $serviceName': '$status
  fi
done
saveLog "====================================="
```