**UNIVERSIDADE FEDERAL DO CEARÁ**

**CENTRO DE TECNOLOGIA**

**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA**

**MESTRADO ACADÊMICO EM ENGENHARIA DE TELEINFORMÁTICA**

**JOARI SANTIAGO LIMA FILHO**

# UM MECANISMO DE *OFFLOADING* DE DADOS COM TOMADA DE DECISÃO

**FORTALEZA**

**2017**

JOARI SANTIAGO LIMA FILHO

# UM MECANISMO DE *OFFLOADING* DE DADOS COM TOMADA DE DECISÃO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Sinais e Sistemas

Orientador: Prof. Dr. Danielo Gonçalves Gomes

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

L698m Lima Filho, Joari Santiago.
Um Mecanismo de Offloading de Dados com Tomada de Decisão / Joari Santiago Lima Filho. – 2017.
63 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2017.
Orientação: Prof. Dr. Danielo Gonçalves Gomes.

1. Mobile Cloud Computing. 2. Offloading de Dados. I. Título.

CDD 621.38

---

JOARI SANTIAGO LIMA FILHO

UM MECANISMO DE *OFFLOADING* DE DADOS COM TOMADA DE DECISÃO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia de Teleinformática do Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia de Teleinformática. Área de Concentração: Sinais e Sistemas

Aprovada em:

BANCA EXAMINADORA

---

Prof. Dr. Danielo Gonçalves Gomes (Orientador)
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Paulo Antonio Leal Rego
Universidade Federal do Ceará (UFC)

---

Prof. Dr. André Luiz Lins de Aquino
Universidade Federal de Alagoas (UFAL)

Aos meus pais que são a base, a sustentação e a inspiração para trilhar meu caminho.

## AGRADECIMENTOS

Ao Prof. Dr. José Marques Soares por me receber, ajudar e me indicar ao professor Danielo.

Ao Prof. Dr. Danielo Gonçalves Gomes por me orientar em minha dissertação de mestrado e acreditar no projeto proposto.

Ao Prof. Dr. Lincoln Souza Rocha que me ajudou a compreender e delinear os problemas em aberto na proposta anterior a este trabalho.

Aos amigos de laboratório, Anderson Almada e Manuel Gonçalves pelas discussões sobre o trabalho proposto e pela paciência nos momentos de dificuldade.

Ao Prof. Dr. Paulo Antônio Leal Rego por me ajudar em minha dissertação de mestrado e propor a base do mecanismo utilizado neste trabalho.

Aos meus amigos Kauã Melo, Leandro Jader, Ricky Persivo e João Paulo Pitombeira que me apoiaram e, pacientemente, me aconselharam a seguir firme nessa jornada.

Aos meus pais, irmãos e sobrinhos, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente.

Aos amigos de graduação em Engenharia de Computação Danilo Leal e Vanessa Viana por incentivarem e esclarecerem dúvidas pertinentes ao mestrado acadêmico.

E à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) pelo financiamento da pesquisa de mestrado via bolsa de estudos.

"Ria e o mundo rirá com você. Chore e você chorará sozinho."

(Ella Wheeler Wilcox)

## RESUMO

De acordo com o IBGE (Instituto Brasileiro de Geografia e Estatística), 92,1% do acesso domiciliar à Internet passou a ser feito pelos telefones móveis celulares. Devido às limitações físicas, o poder de processamento desses dispositivos e o tempo de vida das baterias não têm acompanhado a exigência crescente dos aplicativos móveis. No paradigma de *mobile cloud computing*, as técnicas de *offloading* permitem a extensão das capacidades energética e computacional de dispositivos móveis, bem como a redução do tempo de execução de procedimentos. Nesta dissertação, propomos um mecanismo de *offloading de dados* que seleciona e migra arquivos para uma infraestrutura local (*cloudlet*) auxiliando os *frameworks* de *offloading de processamento* a reduzirem a quantidade de dados enviados pela rede. O mecanismo utiliza-se do histórico de execuções dos métodos dos aplicativos, assim como das condições da rede, para criar árvores de decisão que auxiliam na decisão de quando e quais arquivos utilizados por estes métodos devem ser transferidos. Os resultados dos experimentos indicam que a utilização do mecanismo proposto reduz o tempo do *offloading de processamento* em até 19,5%.



**Palavras-chave:** Mobile Cloud Computing, Offloading de Dados.

**ABSTRACT**

According to IBGE (Brazilian Institute of Geography and Statistics), 92.1% of home access to the Internet was made by mobile phones. Due to the physical limitations, the processing power of these devices and the life of the batteries have not matched the growing demand of mobile applications. In the mobile cloud computing paradigm, offloading techniques are used to augment computation and power capacities of mobile devices as well as to reduce the execution time of tasks. In this dissertation, we propose a data offloading mechanism that selects and migrates files to a local infrastructure (cloudlet), assisting computation offloading frameworks to reduce the amount of data sent over the network. The mechanism uses the application methods execution history, as well as the network condition, to create decision trees that help deciding when and which files used by these methods should be transferred. The experiments results indicate that our mechanism reduces the processing offloading time by up to 19.5%.

**Keywords:** Mobile Cloud Computing. Data Offloading.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| API | *Application Programming Interface* |
| BTS | *Base Transceiver Station* |
| CART | *Classification and Regression Trees* |
| COP | *Contextual data Offloading service with Privacy support* |
| ID3 | *Interactive Dichotomizer 3* |
| IP | *Internet Protocol* |
| MCC | *Mobile Cloud Computing* |
| MFLOPS | *Million Floating-Point Operations Per Second* |
| MpOS | *Multiplatform Offloading System* |
| NoSQL | *Not Only SQL* |
| RTT | *Round Trip Time* |
| SHA-1 | *Secure Hash Algorithm 1* |
| SVM | *Support Vector Machines* |
| VM | *Virtual Machine* |
| WLAN | *Wireless Local Network* |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| $D_d$ | Dados recebidos pelo dispositivo móvel |
| $D_u$ | Quantidade de dados enviados do dispositivo para o servidor |
| $P_m$ | Poder de processamento do dispositivo móvel |
| $P_{sr}$ | Poder de processamento do servidor remoto |
| $V_d$ | Vazão de *download* |
| $V_u$ | Vazão de *upload* |
| $W$ | Total de computação a ser realizada |

**SUMÁRIO**

# 1 INTRODUÇÃO

Este capítulo está organizado em quatro seções. A Seção 1.1 contextualiza o tema abordado, destacando a problemática em foco e alguns artigos da literatura correlata; a partir daí, estabelecemos nosso objetivo central na Seção 1.2; na sequência, a contribuição e o delineamento estratégico estão brevemente descritos na Seção 1.3; por fim, a Seção 1.4 informa a organização dos capítulos restantes.

## 1.1 Contextualização e Justificativa

Uma das características que contribuíram para a popularidade dos dispositivos móveis, em especial os *smartphones*, é o suporte a um grande número de aplicativos, tais como jogos, comércio eletrônico e redes sociais *on-line*. No entanto, apesar destes aplicativos estarem cada vez mais exigentes em termos de processamento, armazenamento e banda, as melhorias no *hardware* dos dispositivos móveis e na vida útil das suas baterias não têm acompanhado a crescente demanda por recursos computacionais (KHAN *et al.*, 2014). Diante desses obstáculos, uma das soluções em perspectiva é migrar o processamento e o armazenamento dos dados para dispositivos remotos ou nuvens computacionais (KEMP *et al.*, 2009).

Neste contexto, um dos benefícios providos pela computação móvel em nuvem (*Mobile Cloud Computing* (MCC)) é o uso das nuvens, as quais disponibilizam serviços para a expansão das capacidades físico-computacionais dos dispositivos móveis (SCHÜRING, 2011; DINH *et al.*, 2013). Através da MCC, os dispositivos móveis podem expandir seus recursos de processamento, memória, armazenamento e melhorar sua autonomia energética (KHAN *et al.*, 2014). A migração das tarefas de processamento de um dispositivo móvel para uma nuvem computacional remota ou para uma infraestrutura computacional de alcance local é conhecida por *offloading de processamento* ou *cyber foraging* (KUMAR *et al.*, 2013).

Para que o *offloading* economize energia ou reduza o tempo de execução das tarefas, de acordo com Kumar *et al.* (2013), é necessário avaliar as características instantâneas da rede em que o dispositivo móvel está inserido, o poder de processamento (*Million Floating-Point Operations Per Second* (MFLOPS)) dos dispositivos envolvidos na operação e a quantidade de dados em potencial de migração. Um sistema de *offloading* eficiente é aquele capaz de inferir onde uma tarefa computacional deve ser executada (se local ou remotamente) de modo que o dispositivo móvel economize seus recursos computacionais (FLORES *et al.*, 2015).

Nesse sentido, uma alternativa para diminuir a quantidade de dados transferidos durante o *offloading de processamento* é o seu emprego conjunto com mecanismos de *offloading de dados* (LEWIS; LAGO, 2015). Através desses mecanismos, os dispositivos móveis podem migrar seus dados para uma infraestrutura computacional local (*cloudlet*) que funciona como uma intermediária entre o dispositivo móvel e a nuvem remota. Entretanto, notamos que os atuais *frameworks* de *offloading de dados* normalmente não filtram quais dados devem ser migrados para uma *cloudlet*, i.e. apesar deles decidirem *quando* migrar os dados, não avaliam *o que* devem migrar (LEWIS; LAGO, 2015).

Em Lewis e Lago (2015), foi realizada uma revisão sistemática para identificar e classificar os trabalhos de *offloading de processamento* e de dados sob o ponto de vista arquitetural. A Tabela 1, retirada do trabalho de Lewis e Lago (2015), elenca características dos *frameworks* de *offloading de dados* quanto ao local para onde são enviados os dados (*cloudlet* desconectada da nuvem, *cloudlet* conectada à nuvem e a nuvem) e em relação a quando os dados são enviados. Podemos observar que, para os trabalhos elencados, não é feita a tomada de decisão de quais dados devem ser migrados (Tabela 1, Coluna "Decisão em Tempo de Execução"). Para esses *frameworks*, sempre é feito o *offloading* dos dados e a migração é total.

Tabela 1 – Algumas propostas de *offloading de dados*.

| **Referência** | **Onde** | | | **Quando** | |
|---|---|---|---|---|---|
| | Cloudlet Descon. | Cloudlet Conect. | Nuvem | Decisão em Tempo de Execução | Sempre Faz Offloading |
| (ARMSTRONG *et al.*, 2006) | | x | | | x |
| (BAHRAMI *et al.*, 2006) | x | | | | x |
| (FLINN *et al.*, 2003) | x | | | | x |
| (IYER; ROOPA, 2012) | | | x | | x |
| (KUNDU *et al.*, 2007) | x | | x | | x |
| (PHOKAS *et al.*, 2013) | x | x | x | | x |
| (XIAO *et al.*, 2013) | | x | | | x |
| (YANG *et al.*, 2012) | x | x | x | | x |

Fonte: (LEWIS; LAGO, 2015).

Na revisão sistemática de Lewis e Lago (2015), foi definida uma *string* de busca com três principais palavras-chave: cyber-foraging, dispositivos móveis e arquitetura de software. Cada uma dessas palavras-chave possui um conjunto de sinônimos alternativos. Com base nessas

palavras-chave e seus termos relacionados, a seguinte *string* de busca foi definida:

*(cyber foraging OR cyber-foraging OR code offload OR code offloading OR computation offload OR computation offloading OR data offload OR data staging) AND (mobile OR handheld OR smartphone) AND (software architecture OR software design OR system architecture)*

A *string* de busca foi executada em setembro de 2013 na base *Google Scholar*, a qual retornou 430 resultados[1]. Para esta dissertação, executamos a mesma *string* de busca em novembro de 2016, cujo retorno foi de 2.770 resultados, i.e. um aumento de mais de 540% com relação ao artigo de Lewis e Lago (2015). Apesar da realização de uma revisão sistemática estar fora do escopo desta dissertação, tal aumento indica interesse no tema e presença de problemas ainda em aberto.

É neste cenário que buscamos investigar se a utilização de um mecanismo capaz de selecionar e migrar dados estáticos (arquivos de imagem, vídeo e áudio) de dispositivos móveis para uma infraestrutura local (*cloudlet*) contribui na diminuição de tempo no processo de *offloading*.

## 1.2 Hipótese e Objetivo Central

A hipótese deste trabalho é a de que os *frameworks* de *offloading de processamento* podem executar mais rapidamente se auxiliados por um mecanismo de *offloading de dados* capaz de selecionar previamente os dados que devem ser migrados para uma *cloudlet*. A partir desta hipótese, podemos afirmar que o objetivo central desta dissertação é diminuir o tempo de execução de procedimentos executados remotamente (*cloudlet*).

## 1.3 Contribuição e Metodologia

A contribuição central deste trabalho é um mecanismo de *offloading de dados* com tomada de decisão que leva em consideração as condições momentâneas da rede e o histórico de execuções dos métodos de aplicativos para selecionar quais arquivos do dispositivo móvel devem ser migrados para uma *cloudlet*. O mecanismo utiliza árvores de decisão para a escolha de quais arquivos e de quando estes arquivos devem ser migrados. Os arquivos selecionados para a migração variam dependendo dos aplicativos utilizados e das diferentes condições de rede. Para a implementação e avaliação do mecanismo, delineamos três atividades:

[1] https://www.andrew.cmu.edu/user/gritter/slr-online-material.pdf

1. Investigar se a migração dos dados do dispositivo móvel para a *cloudlet* diminui o tempo de execução de procedimentos executados remotamente (na *cloudlet*);
2. Desenvolver um mecanismo de tomada de decisão baseado em árvores de decisão que seja capaz de selecionar e migrar arquivos entre o dispositivo móvel e a *cloudlet*;
3. Avaliar a seleção dos arquivos feita pelo mecanismo desenvolvido, analisando se a migração dos aquivos selecionados diminui o tempo gasto para realizar as operações de *offloading de processamento*.

## 1.4 Organização da Dissertação

Esta dissertação está organizada em cinco capítulos. O capítulo atual descreveu uma breve introdução ao tema, contextualizando e motivando os assuntos abordados nesta dissertação. Além disso, foi definido o objetivo e a contribuição, como também, a metodologia utilizada para desenvolver este trabalho.

No Capítulo 2, realizamos um levantamento de conceitos sobre árvores de decisão e sobre o tema de MCC com enfoque nas operações de *offloading*. Ainda nesse capítulo, são apresentados os trabalhos correlatos. No Capítulo 3, apresentamos o mecanismo proposto: sua arquitetura, a criação da árvore de decisão e a tomada de decisão. No Capítulo 4, é apresentada a etapa de projeto dos experimentos que visam validar e verificar a efetividade da proposta e os resultados alcançados. Por fim, no Capítulo 5, mostramos as conclusões e perspectivas futuras oriundas deste trabalho.

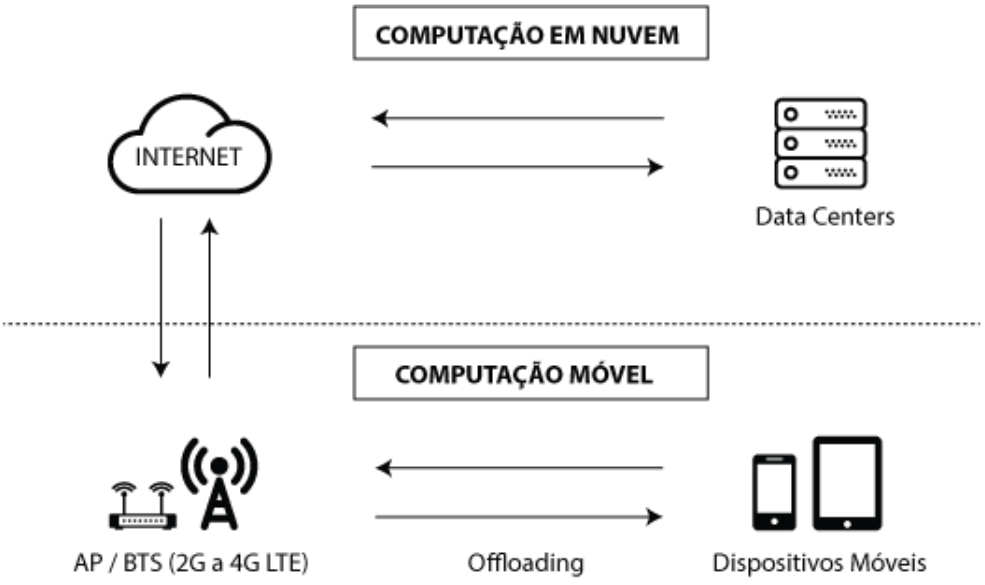
# 2 REVISÃO DA LITERATURA

Neste capítulo, apresentamos uma revisão da literatura sobre os temas abordados nesta dissertação. Iniciamos o capítulo com uma breve descrição dos conceitos de computação móvel em nuvem (MCC), *offloading* e árvores de decisão. Por conseguinte, descrevemos alguns trabalhos correlatos ao mecanismo proposto nesta dissertação.

## 2.1 Mobile Cloud Computing

A computação móvel em nuvem (MCC) integra os paradigmas da computação em nuvem e da computação móvel com o objetivo de mitigar problemas relacionados à interoperabilidade, desempenho, energia, segurança de aplicativos móveis e dos respectivos dispositivos que os hospedam e executam (GOMES *et al.*, 2015) (DINH *et al.*, 2013). A MCC pode ser considerada um modelo de computação distribuída que estende a visão de computação utilitária das nuvens computacionais para os dispositivos móveis, superando assim suas limitações de recursos (SHIRAZ *et al.*, 2015).

Figura 1 – Arquitetura Geral da Mobile Cloud Computing



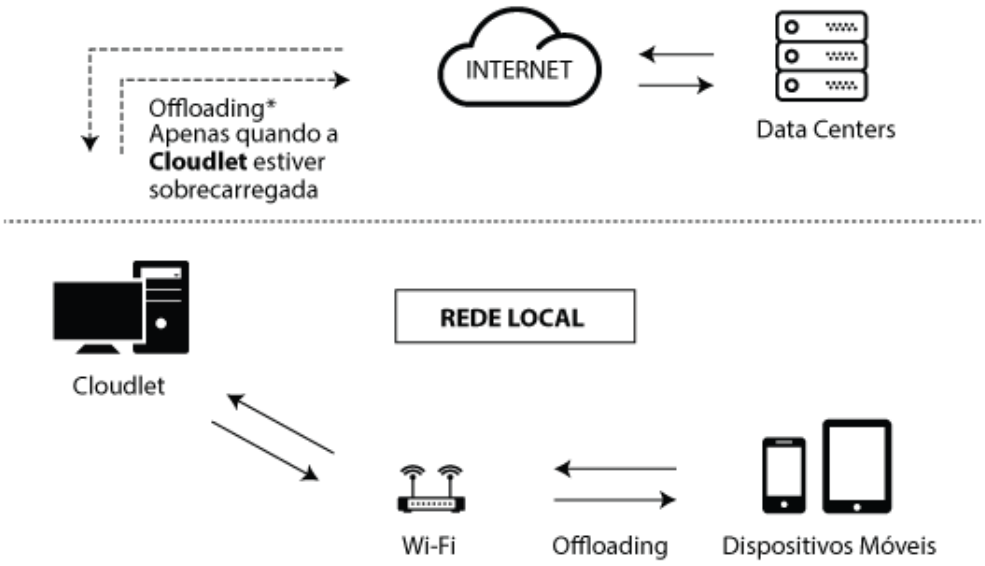

Fonte: (QI; GANI, 2012)

Uma das arquiteturas em perspectiva em MCC pode ser dividida em componentes de computação móvel e computação em nuvem (DINH *et al.*, 2013) (QI; GANI, 2012) (BAHL *et al.*, 2012), como é exibido na Figura 1. Essa arquitetura tem o seguinte funcionamento: os

usuários utilizam dispositivos móveis para interagirem com os serviços de *offloading* que são disponibilizados em uma nuvem. A comunicação entre os dispositivos móveis e a nuvem é feita por meio de conexão Wi-Fi (*Access Point*) ou via uma estação de rádio base (*Base Transceiver Station* (BTS)) que possibilita o uso de Internet móvel (2G a 4G LTE). Para problemas específicos, existem outros modelos arquiteturais como é mostrado a seguir.

No trabalho de Satyanarayanan *et al.* (2009) foi apresentado o modelo arquitetural *cloudlet* (Figura 2). O objetivo desse modelo é disponibilizar os serviços de *offloading* em máquinas que estão na rede local (Wi-Fi) do dispositivo móvel. Com esse modelo, os serviços são executadas em uma infraestrutura local (*cloudlet*) ao invés de utilizarem uma nuvem pública. A *cloudlet* pode ser composta de notebooks, servidores, *desktops* ou uma infraestrutura mais robusta, dependendo da utilização do serviço e da quantidade de usuários.

Figura 2 – Modelo Arquitetural *Cloudlet*




Fonte: (FERNANDO *et al.*, 2013)

As redes Wi-Fi, geralmente, possuem menor latência e velocidades maiores do que as redes de celulares (Internet móvel) por serem menos congestionadas. Ao utilizar a rede Wi-Fi, as *cloudlets* tendem a disponibilizar serviços de melhor qualidade se comparados com a arquitetura que faz o uso da infraestrutura de Internet (Figura 1). De acordo com Satyanarayanan *et al.* (2009), o modelo arquitetural *cloudlet* (Figura 2) poderia ter seus recursos computacionais estendidos para uma nuvem pública na Internet com o objetivo de atender mais usuários. No entanto, por questões de delimitação de escopo, não será considerada, nesta dissertação, a

possibilidade de estender os recursos da *cloudlet* para a nuvem. A arquitetura adotada nesta dissertação é explicada em detalhes na Seção 3.1 e conta apenas com um dispositivo móvel e uma *cloudlet*.

Os serviços de *offloading*, discutidos anteriormente, são tarefas de migração (total ou parcial) de elementos computacionalmente custosos para fora do dispositivo móvel (KUMAR *et al.*, 2013). Os *offloading de dados* e de processamento na computação em nuvem têm sido utilizados para mitigar problemas de limitação de recursos dos dispositivos móveis e são brevemente discutidos na subseção seguinte.

### *2.1.1 Offloading de Processamento e de Dados*

Dos vários temas de pesquisa existentes relacionados a MCC é possível destacar a técnica de *offloading* (FERNANDO *et al.*, 2013). Essa técnica visa estender o poder de processamento e aumentar a capacidade de armazenamento dos dispositivos móveis por meio da migração de tarefas computacionais ou de dados para outras infraestruturas que possuam maior poder computacional e armazenamento. Vale salientar que a técnica de *offloading* difere do modelo cliente-servidor tradicional pois, na execução do modelo cliente-servidor, o cliente sempre delega a responsabilidade de realizar o processamento ao servidor. Já no *offloading*, quando não há conexão entre o dispositivo móvel e servidor, o processamento é realizado localmente no dispositivo móvel. Quando existe a conexão e existem ganhos em se migrar dados ou processamento, o *offloading* é realizado.

Tipicamente, existem dois tipos principais de *offloading*: processamento e dados (FERNANDO *et al.*, 2013). O *offloading de processamento* é a entrega de um processamento computacional do dispositivo móvel para outro ambiente de execução (e.g., *desktop*), a fim de prolongar a vida útil da bateria e aumentar a capacidade computacional. O *offloading de dados* tem por objetivo estender a capacidade de armazenamento do dispositivo móvel. Assim, pode-se retirar o armazenamento do dispositivo e enviar os dados para um ambiente com maior capacidade de armazenamento, com a possibilidade também de enviar dados processados de volta para o dispositivo móvel (GOMES *et al.*, 2016).

Outra aplicação para o *offloading de dados* é a migração de dados para uma infraestrutura computacional local (*cloudlet*) a qual funciona como uma intermediária entre o dispositivo móvel e a nuvem remota. Com a migração para a *cloudlet*, pode-se diminuir a quantidade de dados que precisam ser retransmitidos pelo dispositivo móvel. De acordo com Lewis e Lago

(2015), tal tática aumenta os recursos computacionais provendo maior poder de armazenamento e um aumento da eficiência energética em virtude de se utilizar, por menos tempo, as conexões de rede para envio dos dados. Eles afirmam ainda que, neste tipo de *offloading*, há informações adicionais que mesmo não sendo utilizadas na tomada de decisão pelo *offloading*, são relevantes sob o ponto de vista global do sistema pois podem resultar em um aumento de complexidade no processo de migração, e.g., *(i)* o tipo dos dados que irão sofrer *offloading*; e *(ii)* as operações de cache que o sistema remoto poderá realizar sobre esses dados.

### *2.1.2 Quando fazer offloading?*

Kumar *et al.* (2013) propuseram um modelo analítico para responder à pergunta: "*Quando o offloading de processamento melhora o desempenho do dispositivo móvel?*". O referido modelo, apresentado a seguir de forma adaptada, compara o tempo de execução de uma tarefa no dispositivo móvel ($\frac{W}{P_m}$) com o tempo gasto para enviar os dados e executar o processamento em um servidor remoto ($\frac{D_u}{V_u} + \frac{W}{P_{sr}} + \frac{D_d}{V_d}$), em que $W$ é o total de computação a ser realizada; $P_m$ é o poder de processamento do dispositivo móvel; $P_{sr}$ é o poder de processamento do servidor remoto; $D_u$ é a quantidade de dados enviados do dispositivo para o servidor, e $D_d$ são os dados recebidos pelo dispositivo (i.e., resultado da computação); e $V_u$ e $V_d$ são as vazões de *upload* e *download*, respectivamente.

$$\frac{W}{P_m} > \frac{D_u}{V_u} + \frac{W}{P_{sr}} + \frac{D_d}{V_d}. \tag{2.1}$$

Quanto maior o valor de $W$ e menor o valor da soma ($\frac{D_u}{V_u} + \frac{D_d}{V_d}$), seja pela baixa quantidade de dados a transferir, seja por uma alta vazão, maior a probabilidade da inequação (2.1) ser satisfeita. Nesta perspectiva, o *offloading de dados* pode ser utilizado para reduzir o tempo de gasto com transferência de dados entre dispositivo móvel e servidor remoto, pois os dados já estariam localizados no servidor.

## 2.2 Árvores de Decisão

Classificação é o problema de identificar a qual classe, dado um conjunto destas, uma instância pertence. As técnicas de classificação usam abordagens sistemáticas para construir modelos de classificação a partir de um conjunto de instâncias (chamado conjunto de treinamento) e dentre tais técnicas destacam-se as árvores de decisão, as redes neurais e as máquinas de vetores

de suporte (*Support Vector Machines* (SVM)). Cada técnica utiliza um algoritmo de aprendizagem para identificar o modelo que melhor representa o relacionamento entre os atributos das instâncias e as classes às quais as instancias pertencem. O modelo gerado deve representar bem o conjunto de treinamento e prever corretamente novas classes de instâncias. Portanto, o objetivo principal de um algoritmo de aprendizagem é construir modelos com boa capacidade de generalização

As árvores de decisão são uma técnica clássica de aprendizagem de máquina, bem difundida e amplamente utilizada para resolver problemas de classificação. Utiliza-se de um algoritmo de aprendizagem para identificar o modelo que melhor representa o relacionamento entre os atributos das instâncias e as classes às quais as instâncias pertencem. O modelo gerado deve representar bem o conjunto de treinamento e prever corretamente novas classes de instâncias (QUINLAN, 1990). Através das árvores de decisão, o conhecimento adquirido é representado por meio de regras de fácil compreensão e pouco exigentes do ponto de vista computacional (TAN *et al.*, 2006).

Uma árvore de decisão é composta de três elementos básicos: um nó não-terminal que contém um teste condicional para o valor de algum dos atributos da instância; uma aresta, que corresponde a um dos possíveis resultados do teste condicional; e um nó terminal (folha), ao qual é atribuído um rótulo, que define a classe à qual uma instância pertence. O processo de classificar uma instância começa no nó raiz da árvore, onde o teste condicional é aplicado, e seu resultado define qual aresta da árvore seguir. Caso o próximo nó seja uma folha, a instância em questão é classificada com o rótulo atribuído ao nó. Caso contrário, o processo de aplicar o teste condicional se repete, até alcançar um nó folha.

Dentre os diversos algoritmos para construção de árvores de decisão (e.g. *Interactive Dichotomizer 3* (ID3), C4.5, *Classification and Regression Trees* (CART)), a seleção do atributo mais útil para classificar as instâncias é um fator crítico de escolha. Nesta dissertação, utilizamos as árvores C4.5, as quais baseiam-se no *information gain*, uma propriedade estatística da teoria da informação, a qual permite a seleção do melhor atributo para dividir o conjunto de treinamento (QUINLAN, 1993).

## 2.3 Trabalhos Relacionados

Atualmente, encontra-se na literatura uma variedade de estudos abordando *offloading*, seja ele de processamento ou de dados (ALI *et al.*, 2016), (Silva Jr. *et al.*, 2015), (ENZAI; TANG, 2014), (FERNANDO *et al.*, 2013), (KUMAR *et al.*, 2013), (CHUN *et al.*, 2011), (CUERVO *et*

*al.*, 2010) e (SATYANARAYANAN *et al.*, 2009).

Os trabalhos relacionados foram selecionados levando em consideração as técnicas que esses mecanismos utilizam para mitigarem o acesso à rede e consequentemente diminuírem a quantidade de dados enviados pelos dispositivos móveis. Se encaixam nessa seleção, os mecanismos de *offloading de processamento* que utilizam técnicas de *cache* ou de sincronização de dados e os mecanismos de *offloading de dados*.

Em Kumar *et al.* (2013), foi realizado um estudo do impacto das tarefas de *offloading* no desempenho global do sistema. Foi apresentado um modelo analítico que propõe responder *quando* que a realização do *offloading de processamento* seria realmente viável levando em conta uma visão global do dispositivo móvel e do processo de *offloading*. As equações propostas em seu modelo assumem que mesmo no *offload* voltado a melhoria do processamento, uma quantidade significante de dados deve ser transferida do dispositivo móvel para o dispositivo remoto. Esta dissertação de mestrado utilizou-se da mesma ideia geral de (KUMAR *et al.*, 2013), onde se aponta a possibilidade de realizar previamente o *offloading* dos dados envolvidos nas tarefas de processamento remoto, diminuindo assim esta sobrecarga.

Em Hung *et al.* (2012), foi proposto um *framework* para execução de aplicações móveis em um ambiente virtual na nuvem. Um agente instalado no dispositivo local e no ambiente virtualizado se responsabiliza por orquestrar a sincronização dos dados necessários para correta execução da aplicação na nuvem. Para diminuir a sobrecarga na transferência dos dados, os agentes obtêm de forma prévia informações apenas do estado do dispositivo móvel e do dispositivo virtualizado. Tais informações são então utilizadas para elencar quais dados devem efetivamente passar pelo *offloading*.

### *2.3.1 Pocket Cloudlet*

No trabalho de Koukoumidis *et al.* (2012) é apresentada uma solução para aumentar a eficiência do acesso a dados na nuvem. Isso é feito ao utilizar a memória de dispositivos móveis para aumentar a velocidade de acesso a dados, reduzir latência e melhorar a eficiência energética. A ideia desta solução é construir uma *Pocket Cloudlet* baseada em memória não volátil para armazenar partes específicas ou mesmo os serviços completos da nuvem nos dispositivos móveis. No entanto, nem todos os dados podem ser armazenados no dispositivo móvel. Os autores desenvolveram uma arquitetura para a *Pocket Cloudlet*, incluindo seleção de dados e gerenciamento de dados, para determinar a quantidade de dados a serem armazenados no

dispositivo para cada serviço da nuvem.

Além disso, a arquitetura da *Pocket Cloudlet* introduz um mecanismo para gerenciar e sincronizar dados entre dispositivos móveis e a nuvem para lidar com mudanças de dados na nuvem (por exemplo, conteúdos da Web a serem atualizados ao longo do tempo). Esta é uma arquitetura eficaz para os usuários móveis que acessam dados na nuvem (DINH *et al.*, 2013). No entanto, existem duas questões nesta abordagem. Primeiro, essa arquitetura requer uma memória não volátil cara para armazenar dados de nuvens. Em segundo lugar, para cada serviço em nuvem, os dispositivos móveis precisam determinar quais partes de serviços devem ser armazenados em cache localmente. Assim, isso não é flexível quando se aplica para diversos serviços na nuvem.

### *2.3.2 Spectra*

O Spectra (FLINN *et al.*, 2002) é um exemplo de um sistema de *offloading* que utiliza um sistema de armazenamento que replica seus dados em vários dispositivos. Quando um aplicativo lê ou grava dados, o sistema de armazenamento garante que as modificações feitas em um dispositivo sejam transferidas para outros dispositivos para que cada componente tenha acesso à versão mais recente dos dados, independentemente de onde ele for executado. O trabalho de Flinn *et al.* (2002) usa o sistema de arquivos distribuídos Coda (KISTLER; SATYANARAYANAN, 1992) para persistir os dados.

Os sistemas de replicação de dados, como o Coda, armazenam o conteúdo dos arquivos em cada dispositivo. Os acessos aos dados podem consumir tempo e energia significativos quando os itens não estão disponíveis localmente e, consequentemente, precisam ser obtidos de outro local. Sistemas como Spectra (FLINN *et al.*, 2002), portanto, monitoram o estado do cache do arquivo como um recurso para estimar os custos de replicação ao tomar decisões de particionamento.

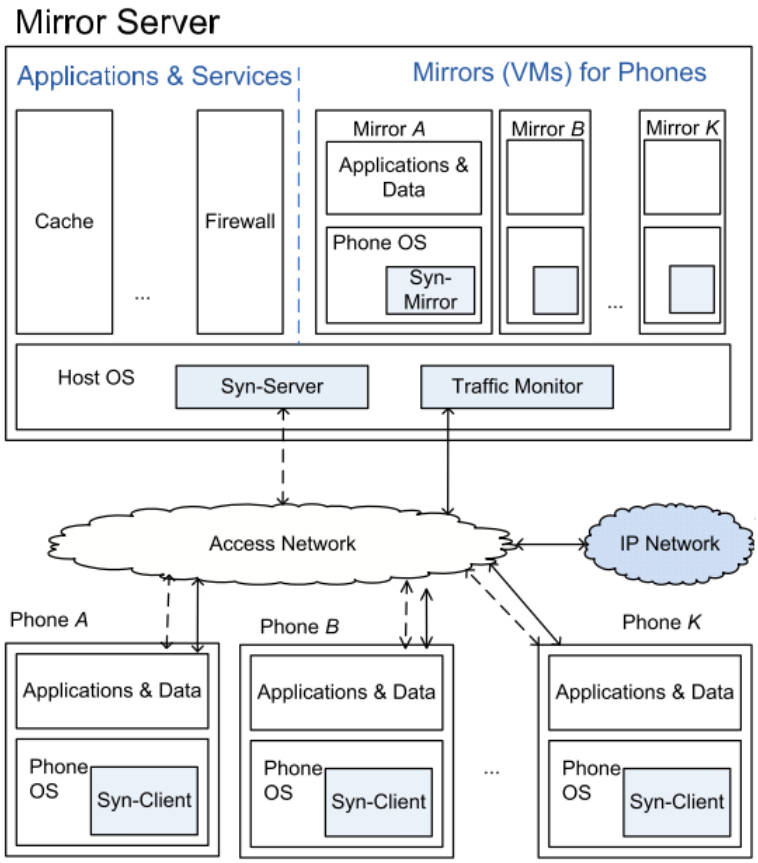
O monitor de estado do Spectra interage com o Coda para prever o estado do *cache* dos arquivos. O monitor se comunica com o Coda para saber que arquivos estão armazenados em cache localmente no computador. Embora seja possível que o estado do cache mude ligeiramente durante uma computação, é improvável que essas mudanças sejam significativas. O monitor Spectra também obtém do Coda uma estimativa da taxa em que os dados não *cached* serão obtidos de um servidor de arquivos.

### *2.3.3 Mirroring smartphones*

No trabalho de Zhao *et al.* (2010) foi proposto um *framework* que mantém clones de *smartphones* em servidores na nuvem. Com os clones, o *framework* migra a carga de trabalho computacional de um *smartphone* para a nuvem. Uma vez que o clone é sincronizado com o seu *smartphone* correspondente, algumas operações, como compartilhamento de arquivos e varredura de vírus, podem ser realizadas diretamente na nuvem.

O *framework* é baseado em *Cloud Computing* e nas técnicas de virtualização. No lado do *smartphone*, um módulo de sincronização é implantado para coletar dados de entrada do usuário e transmiti-los ao servidor. Com as técnicas de *Cloud Computing*, o servidor é capaz de hospedar centenas de clones e cada clone é implementado como uma máquina virtual (*Virtual Machine* (VM)). Para manter a sincronização entre o clone e o *smartphone*, todas as operações realizadas no *smartphone* são repassadas para o servidor exatamente na mesma ordem em que elas aconteceram. A arquitetura do *framework* é exibida na Figura 3.

Figura 3 – Arquitetura do *framework* proposto por Zhao *et al.* (2010)




Fonte: (ZHAO *et al.*, 2010).

Os clones precisam ser atualizados quando o estado do *smartphone* muda. A tarefa de atualização é definida de acordo com o mecanismo de sincronização escolhido, o qual depende dos aplicativos que serão suportados no servidor (ZHAO *et al.*, 2010). Para que seja feito o *upload* de arquivos que estão no *smartphone*, o serviço de cache de dados utiliza o clone, pois este tem os mesmos dados armazenados que o seu *smartphone* correspondente e pode reproduzir todas as ações executadas pelo *smartphone*. Assim, se um usuário quiser enviar um arquivo para várias pessoas, ele não precisa enviar o arquivo várias vezes. Em vez disso, o serviço de armazenamento de dados pode enviar o arquivo diretamente do clone para várias pessoas e evitar que o *smartphone* faça acesso a rede repetidamente para uma mesma ação.

Quando o *smartphone* tenta realizar o *upload* de algum dado, o serviço de armazenamento registra o evento de "upload"nos módulos de sincronização. O módulo *Syn-Client* (exibido na Figura 3) intercepta a ação de upload no *smartphone* e, ao invés de fazer o upload do arquivo imediatamente, substitui a ação de upload padrão por uma ação "fake"impedindo que seja feito o *upload* dos dados. Quando ocorre a próxima sincronização entre o *smartphone* e o clone, o clone faz o *upload* dos dados.

### *2.3.4 COP*

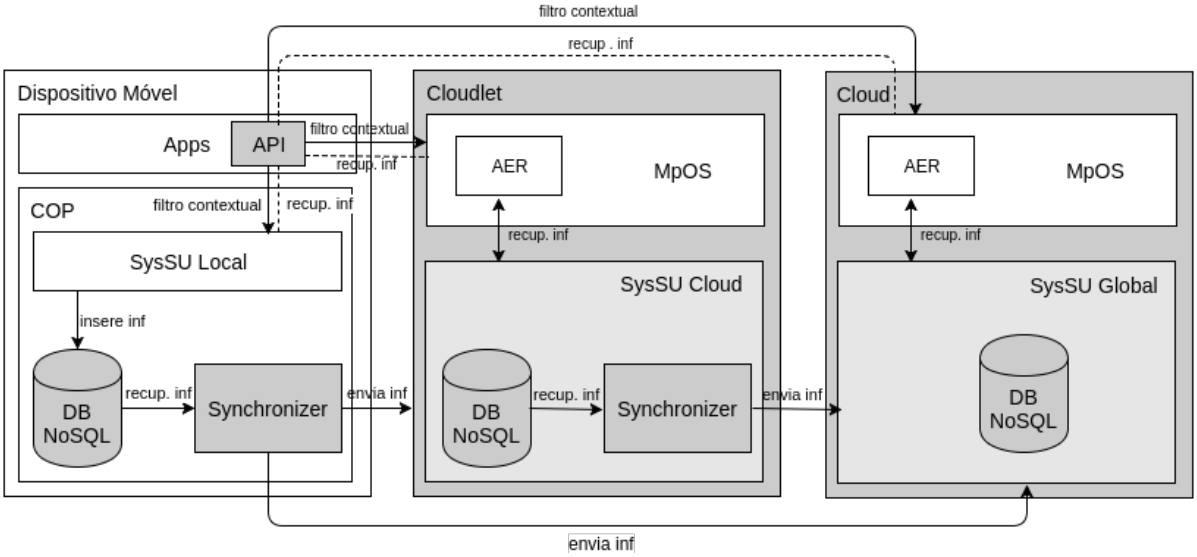
Em Gomes *et al.* (2016), foi proposto um serviço de *offloading de dados* com suporte à privacidade batizado de COP (***C****ontextual data* ***O****ffloading service with* ***P****rivacy support*) a fim de realizar a disseminação de dados contextuais entre os dispositivos móveis e o ambiente da nuvem. Esse serviço se utiliza do conceito de *cloudlets* apresentado em Satyanarayanan *et al.* (2009) para permitir essa disseminação.

O serviço realiza *offloading de processamento* e faz uma tomada de decisão pela execução ou não na nuvem a fim de melhorar o desempenho da aplicação que é realizada com uso de métricas de rede. Ele se utiliza de informações como a qualidade de conexão entre o dispositivo móvel e o servidor remoto, ou seja, a latência da conexão para mensurar a viabilidade das tarefas de *offloading*.

A arquitetura do COP é apresentada na Figura 4 e é composta por 3 camadas. A primeira camada é representada pelo dispositivo móvel, onde os dados contextuais são capturados e armazenados localmente. Nessa camada foi acrescentado um banco de dados *Not Only SQL* (NoSQL), que atua como uma espécie de *cache* local das informações de contexto, até que o processo de *offloading de dados* ocorra. A segunda camada é representada por uma *cloudlet*,

que recebe dados contextuais de usuários conectados a uma mesma *Wireless Local Network* (WLAN). A última camada agrega dados de diferentes *cloudlets* e usuários que compartilham seus dados contextuais (GOMES *et al.*, 2016).

Figura 4 – Arquitetura do COP




Fonte: (GOMES *et al.*, 2016)

Além disso, no trabalho de Gomes *et al.* (2016) foi proposto um conjunto de regras que indicam como os dados contextuais são enviados pelos dispositivos móveis para a nuvem. Estas regras são chamadas de políticas de sincronização. O usuário do sistema pode explicitamente indicar quais políticas de sincronização ele deseja utilizar para o envio dos dados contextuais para a nuvem, e quais desses dados devem ser compartilhados.

O componente *Synchronizer*, presente no dispositivo móvel, é responsável por recuperar os dados contextuais do banco de dados no dispositivo móvel e enviar para o ambiente remoto. O *Synchronizer* utiliza de estratégias de envio definidas nas políticas de sincronização para definir em que momentos os dados contextuais devem ser migrados do dispositivo para um ambiente remoto.

De acordo com Gomes *et al.* (2016) há três estratégias de envio: (*i*) por tempo, (*ii*) por quantidade de tuplas e (*iii*) orientada à conexão *Wi-Fi*. A estratégia de envio por tempo significa que, periodicamente (e.g., a cada 30 segundos), as tuplas são enviadas para o ambiente remoto. O período para o envio dos dados é configurável. Caso o usuário não defina explicitamente outra estratégia a ser usada, esta é a estratégia padrão utilizada. A estratégia de envio por quantidade de tuplas implica que, se uma quantidade pré-estabelecida de tuplas for alcançada, as tuplas serão enviadas à nuvem. Essa quantidade também é configurável. Por fim, a estratégia de envio

orientada à conexão *Wi-Fi* define que apenas quando uma conexão *Wi-Fi* for estabelecida, as tuplas do dispositivo móvel serão enviadas à infraestrutura de nuvem.

## 2.4 Considerações finais do capítulo

Nos trabalhos aqui citados, a problemática da sobrecarga adicional gerada pela transferência dos dados necessários durante as tarefas de *offloading de processamento* foi tratada utilizando a ideia de disponibilizar tais dados de forma prévia, a fim de obter um ganho de desempenho global. O mecanismo que propomos possui dois diferenciais: (*i*) Adicionou-se uma tomada de decisão antes do *offloading de dados*, a qual é feita tomando por base uma árvore de decisão. Essa árvore é criada na infraestrutura local (*cloudlet*) a partir do histórico de execuções local (dispositivo móvel) e remota (*cloudlet*); (*ii*) o *offloading de dados* é utilizado em conjunto com o *offloading de processamento* com intuito de melhorar o seu desempenho global. Esse processo, alinhado ao mecanismo de tomada de decisão adotado, são os diferenciais desta dissertação.

# 3 PROPOSTA

A proposta central deste trabalho consiste na elaboração de um mecanismo de *offloading de dados* com tomada de decisão que objetiva auxiliar no *offloading de processamento*. Para isto, é realizada a persistência de arquivos que são utilizados como parâmetros na chamada de métodos dos aplicativos móveis. Ao final deste capítulo, será possível compreender as entidades que compõem a proposta, bem como as métricas e processos que serão considerados e discutidos para a realização da tomada de decisão de *offloading de dados*.

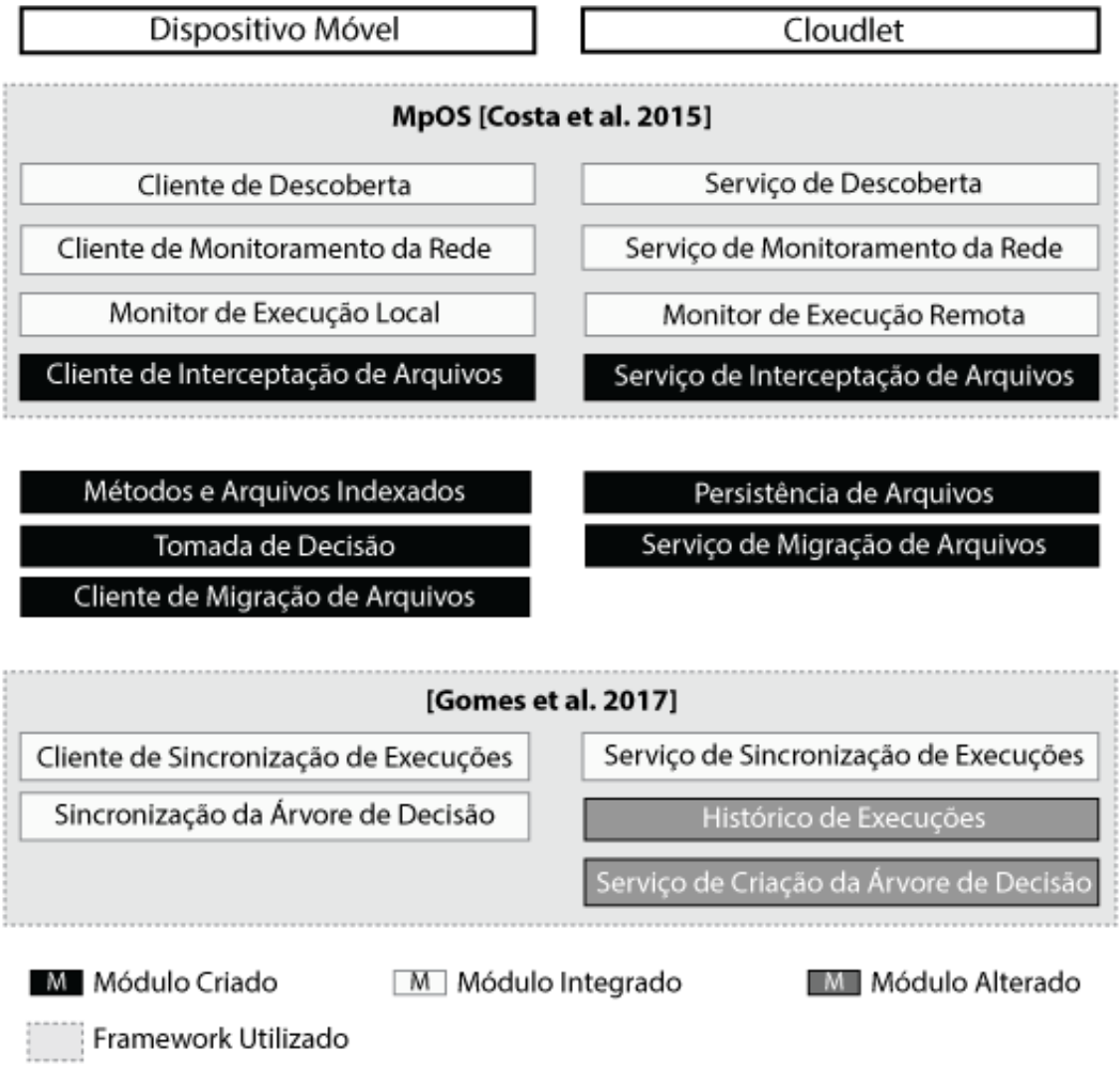
## 3.1 Arquitetura

O desenvolvimento do mecanismo resultou em 2 (dois) aplicativos. O primeiro, chamado de *FileOffApi*, deve ser executado em um dispositivo móvel, e o segundo, chamado de *FileOffCloudlet*, deve ser executado em uma *cloudlet* (como um *desktop*). O mecanismo não realiza comunicação com a nuvem e se restringe apenas aos dispositivos que estão em uma rede local. Para que o mecanismo desempenhe sua função, os dois aplicativos precisam se comunicar periodicamente e a infraestrutura de comunicação utilizada foi a conexão Wi-Fi entre os dois dispositivos, neste caso, entre o dispositivo móvel e a *cloudlet*.

O mecanismo de *offloading de dados* precisa estar integrado a um *framework* que faça *offloading de processamento* para que a estrutura como um todo entre em funcionamento. Ele foi organizado de forma que, é necessária a integração com o código fonte do *framework* para *offloading de processamento* que se deseja auxiliar. Utilizamos o *framework Multiplatform Offloading System* (MpOS) (COSTA *et al.*, 2015) na integração, pois, além de prover a funcionalidade de *offloading de processamento*, este possui os módulos que fazem o monitoramento da rede e o processo de descoberta de serviço que é necessário para o mecanismo. A utilização do MpOS é representada na Figura 5 pelo retângulo com linhas tracejadas.

Para o funcionamento do mecanismo, além da implementação do código responsável pelo *offloading* dos dados, foi necessário o desenvolvimento ou reuso de funcionalidades existentes em outros *frameworks*. Além do uso do *framework* MpOS (COSTA *et al.*, 2015), foram utilizados os trabalhos de Rego *et al.* (2017) e de Gomes *et al.* (2017) para a criação da árvore de decisão e sua sincronização com o dispositivo móvel. A Figura 5 exibe a arquitetura, dividida em módulos, utilizada para a integração dessas funcionalidades. Na Figura 5, os módulos desenvolvidos especificamente para este trabalho estão na cor preta. Os módulos que foram alterados

Figura 5 – Arquitetura do mecanismo proposto. Módulos do dispositivo móvel e da *cloudlet*.




Fonte: (FILHO *et al.*, 2017)

(cor cinza) e os legados (cor branca) foram retirados de *frameworks* já existentes, conforme foi apresentado.

As funcionalidades principais do mecanismo são: *(i)* a descoberta de serviço e o monitoramento das condições da rede, *(ii)* a interceptação e a indexação de parâmetros dos métodos executados, *(iii)* o monitoramento e armazenamento em uma base de dados dos tempos de execução dos métodos, *(iv)* a criação e sincronização da árvore de decisão, *(v)* o mecanismo de tomada de decisão e, por fim, *(vi) a migração dos dados*. O funcionamento dos módulos que implementam essas funcionalidades são explicados em detalhes a seguir.

### *3.1.1 Monitoramento da rede e descoberta de serviço*

Os módulos do mecanismo são instanciados a partir de *threads* que são ativadas e adormecidas em intervalos de 60 segundos. No momento que o mecanismo inicia, *threads* são criadas para cada módulo e são responsáveis pelas funcionalidades do mecanismo de forma isolada. Sendo assim, mesmo que o módulo de descoberta ou de monitoramento parem de funcionar, a aplicação não pára.

Depois da instanciação das *threads*, o funcionamento do mecanismo começa com os módulos de descoberta que são responsáveis pela busca e registro em um serviço de *offloading de dados* que está em execução em uma rede local. O módulo **Cliente de Descoberta**, neste caso, anuncia sua presença por meio de diversas mensagens de *multicast*. O módulo **Serviço de Descoberta** escuta esse tipo de mensagem e responde para o destinatário com o endereço *Internet Protocol* (IP) do serviço, conforme foi definido no arquivo de configuração da *cloudlet*. Com isso, o dispositivo móvel poderá interagir diretamente com a *cloudlet* para descobrir as portas dos outros serviços de rede, como a implantação de serviço, monitoramento de rede e o próprio serviço de *offloading* (COSTA *et al.*, 2015).

Em seguida, os módulos de monitoramento (**Cliente de Monitoramento da Rede** e **Serviço de Monitoramento da Rede**) trocam dados para mensurarem a qualidade de uma conexão de rede entre o dispositivo móvel e a *cloudlet*. As métricas empregadas para mensurarem a qualidade de rede são: *Round Trip Time* (RTT), taxa de *Upload* e taxa de *Download*. A aferição dessas métricas é feita periodicamente a cada 60 segundos e seus valores são armazenados e compartilhados entre o dispositivo móvel e a *cloudlet*. As métricas obtidas por esse processo são utilizadas para a criação da árvore de decisão e a tomada de decisão que serão vistas, em detalhes, nas subseções a seguir.

### *3.1.2 Interceptação e indexação de parâmetros*

Para que o *framework* MpOS possa realizar a operação de *offloading* é necessário que o desenvolvedor marque que métodos de uma aplicação Android ou Windows Phone são candidatos a sofrerem a migração para a *cloudlet*. Os métodos marcados podem conter parâmetros diversos, no entanto, o mecanismo proposto leva em consideração apenas os parâmetros que contém dados referentes a arquivos de imagens, vídeos ou áudios pois, geralmente, esses parâmetros apresentam tamanhos significativamente maiores em comparação a outros tipos de

parâmetros.

As aplicações que trabalham com operações em arquivos se beneficiam diretamente do mecanismo. Como exemplo, pode-se imaginar um usuário que utiliza uma aplicação de filtro de imagem e que, geralmente, aplica mais de um filtro à mesma imagem. Com esse comportamento, a mesma imagem seria enviada diversas vezes para a *cloudlet*. No entanto, com a utilização do mecanismo é feita a migração prévia da imagem para a *cloudlet*. Isso é feito para que haja a diminuição do tempo de envio dos arquivos do dispositivo móvel quando uma operação é feita diversas vezes com o mesmo arquivo ou quando o mesmo arquivo é utilizado em diferentes operações.

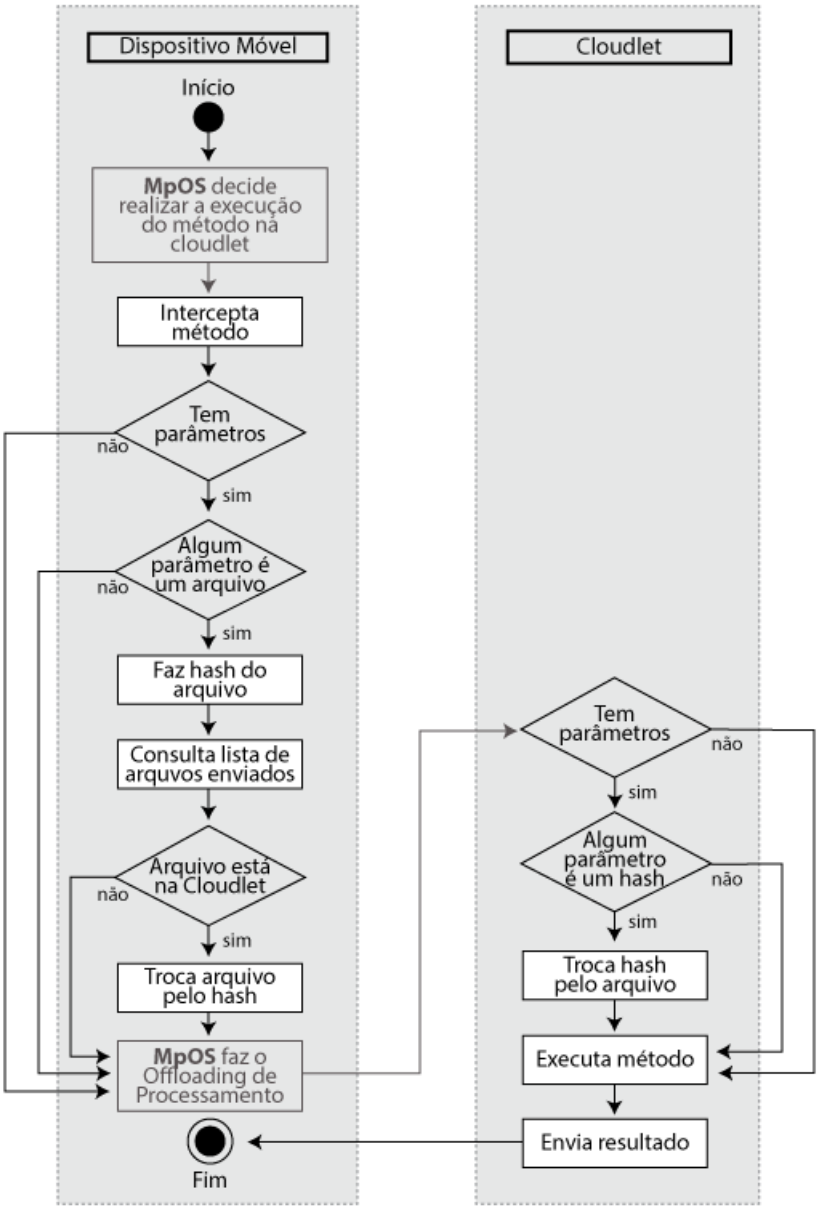
Antes de iniciar o processo de tomada de decisão e de migração dos arquivos, o usuário do dispositivo móvel deve selecionar um diretório que contêm os arquivos que possivelmente serão migrados. Para cada arquivo é gerada sua identificação com o uso de uma função *hash* criptográfica *Secure Hash Algorithm 1* (SHA-1) que utiliza como entrada os *bytes* que compõem o próprio arquivo. Em posse desses dados, o módulo **Métodos e Arquivos Indexados** lista todos os arquivos com suas funções *hash*. Esse módulo também mantém duas listas que relacionam os arquivos que estão apenas no dispositivo móvel e os arquivos que já foram transferidos para a *cloudlet*.

O módulo de indexação também tem a função de manter a listagem dos métodos que foram invocados. Essa listagem é gerada a partir da marcação feita pelo desenvolvedor no momento de integrar o mecanismo ao código-fonte da aplicação desenvolvida. No MpOS, o desenvolvedor deve explicitamente incluir uma *annotation* antes dos métodos que podem sofrer o processo de *offloading de processamento*. O mecanismo verifica que métodos foram marcados com a *annotation @Offloadable* e mantém o nome dos métodos na listagem.

Momentos antes de realizar o *offloading* de um método é necessário inspecionar os parâmetros passados e verificar se os arquivos presentes nos parâmetros estão disponíveis na *cloudlet*. Caso essa situação seja verdadeira, o arquivo que seria transferido junto com o método é substituído pelo seu identificador (função *hash*). Quando a invocação do método, junto com os parâmetros, chega à *cloudlet*, o identificador do arquivo é substituído pelo arquivo e a execução do método prossegue normalmente.

O processo de interceptação e substituição de parâmetros dos métodos é ilustrado na Figura 6 e começa após o *framework* MpOS decidir que um método será executado na *cloudlet*. O método a ser executado é interceptado e é verificado se há parâmetros que representam arquivos

Figura 6 – Fluxograma do processo de interceptação e troca de parâmetros.




Fonte: o autor.

(*arrays* de *bytes*). Após a interceptação dos parâmetros, é feita a identificação (função *hash*) dos arquivos e em seguida é verificado se esse arquivo já foi enviado para a *cloudlet* ao consultar a lista dos arquivos já enviados. Quando o arquivo já está disponível na *cloudlet*, o parâmetro que representa o arquivo é trocado pela sua função *hash* correspondente e o processo de *offloading de processamento* segue normalmente. Na *cloudlet* é feito o processo inverso, onde é verificado se algum dos parâmetros do método a ser executado é uma função *hash* e então é feita a troca pelo arquivo.

O processo descrito anteriormente, ilustrado na Figura 6, começa com o módulo **Cliente de Interceptação de Arquivos** que inspeciona os parâmetros dos métodos e verifica

no módulo **Métodos e Arquivos Indexados** se o arquivo passado nos parâmetros já está na *cloudlet*. O módulo **Serviço de Interceptação de Arquivos** inspeciona os parâmetros do método que chegou na *cloudlet* e substitui o identificador pelo arquivo depois de solicitar ao módulo **Persistência de Arquivos** o arquivo correspondente ao identificador informado.

### *3.1.3 Monitoramento e armazenamento dos tempos de execução*

Os módulos **Monitor de Execução Local** e **Monitor de Execução Remota** são responsáveis pelo monitoramento do tempo de execução de métodos que são executados no dispositivo móvel e na *cloudlet*, respectivamente. Eles calculam quanto tempo decorreu desde a chamada de um método até seu término. O módulo **Monitor de Execução Remota**, além de fazer a medição do tempo de execução, junta esses dados com as condições da rede no momento da medição. As condições da rede são obtidas através do módulo **Serviço de Monitoramento da Rede** que detalhamos anteriormente na subseção 3.1.1.

As informações obtidas pelos módulos de monitoramento são persistidas em uma base de dados que armazena o histórico de execuções dos métodos. Essas informações são utilizadas para o treinamento da árvore. Para que essas informações cheguem à base de dados, foram utilizados dois módulos do trabalho (GOMES *et al.*, 2017): o **Cliente de Sincronização de Execuções** e o **Serviço de Sincronização de Execuções**. O primeiro módulo obtém os tempos de execução dos métodos executados no dispositivo móvel e se comunica com o segundo módulo para repassar as novas informações. O módulo **Serviço de Sincronização de Execuções** recebe as informações do módulo **Cliente de Sincronização de Execuções** e também é responsável por medir os tempos de execução dos métodos executados na *cloudlet*. A persistência na base de dados é feita pelo módulo **Serviço de Sincronização de Execuções**.

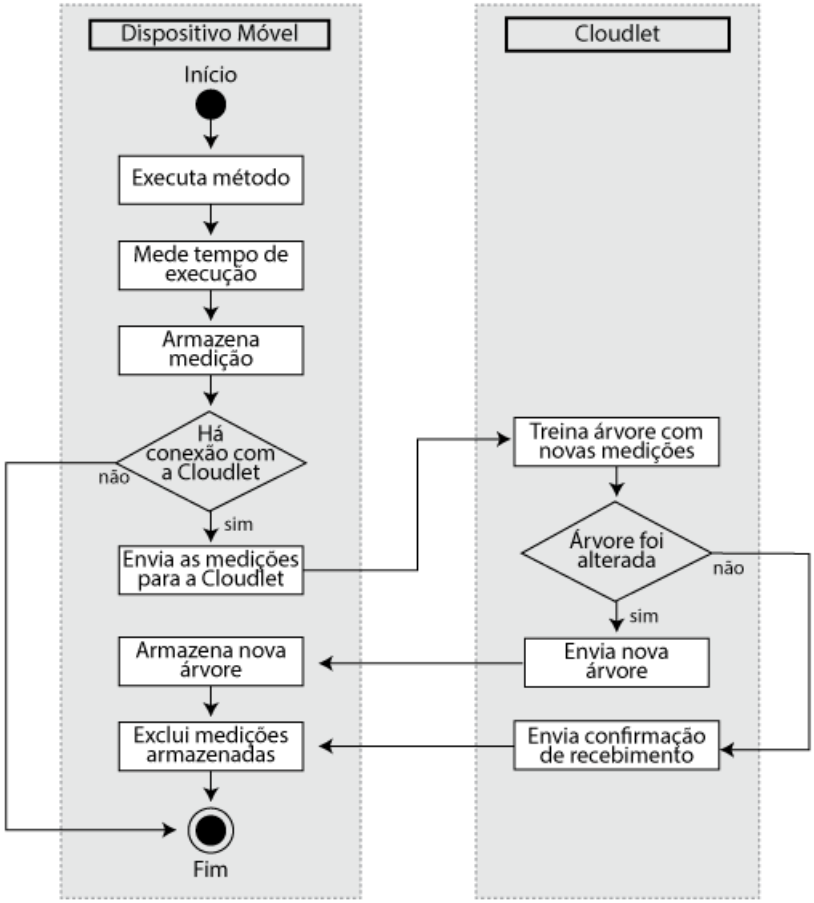
### *3.1.4 Criação e sincronização da árvore de decisão*

O processo de criação da árvore de decisão depende do histórico de execuções do dispositivo móvel e da *cloudlet*. A árvore de decisão sempre é criada e alterada na *cloudlet*, visto que o processo computacional para sua criação pode consumir recursos do dispositivo móvel. Para tal, o histórico de medições do dispositivo móvel deve, sempre que possível, estar sincronizado com a *cloudlet* para que esta verifique se os dados adicionais alterarão a estrutura da árvore.

A Figura 7 ilustra o processo de medição de execuções no dispositivo móvel com

a sincronização das medições e da árvore de decisão com a *cloudlet*. O processo começa com a medição e armazenamento dos tempos de execução dos métodos. Em seguida, é verificado se há conexão com a *cloudlet* para o envio das medições. Quando a *cloudlet* está em posse das medições, é feito o processo de treinamento da árvore que pode resultar em sua alteração. Caso a árvore mude, a *cloudlet* envia a nova árvore para o dispositivo móvel, senão apenas uma mensagem de confirmação de recebimento das medições é enviada para o dispositivo móvel. Em ambas situações, o dispositivo móvel apaga o histórico de medições até o que processo possa ocorrer novamente.

Figura 7 – Fluxograma do processo de execução de métodos e sincronização da árvore de decisão no dispositivo móvel.

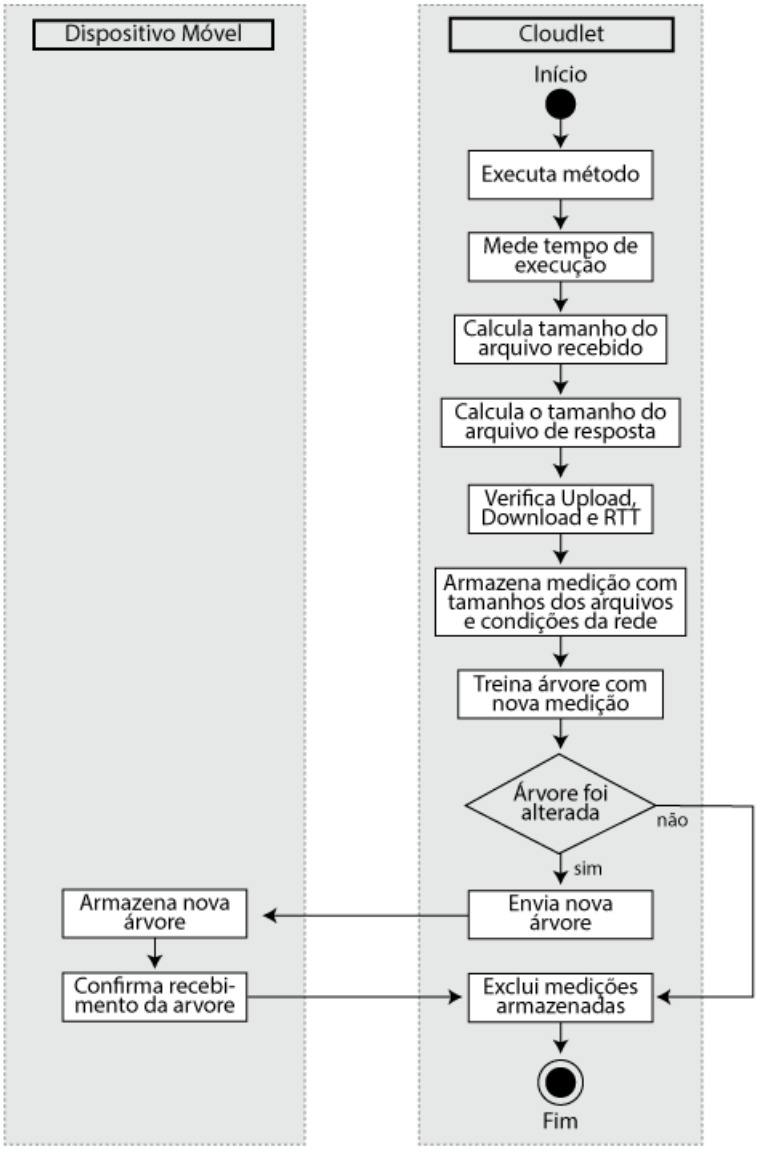



Fonte: o autor.

Na Figura 8 é ilustrado o processo de medição de execuções na *cloudlet* e também a sincronização da árvore de decisão com o dispositivo móvel. O que muda em relação ao processo ilustrado na Figura 7 é a quantidade de métricas adicionais que devem ser armazenadas junto com o tempo de execução dos métodos. Além das métricas de RTT, vazão de *Upload* e vazão de

*Download*, é necessário que seja calculado o tamanho do arquivo que será enviado de volta para o dispositivo móvel. Essas métricas são utilizadas para treinar a árvore de decisão. Caso haja alteração na estrutura da árvore de decisão após o treinamento, é iniciado o processo de envio da árvore para o dispositivo móvel.

Figura 8 – Fluxograma do processo de execução de métodos e sincronização da árvore de decisão na *cloudlet*.




Fonte: o autor.

### *3.1.5 Tomada de decisão e migração dos dados*

A tomada de decisão é o processo de definir que arquivos devem ser migrados. Isso ocorre no módulo **Tomada de Decisão**. Inicialmente, faz-se acesso ao módulo **Métodos e**

**Arquivos Indexados** em busca dos arquivos que estão apenas no dispositivo móvel e os métodos que podem ser executados (*offloaded*) na *cloudlet*. Em seguida, o módulo **Sincronização da Árvore de Decisão** é acessado e é solicitada a última árvore de decisão disponível. O próximo passo é solicitar ao módulo **Cliente de Monitoramento da Rede** as condições da rede. Em posse desses dados, são passadas para a árvore de decisão as condições da rede junto com o tamanho de cada arquivo no dispositivo móvel e os métodos que podem ser executados na *cloudlet*. Com essas informações a árvore é capaz de inferir se um arquivo deve ou não ser migrado para a *cloudlet*.

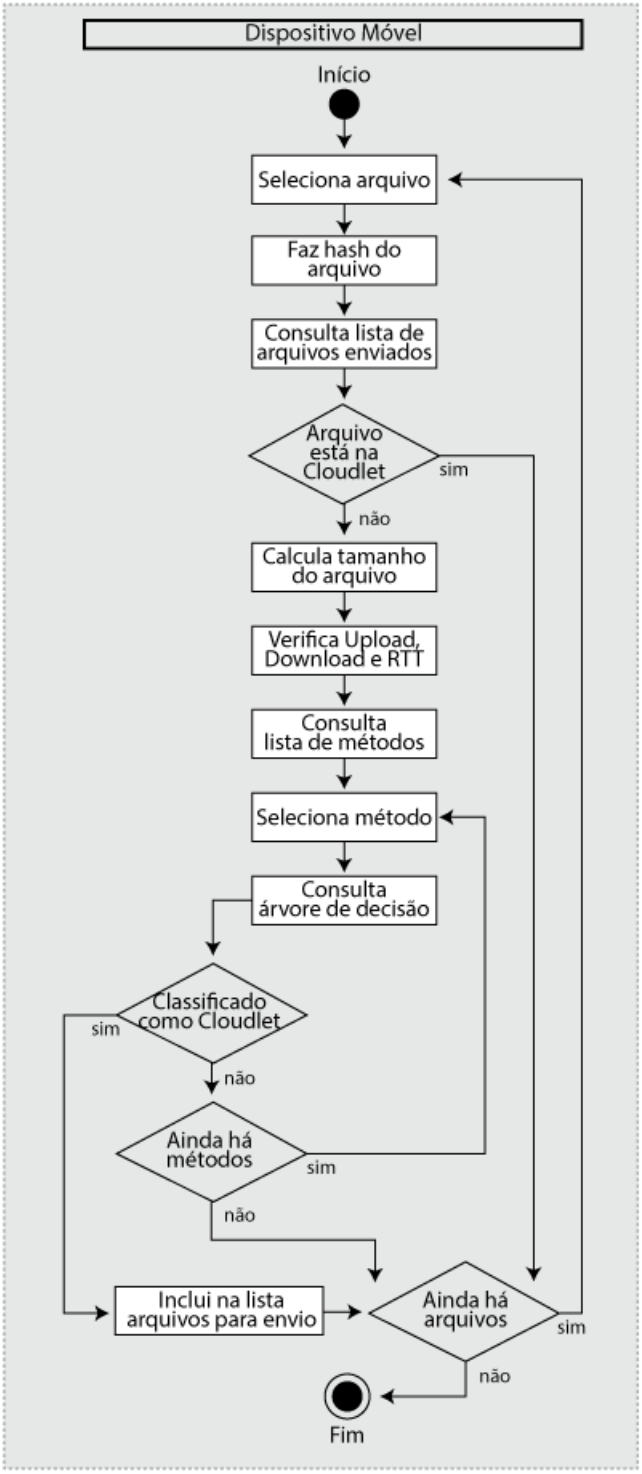
O processo de tomada de decisão é ilustrado na Figura 9. O processo é iniciado quando o dispositivo móvel está em posse da árvore de decisão. Inicialmente, é verificado no diretório definido pelo usuário os arquivos que podem ser migrados. Para cada arquivo é feito o processo de identificação (função *hash*) e verificação da disponibilidade do arquivo na *cloudlet*. Caso o arquivo não tenha sido enviado anteriormente ou seja um arquivo novo, é calculado seu tamanho. A ação seguinte é verificar as condições da rede (RTT, vazão *Upload*, vazão *Download*) e listar os métodos que podem ser executados na *cloudlet*. Então, são passadas para a árvore de decisão as condições da rede com a combinação dos arquivos e dos métodos. A árvore então classifica se o arquivo selecionado, utilizado como parâmetro em um método, é "Local"ou "Cloudlet". Os arquivos que são classificados como "Cloudlet"devem ser migrados para a *cloudlet*, já os classificados como "Local"devem permanecer apenas no dispositivo móvel.

A tomada de decisão resulta na listagem dos arquivos que devem ser migrados para a *cloudlet*. Essa listagem é passada para o módulo **Cliente de Migração de Dados** que faz o *offloading* dos arquivos. O módulo **Cliente de Migração de Dados** faz a conexão com a *cloudlet* através do módulo **Serviço de Migração de Dados** para a transferência dos arquivos. Além de enviar os arquivos, o módulo de transferência também tem a função de sincronizar a lista de arquivos que foram enviados para a *cloudlet*. Esse processo é ilustrado na Figura 10. Na *cloudlet* ainda há o módulo **Persistência de Arquivos** que armazena os arquivos recebidos.

## 3.2 Criação da árvore

O processo de criação da árvore de decisão foi introduzido em (REGO *et al.*, 2017) e adaptado neste trabalho. O **Serviço de Criação da Árvore de Decisão** utiliza o tempo de execução dos métodos no dispositivo móvel e o tempo de execução na *cloudlet* (capturado pelo **Serviço de Sincronização de Execuções**) para classificar cada instância do histórico de

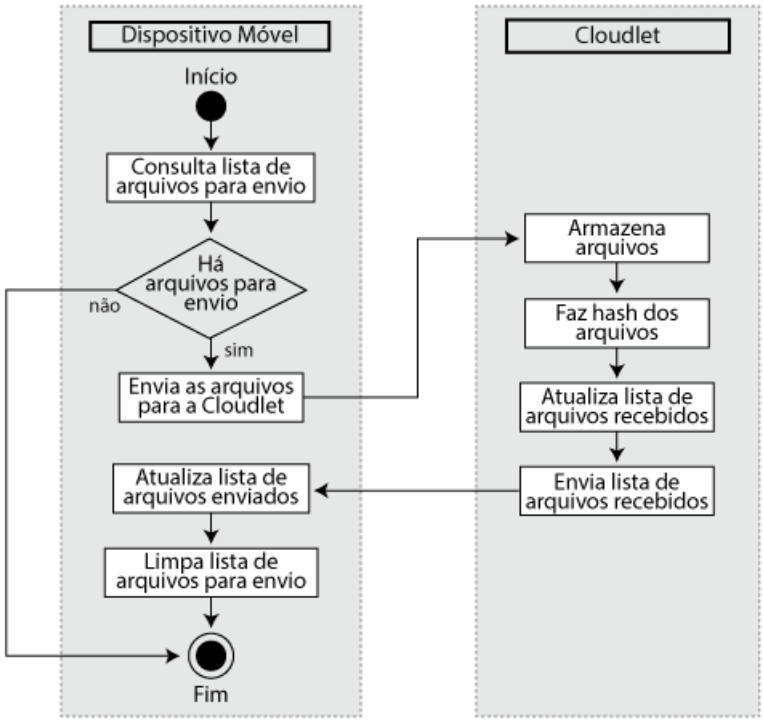
Figura 9 – Fluxograma do processo de seleção dos arquivos do dispositivo móvel para a migração.




Fonte: o autor.

execuções de *offloading* entre as classes *Local* e *Cloudlet* (Figura 11 (I)). Tais classes indicam que, em condição de rede similar, o método deve ser executado, respectivamente, no dispositivo móvel ou na *cloudlet*.

Figura 10 – Fluxograma do processo de migração dos arquivos do dispositivo móvel para a *cloudlet*.




Fonte: o autor.

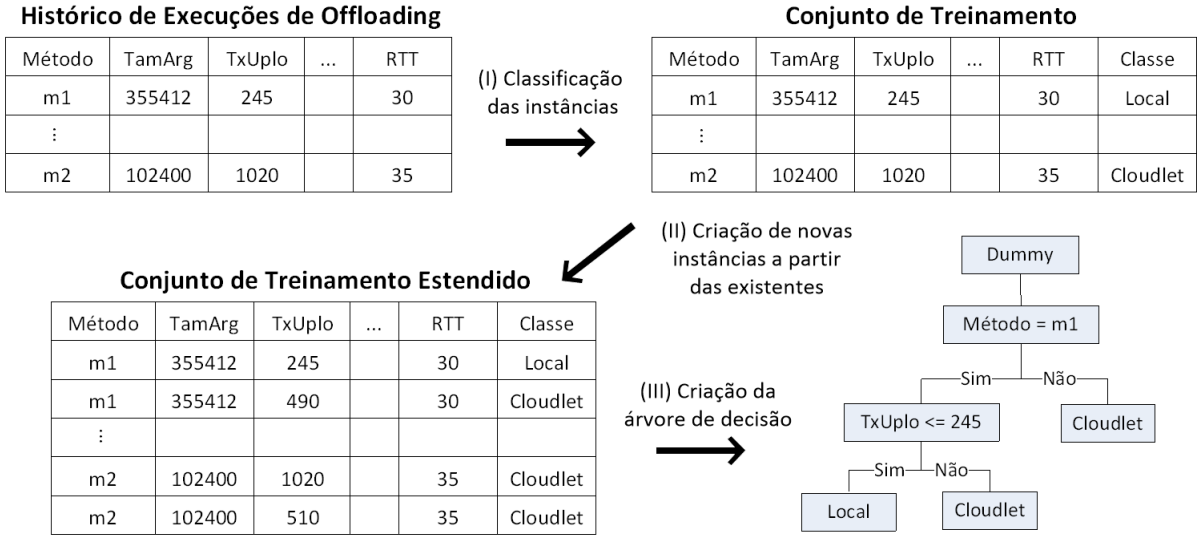
Uma vez que as instâncias são classificadas, os dados formam o conjunto de treinamento (típico de algoritmos de aprendizagem supervisionados) que é utilizado para gerar a árvore de decisão. Como o processo de aprendizagem de árvores de decisão depende do conjunto de treinamento, o número de instâncias do conjunto pode afetar o poder de generalização e precisão da árvore de decisão. Assim, antes de criar a árvore de decisão, o algoritmo analisa o histórico de execuções e considera diferentes condições de rede (Figura 11 (II)) para criar novas instâncias para o conjunto de treinamento (e.g., considerando redes com a metade da taxa de download/upload).

Dentre os diversos algoritmos para construção de árvores de decisão (e.g. ID3, C4.5 e CART), a seleção do atributo mais útil para classificar as instâncias é um fator crítico de escolha. Nesta dissertação utilizamos as árvores C4.5, que baseiam-se na *information gain*, uma propriedade estatística da teoria da informação, a qual permite a seleção do melhor atributo para dividir o conjunto de treinamento (QUINLAN, 1993). C4.5 foi escolhida porque seu algoritmo lida tanto com atributos categóricos (ordinais ou não-ordinais), como com atributos contínuos, além de gerar árvores não necessariamente binárias (diferente do CART) e alcançar ótimos resultados em problemas de classificação. Tais características, fazem do C4.5 um dos algoritmos

de árvore de decisão mais utilizados na literatura (WU *et al.*, 2008).

Uma vez que o conjunto de treinamento está pronto, uma árvore C4.5 é criada utilizando o algoritmo J48 da biblioteca Java Weka[1] (Figura 11 (III)). A Figura 11 apresenta um exemplo de árvore de decisão de *offloading* para a aplicação "Dummy". Neste exemplo, se a taxa de upload for igual a 300 Kbps, o método `m1` deve ser executado no dispositivo móvel. Além disso, independente das métricas, o método `m2` deve ser executado na *cloudlet*.

Figura 11 – Exemplo de árvore de decisão de *offloading*.




Fonte: (REGO, 2016)

O módulo **Serviço de Criação da Árvore de Decisão** periodicamente verifica na base de dados se há novas medições. Nem sempre a inclusão de novas medições vai mudar a estrutura da árvore de decisão, no entanto, quando ocorre esta mudança é preciso notificar o módulo **Sincronização da Árvore de Decisão** e junto com a notificação enviar a nova árvore de decisão.

## 3.3 Considerações finais do capítulo

Este capítulo apresentou o mecanismo de *offloading de dados* proposto nesta dissertação. O mecanismo visa auxiliar os mecanismos de *offloading de processamento* a diminuírem o tempo de *offloading* de seus procedimentos ao migrar dados do dispositivo móvel para a *cloudlet*. O mecanismo faz a tomada de decisão de que arquivos devem ser migrados para a *cloudlet* e os migra. Para isso, foram utilizadas árvores de decisão e o histórico de execuções de métodos.

[1] Weka website: <http://www.cs.waikato.ac.nz/ml/weka>.

Foram apresentados os módulos que compõem o mecanismo assim como a descrição de suas funcionalidades. O próximo capítulo trata da implementação e dos experimentos realizados com o mecanismo proposto para observar sua influência no tempo de *offloading* de procedimentos executados em uma *cloudlet*.

# 4 EXPERIMENTOS E ANÁLISE DOS RESULTADOS

Neste capítulo, são descritos os três experimentos realizados para avaliar o mecanismo proposto. É apresentado o processo de avaliação da abordagem, o ambiente e os experimentos conduzidos. Por fim, são analisados os dados obtidos nos experimentos e é feita a discussão dos resultados. Os detalhes dos experimentos e seus resultados são descritos a seguir.

## 4.1 Planejamento dos experimentos

Para a validação do mecanismo proposto, foi desenvolvido um aplicativo Android que executa filtros de imagem (Figura 12). Os filtros de imagem são procedimentos que alteram as cores e a aparência de fotos com o objetivo basicamente estético. Como exemplo, podemos citar o aplicativo Instagram[1] que disponibiliza filtros de imagem aos usuários para que possam alterar a aparência de suas fotos de forma simples.

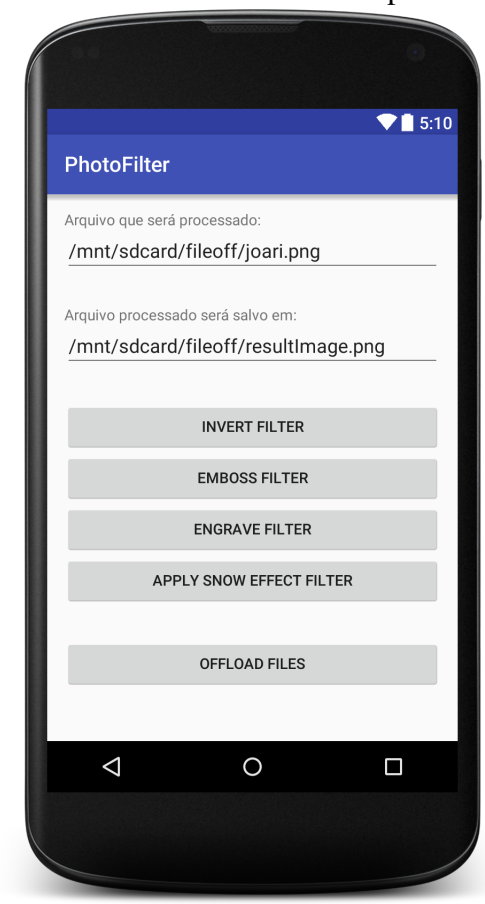
Os filtros de imagem utilizados pertencem à biblioteca de código aberto PhotoFilter[2], a qual possui 30 filtros. Essa biblioteca foi utilizada nos experimentos por dar suporte a versões antigas do sistema operacional Android (versão 2.3 e superiores) e pela disponibilidade do código fonte online. Para nossos experimentos foram usados somente os filtros *Inverter*, *Snow*, *Emboss* e *Engrave* devido aos seus tempos de execução em comparação aos outros da biblioteca. Entre os 30 filtros disponíveis, os filtros *Inverter* e *Snow* foram os que apresentaram menor tempo para serem executados, já os filtros *Emboss* e *Engrave* foram os que demandaram mais tempo.

Além de utilizar os filtros de imagem, o aplicativo é capaz de migrar as fotos que estão no dispositivo móvel para a *cloudlet* e de executar os filtros tanto no dispositivo móvel quanto na *cloudlet* com a *Application Programming Interface* (API) do *framework* MpOS (COSTA *et al.*, 2015). Os filtros do aplicativo necessitam que arquivos de imagem sejam passados como parâmetros. Para verificar como o tempo de execução dos filtros variavam de acordo com o tamanho dos arquivos, foram utilizados 10 arquivos com tamanhos distintos.

Para que o mecanismo proposto fosse capaz de decidir quais arquivos deveriam ser migrados, foram coletados os tempos de execução dos filtros. Os filtros foram executados em um dispositivo móvel e também em uma *cloudlet* (detalhes na Tabela 2). Os tempos de execução foram salvos em um banco de dados com o histórico de todas as execuções.

[1] Instagram website: https://www.instagram.com/?hl=pt-br
[2] Código Fonte PhotoFilter: https://github.com/mukeshsolanki/photofilter

Figura 12 – Aplicativo Android desenvolvido para testar o mecanismo.




Fonte: o autor.

Tabela 2 – Detalhes do ambiente de testes.

| DISPOSITIVOS | HARDWARE | | | SOFTWARE |
|---|---|---|---|---|
| **Dispositivo Móvel**<br>Smartphone Samsung Galaxy S6 (SM-G920i) | **Processador**<br>Quad-core 1.5 GHz Cortex-A53 + Quad-core 2.1 GHz Cortex-A57 | **RAM**<br>3GB | **WIFI**<br>802.11 a/b/g/n/ac | Android 5.0 TouchWiz UI Lollipop |
| **Cloudlet**<br>Ultrabook Samsung (NP53U3B-AD1BR) | **Processador**<br>Intel® Core™ i5-2467M | **RAM**<br>8GB | **WIFI**<br>802.11 b/g/n | Windows® 7 Home Premium Original 64 Bits |
| **Roteador**<br>DLink DSL-2740E | **WIFI**<br>802.11 b/g | | | |

Fonte: o autor.

Nas subseções a seguir, são descritos os estudos de caso utilizados na validação do mecanismo.

## 4.2 Experimento #1: Ganho de tempo com o *offloading de dados*

O primeiro experimento consistiu na implementação do *offloading de dados* e seu teste em um ambiente real. Para o teste foram utilizados os dispositivos descritos na Tabela 2. O objetivo deste experimento é verificar se a operação de *offloading de dados* resulta em ganho de tempo para o usuário quando é feito o *offloading de processamento* dos filtros de imagem. Com a utilização do mecanismo, os arquivos que são utilizados nos filtros de imagem são enviados previamente para a *cloudlet* e persistidos para usos futuros. Desse modo, evita-se que os mesmos arquivos sejam enviados várias vezes ao fazer a execução dos filtros de imagem na *cloudlet*.

Tabela 3 – Métrica e fatores utilizados no experimento de tempo de execução de métodos na *cloudlet*

| MÉTRICA | | DETALHES |
|---|---|---|
| Tempo do *Offloading* de Processamento | | Tempo em milissegundos para o *offloading* de processamento dos métodos de filtro de imagem |
| **FATORES** | **NÍVEIS** | **DETALHES** |
| Métodos de Filtro de Imagem | 4 | Filtro ApplySnow Filtro Inverter Filtro Emboss Filtro Engrave |
| Tamanhos de Arquivos | 10 | 25KB 90KB 192KB 342KB 524KB 870KB 1,2MB 1,5MB 1,9MB 4,2MB |
| Largura Máxima de Banda | 2 | 4Mbps 8Mbps |
| Offloading de Dados | 2 | Ativado Desativado |
| **15 Repetições** | **x** | **160 testes** |

Fonte: o autor.

As etapas do Experimento #1 são ilustradas na Figura 13 e iniciam na etapa 1 onde os arquivos de imagem que estão no dispositivo móvel são migrados para a *cloudlet*. Em seguida, na etapa 2, o dispositivo móvel faz o *offloading de processamento* dos filtros de imagem e passa como parâmetros as referências das imagens que foram migradas na etapa 1. Por fim, na etapa 3, o dispositivo móvel faz novamente o *offloading de processamento* dos filtros de imagem, porém, dessa vez, são passadas as imagens ao invés das referências. Para verificar se houve ganho de tempo, foi comparado o tempo gasto para fazer o *offloading de processamento* quando o mecanismo de *offloading de dados* estava ativado (Figura 13, etapa 2) com o tempo gasto

quando o mecanismo estava desativado (Figura 13, etapa 3).

Figura 13 – Etapas do Experimento #1.


Fonte: o autor.

A métrica e fatores utilizados para a execução do Experimento #1 estão descritos na Tabela 3. Além de serem utilizados 4 métodos de filtro de imagem e 10 arquivos de imagem de tamanhos distintos, foram utilizadas duas larguras de banda para verificar o quanto a vazão da rede impactou no tempo para se realizar o *offloading de processamento*. Por último, variou-se a ativação do mecanismo proposto para verificar em que situação o *offloading de processamento* se beneficiou do mecanismo e apresentou menor tempo. Foram realizados 160 testes e para cada teste foram feitas 15 repetições.

## 4.3 Experimento #2: Tempo de execução no dispositivo móvel e na *cloudlet*

O segundo experimento tem o objetivo de comparar os tempos obtidos no Experimento #1 (execução na *cloudlet*) com o tempo de execução dos mesmos métodos no dispositivo móvel. Como a tomada de decisão do mecanismo depende do histórico de execuções tanto no dispositivo móvel quanto na *cloudlet*, então, com este experimento, foi possível analisar quando

o *offloading de dados* seria benéfico. A métrica e os fatores utilizados no experimento estão descritos na Tabela 4.

Tabela 4 – Métrica e fatores utilizados no experimento de tempo de execução de métodos no dispositivo móvel.

| MÉTRICA | | DETALHES |
|---|---|---|
| Tempo de Execução no Dispositivo Móvel | | Tempo em milissegundos para a execução dos métodos de filtro de imagem |
| **FATORES** | **NÍVEIS** | **DETALHES** |
| Métodos de Filtro de Imagem | 4 | Filtro ApplySnow Filtro Inverter Filtro Emboss Filtro Engrave |
| Tamanhos de Arquivos | 10 | 25KB 90KB 192KB 342KB 524KB 870KB 1,2MB 1,5MB 1,9MB 4,2MB |
| **30 Repetições** | **x** | **40 testes** |

Fonte: o autor.

Ao comparar o Experimento #1 com o Experimento #2, é possível observar em que situações a execução dos filtros no dispositivo móvel é mais rápida do que realizar o *offloading de processamento* e, consequentemente, em que circunstâncias não é vantajoso fazer o *offloading* dos dados.

## 4.4 Experimento #3: Seleção dos arquivos utilizando a árvore de decisão

O terceiro experimento utiliza os dados gerados nos experimentos anteriores para criar a árvore de decisão. O objetivo deste experimento é verificar se o mecanismo de tomada de decisão seleciona corretamente que arquivos devem ser migrados para a *cloudlet*. Os fatores utilizados foram os 4 filtros de imagem e os 10 tamanhos distintos de arquivos utilizados nos experimentos anteriores, como é observado na Tabela 5. A migração ou não dos arquivos é a métrica do experimento.

Como é exibido no fluxograma da Figura 9, o funcionamento normal do mecanismo evita que arquivos que já foram migrados para um filtro sejam analisados novamente e assim evita-se o processamento desnecessário. No entanto, para a execução deste experimento, foi feita a tomada de decisão para todos os filtros mesmo quando o arquivo já tinha sido enviado. Isso foi feito com o objetivo de avaliar como se comportaria a tomada de decisão se apenas um dos filtros fosse utilizado e assim ser possível avaliar se a tomada de decisão funciona corretamente.

Tabela 5 – Métrica e fatores utilizados no experimento de seleção de arquivos para o *offloading de dados*.

| MÉTRICA | | DETALHES |
|---|---|---|
| Migração de arquivo | | Houve ou não a migração do arquivo para a *cloudlet* |
| **FATORES** | **NÍVEIS** | **DETALHES** |
| Métodos de Filtro de Imagem | 4 | Filtro Snow, Filtro Inverter, Filtro Emboss, Filtro Engrave |
| Tamanhos de Arquivos | 10 | 25KB, 90KB, 192KB, 342KB, 524KB, 870KB, 1,2MB, 1,5MB, 1,9MB, 4,2MB |
| Largura Máxima de Banda | 1 | 8Mbps |

Fonte: o autor.

## 4.5 Resultados

Os resultados do Experimento #1 foram compilados na Tabela 6. Os dados exibidos na Tabela 6 são as médias dos tempos gastos em milissegundos para o *offloading de processamento* dos quatro filtros em diferentes condições da rede, com o mecanismo de *offloading de dados* ativado ou desativado e com diferentes tamanhos de arquivos.

Nos testes realizados, quando o *offloading de dados* estava ativado, os arquivos utilizados nos filtros não precisavam ser enviados pela rede a cada requisição já que era feita a persistência na *cloudlet*. No início dos testes esperava-se que o tempo para o *offloading de processamento* dos filtros fosse menor quando o mecanismo estivesse ativado pelo motivo da persistência dos arquivos. No entanto, nos testes foi possível observar que há situações em que fazer a persistência de arquivos não é vantajoso (tempos em vermelho na Tabela 6). Os tempos na cor vermelha indicam que, para esses casos, é desvantajoso deixar o mecanismo ligado pois demorou-se mais tempo para serem executados se comparado com o mecanismo desligado.

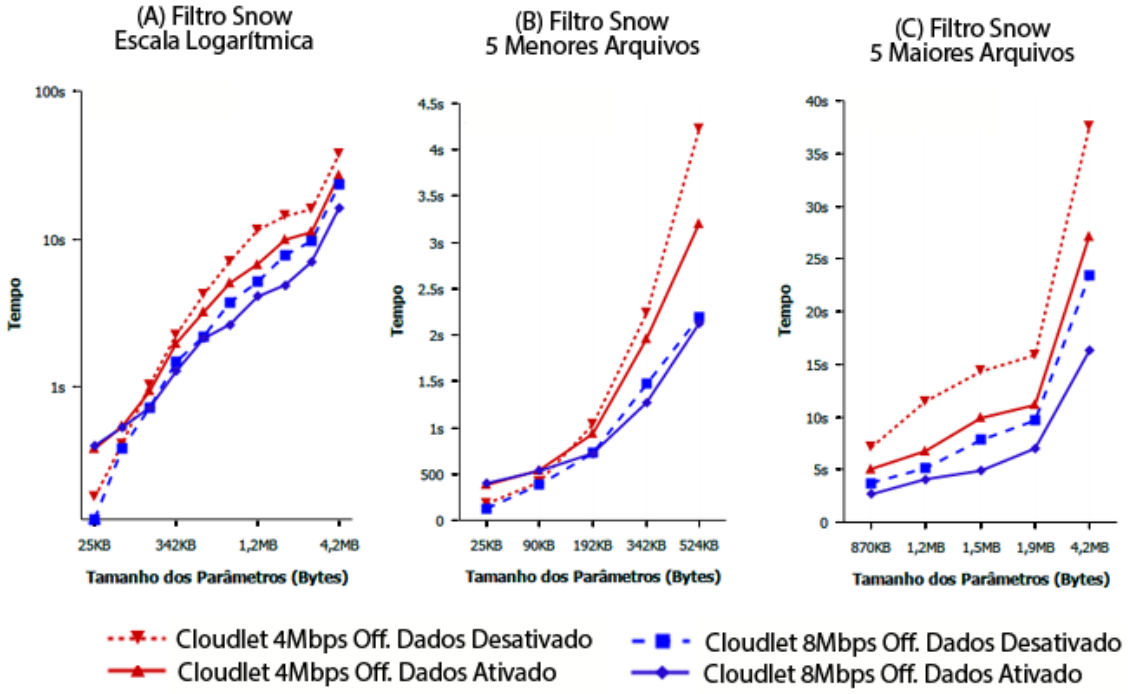
Os resultados do Experimento #1 mostraram que a migração dos arquivos economiza tempo em 80% dos casos. No entanto, podemos observar nas colunas 25KB e 90KB da Tabela 6 que o *offloading de processamento* dos filtros de imagem é mais lento quando o mecanismo estava ativado (tempos em vermelho). O funcionamento do mecanismo introduz na operação de *offloading de processamento* uma sobrecarga para realizar o *hash* dos arquivos e em seguida de trocar o *hash* pelo arquivo correspondente na *cloudlet*. O tempo necessário para realizar a função *hash* impactou no tempo de *offloading de processamento* dos filtros com arquivos de tamanho 25KB e 90KB, e assim, a operação de *offloading de dados* se tornou desvantajosa para estes testes.

Tabela 6 – Tempos de *offloading de processamento* (em milissegundos) com o mecanismo de *offloading de dados* ativado e desativado.

| **CLOUDLET** | | | **TAMANHO DOS ARQUIVOS** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **FILTRO** | **BANDA** | **OFFLOADING DE DADOS** | **25KB** | **90KB** | **192KB** | **342KB** | **524KB** | **870KB** | **1,2MB** | **1,5MB** | **1,9MB** | **4,2MB** |
| INVERTER | 4Mbps | DESATIVADO | 296 | 474 | 1272 | 3164 | 4168 | 9033 | 12307 | 17720 | 20799 | 45158 |
| | 4Mbps | ATIVADO | **416** | **673** | 1232 | 2524 | 3816 | 7852 | 9174 | 12363 | 14597 | 32311 |
| | 8Mbps | DESATIVADO | 176 | 458 | 1017 | 1808 | 2963 | 4648 | 6515 | 10121 | 11508 | 28696 |
| | 8Mbps | ATIVADO | **381** | **620** | 855 | 1765 | 2586 | 3676 | 6206 | 8392 | 9853 | 22660 |
| EMBOSS | 4Mbps | DESATIVADO | 386 | 842 | 2056 | 4256 | 7176 | 12800 | 18701 | 23522 | 26865 | 64780 |
| | 4Mbps | ATIVADO | **550** | **966** | 2012 | 3498 | 5703 | 9975 | 15281 | 19356 | 22285 | 51985 |
| | 8Mbps | DESATIVADO | 314 | 823 | 1426 | 2748 | 4361 | 7236 | 10418 | 15019 | 18868 | 39425 |
| | 8Mbps | ATIVADO | **565** | **970** | 1421 | 2621 | 4082 | 6452 | 8812 | 12857 | 16256 | 37542 |
| ENGRAVE | 4Mbps | DESATIVADO | 386 | 753 | 1806 | 4314 | 5896 | 10585 | 16674 | 20305 | 25092 | 57806 |
| | 4Mbps | ATIVADO | **560** | **905** | 1678 | 3297 | 5363 | 9775 | 13111 | 18498 | 21037 | 44678 |
| | 8Mbps | DESATIVADO | 325 | 813 | 1431 | 2699 | 4544 | 6875 | 10628 | 13144 | 17045 | 39034 |
| | 8Mbps | ATIVADO | **590** | **944** | 1364 | 2625 | 3833 | 5653 | 9557 | 11619 | 14512 | 32901 |
| SNOW | 4Mbps | DESATIVADO | 178 | 411 | 1033 | 2236 | 4218 | 7071 | 11450 | 14297 | 15852 | 37499 |
| | 4Mbps | ATIVADO | **381** | **539** | 934 | 1962 | 3203 | 5039 | 6745 | 9879 | 11119 | 27168 |
| | 8Mbps | DESATIVADO | 127 | 381 | 725 | 1471 | 2193 | 3730 | 5117 | 7795 | 9655 | 23398 |
| | 8Mbps | ATIVADO | **399** | **532** | 723 | 1271 | 2125 | 2642 | 4074 | 4884 | 6995 | 16280 |

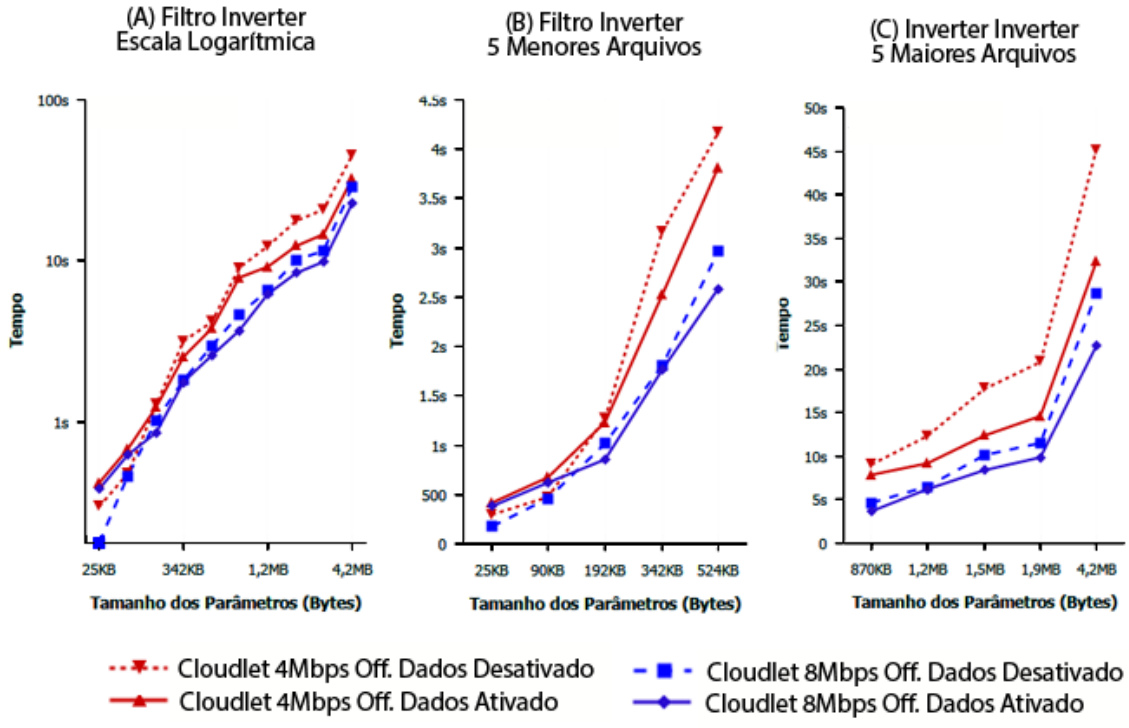
Fonte: o autor.

Figura 14 – Comparativo do tempo de *offloading de processamento* do filtro *Snow* com o *offloading de dados* ativado e desativado.



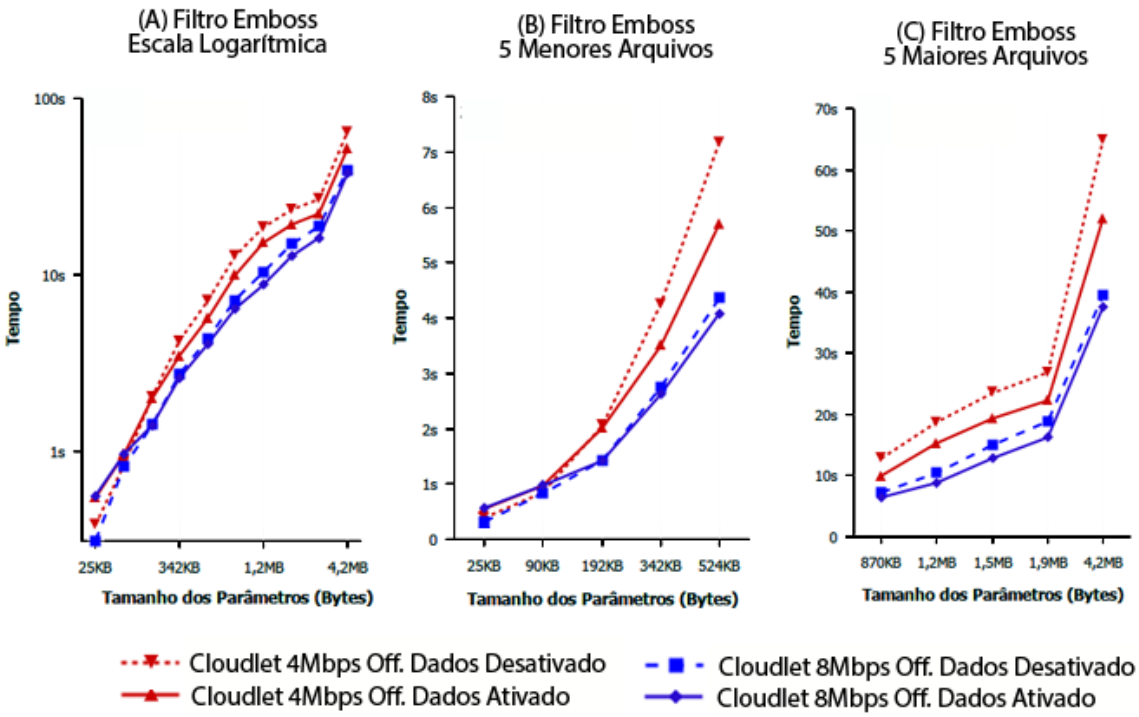

Fonte: o autor.

Figura 15 – Comparativo do tempo de *offloading de processamento* do filtro *Inverter* com o *offloading de dados* ativado e desativado.



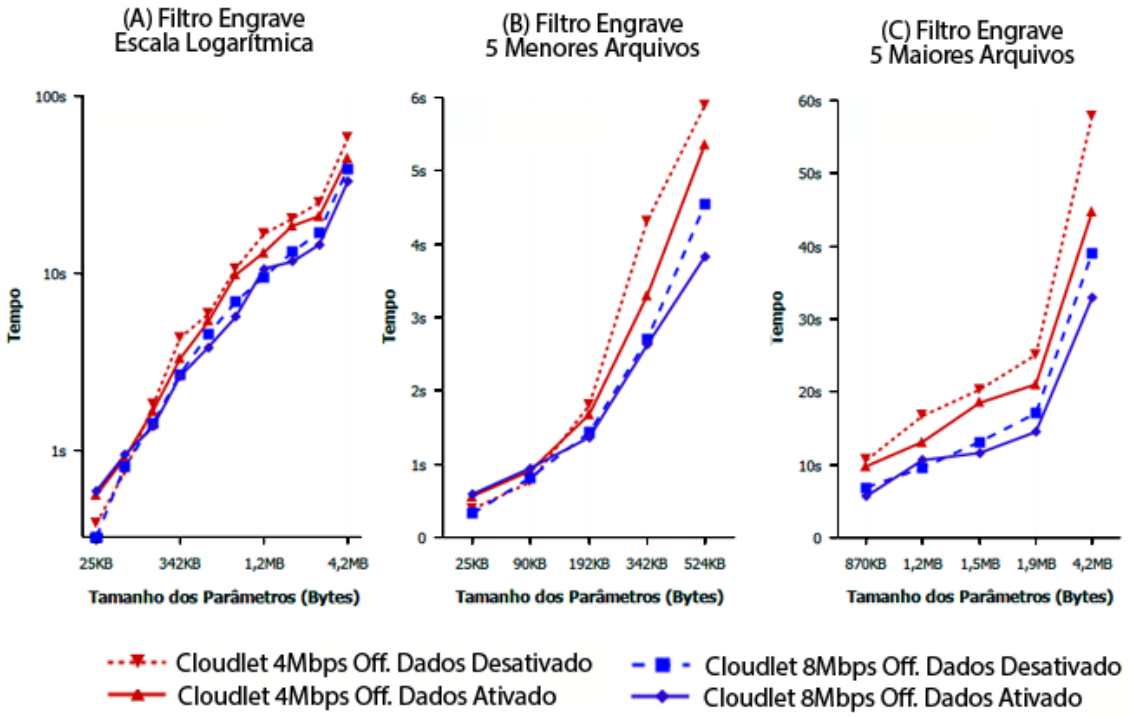

Fonte: o autor.

Figura 16 – Comparativo do tempo de *offloading de processamento* do filtro *Emboss* com o *offloading de dados* ativado e desativado.




Fonte: o autor.

Figura 17 – Comparativo do tempo de *offloading de processamento* do filtro *Engrave* com o *offloading de dados* ativado e desativado.




Fonte: o autor.

Os dados obtidos no Experimento #1 foram utilizados para gerar os gráficos da Figura 14 que representa o filtro *Snow*, a Figura 15 que representa o filtro *Inverter*, a Figura 16 que representa o filtro *Emboss* e a Figura 17 que representa o filtro *Engrave*. As figuras exibem a diferença de tempos do *offloading de processamento* dos filtros quando o *offloading de dados* está ativado e desativado para duas larguras de banda distintas.

Podemos observar nas figuras e na Tabela 6 que para as execuções com tamanho de arquivos maiores ou iguais a 192KB é vantajoso utilizar o *offloading de dados*. Observamos que a utilização do mecanismo se torna mais vantajosa de acordo com o aumento do tamanho dos arquivos. Quando comparamos os tempos de execução com e sem o mecanismo, em média, o tempo ganho com a migração dos dados foi de 19,5%.

Apesar do ganho médio observado no Experimento #1, foi necessário comparar o tempo do *offloading de processamento* com o tempo de execução no dispositivo móvel para averiguar onde é mais vantajoso realizar a execução dos filtros de imagem. Essa comparação foi feita no Experimento #2 e os dados obtidos foram compilados na Tabela 7, na Figura 18 e na Figura 19.

Os resultados do Experimento #2 mostram que o *offloading de processamento* dos filtros consome menos tempo do que a execução no dispositivo móvel em 40% dos casos (tempos na cor verde). No entanto, quando analisamos a Tabela 7 percebemos que o filtro *Inverter* e *Snow* são executados mais rapidamente no dispositivo móvel do que na *cloudlet*. Nessa situação, assumimos que não existe ganho em fazer a migração dos arquivos quando os filtros são executados mais rapidamente no dispositivo móvel. Já que não haverá execução na *cloudlet* para essas situações, então não há necessidade de migrar arquivos para essas operações.

Na Tabela 7, podemos observar que a execução na *cloudlet* (com o *offloading de dados* ativado) depende do tamanho do arquivo e do filtro selecionado para que seja vantajosa. Para os filtros *Emboss* e *Engrave* só devem ser migrados os arquivos de tamanho maiores ou iguais a 192KB. O tempos na cor vermelha indicam que para dado filtro e para dado tamanho de arquivo não vale a pena realizar a sua migração para a *cloudlet*. Já os tempos na cor verde indicam que para esses filtros e esses tamanhos de arquivos a migração contribui para o ganho de tempo.

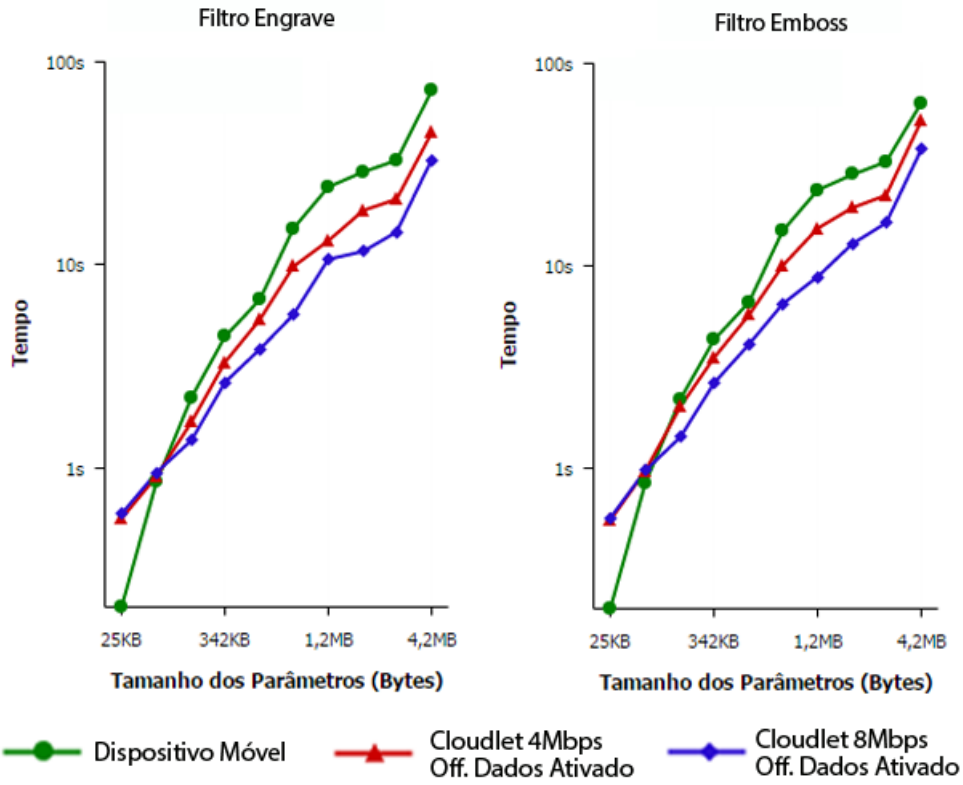
Pode-se observar nos dois primeiros experimentos que a migração de todos os arquivos não é vantajosa, pois existem situações onde a execução dos filtros no dispositivo móvel é mais rápida ou a própria sobrecarga do mecanismo torna o *offloading de processamento* mais

Tabela 7 – Comparação dos tempos de *offloading de processamento* (em milissegundos) com a execução no dispositivo móvel.

| **EXECUÇÃO** | | **TAMANHO DOS ARQUIVOS** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **FILTRO** | **DISPOSITIVO** | **25KB** | **90KB** | **192KB** | **342KB** | **524KB** | **870KB** | **1,2MB** | **1,5MB** | **1,9MB** | **4,2MB** |
| INVERTER | Cloudlet (4Mbps Offloading de Dados Ativado) | 416 | 673 | 1232 | 2524 | 3816 | 7852 | 9174 | 12363 | 14597 | 32311 |
| | Cloudlet (8Mbps Offloading de Dados Ativado) | 381 | 620 | 855 | 1765 | 2586 | 3676 | 6206 | 8392 | 9853 | 22660 |
| | Dispositivo Móvel | **47** | **202** | **528** | **1063** | **1608** | **3615** | **5676** | **6703** | **7552** | **18012** |
| EMBOSS | Cloudlet (4Mbps Offloading de Dados Ativado) | 550 | 966 | 2012 | 3498 | 5703 | 9975 | 15281 | 19356 | 22285 | 51985 |
| | Cloudlet (8Mbps Offloading de Dados Ativado) | 565 | 970 | **1421** | **2621** | **4082** | **6452** | **8812** | **12857** | **16256** | **37542** |
| | Dispositivo Móvel | **201** | **840** | 2192 | 4290 | 6603 | 14802 | 23631 | 28209 | 32615 | 63758 |
| ENGRAVE | Cloudlet (4Mbps Offloading de Dados Ativado) | 560 | 905 | 1678 | 3297 | 5363 | 9775 | 13111 | 18498 | 21037 | 44678 |
| | Cloudlet (8Mbps Offloading de Dados Ativado) | 590 | 944 | **1364** | **2625** | **3833** | **5653** | **9557** | **11619** | **14512** | **32901** |
| | Dispositivo Móvel | **205** | **849** | 2218 | 4410 | 6734 | 15050 | 24138 | 28659 | 32688 | 72077 |
| SNOW | Cloudlet (4Mbps Offloading de Dados Ativado) | 381 | 539 | 934 | 1962 | 3203 | 5039 | 6745 | 9879 | 11119 | 27168 |
| | Cloudlet (8Mbps Offloading de Dados Ativado) | 399 | 532 | 723 | 1271 | 2125 | 2642 | 4074 | 4884 | 6995 | 16280 |
| | Dispositivo Móvel | **6** | **29** | **78** | **156** | **235** | **549** | **870** | **1040** | **1186** | **2683** |

Fonte: o autor.

Figura 18 – Comparativo do tempo de processamento dos filtros *Engrave* e *Emboss* com o *offloading de dados* ativado e no dispositivo móvel.




Fonte: o autor.

lento. É nesse cenário que a seleção dos arquivos se faz necessária. Assim, a fim averiguar a eficácia da tomada de decisão do mecanismo, foi gerada uma árvore com os dados extraídos dos Experimentos #1 e #2 (que pode ser observada na Figura 20).

Pelos resultados dos dois experimentos, esperava-se que o mecanismo fizesse a seleção de arquivos maiores ou iguais a 192KB e apenas para os filtros *Emboss* e *Engrave*. A Tabela 8 apresenta o resultado do Experimento #3, onde pode-se perceber que a seleção foi feita de forma esperada.

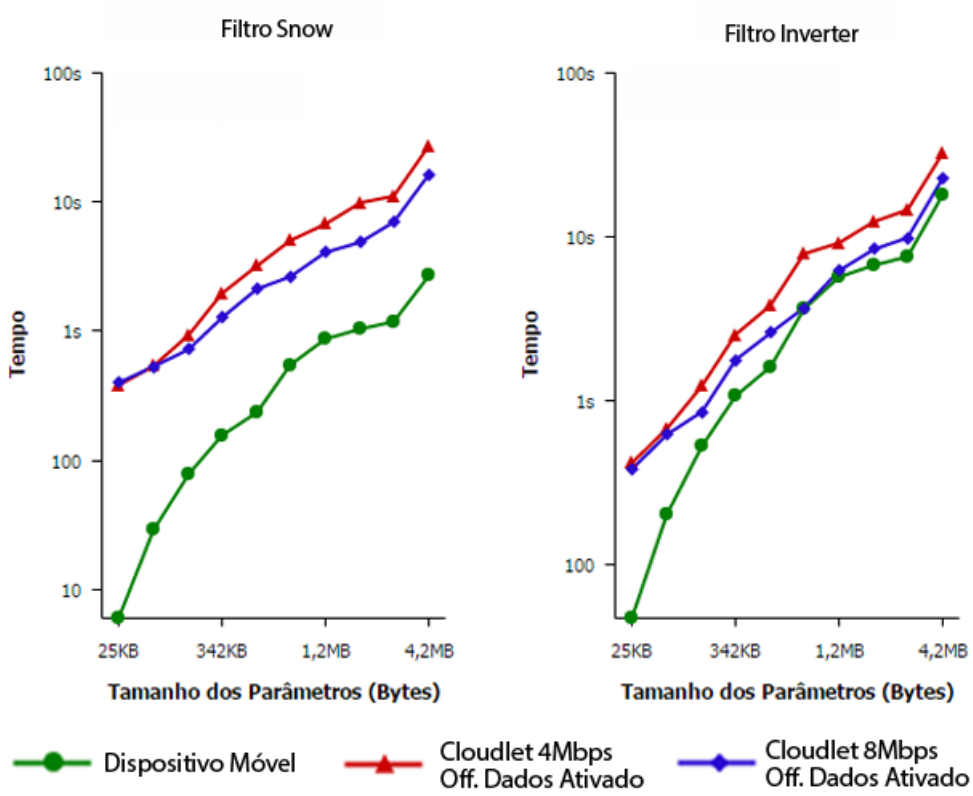
Tabela 8 – Arquivos selecionados pelo mecanismo de tomada de decisão.

| | Tamanho dos Arquivos | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Filtro** | **25KB** | **90KB** | **192KB** | **342KB** | **524KB** | **870KB** | **1,2MB** | **1,5MB** | **1,9MB** | **4,2MB** |
| Inverter | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO |
| Emboss | NÃO | NÃO | SIM | SIM | SIM | SIM | SIM | SIM | SIM | SIM |
| Engrave | NÃO | NÃO | SIM | SIM | SIM | SIM | SIM | SIM | SIM | SIM |
| Snow | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO | NÃO |

Fonte: o autor.

Os seguintes itens podem ser caracterizados como as principais ameaças à validação

Figura 19 – Comparativo do tempo de processamento dos filtros *Snow* e *Inverter* com o *offloading de dados* ativado e no dispositivo móvel.



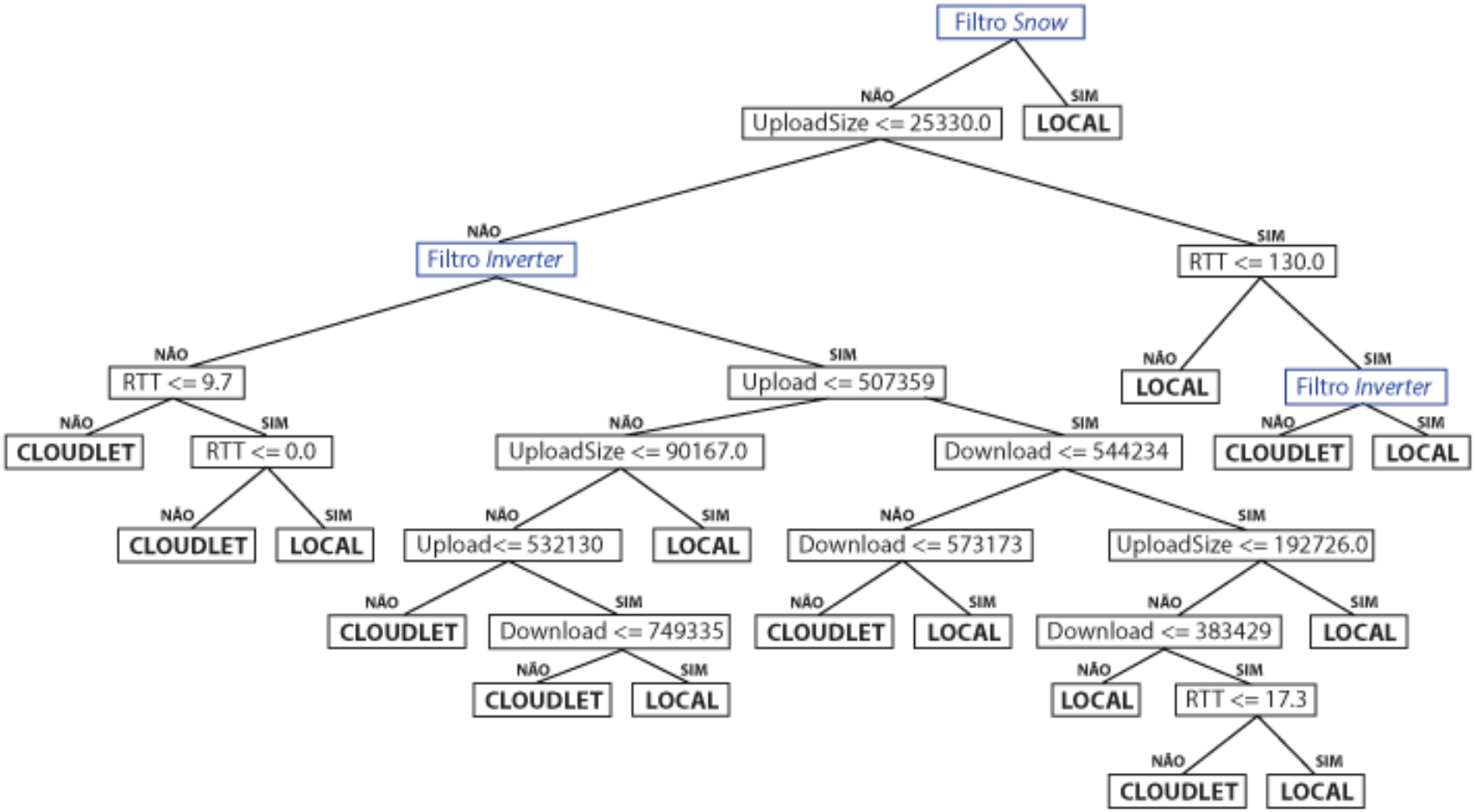

Fonte: o autor.

dos resultados deste trabalho: *(i)* Existe a necessidade de ampliar o ambiente de testes com uma maior variação de configurações e quantidade de dispositivos móveis; *(ii)* Para fins de redução no escopo, o experimento utilizou aplicativos que realizam operações em imagens. Ampliar os testes com outros tipos de aplicativos proporcionaria uma visão mais precisa dos possíveis ganhos ao utilizar o método proposto.

## 4.6 Considerações finais do capítulo

Neste capítulo, mostramos os resultados obtidos com a execução dos experimentos propostos. No experimento #1 observamos que a migração dos arquivos economiza tempo em 80% dos casos, economizando, em média 19,5% do tempo de *offloading de processamento*. No entanto, observamos que para arquivos menores que 192KB não é vantajoso realizar a migração dos arquivos. No Experimento #2 observamos que o *offloading de processamento* dos filtros consome menos tempo do que a execução no dispositivo móvel em 40% dos casos implicando assim na necessidade de seleção de quais arquivos devem ser migrados. Por fim, temos o Experimento #3 que testa a eficácia da tomada de decisão do mecanismo usando como base os

Figura 20 – Árvore gerada a partir do histórico de execuções dos filtros de imagem no dispositivo móvel e na *cloudlet*




Fonte: o autor.

resultados dos experimentos #1 e #2 para selecior corretamente os arquivos que deveriam ser migrados.

Para a evolução dos experimentos, consideramos incluir, em trabalhos futuros, variações no tráfego da rede já que os dispositivos utilizados para os experimentos fucionaram exclusivamente para os testes propostos. Tendo em mente a avaliação do mecanismo em outros aspectos, pode-se analisar a eficiência energética do mecanismo ou, também, verificar se a realização de processamento distribuído do mecanismo pode contribuir para a diminuição do tempo das operações de *offloading de processamento*.

# 5 CONCLUSÃO

Nesta dissertação, propomos um mecanismo de auxílio a *frameworks* de *offloading de processamento* na persistência dos arquivos utilizados como parâmetros na invocação de métodos. Com base nos resultados discutidos na Seção 4.5, a hipótese levantada na Seção 1.2 pode ser confirmada pois o mecanismo proposto diminuiu o tempo de *offloading* em 19,5% (em média). O mecanismo proposto leva em conta a tomada de decisão sobre quais arquivos devem ser migrados e, de acordo com os resultados obtidos, podemos concluir que não se deve migrar tráfego do tipo "rato"(i.e. tráfego composto por uma pequena quantidade de pacotes). Esse tipo de tráfego ocorre com maior frequência mas representa uma pequena parcela do tráfego total (HE *et al.*, 2015). De forma complementar e suportando nossa conclusão, notamos que (i) o *offloading de processamento* é mais rápido quando o mecanismo de *offloading de dados* encontra-se desativado, e *(ii)* os filtros de imagem para envolvem esses tamanhos de arquivos são executados mais rapidamente no dispositivo móvel.

A principal contribuição desta dissertação é a utilização de técnicas de seleção e tomada de decisão no *offloading de dados* para dar suporte ao *offloading de processamento*, o que propicia redução no tempo de execução das tarefas executadas fora do dispositivo móvel. Como foi observado na Seção 4.5, o mecanismo foi capaz de selecionar e migrar os arquivos que representavam ganho de tempo para o mecanismo de *offloading de processamento*.

## 5.1 Publicação

JOARI S. L. FILHO, MANUEL G. da S. NETO, PAULO A. L. REGO, DANIELO G. GOMES. Um Mecanismo de Offloading de Dados com Tomada de Decisão. Anais do 16º Workshop em Desempenho de Sistemas Computacionais e de Comunicação. São Paulo, CSBC, 2017 (FILHO *et al.*, 2017).

## 5.2 Trabalhos futuros

Como perspectiva de estudos futuros, sugere-se a criação de perfis de acesso aos arquivos e perfis de funcionalidades utilizadas para que a tomada de decisão leve em consideração não só o histórico de todo o período de acesso, mas também o comportamento do usuário em intervalos de tempo mais curtos. Sugere-se ainda, com base nos resultados obtidos e nas ameaças à validação, a ampliação dos casos de uso e quantitativo de experimentos a fim de proporcionar

uma visão mais ampla e precisa dos ganhos obtidos pelo uso do mecanismo e de sua aplicação em ambientes não controlados.

# REFERÊNCIAS

ALI, F. A.; SIMOENS, P.; VERBELEN, T.; DEMEESTER, P.; DHOEDT, B. Mobile device power models for energy efficient dynamic offloading at runtime. **Journal of Systems and Software**, Elsevier, v. 113, p. 173–187, 2016.

ARMSTRONG, T.; TRESCASES, O.; AMZA, C.; LARA, E. de. Efficient and transparent dynamic content updates for mobile clients. In: ACM. **Proceedings of the 4th international conference on Mobile systems, applications and services**. [S.l.], 2006. p. 56–68.

BAHL, P.; HAN, R. Y.; LI, L. E.; SATYANARAYANAN, M. Advancing the state of mobile cloud computing. In: ACM. **Proceedings of the third ACM workshop on Mobile cloud computing and services**. [S.l.], 2012. p. 21–28.

BAHRAMI, A.; WANG, C.; YUAN, J.; HUNT, A. The workflow based architecture for mobile information access in occasionally connected computing. In: IEEE. **Services Computing, 2006. SCC'06. IEEE International Conference on**. [S.l.], 2006. p. 406–413.

CHUN, B.-G.; IHM, S.; MANIATIS, P.; NAIK, M.; PATTI, A. Clonecloud: elastic execution between mobile device and cloud. In: ACM. **Proceedings of the sixth conference on Computer systems**. [S.l.], 2011. p. 301–314.

COSTA, P. B.; REGO, P. A. L.; ROCHA, L. S.; TRINTA, F. A. M.; SOUZA, J. N. de. Mpos: A multiplatform offloading system. In: **30th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2015. p. 577–584.

CUERVO, E.; BALASUBRAMANIAN, A.; CHO, D.-k.; WOLMAN, A.; SAROIU, S.; CHANDRA, R.; BAHL, P. Maui: making smartphones last longer with code offload. In: ACM. **Proceedings of the 8th international conference on Mobile systems, applications, and services**. [S.l.], 2010. p. 49–62.

DINH, H. T.; LEE, C.; NIYATO, D.; WANG, P. A survey of mobile cloud computing: architecture, applications, and approaches. **Wireless communications and mobile computing**, Wiley Online Library, v. 13, n. 18, p. 1587–1611, 2013.

ENZAI, N. I. M.; TANG, M. A taxonomy of computation offloading in mobile cloud computing. In: IEEE. **Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on**. [S.l.], 2014. p. 19–28.

FERNANDO, N.; LOKE, S. W.; RAHAYU, W. Mobile cloud computing: A survey. **Future Generation Computer Systems**, v. 29, p. 84–106, 2013.

FILHO, J. S. L.; NETO, M. G. da S.; REGO, P. A. L.; GOMES, D. G. Um mecanismo de offloading de dados com tomada de decisão. In: **Anais do 16º Workshop em Desempenho de Sistemas Computacionais e de Comunicação – WPeformance**. [S.l.: s.n.], 2017.

FLINN, J.; PARK, S.; SATYANARAYANAN, M. Balancing performance, energy, and quality in pervasive computing. In: IEEE. **Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on**. [S.l.], 2002. p. 217–226.

FLINN, J.; SINNAMOHIDEEN, S.; TOLIA, N.; SATYANARYANAN, M. Data staging on untrusted surrogates. In: **Proceedings of the 2Nd USENIX Conference on File and Storage Technologies**. Berkeley, CA, USA: [s.n.], 2003. p. 15–28.

FLORES, H.; HUI, P.; TARKOMA, S.; LI, Y.; SRIRAMA, S.; BUYYA, R. Mobile code offloading: from concept to practice and beyond. **IEEE Communications Magazine**, IEEE, v. 53, n. 3, p. 80–88, 2015.

GOMES, D. G.; TOLOSANA-CALASANZ, R.; AGOULMINE, N. Introduction to special issue on green mobile cloud computing (green mcc). **Sustainable Computing: Informatics and Systems**, v. 8, p. 37 –, 2015.

GOMES, F. A. A.; REGO, P. A. L.; ROCHA, L.; SOUZA, J. N. de; TRINTA, F. Caos: A context acquisition and offloading system. In: **2017 IEEE 41th Annual Computer Software and Applications Conference (COMPSAC)**. [S.l.: s.n.], 2017. v. 1.

GOMES, F. A. A.; VIANA, W.; ROCHA, L. S.; TRINTA, F. A Contextual Data Offloading Service With Privacy Support. In: **WebMedia**. Teresina-PI , Brazil: Sociedade Brasileira de Computação, 2016.

HE, K.; ROZNER, E.; AGARWAL, K.; FELTER, W.; CARTER, J.; AKELLA, A. Presto: Edge-based load balancing for fast datacenter networks. **SIGCOMM Computer Communications**, ACM, New York, NY, USA, v. 45, n. 4, p. 465–478, 2015.

HUNG, S.-h.; SHIH, C.-s.; SHIEH, J.-p.; LEE, C.-p.; HUANG, Y.-h. Executing mobile applications on the cloud : Framework and issues. **Computers and Mathematics with Applications**, Elsevier Ltd, v. 63, n. 2, p. 573–587, 2012.

IYER, A. N.; ROOPA, T. Extending android application programming framework for seamless cloud integration. In: IEEE. **Mobile Services (MS), 2012 IEEE First International Conference on**. [S.l.], 2012. p. 96–104.

KEMP, R.; PALMER, N.; KIELMANN, T.; SEINSTRA, F.; DROST, N.; MAASSEN, J.; BAL, H. eyedentify: Multimedia cyber foraging from a smartphone. In: IEEE. **Multimedia, 2009. ISM'09. 11th IEEE International Symposium on**. [S.l.], 2009. p. 392–399.

KHAN, A. R.; OTHMAN, M.; MADANI, S. A.; KHAN, S. U. A survey of mobile cloud computing application models. **Communications Surveys & Tutorials**, IEEE, v. 16, n. 1, p. 393–413, 2014.

KISTLER, J. J.; SATYANARAYANAN, M. Disconnected operation in the coda file system. **ACM Transactions on Computer Systems (TOCS)**, ACM, v. 10, n. 1, p. 3–25, 1992.

KOUKOUMIDIS, E.; LYMBEROPOULOS, D.; STRAUSS, K.; LIU, J.; BURGER, D. Pocket cloudlets. **ACM SIGPLAN Notices**, v. 47, n. 4, p. 171, 2012.

KUMAR, K.; LIU, J.; LU, Y.-H.; BHARGAVA, B. A survey of computation offloading for mobile systems. **Mobile Networks and Applications**, Springer, v. 18, n. 1, p. 129–140, 2013.

KUNDU, S.; MUKHERJEE, J.; MAJUMDAR, A. K.; MAJUMDAR, B.; RAY, S. S. Algorithms and heuristics for efficient medical information display in pda. **Computers in Biology and Medicine**, Elsevier, v. 37, n. 9, p. 1272–1282, 2007.

LEWIS, G.; LAGO, P. Architectural tactics for cyber-foraging: Results of a systematic literature review. **Journal of Systems and Software**, Elsevier, v. 107, p. 158–186, 2015.

PHOKAS, T.; EFSTATHIADES, H.; PALLIS, G.; DIKAIAKOS, M. D. Feel the world: A mobile framework for participatory sensing. In: SPRINGER. **International Conference on Mobile Web and Information Systems**. [S.l.], 2013. p. 143–156.

QI, H.; GANI, A. Research on mobile cloud computing: Review, trend and perspectives. In: IEEE. **Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on**. [S.l.], 2012. p. 195–202.

QUINLAN, J. R. Decision trees and decision-making. **IEEE Transactions on Systems, Man, and Cybernetics**, IEEE, v. 20, n. 2, p. 339–346, 1990.

QUINLAN, J. R. C4. 5: Programming for machine learning. **Morgan Kauffmann**, p. 38, 1993.

REGO, P. A. L. **Applying Smart Decisions, Adaptive Monitoring and Mobility Support for Enhancing Offloading Systems**. Tese (Doutorado) — Universidade Federal do Ceará, Fortaleza, Brasil, 12 2016.

REGO, P. A. L.; CHEONG, E.; COUTINHO, E. F.; TRINTA, F. A.; HASAN, M. Z.; SOUZA, J. N. de. Decision tree-based approaches for handling offloading decisions and performing adaptive monitoring in MCC systems. In: **2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)**. [S.l.: s.n.], 2017.

SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. The case for vm-based cloudlets in mobile computing. **IEEE Pervasive Computing**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 8, n. 4, p. 14–23, 2009.

SCHÜRING, M. Mobile cloud computing–open issues and solutions. In: **15thTwente Student Conference on IT, Enschede, The Netherlands**. [S.l.: s.n.], 2011.

SHIRAZ, M.; SOOKHAK, M.; GANI, A.; SHAH, S. A. A. A Study on the Critical Analysis of Computational Offloading Frameworks for Mobile Cloud Computing. **Journal of Network and Computer Applications**, v. 47, p. 47–60, 2015.

Silva Jr., L. S. d.; MAGALHÃES, D. M. V.; GOMES, D. G. Modelagem e simulação de offloading para computação móvel em nuvem. In: **Anais / XXXV Congresso da Sociedade Brasileira de Computação**. Porto Alegre: SBC, 2015. p. 91–100.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. Introduction to data mining. v. 74, 2006.

WU, X.; KUMAR, V.; QUINLAN, J. R.; GHOSH, J.; YANG, Q.; MOTODA, H.; MCLACHLAN, G. J.; NG, A.; LIU, B.; PHILIP, S. Y. *et al.* Top 10 algorithms in data mining. **Knowledge and information systems**, Springer, v. 14, n. 1, p. 1–37, 2008.

XIAO, Y.; SIMOENS, P.; PILLAI, P.; HA, K.; SATYANARAYANAN, M. Lowering the barriers to large-scale mobile crowdsensing. In: ACM. **Proceedings of the 14th Workshop on Mobile Computing Systems and Applications**. [S.l.], 2013. p. 9.

YANG, F.; QIAN, Z.; CHEN, X.; BESCHASTNIKH, I.; ZHUANG, L.; ZHOU, L.; SHEN, J. Sonora: A platform for continuous mobile-cloud computing. **Technical Report. Microsoft Research Asia, Tech. Rep**, 2012.

ZHAO, B.; XU, Z.; CHI, C.; ZHU, S.; CAO, G. Mirroring smartphones for good: A feasibility study. In: SPRINGER. **International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services**. [S.l.], 2010. p. 26–38.