Uma abordagem sensível a contexto para tomada de decisão de *offloading* de processamento

Pedro P. Alves[1], **Paulo A. L. Rego** [1], **Fernando A. M. Trinta**[1]

[1]Mestrado em Ciência da Computação – Universidade Federal do Ceará (UFC)
Campus do Pici - Bloco 910 - Fortaleza - CE - 60440-900

pedroalves@great.ufc.br,{paulo,fernando.trinta}@dc.ufc.br

Abstract. *Applications for mobile devices have evolved to enable the most diverse activities. To assist the devices, the computational offloading technique has been used to migrate tasks to cloud servers with superior processing power or greater energy capacity. However, when such infrastructure is unavailable or when communication latency is an obstacle to the execution of offloading, a strategy that can be used is offloading for a remote execution environment (REE), which may include other nearby mobile devices. The choice of a REE is a complex process, as the heterogeneity of the devices interferes with the completion time of the migrated tasks. This work presents a method to select a REE that considers the user's context. To validate the developed solution, a software architecture was designed and implemented, and experiments were performed.*

Resumo. *As aplicações para dispositivos móveis tem evoluído possibilitando a realização das mais diversas atividades. Para auxiliar os dispositivos, a técnica de offloading computacional tem sido utilizada para migrar tarefas para servidores em nuvem com poder de processamento superior ou maior capacidade energética. Entretanto, quando indisponível tal infraestrutura ou quando a latência da comunicação é um empecilho para a execução do offloading, uma estratégia que pode ser utilizada é o offloading para um ambiente remoto de execução (do inglês, remote execution environment - REE), que pode ser inclusive outros dispositivos móveis próximos. A escolha de um REE é um processo complexo, pois a heterogeneidade dos dispositivos interfere no tempo de conclusão das tarefas migradas. Este trabalho apresenta um método para selecionar um REE que considera o contexto do usuário. Para validar a solução desenvolvida, uma arquitetura de software foi projetada e implementada, e experimentos foram executados.*

1. Introdução

Já há algum tempo, a computação móvel é uma realidade no cotidiano da sociedade moderna. Smartphones e tablets permitem acesso fácil e rápido a dados e aplicações para auxílio de diversas tarefas. Ao longo dos anos, melhorias no hardware e a diversificação de sensores têm permitido a execução de aplicações inovadoras e complexas, como realidade aumentada e jogos. Apesar desta evolução e aprimoramentos em dispositivos móveis, estes equipamentos ainda são considerados dispositivos de computação limitados.

Em resposta a esta questão, uma técnica que cada vez mais tem sido estudada é o *offloading* de processamento. De acordo com [Akherfi et al. 2018], o *offloading* supera

diversas limitações de dispositivos móveis como baixa vida útil pelo uso de bateria, recursos de processamento e capacidade de armazenamento limitada, através da transferência da execução de tarefas de dispositivos mais fracos para outros com melhor desempenho e recursos mais avançados.

Atualmente, a grande popularidade dos *smartphones* e a crescente melhoria na qualidade das conexões sem fio possibilita um cenário onde ao estar indisponíveis servidores que atendam solicitações de offloading ou onde a latência seja um empecilho, possamos utilizar recursos computacionais advindos dos próprios dispositivos na mesma WLAN para auxiliar a execução de tarefas por parte dos dispositivos mais fracos. [Varghese and Buyya 2018]. Dentre os desafios presentes para que esse cenário seja plausível estão os de: (1) Segurança na comunicação, (2) Motivação para a particiáção e a (3) Escolha do melhor dispositivo, dada a heterogeneidade e dinamicidade dos dispositivos.

Este trabalho tem foco no terceiro desafio citado, abordando tal problemática ao propor um algoritmo sensível ao contexto para a seleção do melhor local para realizar o *offloading* de métodos em aplicações móveis. Aqui, a sensibilidade ao contexto se refere à capacidade da solução interagir com ambientes dinâmicos, nos quais recursos e dispositivos computacionais mudam com o tempo, exigindo que o algoritmo se adapte a essa dinamicidade para continuar atingindo seus objetivos. Com base no algoritmo proposto, uma arquitetura de software foi projetada para a plataforma Android a partir de duas soluções existentes, chamadas CAOS [Gomes et al. 2017] e CAOS D2D [Dos Santos et al. 2018], o que permitiu realizar experimentos que comprovaram a eficácia da solução proposta.

2. Tomada de Decisão de *Offloading* Sensível ao Contexto

A realização de *offloading* entre dois dispositivos pode ter diversos objetivos. Em geral, boa parte dos trabalhos propostos na literatura utilizam *offloading* para a melhoria no desempenho na execução de tarefas ou processos. Porém, é possível encontrar também soluções que objetivam economia de energia, compartilhamento de dados, realização de tarefas comuns, dentre outros [Trinta et al. 2017].

Estabelecer o objetivo de um processo de *offloading* é de fundamental importância para a tomada de decisão em um cenário com múltiplos possíveis locais para migração de uma tarefa. No contexto deste trabalho, foram definidos dois possíveis objetivos: (*i*) minimizar o tempo de execução de uma determinada tarefa e (*ii*) preservar a autonomia (tempo de vida) da bateria dos dispositivos REEs.

A escolha destes objetivos permite a definição de parâmetros ou métricas de interesse que impactam na escolha do local para migração de uma tarefa. Porém, é importante frisar que estes valores são voláteis. Por exemplo, dependendo da carga de trabalho que um dispositivo esteja executando (seja por utilização do usuário do dispositivo ou atendimento a solicitações de *offloading*), este pode apresentar variações na sua velocidade de processamento para finalizar certa atividade. Com isso, informações do contexto de execução dos dispositivos devem ser levadas em consideração para escolha do local de migração de uma tarefa.

Neste trabalho, foram estabelecidos seis parâmetros que devem ser considerados na escolha do *offloading*. São eles: (*i*) Poder de processamento do dispositivo, (*ii*)

Métricas de conexão de rede, (*iii*) Número de requisições de *offloading*, (*iv*) Nível de Bateria, (*v*) Decisão do usuário e (*vi*) Decisão do desenvolvedor. Tais parâmetros são mais detalhados nas subseções a seguir.

Poder de Processamento

Benchmarks são comumente adotados quando se necessita realizar comparações entre diversos objetos (e.g., para verificar o poder de processamento de dispositivos móveis) [Guo et al. 2017]. Em geral, tais métodos funcionam através do cálculo do tempo de execução de uma tarefa atribuída aos dispositivos, com posterior classificação dos dispositivos de acordo com cada tempo de finalização, onde os dispositivos que finalizarem primeiro são considerados os de maior poder de processamento.

Baseado na ideia de *benchmarking*, este trabalho utilizou a velocidade de processamento da função de criptografia SHA-1 para classificar os dispositivos. O SHA-1 funciona de forma a transformar uma mensagem de qualquer tamanho em 40 dígitos hexadecimais, por meio da aplicação de diversos procedimentos nesta mensagem [Eastlake 3rd and Jones 2001]. A escolha do SHA-1 se deu pela necessidade de um método de rápida execução, uma vez que ocupar o processamento do dispositivo por um longo período de tempo prejudicaria o atendimento de requisições de *offloading*. Portanto, através da execução 30 mil vezes desse método é obtido um valor em nanosegundos que é então atribuído em forma de *score* de dois dígitos para o dispositivo.

Métricas de rede

No processo de *offloading*, a qualidade da conexão entre dispositivo cliente e REE é fundamental para se determinar o quão eficiente será a migração. Para o monitoramento da conexão de rede, dois parâmetros foram utilizados. O primeiro é o *Round Time Trip* (RTT) entre cada REE e uma entidade pré-estabelecida. Para aferição deste valor, um conjunto de mensagens é trocado entre cada REE e esta entidade. Os tempos de envio das mensagens são salvos e auxiliam no calculo da taxa de transferência. Além disso, também é calculada a média entre o tempo decorrido do envio das mensagens, considerando tal tempo como a taxa de RTT do momento.

O segundo parâmetro de conexão de rede tem o objetivo de abordar a dinamicidade dos REEs na rede, através do *Received Signal Strength Indicator* (RSSI) do dispositivo. O monitoramento do RSSI é realizado tanto com o propósito de auxiliar o método de seleção na escolha de um REE com melhor qualidade de conexão, como para estabelecer uma taxa de confiança com a qual pode-se verificar se um REE está ou não na "borda" da rede, e com isso possa subitamente se tornar indisponível aos clientes de *offloading*. Portanto, o valor em *dBm* da conexão *Wi-Fi* é capturado e normalizado em uma escala de 0 a 100 para identificação da qualidade do sinal. O valor de 30% foi definido como limite de aceitação do dispositivo na rede, pois este valor é próximo de -90 dBm o que torna extremamente falha qualquer comunicação na rede.

Número de requisições em um REE

Em um ambiente com múltiplos clientes e REEs, é possível que solicitações de *offloading* concorrente sejam alocadas a um mesmo REE. Nesse caso, a possibilidade de decaimento no tempo de processamento é grande. Com isso, busca-se evitar enviar solicitações concorrentes a um mesmo REE.

Aqui, a carga de trabalho para cada REE é monitorada não apenas pelo `score`, mas também pelo número de requisições que este está atendendo. Este monitoramento é feito por meio de um contador (chamado de `NProcessos`) que mantém o mapeamento do número de requisições atendidas por cada dispositivo. Sempre que um REE é selecionado para atender a uma requisição, seu contador é incrementado. Quando este REE retorna a resposta do *offloading* para o dispositivo cliente, decrementa-se o contador.

Nível de Bateria

O tempo de funcionamento de um dispositivo móvel é uma das maiores preocupações dos usuários. Apesar dos dispositivos mais modernos possuírem recursos sofisticados para economizar energia, esta ainda é um recurso finito que precisa ser utilizado com moderação [Li et al. 2018].

Usar um dispositivo móvel como REE implica em um consumo de bateria a mais. Com isso, é importante que se monitore o nível de energia dos dispositivos de modo a considerar este valor no processo de seleção do REE mais adequado para o envio de solicitações de *offloading*. Portanto, dentre os parâmetros monitorados, inclui-se também o nível de bateria do dispositivo. Além disso, assume-se que uma estratégia a não penalizar dispositivos com baixa carga de bateria é priorizar REEs de dispositivos móveis conectados à uma fonte de energia (ie, em carregamento). Com isso, tais dispositivos são identificados atribuindo aos mesmos nível de bateria igual a 101%.

Escolha dos Usuários

Por envolver dispositivos de outros usuários para a execução do *offloading*, é importante atribuir poder de decisão a estes usuários, uma vez que seus dispositivos serão utilizados para o processamento de tarefas de terceiros, causando um maior consumo energético. Para tanto, duas decisões básicas foram implementadas para dar o direito de participação ou não destes colaboradores na rede.

Primeiramente, a possibilidade de ativação e desativação do serviço foi implementada, permitindo que o proprietário do dispositivo decida quando estar disponível ou não como REE. Desta forma, pode-se realizar programaticamente o envio de um alerta para ter ciência da desconexão do dispositivo. Uma segunda possibilidade é a atribuição, através da interface do usuário, de um limite mínimo de bateria no qual o usuário estaria disposto a oferecer seu dispositivo para processamento de métodos advindos via *offloading*. Deste modo, o proprietário não precisa se preocupar que o serviço consuma sua bateria quando esta já estiver num percentual considerado baixo pelo mesmo. Este limite é utilizado no algoritmo de decisão para remoção dos dispositivos que já atingiram o percentual de bateria indicado pelo seu proprietário, respeitando sua decisão.

Escolha do Desenvolvedor

O presente trabalho também possibilita ao desenvolvedor fazer escolhas sobre o objetivo a ser atingido quando a aplicação deseja realizar *offloading* de uma tarefa. Com esse propósito, a *flag* (`ASAP`) foi criada para permitir que o desenvolvedor da aplicação indique sua intenção. Caso esta *flag* seja definida como verdadeira, o critério de seleção será a escolha do REE com capacidade de processar mais rapidamente o pedido de *offloading*. Caso contrário, a escolha deve levar em conta os níveis de bateria de cada dispositivo disponível, ou se os mesmos encontram-se conectados a uma fonte de energia.

2.1. Algoritmo de Seleção do REE

O Algoritmo 1 descreve o processo de seleção proposto para escolha do melhor REE para atender à solicitação de *offloading*. Para realizar esta escolha, o processo recebe como entrada a lista de todos os REEs capazes de processar pedidos de *offloading* (`REEsDisponíveis`), e a *flag* que determina o objetivo geral do *offloading* (`ASAP`).

Algoritmo 1: Seleção de REE para *offloading*

```
Function selecionarMelhorREE (REEs_Disponíveis, ASAP) :
    REEs_Conectados = new List();
    for each dispositivo ∈ REEs_Disponíveis do
        if dispositivo.bateria < LIMITE_BATERIA then
            REEs_Disponíveis.remove(dispositivo);
        end
        if dispositivo.rssi < LIMITE_RSSI then
            REEs_Disponíveis.remove(dispositivo);
        end
        if dispositivo.conectado then
            REEs_Conectados.add(dispositivo);
        end
    end
    pesos = vetor_Pesos_Processamento;
    lista_REE = REEs_Disponíveis;
    if not ASAP then
        if listaDM_Conectados.size() > 0 then
            lista_REE = REEs_Conectados;
        end
        else
            pesos = vetor_Pesos_Bateria;
        end
    end
    return REE_escolhido = TOPSIS(pesos, lista_REE);
EndFunction
```

A primeira etapa do algoritmo faz uma varredura na lista de REEs e descarta aqueles cujos dispositivos atingiram o limite de bateria determinado pelo proprietário e aqueles com baixo RSSI. Esta varredura também separa os dispositivos que estejam conectados à energia em uma lista separada (`REEs_Conectados`), que inicialmente encontra-se vazia.

Com a lista atualizada, a *flag* `ASAP` definirá os parâmetros que serão repassados à função `TOPSIS` para escolha do melhor `REE`. Se o objetivo for o processamento mais rápido possível (`ASAP` = *true*), devem ser considerados todos os dispositivos, sendo o critério de escolha aquele de melhor processamento dentre todos (`vetor_Pesos_Processamento`). Caso contrário (`ASAP` = *false*), o critério de escolha será aquele de melhor processamento entre os REEs conectados (`vetor_Pesos_Processamento`). Se não houver nenhum REE conectado, então o critério será aquele com maior reserva de energia (`vetorPesos_Bateria`) entre os REEs que não foram descartados na primeira varredura.

O `TOPSIS` (*Technique for Order Preference by Similarity to Ideal Solution*) é um método que busca resolver problemas de decisão através da classificação das alternativas e da seleção que mais chega próxima de uma solução ideal [Hwang et al. 1993]. Este

método funciona a partir da construção de uma matriz relacionando cada objeto que se deseja avaliar com suas características. Desta forma, os objetos avaliados neste trabalho são os dispositivos (REEs) e as suas características são os parâmetros coletados anteriormente (Nprocessos, Score, Bateria, RTT, RSSI). A cada parâmetro é atribuído um peso que indica sua importância na seleção. Para isso, recorremos ao método `AHP` (*Analytic Hierarchy Process*), bastante utilizado na literatura em conjunto com o `TOPSIS` para resolver problemas de tomada de decisão nas mais diversas áreas [Bangui et al. 2017].

O método `AHP` é utilizado para auxílio na tomada de decisão baseada em multicritérios, em que os atributos dos objetos a serem comparados possuem uma relação de hierarquia entre si [Saaty 1990]. Neste trabalho, tal método nos fornece os vetores de pesos que são utilizados no algoritmo de seleção (`vetor_Pesos_Processamento` e `vetorPesos_Bateria`), que servem de entrada para o método `TOPSIS`, sendo estes construídos de acordo com o objetivo da execução do *offloading*.

As Tabelas 1 e 2 apresentam os valores para a priorização do tempo de processamento e do balanceamento energético, respectivamente. Elas foram elaboradas de acordo com o método AHP utilizando os atributos de nossa solução, priorizando-se um sobre o outro para obtenção dos autovetores normalizados, que são utilizados no Algoritmo 1.

Tabela 1. Parametrização que prioriza o tempo de processamento.

	NProcessos	**Score**	**RTT**	**RSSI**	**Bateria**		**Autovetor**		**Autovetor Normalizado**
NProcessos	1	2	5	7	9		3,629678		0,445885377
Score	1/2	1	5	7	9		2,750782		0,337917926
RTT	1/5	1/5	1	3	9		1,015511		0,124749804
RSSI	1/7	1/7	1/3	1	9		0,571986		0,070265179
Bateria	1/9	1/9	1/9	1/9	1		0,172427		0,021181715
Soma	1,9540	3,4540	11,4444	18,1111	37,0000		8,1404		1

Tabela 2. Parametrização que prioriza o balanceamento energético.

	Bateria	**NProcessos**	**Score**	**RTT**	**RSSI**		**Autovetor**		**Autovetor Normalizado**
Bateria	1	5	5	7	9		4,359695		0,54356335
NProcessos	1/5	1	2	5	7		1,695218		0,211358473
Score	1/5	1/2	1	5	7		1,284735		0,160179769
RTT	1/7	1/5	1/5	1	3		0,443421		0,055285431
RSSI	1/9	1/7	1/7	1/3	1		0,237513		0,029612978
Soma	1,6540	6,8429	8,3429	18,3333	27,0000		8,0206		1

Os pesos atribuídos seguiram a escala definida por [Saaty 1990], onde o valor de relevância de um atributo sobre outro deve variar de 1 a 9. Deste modo, os valores foram ajustados nesse intervalo de acordo com diversas execuções da solução e observação de sua eficácia na seleção dos dispositivos de acordo com o objetivo a ser alcançado.

2.2. Infraestrutura para *Offloading* entre múltiplos dispositivos móveis

O suporte ao algoritmo proposto na Subseção 2.1 teve como ponto de partida duas soluções desenvolvidas na Universidade Federal do Ceará. A primeira, denominada CAOS (*Context Acquisition and Offloading System*) [Gomes et al. 2017] é um *framework* para o desenvolvimento de aplicações sensíveis ao contexto utilizando a plataforma Android, e que permite o *offloading* de dados contextuais e de processamento para um ambiente em nuvem, ou para um *cloudlet* na mesma WLAN do dispositivo móvel cliente.

Já o CAOS D2D (*CAOS Device to Device*) [Dos Santos et al. 2018] é uma extensão proposta ao *framework* CAOS que permite efetuar *offloading* entre dispositivos móveis, sem a figura de um servidor central. A Figura 1 apresenta a arquitetura do sistema a partir da integração das duas soluções, dividida em três partes: *CAOS API*, *CAOS Controller* e *CAOS REE*. Esta divisão reflete os três principais componentes da solução proposta, respectivamente: (*i*) o dispositivo cliente que solicita o *offloading* de um método, (*ii*) o servidor que decide qual o melhor local para executar remotamente este processamento, e (*iii*) os dispositivos-alvo que ofertam ambientes para execução remota dos métodos.

O *CAOS API* utiliza a mesma estratégia do CAOS e CAOS D2D, onde o desenvolvedor pode especificar a intenção de *offloading* de um método por meio de anotações (*@Offloadable*) nos métodos das classes da aplicação. As demais funções dos módulos da *CAOS API* vão desde a descoberta de uma infraestrutura servidora de *offloading* compatível com o CAOS/CAOS D2D na mesma WLAN que o dispositivo móvel, até o monitoramento da execução da aplicação móvel. Quando um método marcado for chamado, é disparado uma solicitação de *offloading* ao *CAOS Controller*.

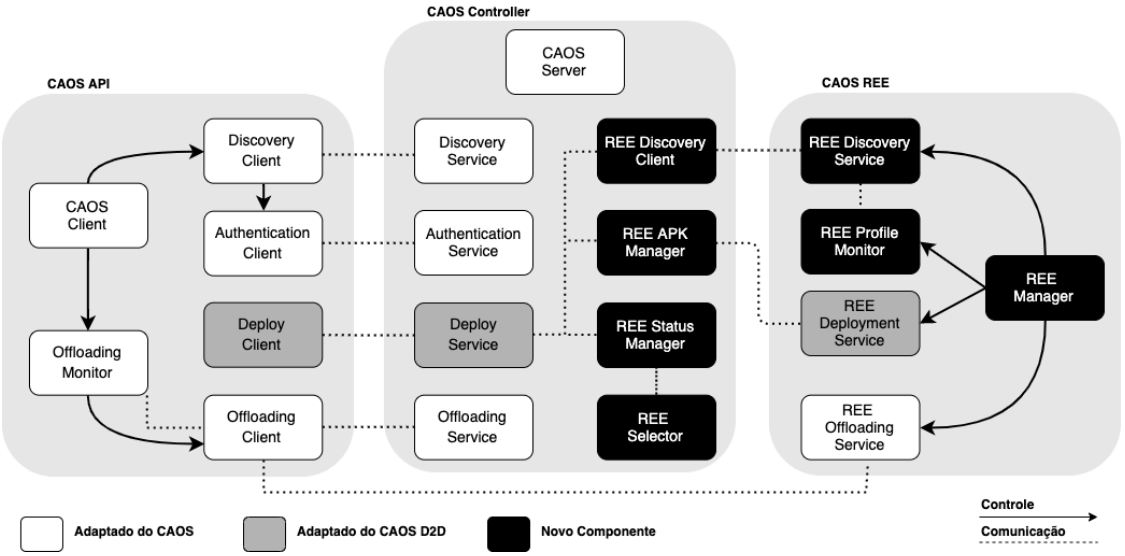

Figura 1. Arquitetura para Seleção de Dispositivos para *Offloading*

O módulo *Deploy Client*, advindo do CAOS D2D, é responsável pelo envio das dependências da aplicação cliente para o servidor. Estas dependências incluem os arquivos binários (.APK) que representam a aplicação. Os demais módulos do *CAOS API* foram herdados do projeto CAOS e são responsáveis pela comunicação da aplicação com o servidor. Tais módulos sofreram pequenas modificações para se adaptarem ao novo ambiente de execução, com destaque para o módulo *Offloading Client*. Este módulo é responsável por execução do *offloading*, e na versão aqui proposta, se comunica tanto com o *CAOS Controller* quanto com o dispositivo escolhido para realizar o *offloading*.

Já o *CAOS Controller* é responsável por monitorar os dispositivos do ambiente que oferecem um REE para *offloading*. Este componente tem o papel central de decidir qual o melhor dispositivo que pode atender ao pedido de *offloading* de um cliente. O *CAOS Controller* possui módulos correspondentes e de suporte aos serviços de descoberta, autenticação, migração de métodos e instalação de dependências existentes no

CAOS API. Para possibilitar a tomada de decisão, quatro novos módulos foram implementados. O *REE Discovery* permite que dispositivos que queiram ofertar seus recursos na forma de um REE possam se registrar no *CAOS Controller*. O *REE APK Manager* gerencia se os REEs disponíveis já possuem as dependências necessárias para execução remota dos métodos, e se necessário, faz o envio destes arquivos para cada REE. O *REE Status Manager* é responsável por receber informações de cada REE periodicamente, como o *score* obtido pelo dispositivo, nível de carga de bateria, dentre outros, e que serão utilizadas para a tomada de decisão pelo *REE Selector*.

Por fim, o *CAOS REE* é o responsável pela execução do método e envio do resultado para a aplicação cliente. Dentre seus principais módulos, o *REE Manager* é responsável por controlar a inicialização dos outros componentes. O *REE Discovery Service* utiliza uma estratégia semelhante aos clientes (i.e., *broadcast*) para encontrar o *CAOS Controller* e se registrar como REE para operações de *offloading*. O *REE Profile Monitor* coleta internamente dados sobre o dispositivo e os envia ao *CAOS Controller*. O *REE Deployment Service* tem a função de receber os arquivos de dependências das aplicações para realizar a execução dos métodos remotos. Finalmente, quando um dispositivo-alvo é selecionado para executar um método, o *REE Offloading Service* recebe a solicitação diretamente do dispositivo cliente, a executa e repassa o resultado obtido.

3. Avaliação

Para avaliar a solução proposta, foram realizados três experimentos que buscaram: (1) verificar a assertividade do algoritmo de seleção na escolha do REE; (2) demonstrar a adaptação do algoritmo de seleção do REE à mobilidade dos dispositivos; e (3) demonstrar o funcionamento do algoritmo de seleção ao priorizar o balanceamento energético.

Durante os três experimentos foram utilizados os dispositivos apresentados na Tabela 3 e utilizou-se uma aplicação que permite a execução de multiplicação de matrizes criadas aleatoriamente com dimensões variadas. Esse tipo de aplicação tem sido utilizado em diversos trabalhos da área [Shiraz et al. 2015] [Dos Santos et al. 2018] [Rego et al. 2019] por permitir alterar a quantidade de dados trocados entre os dispositivos, além de exigir tempo de computação variável, de acordo com a dimensão da matriz selecionada (i.e., quanto maior a dimensão das matrizes, maior é o seu tempo de transferência pela rede, e maior é o tempo necessário para realizar a operação).

Tabela 3. Dispositivos utilizados nos experimentos.

Dispositivo	Modelo	Android	Processador	Memória	Disco
Servidor	Dell Inspiron 5557	-	Intel Core I7 2.5GHZ	8 GB	1 TB
Cliente	Galaxy S3 Slim Duos	4.2.2	Quad-Core 1.2GHZ	1 GB	64 GB
Dispositivo A	Xiaomi Mi 8 Lite	9.0	Octa-Core 2.2GHZ	6 GB	128 GB
Dispositivo B	Galaxy J7 Neo	9.0	Octa-Core 1.3GHZ	2 GB	16 GB
Dispositivo C	Moto Z2 Play	9.0	Octa-Core 2.2GHZ	4 GB	64 GB
Dispositivo D	Tablet Samsung SM-T560	4.4.4	Octa-Core 1.3GHZ	1,5 GB	8 GB
Dispositivo E	One Plus 5T	9.0	Octa-Core 2.2GHZ	8 GB	128 GB
Dispositivo F	Xiaomi Pocophone F1	9.0	Octa-Core 2.3GHZ	6 GB	128 GB

3.1. Experimento 1

O objetivo deste experimento é verificar a assertividade do algoritmo proposto quanto à seleção do melhor REE para atender à requisição de *offloading* quando se deseja executar

o método o mais rápido possível. Para isso, foi criada uma variação da API do CAOS com a capacidade de enviar solicitações simultâneas de *offloading* para múltiplos dispositivos. Desta forma, pôde-se verificar se o REE escolhido pelo algoritmo de seleção foi realmente aquele com capacidade de atender mais rápido à solicitação do dispositivo cliente.

Durante o experimento, o dispositivo cliente executou 30 vezes a operação de multiplicação com matrizes de dimensões 600×600 em duas baterias de testes. Na primeira bateria, os dispositivos foram dispostos lado a lado, com telas desligadas e sem interação do usuário. Na segunda bateria, os proprietários dos dispositivos ficaram livres para utilizarem os dispositivos, simulando um ambiente real, somente com a restrição de não saírem da rede de cobertura. Em ambas as baterias, os testes realizados foram executados em dois ambientes distintos, nos quais os dispositivos que atuaram como REE são representados por dois conjuntos. No Conjunto 1 estão contidos os dispositivos A, B, C e D, enquanto os dispositivos C, D, E e F representam o Conjunto 2.

Nesse experimento, foram utilizadas três métricas: (*i*) acerto, quando o algoritmo seleciona corretamente o melhor dispositivo para atender à requisição de *offloading*; (*ii*) acerto perfeito, onde é observado se além do primeiro colocado o algoritmo consegue acertar a ordem exata em que os REEs irão responder às requisições; e (*iii*) exceções, quando ocorre alguma falha de comunicação com algum dos REEs, o que impossibilita a definição da ordem em que os dispositivos respondem às requisições de *offloading*.

A Figura 2 apresenta o resultado da primeira bateria de testes do experimento (dispositivos dedicados), onde pode-se observar que com o primeiro conjunto de dispositivos foi alcançada uma taxa de acerto de 96,66%, enquanto a taxa de acerto perfeito ficou em 80%, e em 2,5% dos casos ocorreu alguma exceção. Já o segundo conjunto de testes foi mais desafiador, pois dois dos REEs mais potentes possuíam poder de processamento bem similar, de modo a deixar a disputa bem acirrada. Como resultado, obteve-se 83% de acerto do melhor REE, 76,6% de acerto perfeito e repetiu-se 2,5% de exceções causadas por alguma falha de comunicação.

A diminuição da porcentagem de acertos no segundo conjunto de testes é atribuída à similaridade entre os dispositivos E e F da Tabela 3. Entretanto, em todos os casos em que o algoritmo não acertou o melhor REE, o REE selecionado foi aquele com o segundo melhor desempenho. A diferença média de tempo de execução entre o primeiro REE a responder à requisição de *offloading* e o REE selecionado pelo algoritmo foi de aproximadamente 0,7 segundos.

A Figura 3 apresenta o resultado da segunda bateria de testes (REEs não dedicados), onde pode-se observar que, com o primeiro conjunto de dispositivos, obteve-se uma taxa de 93,33% de acerto na seleção do melhor REE, 43,33% de acertos perfeitos e 5,83% de exceções causadas por alguma falha de comunicação. Já com os dispositivos do segundo conjunto, obteve-se 56,66% de acerto, 43,33% de acerto perfeito e 3,33% de exceções causadas por alguma falha de comunicação.

A diminuição do percentual de acerto na segunda bateria de testes é atribuída ao intervalo de aquisição do contexto dos dispositivos (60 segundos), pois dentro desse tempo, devido à atividade do usuário, o contexto do dispositivo pode não ser mais o mesmo do adquirido, quando solicitada uma requisição de *offloading*. Desta forma, os cálculos para decisão do melhor dispositivo são realizados com valores desatualizados,

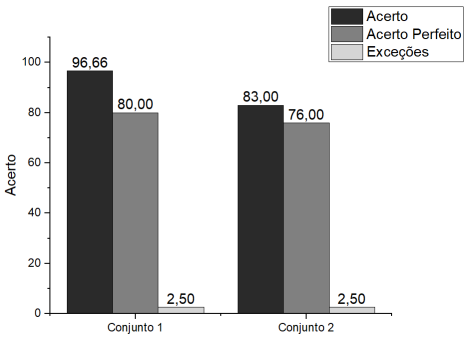

Figura 2. REEs dedicados.

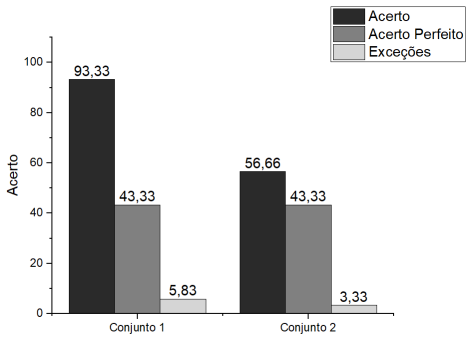

Figura 3. REEs não dedicados.

provocando uma decisão errônea. No segundo conjunto de dispositivos, a redução do número de acertos foi mais atenuada devido à similaridade entre os dispositivos E e F. Ainda assim, quando errônea a seleção, o dispositivo selecionado pelo algoritmo esteve sempre em segundo lugar em tempo de resposta, sendo a diferença do tempo médio de resposta entre o primeiro e o segundo dispositivo de apenas 2,2 segundos.

3.2. Experimento 2

O objetivo deste experimento é demonstrar que a solução proposta se adapta à presença dos REEs de acordo com suas conexões e desconexões com o dispositivo servidor, de modo a indicar o melhor REE disponível no momento para realização do *offloading*.

Para isto, foi considerado o resultado do experimento anterior para o conjunto de dispositivos A, B, C e D, em que pôde-se observar que a seleção para *offloading* acompanha o poder de processamento dos 4 dispositivos, onde $P(A) > P(C) > P(B) > P(D)$, sendo $P(X)$ o poder de processamento do dispositivo X. Dessa forma, intercalamos aleatoriamente a presença dos dispositivos no ambiente com o intuito de verificar se a solução se adapta à mobilidade (disponibilidade) dos dispositivos para escolher o melhor REE dentre os que estão presentes em dado momento.

A Figura 4 representa um gráfico temporal, onde são destacadas as conexões e desconexões entre os REEs e o dispositivo `servidor`, e a adaptação do *REE* escolhido de acordo com as requisições de *offloading* a cada momento. Para facilitar a visualização, os dispositivos estão representados de acordo com seu poder de processamento no eixo Y, sendo o mais alto o de maior poder computacional. Cada dispositivo representa seu tempo de conexão com o `servidor` através de uma linha, enquanto um hexágono presente na linha indica que o REE foi escolhido para atender a uma solicitação de *offloading*, representada pela esfera na dimensão tempo. O hexágono presente no início do eixo X, dentro da esfera de solicitação de *offloading*, representa que o processamento foi realizado localmente na primeira solicitação, pois não haviam dispositivos presentes naquele momento para atender à requisição, como pode ser observado na figura.

3.3. Experimento 3

O objetivo deste experimento é demonstrar o funcionamento do algoritmo de seleção do REE ao priorizar o balanceamento do consumo energético. Para isso, foram utilizados os dispositivos cliente, A, B, C e D, e a aplicação de multiplicação de matrizes (com a *flag*

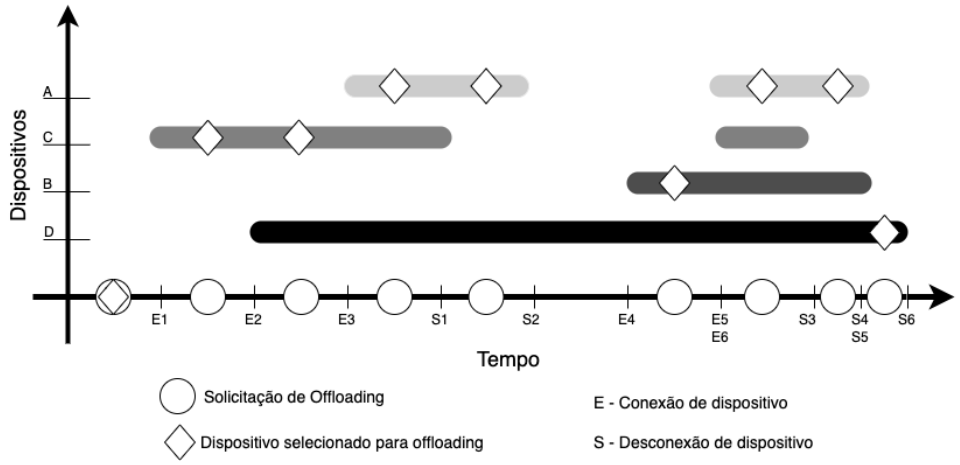

Figura 4. Escolha do REE de acordo com o contexto.

$ASAP = false$) foi executada 30 vezes. Os seguintes casos foram considerados para a avaliação do algoritmo: (*i*) somente um REE conectado a uma fonte de energia; (*ii*) mais de um REE conectado a uma fonte de energia; e (*iii*) nenhum REE conectado a uma fonte de energia.

O primeiro caso de testes é trivial, pois o algoritmo não precisa calcular qual é o melhor dispositivo naquele momento, pois basta selecionar o único em estado "carregando". No segundo caso, é selecionado o REE com maior poder de processamento dentre aqueles em estado "carregando". Dessa forma, o algoritmo de seleção utilizou os pesos obtidos do método AHP com foco em velocidade de processamento, e comportou-se semelhante ao Experimento 1 na seleção dos dispositivos.

No terceiro e último caso, o algoritmo utiliza o vetor de pesos para balancear a energia em todo o conjunto de REEs disponíveis. Através dos testes, percebeu-se que nem sempre aquele de maior percentual de bateria era o escolhido, pois no caso em que dois ou mais dispositivos possuem valores próximos de bateria, o escolhido é o de melhor poder de processamento. Esse comportamento era esperado devido a seleção multicritério que está sendo aplicada, uma vez que cada parâmetro tem seu poder de decisão no algoritmo. Portanto, para valores próximos de bateria, outros parâmetros irão fazer a diferença na escolha. Neste trabalho, este comportamento é considerado válido uma vez que une o balanceamento de consumo energético e a melhoria da velocidade de processamento. A Tabela 4 representa os dados de um caso ocorrido durante os testes, onde o dispositivo selecionado pelo algoritmo foi o C. Em valores absolutos, o dispositivo C não é o de maior carga de bateria (93%), porém este foi selecionado por possuir capacidade energética próxima ao do dispositivo com maior capacidade (96%) e por superar o mesmo em capacidade de processamento, pois seu *score* é menor.

Tabela 4. Parâmetros dos REEs no caso de teste (iii) do Experimento 3

Dispositivo	Bateria	NProcessos	Score	RTT	RSSI	Valor TOPSIS
A	56	0	5	95,2	100	0,41
B	81	0	15	69,5	100	0,47
C	93	0	11	84,0	100	0,72
D	96	0	17	64,2	100	0,58

4. Trabalhos Relacionados

Alguns trabalhos encontrados na literatura [Habak et al. 2015] [Zhou et al. 2016] [Chilukuri et al. 2017] [Basic et al. 2019] têm abordado a seleção do melhor REE para *offloading* levando em consideração diversas métricas, sendo as principais: o cálculo da capacidade computacional, consumo energético, escolhas de usuários e desenvolvedores, e a heterogeneidade e mobilidade (dinamicidade) dos dispositivos.

Em [Habak et al. 2015], os autores desenvolveram um protótipo de sistema com o objetivo de prover uma nuvem móvel dinâmica e autoconfigurável formada por um conjunto de dispositivos móveis. O sistema estima a capacidade computacional de cada REE ao capturar o número de núcleos de processamento e processos em execução, além de considerar o histórico de utilização do dispositivo. Informações de cada REE são compartilhadas com um dispositivo de controle na rede, que é responsável por estimar o tempo de presença do usuário e configurar a participação do REE como um oferecedor de serviço. O trabalho depende de uma medição prévia da capacidade computacional, que é considerada sempre a mesma e não depende do contexto do dispositivo. Além disso, os autores não consideram consumo energético e as escolhas do desenvolvedor.

Os autores de [Zhou et al. 2016] propõem um modelo baseado em pesos para selecionar o melhor REE considerando seu status e estabilidade na rede para o provimento de serviços, onde os REEs são classificados de acordo com sua distância ao dispositivo central, sua cobertura de rede e sua dispersão na rede. O objetivo do trabalho é melhorar a qualidade de serviço provido pelos dispositivos móveis, uma vez que a saída deste dispositivo da rede antes do termino do processamento que lhe foi designado impacta negativamente no serviço. A solução considera apenas as escolhas do usuário, com base no horário de partida informado, não levando em consideração as opções do desenvolvedor, consumo de energia ou poder computacional dos dispositivos envolvidos.

Os autores de [Chilukuri et al. 2017] propõem uma heurística para seleção do REE que executará o *offloading*, onde uma pontuação é calculada para os dispositivos de acordo com seus recursos disponíveis, estabilidade na rede e necessidades da aplicação que solicita o *offloading*. O cálculo da pontuação utiliza pesos para os parâmetros de configuração, definidos pelo desenvolvedor da aplicação, que tem a liberdade de priorizar o que é mais importante para a execução de sua aplicação. Nosso trabalho utilizou o método AHP para fazer com que a escolha dos parâmetros pudesse ser mais genérica ao ponto de atender diferentes tipos de aplicações, além de considerar o consumo energético e as escolhas dos donos dos dispositivos.

O trabalho [Basic et al. 2019] propõe um algoritmo baseado em lógica *Fuzzy* para selecionar um REE para realização do processo de *offloading* computacional com base nos parâmetros de largura de banda, velocidade do processador e latência. É proposto também um controlador de transferência do processo de *offloading*, que leva em consideração o tempo de resposta da aplicação como indicador para decidir se deve mover a execução da tarefa para outro REE. Apesar de considerarem a dinamicidade dos dispositivos, os autores não especificam a abordagem utilizada para evitar falhas na comunicação. Além disso, as escolhas de usuários e desenvolvedores não são consideradas.

Para facilitar a comparação entre os trabalhos relacionados, a Tabela 5 sumariza os principais parâmetros considerados para fazer a seleção do REE para atender às

requisições de *offloading*.

Tabela 5. Comparação com os trabalhos relacionados

	Consumo energético	Capacidade computacional	Escolha do usuário	Escolha do desenvolvedor	Dinamicidade dos dispositivos	Heterogeneidade dos dispositivos
[Habak et al. 2015]		X	X		X	X
[Zhou et al. 2016]			X		X	X
[Chilukuri et al. 2017]		X		X	X	X
[Basic et al. 2019]		X			X	X
Este Trabalho	X	X	X	X	X	X

5. Conclusões

Apesar da melhoria na capacidade de processamento e no uso de estratégias mais eficientes para economia no consumo de bateria, dispositivos móveis ainda necessitam de auxílio para execução de certas aplicações móveis. O *offloading* de processamento é uma alternativa para situações em que haja a possibilidade de migrar tarefas de dispositivos mais fracos para infraestruturas mais poderosas. Em um ambiente composto por diversas opções para execução remota de algum processamento, escolher o melhor destino para migração não é uma tarefa simples, uma vez que além da heterogeneidade de tais dispositivos, o contexto de sua execução (e.g., quantidade de processos em execução, latência entre dispositivos, escolhas do usuário, dentre outros fatores) é relevante nesta seleção.

Este trabalho aborda este tema ao propor uma a infraestrutura onde possa ser possível a realização de *offloading* entre dispositivos móveis, com base em um algoritmo de seleção para escolher um ambiente de execução remoto a partir de objetivos pré-definidos pelos desenvolvedores de aplicações móveis, e considerando informações contextuais dos dispositivos heterogêneos que compõem o ambiente.

Como trabalhos futuros, pretende-se adaptar a solução proposta a outros dispositivos, como aqueles presentes em cenários de Internet das Coisas, uma vez que tais dispositivos também possuem restrições de capacidade computacional e dependem de fontes de energia limitadas.

Agradecimentos

Os autores gostariam de agradecem à Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) pelo apoio financeiro (Processo 6945087/2019).

Referências

Akherfi, K., Gerndt, M., and Harroud, H. (2018). Mobile cloud computing for computation offloading: Issues and challenges. *Applied Computing and Informatics*, 14:1–16.

Bangui, H., Ge, M., Buhnova, B., Rakrak, S., Raghay, S., and Pitner, T. (2017). Multi-criteria decision analysis methods in the mobile cloud offloading paradigm. *Journal of Sensor and Actuator Networks*, 6(4):25.

Basic, F., Aral, A., and Brandic, I. (2019). Fuzzy handoff control in edge offloading. In *2019 IEEE International Conference on Fog Computing (ICFC)*, pages 87–96. IEEE.

Chilukuri, S., Bollapragada, S., Kommineni, S., and Chakravarthy, K. (2017). Raincloud-cloudlet selection for effective cyber foraging. In *Wireless Communications and Networking Conference (WCNC), 2017 IEEE*, pages 1–6. IEEE.

Dos Santos, G. B., Trinta, F. A., and Rego, P. A. (2018). Impactos do offloading de processamento no tempo de execução e consumo energético de dispositivos m óveis. In *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

Dos Santos, G. B., Trinta, F. A. M., Rego, P. A. L., Silva, F. A., and De Souza, J. N. (2018). Performance and energy consumption evaluation of computation offloading using caos d2d. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7.

Eastlake 3rd, D. and Jones, P. (2001). Rfc3174: Us secure hash algorithm 1 (sha1).

Gomes, F. A., Rego, P. L., Rocha, L., Souza, J., and Trinta, F. (2017). Caos: A context acquisition and offloading system. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, pages 957–966, Los Alamitos, CA, USA. IEEE Computer Society.

Guo, Y., Xu, Y., and Chen, X. (2017). Freeze it if you can: Challenges and future directions in benchmarking smartphone performance. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*, pages 25–30. ACM.

Habak, K., Ammar, M., Harras, K. A., and Zegura, E. (2015). Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, pages 9–16. IEEE.

Hwang, C.-L., Lai, Y.-J., and Liu, T.-Y. (1993). A new approach for multiple objective decision making. *Computers & operations research*, 20(8):889–899.

Li, H., Liu, X., and Mei, Q. (2018). Predicting smartphone battery life based on comprehensive and real-time usage data. *arXiv preprint arXiv:1801.04069*.

Rego, P. A., Trinta, F. A., Hasan, M. Z., and de Souza, J. N. (2019). Enhancing offloading systems with smart decisions, adaptive monitoring, and mobility support. *Wireless Communications and Mobile Computing*, 2019.

Saaty, T. L. (1990). How to make a decision: the analytic hierarchy process. *European journal of operational research*, 48(1):9–26.

Shiraz, M., Gani, A., Shamim, A., Khan, S., and Ahmad, R. W. (2015). Energy efficient computational offloading framework for mobile cloud computing. *Journal of Grid Computing*, 13(1):1–18.

Trinta, F., Rego, P. A., Gomes, F. A., Rocha, L., and de Souza, J. N. (2017). Using mobile cloud computing for developing context-aware multimedia applications: A case study of the caos framework. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, WebMedia '17, pages 37–40, New York, NY, USA. ACM.

Varghese, B. and Buyya, R. (2018). Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861.

Zhou, A., Wang, S., Li, J., Sun, Q., and Yang, F. (2016). Optimal mobile device selection for mobile cloud service providing. *The Journal of Supercomputing*, 72(8):3222–3235.