Acelerando a identificação de impressões digitais através da técnica de offloading computacional

Renan A. Barbosa[1], João P. B. Andrade[1], Mauro R. C. da Silva[2], Francisco I. da S. Lima[1], Fernando A. M. Trinta[1] e Paulo A. L. Rego[1]

[1]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Universidade Federal do Ceará (UFC)
Fortaleza, Ceará, Brasil

[2]Instituto de Computação
Universidade Estadual de Campinas (Unicamp)
Campinas, São Paulo, Brasil

{renan.alves, igorlima_es, jpandrade}@alu.ufc.br,

mauro.silva@ic.unicamp.br, {fernando.trinta, paulo}@dc.ufc.br

Abstract. *Fingerprint recognition has been widely used to control people's access to controlled environments or devices, such as smartphones and laptops. Among other problems, the resources present in smartphones and smart locks (e.g., computational power and storage) are limited and make it impossible to use them for fingerprint identification tasks where the database to be compared is numerous. In this context, this work presents a case study of the adoption of the computational offloading technique to identify people through fingerprint readers and smartphones. We present the solution architecture and the experiments with Cloud and Cloudlet's support to improve the identification process. The results demonstrate that using the offloading technique compared to the execution on the device speeds up the fingerprint identification process by at least 85%.*

Resumo. *O reconhecimento de impressões digitais tem sido amplamente utilizado para liberar o acesso de pessoas a ambientes controlados ou dispositivos, como smartphones e notebooks. Dentre outros problemas, os recursos presentes em smartphones e fechaduras inteligentes (e.g., poder computacional e armazenamento) são limitados e inviabilizam a utilização destes para tarefas de identificação onde a base de dados a ser comparada é numerosa. Nesse contexto, este trabalho apresenta um estudo de caso da adoção da técnica de offloading computacional para a identificação de pessoas através do uso de leitores de impressões digitais e smartphones. São apresentados a arquitetura da solução e os experimentos realizados com apoio da Nuvem e Cloudlet para acelerar o processo de identificação. Os resultados são promissores e indicam que a técnica de offloading reduz o tempo de identificação de impressões digitais em pelo menos 85% quando comparado à execução no dispositivo móvel.*

1. Introdução

O uso de dispositivos móveis e suas aplicações trouxeram impacto em diversas atividades da sociedade. A Internet das Coisas e os sistemas ciber-físicos promovem uma

computação cada vez mais integrada ao ser humano, com cenários outrora vistos apenas em filmes de ficção-científica. Estes cenários integram dados obtidos por meio de diversos sensores, processamento distribuído entre diversos nós e integração de redes heterogêneas [Bordel et al. 2017].

Apesar dos avanços dos dispositivos móveis, boa parte destes equipamentos ainda possui limitações de processamento e armazenamento para lidar com aplicações que processem grandes quantidades de dados e necessitem fornecer respostas rápidas ao seus usuários. Além disso, dispositivos móveis ainda sofrem com questões de consumo de energia, uma vez que são alimentados por baterias com carga finita.

Um exemplo desta dificuldade é a identificação de indivíduos por biometria digital em um dispositivo móvel. Sistemas automatizados de identificação por impressão digital (AFIS, do inglês *Automated Fingerprint Identification Systems*) são exemplos de aplicativos de computação intensiva, uma vez que precisam processar grandes volumes de dados biométricos. Caso o número de indivíduos seja muito grande, possivelmente a base de dados não caberá completamente no dispositivo móvel. Mesmo cabendo, a identificação pode levar muito tempo para ser realizada [Ratha and Bolle 2003]. Esta questão vem sendo estudada em um projeto de pesquisa sobre segurança pública com objetivo de dotar policiais com recursos para identificar indivíduos durante suas rotinas de patrulhamento no estado do Ceará.

Uma proposta de solução ao problema é o uso da técnica de *offloading* computacional, em que dispositivos móveis com recursos limitados enviam suas tarefas de computação intensiva para uma infraestrutura com mais recursos, localizada na nuvem ou na borda da rede, para minimizar o tempo de processamento, bem como economizar energia [Rego et al. 2017]. Esta união entre computação móvel e recursos de nuvem é conhecida na literatura como *Mobile Cloud Computing* (MCC) [Kumar et al. 2013]. Devido ao impacto que a latência na transferência de processos e dados traz à MCC, diversos estudos tem proposto a utilização de infraestruturas mais próximas à borda da rede de modo a aumentar os ganhos com ações de *offloading*. Um deles é o conceito de *cloudlet* [Satyanarayanan et al. 2009], que representa um nó para execução ou armazenamento de dados repassados por dispositivos móveis conectados na mesma rede local sem fio.

Dada esta contextualização, este trabalho apresenta um estudo onde é proposta e avaliada uma arquitetura de software que aproveita a técnica de *offloading* computacional para acelerar o processo de identificação de impressões digitais em dispositivos móveis. O objetivo é verificar os ganhos de desempenho nas operações de identificação quando o *offloading* computacional realizado tanto com uso de *cloudlets* quanto de uma infraestrutura de nuvem pública, com tecnologias de comunicação distintas (Wi-Fi e LTE). Os experimentos realizados indicam resultados promissores, onde o tempo de identificação foi reduzido em 85% com o uso do *offloading* comparado ao mesmos casos de execução onde o *offloading*não foi realizado. Ademais, os resultados indicam que a solução proposta tem um bom desempenho ainda que 16 dispositivos móveis façam requisições de forma concorrente.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2, apresenta-se o conceito de identificação de pessoas por meio de impressões digitas. Na Seção 3 são discutidos os trabalhos relacionados ao reconhecimento biométrico e ao uso

da técnica de *offloading* computacional. Nas Seções 4 e 5, é descrita a solução proposta e os experimentos. Finalmente, as conclusões e trabalhos futuros são abordados na Seção 6.

2. Usando Impressões Digitais para Identificar Pessoas

Nesta Seção são introduzidos os principais conceitos para o entendimento do processo de identificação de pessoas através de impressões digitais e como esses conceitos se relacionam. Também apresenta-se um fluxo de trabalho para um sistema genérico de identificação de impressão digital.

O uso de impressões digitais como um tipo de biometria tem sido cada vez mais prevalente. Dentre as aplicações dessa tecnologia amplamente utilizada, é destacável a proteção de residências, investigações criminais e bloqueio/desbloqueio de *smartphones*. Algumas das vantagens do uso de impressões digitais são a precisão no reconhecimento e a capacidade de diferenciar indivíduos, o baixo custo dos dispositivos de leitura e a facilidade de uso [Alonso-Fernandez et al. 2009].

As impressões digitais são compostas por cristas e vales e, neste padrão natural, as minúcias são características discriminatórias entre os sujeitos [Moses et al. 2011]. Tais minúcias são anomalias nas cristas, onde duas principais se destacam na maioria dos algoritmos de verificação (ou *matching*): terminação e bifurcação da crista, que ser podem observadas na Figura 1.

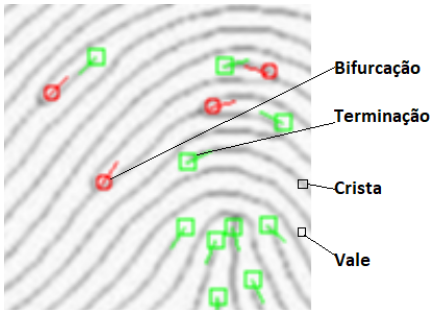


Figura 1. Tipos de minúcias em impressões digitais

O modelo ou *template* de minúcias é a representação mais amplamente usada de impressões digitais [Zaeri 2011]. Esta representação é possível devido aos algoritmos de extração de minúcias que processam a imagem da impressão digital para detectar minúcias e extrair informações como coordenadas (x, y), ângulo de direção entre a crista e o eixo horizontal, confiabilidade de minúcias e o tipo (bifurcação ou terminação). Essas informações podem ser relevantes para diferentes algoritmos de matching de impressão digital. A coleção de informações obtidas a partir de uma imagem de impressão digital e suas minúcias é chamada de *template* de impressão digital. Quando um *template* é extraído de uma impressão digital de boa qualidade, ele geralmente contém entre setenta e oitenta minúcias [Xiao and Raafat 1991].

A partir da extração dos *templates* de duas capturas de impressões digitais diferentes, é possível avaliar a similaridade entre elas usando os *templates* extraídos. Os algoritmos de *matching* usam as informações das minúcias para comparar ponto a ponto, usando diferentes estratégias de acordo com o algoritmo, e retornam uma pontuação para medir a similaridade entre as impressões digitais.

Um Sistema Automatizado de Identificação de Impressão Digital (*Automated Fingerprint Identification Systems* - AFIS) é uma metodologia de identificação biométrica que automaticamente adquire, armazena e compara imagens de impressão digital. Um AFIS exige rapidez e confiabilidade, utilizando diversos tipos de equipamentos e bases de impressão digital cada vez maiores. Este tipo de sistema é construído a partir de uma composição de componentes, como aquisição de impressão digital, pré-processamento na imagem para aplicar adaptações e aprimoramentos na impressão digital, extração de minúcias para geração, armazenamento e matching de *templates*.

Depois de registrar as impressões digitais e preencher o banco de dados, os principais usos de um AFIS são os processos de verificação $(1:1)$ e de identificação $(1:N)$. A verificação é a comparação de uma impressão digital com apenas uma outra impressão digital previamente armazenada para verificar a identidade de um indivíduo. Por sua vez, identificação é a comparação de uma impressão digital com um conjunto de impressões digitais armazenadas, com o objetivo de encontrar a identidade do sujeito comparado [Ratha and Bolle 2003].

O processo de identificação é a tarefa mais complexa e computacionalmente cara de um AFIS. A Figura 2 ilustra um fluxo genérico para realizar uma identificação de impressão digital, que consiste em cinco etapas: (1) captura da impressões digitais usando um dispositivo de leitura adequado, (2) pré-processamento, em que a área de interesse da imagem é definida e alguns aprimoramentos são realizados usando técnicas como binarização e afinamento. Na geração de template (3), as minúcias são detectadas e as informações são extraídas usando um extrator para comparação em (4) com a lista de *templates*, utilizando um algoritmo de matching na base de dados armazenada. Finalmente, (5) classifica por similaridade para retornar as N impressões digitais (valor definido pelo administrador do sistema) com pontuação mais alta, e consequentemente mais similares.

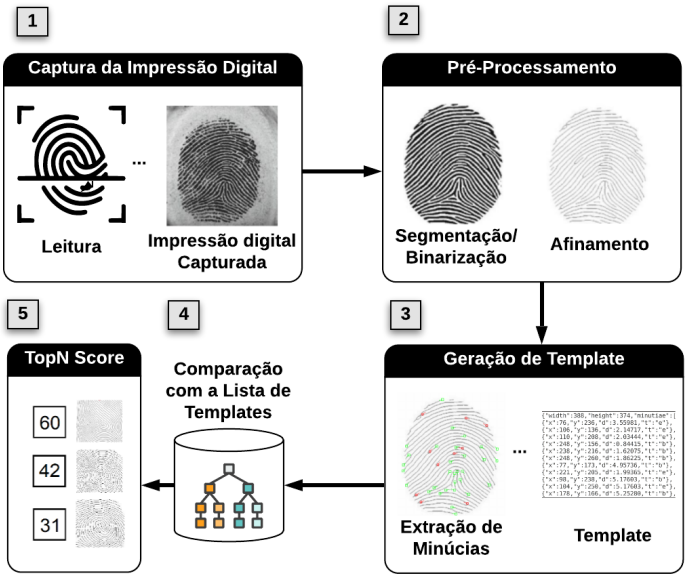


Figura 2. Fluxo do processo de identificação em um AFIS

Os experimentos desta pesquisa estão concentrados na tarefa de identificação, ignorando o estágio inicial de cadastro dos indivíduos no sistema. Portanto, presume-se que uma base de dados de impressão digital foi registrado previamente. Além disso, no AFIS aqui utilizado nos experimentos, foi acoplado uma versão levemente modificada do extrator de minúcias e do algoritmo de matching do *SourceAFIS* [1], modificações estas no sentido de adequar os algoritmos ao AFIS desenvolvido e utilizado.

3. Trabalhos Relacionados

Esta Seção discute alguns trabalhos sobre identificação de impressões digitais, em especial aqueles que propõem formas de acelerar a execução de tal tarefa, além de relatar trabalhos que utilizam *offloading* e computação móvel em aplicações biométricas.

Em [Peralta et al. 2014], os autores propuseram uma estrutura distribuída para *matching* de impressão digital no intuito de lidar com grandes bancos de dados em um tempo razoável, fornecendo um esquema geral para qualquer algoritmo de *matching* para diminuir o tempo de resposta, garantindo a precisão. O *framework*, baseado em paralelização de tarefas, foi amplamente testado com um extenso banco de dados de impressões digitais, envolvendo 400.000 imagens. Os resultados mostraram que o *framework* apresentou escalabilidade linear em relação ao banco de dados de impressões digitais, além de excelente adaptabilidade ao hardware subjacente.

Em [Zhao et al. 2016], é descrito um sistema de identificação de impressão digital distribuído com balanceamento de carga para lidar com a extração, armazenamento e acesso simultâneo aos *templates* armazenados. Para isso, foi desenvolvido um algoritmo de extração utilizando a biblioteca *Hadoop Image Processing Interface* para extrair minúcias de forma rápida e paralela, além do balanceamento de carga por meio do *cluster MongoDB*, permitindo acesso rápido a um grande número de *templates*. Por fim, um algoritmo de *matching* proposto em [Xu et al. 2014] foi utilizado em conjunto com a base de dados FVC2006.

O uso de informações biométricas para identificar pessoas é um tópico de pesquisa que vem sendo estudado há algum tempo sob diferentes abordagens, sendo que as mais conhecidas são as impressões digitais, reconhecimento da face e o da íris. No entanto, as tarefas de reconhecimento tendem a ser operações intensivas de computação, evitando que sejam executadas em dispositivos móveis de baixa potência. O *offloading* pode ajudar a aliviar a execução de tarefas de computação pesadas, enviando-as para serem processadas em uma infraestrutura diferente e mais robusta.

Para permitir o reconhecimento facial em tempo real em dispositivos móveis, os autores em [Powers et al. 2015] propuseram uma arquitetura que usa *cloudlet* e nuvem. Eles separam a tarefa de reconhecimento facial em diferentes fases, permitindo que cada uma seja executada em um componente diferente da arquitetura de acordo com os recursos disponíveis e a complexidade do processo.

Os autores em [Hassan and Elgazzar 2016] definiram uma arquitetura baseada em nuvem para aumentar a precisão e o tempo de desempenho de um aplicativo de reconhecimento de rosto para dispositivos móveis. Em sua arquitetura, as imagens obtidas pelos usuários foram pré-processadas localmente no dispositivo móvel e depois transfe-

[1] https://sourceafis.machinezoo.com/

ridas para um servidor em execução na nuvem. A comparação dos resultados dos testes que executam toda a tarefa de reconhecimento no dispositivo móvel com aqueles em que a tarefa usou *offloading* mostrou uma melhora no tempo de resposta geral e na economia de energia.

Em [Sajjad et al. 2020], os autores desenvolveram um *framework* para reconhecimento facial em tempo real, usando uma câmera anexada ao uniforme de policiais para transmitir vídeo para um *Raspberry Pi* próximo. Depois de receber os dados, o *Raspberry Pi* localiza o rosto do sujeito e executa uma extração de características na imagem. Dependendo da capacidade de computação estimada necessária para executar o processo de reconhecimento, o *framework* pode decidir executá-lo localmente ou transferi-lo para a nuvem.

Na Tabela 1, há uma comaparação dos trabalhos apresentados nesta Seção com o trabalho descrito neste artigo, consideranto três pontos chave: o tipo de biometria utilizada no trabalho, se houve utilização da técnica de *offloading* e, em caso positivo, onde ele foi realizado. Como é possível ver, este trabalho analisa um cenário que não foi contemplado por nenhum dos autores citados.

Tabela 1. Comparação com os trabalhos relacionados

Trabalho	Biometria	Usa Offloading?	Alvo Offloading
[Peralta et al. 2014]	Impressão Digital	Não	-
[Zhao et al. 2016]	Impressão Digital	Não	-
[Powers et al. 2015]	Facial	Sim	*Cloudlet* e Nuvem
[Hassan and Elgazzar 2016]	Facial	Sim	Nuvem
[Sajjad et al. 2020]	Facial	Sim	Nuvem
Este trabalho	Impressão Digital	Sim	*Cloudlet* e Nuvem

Apesar da importância da biometria facial ser relevante, o uso de impressões digitais ainda é a mais popular forma de biometria atualmente, o que motiva seu uso neste trabalho. Além disso, como vários estudos indicam a forte influência da latência de rede em soluções de *offloading*, o suporte a *cloudlets* é um requisito imprescindível. Além disso, este estudo se insere em um contexto já explicado de segurança pública, onde a biometria digital já é utilizada, e o conceito de *cloudlet* atende ao cenário de viaturas dotadas com recursos computacionais mais poderosos e que podem auxiliar policiais que trabalhem com dispositivos móveis com menos recursos, mais baratos e suscetíveis a substituição.

4. Identificação de impressões digitais com suporte de *offloading*

Esta Seção introduz a arquitetura do sistema desenvolvido e apresenta os detalhes do funcionamento do processo de *offloading* para a identificação de impressões digitais.

4.1. Visão geral da arquitetura

Inicialmente é preciso dividir a solução entre dois ambientes, sendo local e remoto, o ambiente local é composto de um ou mais dispositivos móveis, cada um deles conectado a um leitor de impressões digitais, e o ambiente remoto é composto de *cloudlets* virtualizadas e/ou instâncias de máquinas virtuais da nuvem, esta divisão é ilustrada da Figura 3.

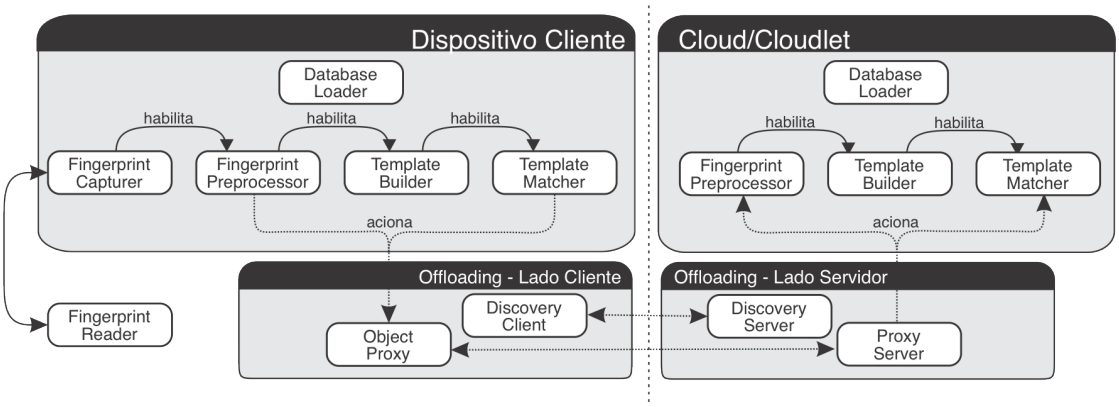


Figura 3. Visão geral da arquitetura proposta

Ainda sobre os diversos componentes englobados pela solução e que são ilustrados na Figura 3, o `Fingerprint Capturer` depende de um leitor de digitais externo conectado ao dispositivo móvel para capturar a imagem da digital, como ilustrado na Figura 4. O uso de dispositivos externos para leitura de digitais é necessário pois, por questões de segurança, a captura da imagem da impressão digital não é permitida em dispositivos Android. Ainda que fosse possível, os leitores embutidos apresentam uma menor área de captura quando comparados a leitores externos, o que dificulta a extração de muitas minúcias das impressões digitais.

Como descrito na Seção 2, a imagem capturada passa por diversos processos antes da criação do *template* de minúcias. O componente `Fingerprint Preprocessor` executa diversos algoritmos para preparar a imagem para o `Template Builder`, componente responsável por extrair as minúcias das digitais e gerar o respectivo *template*. Depois disso, o componente `Template Matcher` compara o *template* gerado com uma base local de *templates* candidatos. O componente `Database Loader` possui instâncias tanto no dispositivo móvel quanto na infraestrutura remota, e tem por objetivo carregar em memória uma base de *templates* de impressões digitais. A principal diferença entre estas instâncias é a quantidade de informações que cada uma armazena, onde espera-se que a instância do lado servidor tenha maior capacidade de armazenamento que nos dispositivos móveis.

Figura 4. Leitor de impressão digital externo conectado ao smartphone

A fim de permitir que a aplicação realize o *offloading* de diferentes tarefas, componentes específicos para dar suporte à migração de tarefas foram desenvolvidos. No ambiente local, o `Discovery Client` e o `Object Proxy` são os componentes com maior importância no processo de *offloading*.

O `Discovery Client` é responsável por estabelecer uma conexão com o servidor remoto. A conexão com os servidores pode ser feita automaticamente, usando um protocolo baseado em mensagens *broadcast* para descobrir os endereços IP dos servidores disponíveis, que estejam executando na mesma rede, ou pode ser feito a nível de programação onde será definido um nome ou endereço IP específico. Uma vez que a conexão é estabelecida, o componente `Object Proxy` é responsável por gerenciar a migração de tarefas do dispositivo móvel para o servidor remoto e também gerenciar a chegada dos resultados das tarefas recebidas do ambiente remoto.

O ambiente remoto consiste em um servidor com maior poder computacional e acesso a uma base maior de digitais candidatas. Nele, o `Discovery Server` responde às requisições de descoberta de dispositivos móveis, enquanto o `Proxy Server` recebe requisições de *offloading* e executa o componente apropriado para processar a requisição. Dependendo da etapa da identificação alvo do processo de *offloading*, o `Proxy Server` executa um dos componentes: `Fingerprint Preprocessor` em associação com o `Template Builder`, ou o `Template Matcher`, que executam as mesmas tarefas de suas versões do ambiente local.

4.2. Realizando o offloading do processo de identificação de digitais

Diante das limitações dos dispositivos móveis, não é possível salvar grandes bases de dados de *templates* em memória, e o processo de identificação pode demorar minutos para executar por completo de acordo com o equipamento utilizado. Além disso, o tempo necessário para carregar a base de *templates* na memória de um dispositivo móvel qualquer pode levar vários minutos para apenas algumas milhares de digitais.

A arquitetura proposta permite realizar o *offloading* de diferentes partes do processo. Como exemplo, é possível o dispositivo móvel executar os componentes `Fingerprint Preprocessor` e `Template Builder` localmente e executar o `Template Matcher` remotamente, ou ele pode enviar a imagem capturada para o servidor, e este realizar o preprocessamento, a geração do *template* e a comparação das impressões digitais remotamente.

Para implementar a arquitetura de software proposta e tornar o *offloading* possível, foi utilizado o *framework* de *offloading* MpOS [Costa et al. 2015]. Os componentes `Proxy Server` e `Discovery Server` do *framework* são executados em um ambiente remoto, enquanto no ambiente local foi implementado uma aplicação usando a biblioteca do MpOS para o sistema Android. A biblioteca permite que métodos sejam marcados para execução local ou remota via `Object Proxy` e também habilita o servidor de descoberta através do `Discovery Client`.

A biblioteca do MpOS usa a anotação `@Remotable` para marcar os métodos candidatos para serem executados remotamente. O Código 1 mostra um exemplo de dois métodos marcados com a anotação proposta. O método **identify** executa o processo de comparação das impressões digitais e leva dois argumentos: (*i*) um *template* obtido do `Template Builder` e, quando executado localmente, (*ii*) uma lista de *templates* carregados pelo componente `Database Loader`. O método **preprocessExtractAndIdentify** executa o preprocessamento da impressão digital e chama os componentes `Template Builder` e `Template Matcher`, passando a imagem obtida pelo componente `Fingerprint Capturer`.

A escolha do MpOS se deve ao fato dele ter sido desenvolvido no grupo de pesquisa dos autores. Mesmo assim, a arquitetura pode ser implementada usando outros *frameworks* de *offloading*, contanto que possuam funcionalidades similares às dos componentes descritos.

Código 1. Exemplo de anotação de métodos usando o framework MpOS

```
1 public interface OffloadingAFIS {
2
3     @Remotable
4     int identify(
5         String jsonTemplate,
6         List<FingerprintTemplate> candidateFingerprints
7     );
8
9     @Remotable
10    int preprocessExtractAndIdentify(
11        bytes[] image,
12        List<FingerprintTemplate> candidateFingerprints
13    );
14 }
```

5. Avaliação

Para avaliar a solução proposta, três cenários de testes foram definidos. O cenário descrito na Seção 5.1 é composto de um dispositivo real, um *smartphone* Motorola Moto E5 plus com 2GB de RAM, Cortex-A53 Qualcomm Snapdragon 425 MSM8917 (1.4 GHz Quad-Core), que executa o processo de identificação localmente.

Na Seção 5.2, usa-se o mesmo dispositivo e faz o *offloading* do processo de identificação para dois ambientes remotos, sendo uma instância de máquina virtual *t2.medium* (2 vCPUs, 4 GB de RAM e sistema operacional Ubuntu Server 18.04) executada em um centro de dados da Amazon na cidade de São Paulo e um *cloudlet*, usando uma rede LTE (58.000 kbps de *upload* e 173.000 kbps de *download*) e uma conexão *Wi-Fi*, respectivamente. Como *cloudlet*, foi utilizado um notebook com o sistema operacional Ubuntu Desktop 18.04, 4GB de RAM e Intel Core i5-7200U (2.50GHz e 4 cores).

Por último, na Seção 5.3, foi utilizado um *testbed* virtual de MCC [Barbosa and Rego 2019] para criar de 2 a 16 dispositivos móveis virtuais e executar a identificação de forma concorrente, a fim de estressar o servidor. O experimento foi executado em um servidor com sistema operacional Ubuntu Server 16.04, 32 GB de RAM e um processador Intel Xeon E5645.

Em cada cenário, foram realizados testes usando *templates* e imagens como dados de entrada do método de identificação. Ao utilizar *templates*, o preprocessamento da digital foi executado no dispositivo e a identificação foi realizada nos ambientes remotos. Por outro lado, ao utilizar imagens, ambas as tarefas foram executadas fora do dispositivo. O *templates* gerados tinham, em média, 6570 *bytes* e a imagens de impressões digitas tinham, em média, 297300 *bytes*. Por fim, o resultado da identificação tinha 107 *bytes*. Ademais, variou-se também o tamanho da base de digitais para comparação. No primeiro cenário, o número de digitais candidatas variou de 1000 a 6000 (máximo suportado pelo dispositivo móvel), enquanto nos cenários dois e três variou-se o tamanho da base de 2000 a 16000 digitais. Em todos os cenários, cada dispositivo executou o método de

identificação 30 vezes, depois foram calculados a média e o intervalo de confiança (com nível de confiança de 95%).

5.1. Primeiro Cenário - Dispositivo real executando a identificação localmente

A Figura 5 mostra a média e intervalo de confiança do tempo de execução para a tarefa de identificação executada localmente. Devido às limitações de memória do dispositivo móvel, no máximo 6000 digitais candidatas foram utilizadas. Nesse cenário, todo o processo é executado no dispositivo móvel a partir das imagens das impressões digitais.

Como pode-se observar, quão maior é o tamanho da base a ser comparada, mais lenta é a tarefa de identificação. Em todos os casos, realizar a tarefa de identificação no dispositivo móvel demorou mais de 15 segundos.

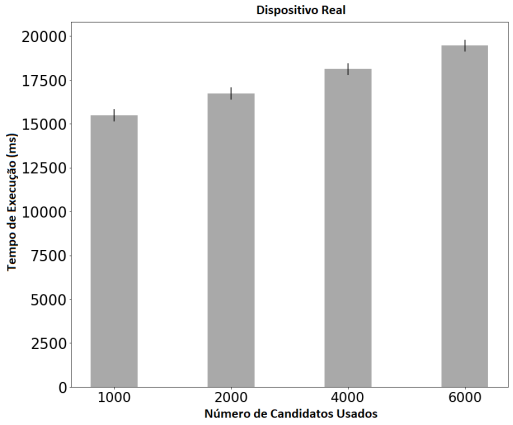


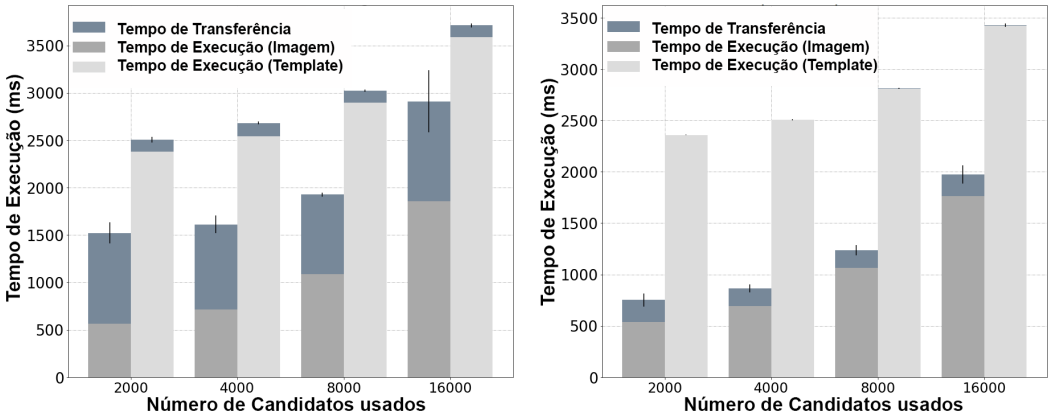
Figura 5. Tempo de execução do processo de identificação no dispositivo móvel

5.2. Segundo Cenário - Dispositivo real fazendo offloading para ambiente remoto

A Figura 6(a) apresenta o tempo total de execução, que é composto pelo tempo de transferência (*download* e *upload*) somado ao tempo de execução (tempo no processador), da tarefa de identificação ao realizar *offloading*. Como a máquina virtual tem maior poder de armazenamento e memória, os testes consideraram uma base maior, que também variou para que fosse possível avaliar seu impacto no tempo de execução.

Como esperado, quanto maior o número de digitais candidatas, maior o tempo total para realizar a identificação. Além disso, os resultados mostram que, embora uma fração maior do tempo total seja gasta com transferência de dados entre o dispositivo real e o ambiente remoto, o tempo médio de execução é menor quando a imagem da impressão digital é utilizada como entrada para o método de identificação. Portanto, quando o preprocessamento da imagem também é executado no ambiente remoto, a execução da identificação é de 25% à 40% mais rápida do que quando a transformação de imagem para *template* é executada no dispositivo real e apenas o *template* é enviado durante o *offloading* nos casos estudados.

A Figura 6(b) apresenta o tempo total de execução da identificação quando o ambiente remoto utilizado é um *cloudlet*. Ao realizar o *offloading* com a imagem da impressão digital, obteve-se um tempo de execução entre 42% e 69% menor.



(a) Executando em uma instância da AWS (b) Executando em um *cloudlet*

Figura 6. Executando o *offloading* da tarefa de Identificação

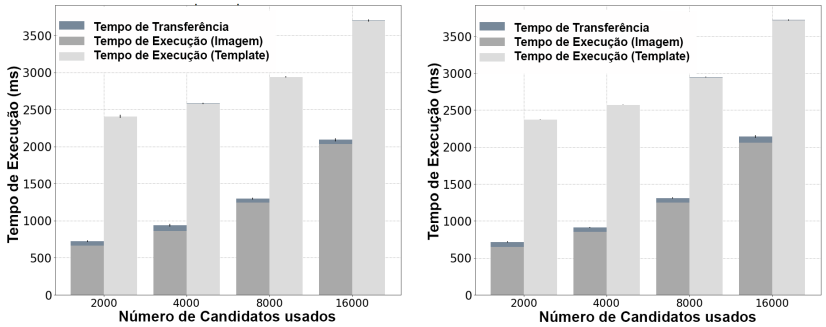
Ao comparar o tempo médio de execução nos dois casos, é possível observar que fazer *offloading* para o *cloudlet* apresenta um desempenho levemente superior à AWS. Essa pequena diferença é causada principalmente pelo tempo de transferência menor ao utilizar o *cloudlet*, o que pode ser explicado pelo fato dos dois dispositivos estarem conectados à mesma rede sem fio.

Também é importante ressaltar que, ao comparar a execução local do Primeiro Cenário e a execução com *offloading* do Segundo Cenário, os resultados mostram que, independente do tamanho da base, é melhor fazer *offloading* do processo de identificação, pois, mesmo no pior caso, o tempo de identificação foi reduzido em aproximadamente 85%. Indo além, pode-se destacar que mesmo o *offloading* utilizando *templates* é mais vantajoso que executar localmente. Sendo assim, a abordagem baseada em *templates* pode ser adotada em cenários onde o dispositivo móvel está conectado a uma rede com baixa taxa de transferência, pois uma quantidade menor de dados precisará trafegar pela rede.

5.3. Terceiro Cenário - Múltiplos dispositivos virtuais fazendo offloading

Para avaliar a solução do ponto de vista de escalabilidade, foi utilizado um *testbed* virtual de MCC para criar cenários com 2, 4, 8 e 16 dispositivos emulados. A Figura 7 apresenta os resultados onde os dispositivos fazem *offloading* para um *cloudlet* usando uma conexão Wi-Fi e a Figura 8 apresenta os resultados onde o *offloading* é realizado para uma instância da AWS usando uma conexão LTE. Neste cenário, tanto a conexão Wi-Fi quanto a LTE são emuladas e mais detalhes podem ser encontrados em [Barbosa and Rego 2019].

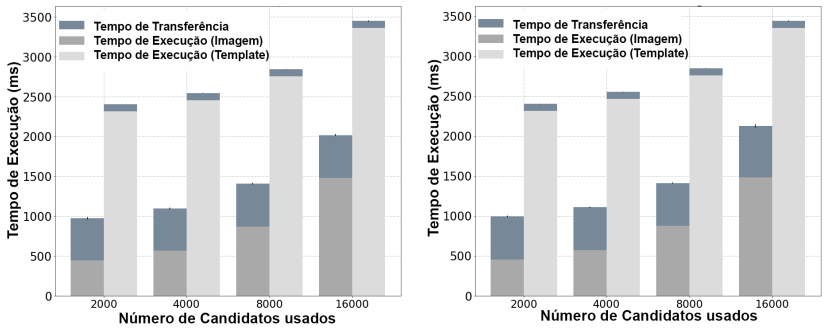
Como pode-se ver, quão maior é a quantidade de digitais candidatas, maior é o tempo total de *offloading* para realizar a identificação, o que é um resultado esperado. Diferente do Segundo Cenário, o tempo de execução da tarefa é um pouco menor quando o *offloading* é realizado para a AWS, o que pode ser explicado pelo fato do *cloudlet* ter sido executado em um container Docker, dentro do *testbed* virtual e não no notebook, tendo assim um menor poder computacional. Entretanto, ressalta-se que devido ao tempo de transferência ser maior para a AWS, o tempo médio de execução em ambos os casos é praticamente o mesmo.



(a) 2 dispositivos (b) 4 dispositivos

(c) 8 dispositivos (d) 16 dispositivos

Figura 7. Executando o processo de Identificação em uma *cloudlet* usando Wi-Fi



(a) 2 dispositivos na AWS (LTE) (b) 4 dispositivos na AWS (LTE)

(c) 8 dispositivos na (LTE) (d) 16 dispositivos na (LTE)

Figura 8. Executando o processo de Identificação em uma instância da AWS usando conexão LTE

Salienta-se ainda que a solução suportou até 16 dispositivos fazendo identificação concorrentemente e, ainda assim, foi capaz de responder em menos de 3 segundos. Por fim, assim como no cenário anterior, usar a imagem da impressão digital em vez do *template* gera uma redução de até 70% no tempo médio de identificação.

6. Conclusões

Foi apresentada neste trabalho uma arquitetura de software que usa *offloading* e os recursos disponibilizados pela computação de borda/*cloudlet* e computação em nuvem para permitir a execução do processo de identificação de impressões digitais em dispositivos com poder computacional limitado. A arquitetura apresentada também permite o *offloading* de diferentes partes do processo de identificação, o que possibilitou a comparação entre realizar o *offloading* de um *template* (resultado do preprocessamento da imagem e extração das minúcias no dispositivo) e realizar o *offloading* da imagem sem preprocessamento. Ademais, um *testbed* virtual foi utilizado para criar cenários com diferentes quantidades de dispositivos móveis realizando *offloading* do processo de identificação para obter resultados mais próximos de um caso de uso real.

A arquitetura proposta propiciou acelerar o processo de identificação de pessoas utilizando impressões digitais em pelo menos 85%. A solução pode ser aplicada em diferentes cenários, como por exemplo no controle de acesso dentro de instituições como escolas, grandes empresas e condomínios. Outra área em que a arquitetura pode ser aplicada é a de segurança pública, permitindo que as forças de segurança tenham um dispositivo móvel capaz de realizar operações de identificação de pessoas remotamente ou em uma base local. Os autores não tem conhecimento de outro trabalho que utilize *offloading* no contexto de identificação de impressões digitais.

É importante ressaltar que o desempenho de outras técnicas de identificação que usam diferentes informações biométricas, como faces e íris, não foi avaliado. Além disso, uma limitação deste trabalho foi o tamanho da base utilizada nos experimentos, no entanto, bases abertas possuem tamanho limitado. Como trabalhos futuros, pretende-se refazer os experimentos utilizando uma base de digitais mais numerosa, dispositivos móveis com maior poder computacional e avaliar se a arquitetura proposta pode auxiliar no processo de identificação de pessoas utilizando diferentes informações biométricas, além de avaliar o uso de diferentes algorítimos de identificação.

Agradecimentos

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

Referências

Alonso-Fernandez, F., Bigun, J., Fierrez, J., Fronthaler, H., Kollreider, K., and Ortega-Garcia, J. (2009). Fingerprint recognition. In *Guide to biometric reference systems and performance evaluation*, pages 51–88. Springer.

Barbosa, R. A. and Rego, P. A. L. (2019). A mobile cloud computing testbed based on lightweight virtualization. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Bordel, B., Alcarria, R., Robles, T., and Martín, D. (2017). Cyber–physical systems: Extending pervasive sensing from control theory to the internet of things. *Pervasive and Mobile Computing*, 40:156–184.

Costa, P. B., Rego, P. A. L., Rocha, L. S., Trinta, F. A., and de Souza, J. N. (2015). Mpos: A multiplatform offloading system. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 577–584. ACM.

Hassan, G. and Elgazzar, K. (2016). The case of face recognition on mobile devices. In *2016 IEEE Wireless Communications and Networking Conference*, pages 1–6.

Kumar, K., Liu, J., Lu, Y.-H., and Bhargava, B. (2013). A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, 18(1):129–140.

Moses, K. R., Higgins, P., McCabe, M., Prabhakar, S., and Swann, S. (2011). Automated fingerprint identification system (afis). *Scientific Working Group on Friction Ridge Analysis Study and Technology and National institute of Justice (eds.) SWGFAST-The fingerprint sourcebook*, pages 1–33.

Peralta, D., Triguero, I., Sanchez-Reillo, R., Herrera, F., and Benítez, J. M. (2014). Fast fingerprint identification for large databases. *Pattern Recognition*, 47(2):588–602.

Powers, N., Alling, A., Osolinsky, K., Soyata, T., Zhu, M., Wang, H., Ba, H., Heinzelman, W., Shi, J., and Kwon, M. (2015). The cloudlet accelerator: Bringing mobile-cloud face recognition into real-time. In *2015 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7.

Ratha, N. and Bolle, R. (2003). *Automatic fingerprint recognition systems*. Springer Science & Business Media.

Rego, P. A., Costa, P. B., Coutinho, E. F., Rocha, L. S., Trinta, F. A., and de Souza, J. N. (2017). Performing computation offloading on multiple platforms. *Computer Communications*, 105:1–13.

Sajjad, M., Nasir, M., Muhammad, K., Khan, S., Jan, Z., Sangaiah, A. K., Elhoseny, M., and Baik, S. W. (2020). Raspberry pi assisted face recognition framework for enhanced law-enforcement services in smart cities. *Future Generation Computer Systems*, 108:995 – 1007.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23.

Xiao, Q. and Raafat, H. (1991). Fingerprint image postprocessing: a combined statistical and structural approach. *Pattern Recognition*, 24(10):985–992.

Xu, J., Jiang, J., Dou, Y., and Shen, X. (2014). A low-cost fully pipelined architecture for fingerprint matching. In *2014 12th International Conference on Signal Processing (ICSP)*, pages 413–418. IEEE.

Zaeri, N. (2011). Minutiae-based fingerprint extraction and recognition. *Biometrics*.

Zhao, Y.-x., Zhang, W.-x., Li, D.-s., Huang, Z., Li, M.-n., and Lu, X.-c. (2016). Pegasus: a distributed and load-balancing fingerprint identification system. *Frontiers of Information Technology & Electronic Engineering*, 17(8):766–780.