ANGELO FERREIRA ASSIS

Orientador: Ricardo Augusto Rabelo Oliveira

PROPOSTA DE MIDDLEWARE PARA COMPRESSÃO ADAPTATIVA DE DADOS EM AMBIENTES ANDROID

Ouro Preto

Novembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

PROPOSTA DE MIDDLEWARE PARA COMPRESSÃO ADAPTATIVA DE DADOS EM AMBIENTES ANDROID

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

ANGELO FERREIRA ASSIS

Ouro Preto
Novembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Proposta de Middleware para Compressão Adaptativa de Dados em Ambientes Android

ANGELO FERREIRA ASSIS

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Ricardo Augusto Rabelo Oliveira – Orientador
Universidade Federal de Minas Gerais

Dr. Daniel Fernandes Macedo
Universidade Pierre et Marie Curie

Dr. Fabrício Benevenuto de Souza
Universidade Federal de Minas Gerais

Ouro Preto, Novembro de 2010

Resumo

A expansão do poder computacional em dispositivos portáteis, associado a um maior acesso à Internet através de redes sem fio resulta em uma situação na qual os usuários podem gastar mais tempo online, usando toda a cobertura disponível. O sistema operacional desenvolvido pela Google para dispositivos móveis, chamado Android, é um exemplo de sistema que explora toda a evolução de recursos desenvolvidos e adaptados para dispositivos móveis, e é o sistema utilizado no trabalho. Em dispositivos móveis, existem várias limitações para a interface, tempo de baterias, capacidade de processamento e limitações gerais da configuração de hardware. Estas restrições apresentam desafios no desenvolvimento de aplicações móveis para ambientes sem fio. Assim, apresentaremos uma solução para economia de energia, a adaptabilidade. A adaptação consiste em ajustar o comportamento do aplicativo de acordo com sua consciência do ambiente, com o objetivo de minimizar o consumo de recursos escassos, como a energia e largura de banda, aperfeiçoando as propriedades que são mais visíveis para os usuários. A adaptação discutida é a compressão dos dados a serem enviados pelo canal de rede sem fio, que diminui o tempo de transmissão, porém aumenta o tempo de processamento para compressão / descompressão. Esse tipo de adaptação introduz um estudo que avalia quando a compressão deve ocorrer. Devem ser avaliados vários parâmetros que interferem na escolha, como largura de banda, tamanho do arquivo a ser transmitido, taxa de compressão, consumo de energia para transmissão e processamento, entre outros.

Abstract

The expansion of computing power in portable devices, coupled with greater access to the Internet through wireless networks results in a situation in which users can spend more time online, using all available network. The operating system developed by Google for mobile devices called Android, is an example of a system that exploits all the evolution of developed and adapted resources for mobile devices, and is the system used in this work. On mobile devices, there are several limitations to the interface, battery time, processing power and limitations of the general hardware configuration. These constraints present challenges in developing mobile applications for wireless environments. Thus, we present a solution to energy saving, adaptability. The adaptation consists of adjusting the application's behavior according to its environment awareness, in order to minimize consumption of scarce resources such as energy and bandwidth, improving the properties that are more visible to users. The adaptation discussed is to compress data to be sent through the wireless channel, which reduces transmission time, but increases the processing time for compression / decompression. This kind of adaptation introduces a study that evaluates when compression should occur. Should be evaluated various parameters that influence this choice, such as bandwidth, file size to be transmitted, compression ratio, power consumption for transmission and processing, among others.

Dedico este trabalho aos meus pais, por todo o amor e dedicação para comigo, por terem sido a peça fundamental para que eu tenha me tornado a pessoa que hoje sou.

Agradecimentos

Toda lista de agradecimentos peca pela ausência de uma ou outra pessoa que teve papel, ainda que pequeno, na conclusão de um trabalho como este. Mesmo assim, tento registrar aqui, em poucas linhas, os meus agradecimentos, sabendo que não são suficientes para expressar a importância de todos nessa minha jornada. Agradeço a todos que contribuíram direta, ou indiretamente, para a concretização deste trabalho:

- A Deus pela minha vida, minha saúde e por guiar meu caminho na conquista de mais um objetivo.
- A minha mãe, que sempre acreditou na minha capacidade e com muita e paciência soube aturar as noites em claro que passei durante toda a graduação.
- Ao meu pai, que apesar de morar longe, nunca deixou de se preocupar com meus trabalhos, meus estudos, minha vida.
- A Flavinha, pelo carinho e apoio em todos os momentos que precisei, além da contribuição em muitos de meus trabalhos.
- Aos meus amigos da UFOP, em especial Rodrigo, Marcão, Rodolfo, Thiago, Larissa, Cecília, Tales e todos os outros que participaram comigo dessa caminhada.
- Aos meus amigos que durante a jornada me acompanharam nos momentos de descontração e sempre torceram pelo meu sucesso.
- Aos professores David Menotti, Joubert Lima, José Maria, Jeroen Van De Graaf, Luiz Merschmann, Sica. Em especial ao meu orientador, Ricardo Rabelo, pela dedicação, preocupação e esforço para produzir um bom trabalho. Todos participaram dessa jornada comigo, ou antes dela forneceram-me os meios para a travessia.

"Nunca deixe que lhe digam que não vale a pena acreditar nos sonhos que se tem, ou que os seus planos nunca vão dar certo, ou que você nunca vai ser alguém"

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a evolução de aplicativos e um maior investimento por parte de empresas, os aparelhos celulares muitas vezes não são mais usados somente para conversa entre pessoas, mas sim com várias outras funções como a de GPS, acesso à internet, para uma possível consulta ao email, entre outros. Não só os celulares, mas diversos dispositivos móveis com comunicação sem fio já estão presentes em nossas atividades diárias como PDAs (Personal Digital Assistant), notebooks, netbooks, smartphones. Além da evolução dos dispositivos e aplicativos, existe a evolução das tecnologias de rede sem fio, que estão cada vez mais poderosas com maior largura de banda, maior alcance, etc.

Um bom exemplo da evolução das redes sem fio é o IEEE 802.11 (IEEE, 2010), conhecida como Wifi. A cronologia de seus principais padrões mostra claramente o progresso dessa tecnologia. Quando aprovado, em 1997, a taxa de transmissão desse padrão atingia entre 1 e 2 Mbps. Após algum tempo foram aprovados também os padrões 802.11b e 802.11a, que usam as frequências de 2,4 e 5 GHz e atingem uma taxa de transmissão de até 11 e 54 Mbps respectivamente. Com a popularização e evolução de protocolos de segurança, em 2003 surgiu o 802.11g, que opera na frequência de 2,4 GHz, assim como o 802.11b, porém com taxa de transmissão de até 54 Mbps. Em 2009 foi aprovado o padrão 802.11n, um aperfeiçoamento dos padrões anteriores 802.11a e 802.11g. Esse último usa as frequências 2,4 e/ou 5 GHz e sua taxa de transmissão varia de 54 Mbps a 600 Mbps. A evolução dessas diversas tecnologias da computação móvel tornou possível o acesso à Internet de qualquer lugar e a qualquer instante.

1.1 Limitações dos dispositivos móveis

Apesar da evolução das tecnologias e dispositivos, a transmissão de dados e o acesso à internet em um ambiente de comunicação sem fio a partir de um dispositivo móvel é uma tarefa diferenciada se comparada a computadores de mesa, uma vez que os recursos dos dispositivos móveis são inerentemente escassos. Existem várias limitações para a interface, tempo de baterias, capacidade de processamento e limitações gerais da configuração de hardware. As principais restrições no ambiente de computação móvel são os dispositivos, com suas limitações de hardware, as redes sem fio, que apesar das evoluções continuam com menores taxas de transmissão que as redes com fio, e a mobilidade (Gupta, 2008). Essas restrições apresentam desafios reais no desenvolvimento de aplicações para esse tipo de ambiente. Quanto à limitação do hardware, um dos maiores problemas é o gasto de bateria. Primeiramente, várias aplicações sensíveis ao contexto executam continuamente em background coletando informações ou esperando por uma condição de ativação e assim, uma pequena necessidade de energia tem o potencial para afetar o dispositivo mais que outras aplicações mais pesadas, porém que executam em menor tempo. Além disso, várias aplicações são consideradas úteis por um grande número de usuários. Com isso, o impacto de consumo dessas aplicações deve ser minimizado. A complexidade desses dispositivos torna difícil para os desenvolvedores compreender o consumo de energia nas aplicações.

Para superar as limitações dos dispositivos móveis, podemos utilizar informações do contexto no qual o dispositivo está inserido afim de realizar adaptações dinâmicas na aplicação.

1.2 Aplicações cientes de contexto

A noção de contexto se refere à situação na qual um dispositivo está inserido. Computação ciente de contexto é um paradigma computacional no qual aplicações podem descobrir e se aproveitar de informações contextuais, tal como a localização do usuário, nível de bateria do dispositivo, tipos de rede sem fio disponíveis, etc. O objetivo de uma aplicação ciente de contexto é elaborar uma maneira de obter informações do contexto do usuário, ambiente e dispositivo computacional, considerando tanto suas características de hardware, como também de software e de comunicação para prover serviços apropriados para um determinado usuário em um certo momento. Adicionar a ciência de contexto em uma aplicação móvel pode incrementar sua usabilidade de forma a aumentar a satisfação do usuário, além de permitir que sejam adicionadas novas funcionalidades à aplicação.

Dey (2001) formalizou a definição de contexto como sendo

"Qualquer informação que possa ser utilizada para caracterizar a situação de entidades (pessoa, lugar ou objeto) que sejam consideradas relevantes para interação entre um usuário e uma aplicação (incluindo o usuário e a aplicação)."

Essa definição é bastante imparcial quanto aos tipos e variedades de dados que podem ser considerados como contextos, sendo ampla o suficiente para aceitar as diversas necessidades específicas de cada aplicação. Embora possa ser utilizada por qualquer aplicação computacional para a qual seja relevante, o uso da sensibilidade ao contexto é especialmente importante para uma área em particular: a computação ubíqua e pervasiva. Atualmente, o computador está inserido cada vez mais na vida das pessoas e essa interação é a base para o conceito de computação pervasiva. Esse conceito implica que o computador está inserido no ambiente de forma totalmente transparente ao usuário. Assim, tem a capacidade de obter informações para se adaptar a este meio e atender melhor as necessidades do usuário. A computação ubíqua surge da necessidade de integração de mobilidade com computação pervasiva, o que significa que qualquer dispositivo computacional móvel também pode obter informações do meio o qual está inserido e com isso adaptar-se às necessidade do usuário proporcionando os benefícios da computação de forma natural.

Especificamente, a mobilidade permitida aos usuários e seus objetos computacionais em aplicações ubíquas, faz com que o uso de contexto seja ainda mais importante. As características utilizadas como entradas em tais soluções poderão se modificar a qualquer instante, e essa informação é crucial para que os melhores serviços sejam oferecidos aos usuários. Tais modificações podem ser observadas em situações nas quais o próprio usuário se movimenta de um local para outro, modificando-se assim as condições do ambiente corrente em variados aspectos como o tipo de rede a qual está conectado, ou a velocidade de transmissão. A restrição de recursos disponíveis para execução das aplicações é outro fator importante a ser considerado, sendo bastante afetado pela característica dinâmica das aplicações ubíquas e pervasivas. Uma vez que os recursos e serviços disponibilizados para os usuários se modificam ao longo do tempo, e os dispositivos utilizados são em geral pequenos, com poucas capacidades computacionais, de energia e memória, os sistemas desse tipo devem realizar um gerenciamento de recursos consistente de maneira a prolongar o tempo de vida da aplicação, mantendo os níveis de qualidade de serviço desejados pelos usuários. O uso de contextos é mais uma vez importante como forma de manter as aplicações conscientes dos recursos disponíveis.

O estudo de Leichtenstern e Andre (2008) descreve os principais aspectos a serem considerados em uma aplicação móvel centrada no usuário para um ambiente de computação

móvel. Esses aspectos são alguns "modelos" e "conjuntos de regras". O primeiro aspecto a ser observado é o personagem, ou usuário da aplicação. Sempre que for necessário o desenvolvimento de uma aplicação ciente de contexto devemos cuidadosamente identificar e analisar os interesses e objetivos do usuário ao utilizar tal aplicação. Além do usuário, deve-se considerar interfaces com capacidade adaptação de apresentação e interação, de acordo com os objetivos. Outro aspecto é contexto do ambiente, que pode sofrer influência de vários fatores, por exemplo, os recursos técnicos disponíveis, que são basicamente os recursos disponíveis no dispositivo móvel, além das tecnologias disponíveis no local, como por exemplo uma rede sem fio.

1.3 Adaptação de aplicativos

Os recursos dos dispositivos que mais consomem energia são as redes sem fio, geralmente 3G, Bluetooth e Wifi (Rice e Hay, 2010). Com isso, é necessário introduzir algum processo de adaptação do conteúdo da Web, fazendo com que a comunicação e transmissão dos dados economize energia. Uma solução para superar esse problema é a adaptabilidade. A adaptação consiste em alterar ou ajustar o comportamento do aplicativo de acordo com o seu contexto. Essa alteração tem o objetivo de diminuir o tempo de transmissão dos dados e aumentar o tempo de duração da bateria.

A adaptação pode ser feita de várias maneiras, dependendo da aplicação, do ambiente do usuário, ou de dispositivos de comunicação. Essa adaptação é mais eficaz quando ocorre de forma dinâmica, isto é, quando o processo de adaptação analisa o ambiente e decide que ação deve ser tomada durante o tempo de execução. O objetivo é economizar ou minimizar o consumo de recursos escassos, como a energia e largura de banda, ou otimizar as propriedades que são mais visíveis para os usuários - como o tempo de resposta e qualidade dos dados. O processo de adaptação pode ser realizado na aplicação, no sistema, ou em um middleware específico. Geralmente, um middleware projetado para um ambiente sem fio é dividido em dois componentes - um no dispositivo e outro em um ponto entre o dispositivo e seu par na rede com fio (por exemplo, a estação de base).

Esse tipo de middleware pode ser usado para alterar, remover, ou fornecer conteúdo multimídia para dispositivos móveis, ou seja, é possível realizar diferentes esquemas de adaptação. Por exemplo, pode-se imaginar uma situação onde imagens podem ser removidas dos e-mails enviados aos usuários antes de sua transmissão através do canal sem fio, ou então a qualidade do vídeo pode ser alterada para se ajustar ao tamanho da tela, ou dados frequentemente acessados podem ser armazenados mais próximos aos usuários, etc.

1.4 Motivação

Dessa forma percebemos que existe uma evolução dos dispositivos e seus recursos. Porém ainda existem várias limitações, como por exemplo a bateria, que ainda restringem o desenvolvimento e usabilidade de certas aplicações. Um dos maiores problemas hoje em dia é o uso de redes sem fio em dispositivos móveis operados por bateria, já que a popularização de dispositivos como smartphones contribui para que um número crescente de usuários use a Internet frequentemente. Como rede sem fio é o recurso que atualmente mais consome energia, é preciso que exista uma solução para diminuir esse gasto, fazendo com que os usuários possam ficar um maior tempo conectados sem se preocupar tanto com o nível de bateria. Então, supõe-se que para amenizar os problemas desse tipo podemos usar informações do contexto do dispositivo e propor adaptações como a compressão de dados a serem transmitidos pela rede sem fio.

A partir do estudo das condições que definem quando e como a compressão deve ocorrer, o objetivo desse trabalho é desenvolver um modelo decisório que define se um arquivo deve ser comprimido ou não antes de ser transmitido para um dispositivo móvel por uma rede sem fio. O foco inicial do trabalho é para arquivos de texto, mas a técnica desenvolvida pode ser estendida para outros tipos de arquivo, por exemplo, multimídia. Além do modelo, ainda existe o objetivo de desenvolver uma aplicação em Android que faça o uso desse modelo, para testar e validar a implementação do trabalho.

Diante desse objetivo, podemos enunciar alguns passos que devem ser seguidos para o desenvolvimento do trabalho. Esses passos são considerados os objetivos específicos:

- Identificar nos trabalhos relacionados as vantagens e desvantagens de cada abordagem;
- Verificar os algoritmos de compressão na plataforma Android;
- Desenvolver o modelo de adaptação que prevê quando a compressão dos dados deve ocorrer;
- Testar o modelo em dispositivos reais, com diferentes tipos de rede.

1.5 Organização do trabalho

O restante deste trabalho está organizado da seguinte forma: O capítulo 2 apresenta alguns trabalhos relacionados com o assunto, além de críticas e considerações sobre cada um. O capítulo 3 fornece uma base teórica do trabalho, com conceitos importantes para

o entendimento de toda a implementação e do modelo desenvolvido. No capítulo 4 são descritos detalhes da implementação do modelo, além da descrição dos experimentos. Finalmente, no capítulo 5, apresentam-se os resultados obtidos, as conclusões alcançadas e estimativas de trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

O trabalho de Couto (2003) avalia quando e como deve ocorrer a compressão de arquivos HTML para uma possível transmissão por uma rede sem fio. Seus resultados foram obtidos através de testes simulados considerando diferentes situações e arquivos HTML. Foram estudados vários algoritmos de compressão, com objetivo de avaliar a desempenho de cada um com arquivos HTML e XML. Assim, foi proposto um modelo adaptativo (Ver figura 2.1) que define quando a compressão deve ser usada e considera alguns fatores do contexto para tomar essa decisão.

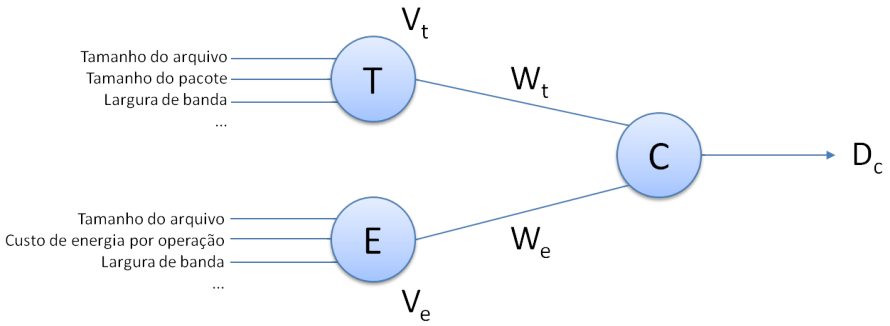

Figura 2.1: Modelo preditivo

Esses fatores são considerados em dois módulos, um módulo de tempo e outro de energia. Cada um desses módulos passa um valor V_t e V_e respectivamente para um terceiro módulo, D_c, responsável pela decisão final. Os valores de V_t e V_e são definidos a partir de algumas propriedades como o tamanho do arquivo, taxa de compressão, largura de banda, etc. Um fator de peso, W_t e W_e é associado a cada módulo, representando sua confiança, sendo: $W_t, W_e \in [0, 1]$ A decisão final é tomada a partir do uso da fórmula

2.1:

$$D_c = \begin{cases} 0 & \text{, se } W_t = W_e = 0 \\ \frac{W_t \cdot V_t + W_e \cdot V_e}{2.0} & \text{, senão} \end{cases} \tag{2.1}$$

Um arquivo é enviado em sua forma original quando $Dc < 0,5$ e é enviado comprimido se $Dc \geq 0,5$. Isso significa que a compressão ocorre se a decisão dos módulos for de alta confiança e concordarem com a compressão ou quando um valor de confiança é alto o bastante para compensar o outro. O problema nesse caso é o valor de Dc que deve ser definido para tomar ou não a decisão. A definição desse valor $(0,5)$ influencia diretamente no desempenho do sistema, podendo prejudicar ou melhorar bastante o resultado, por isso, é um parâmetro que deve ser bem analisado.

Nesse modelo existe uma realimentação, que recalcula os valores dos pesos W_t e W_e. Os novos valores são baseados nos valores reais obtidos na transmissão e com os valores estimados para esta transmissão. Se realmente houve um ganho devido à predição dos modelos de tempo e de energia, esses valores são reforçados (aumentados); caso contrário, há uma penalização daquele peso que trouxe uma predição incorreta. Isso significa que o recurso utilizado para atualização dos valores de W_t e W_e depende de uma estimativa, que pode em algum momento apresentar um resultado diferente da situação real que o usuário viveria. A fórmula 2.2 descreve como funciona o recálculo dos pesos:

$$w = \begin{cases} w + \frac{(1-w)}{2} & \text{, se a predição foi correta} \\ w - \frac{w}{2} & \text{, caso contrário} \end{cases} \tag{2.2}$$

Testes experimentais e simulados com os ambientes IEEE 802.11 e Bluetooth mostraram que este método é eficiente quando comparado a técnicas mais simples de adaptação, porém as conclusões e os resultados obtidos se basearam apenas em simulações.

Os estudos de Xiao et al. (2010) mostraram que a eficiência da economia depende do equilíbrio entre a energia gasta na compressão/descompressão do dado e na energia economizada na transmissão. A economia na transmissão depende da redução dos dados, que por sua vez depende da taxa de compressão, que varia de acordo com o algoritmo de compressão e o tipo do dado. O trabalho propõe um framework dividido em duas partes, uma responsável pela decisão e execução da compressão, e clientes móveis responsáveis pela descompressão. A contribuição inclui três aspectos básicos:

- Propor uma solução prática, que adota um mecanismo de decisão ciente de con-

texto, baseado em algoritmos de compressão e custo de computação e comunicação.

- Levar em consideração a taxa de compressão e descompressão para diferentes tipos de dados na estimativa do custo computacional.
- Levar em consideração as características da rede ao estimar custo de comunicação.

O objetivo do framework é reduzir o consumo de energia através da compressão de dados em dispositivos móveis. Neste trabalho, são focadas as situações em que os dispositivos móveis recebem dados transmitidos pela rede. Assim, o servidor Proxy realiza a compressão antes de enviar os dados para os clientes e é responsável pela escolha do algoritmo de compressão a ser utilizado. A escolha leva em consideração a energia gasta pelo cliente na transmissão e descompressão dos dados. Foi assumido um sistema de servidores como email e arquivos de mídia, e clientes que requisitam o conteúdo dos servidores através dos servidores Proxy intermediários. Os clientes são alimentados por baterias, e os servidores por fontes de alimentação externas.

O modelo do sistema inclui então duas partes, chamadas servidor Proxy e cliente móvel. O cliente fornece suas informações de contexto ao servidor que as usa para decidir se realiza ou não a compressão. Existem dentro do sistema componentes responsáveis pela coleta e gerenciamento dos dados de contexto.

Quanto à escolha do algoritmo de compressão a ser utilizado deve-se realizar cálculos que envolvem tamanho do arquivo, taxa de compressão (dependente do tipo de dado), energia gasta na transmissão (dependente do tipo de rede), energia gasta na compressão e descompressão (dependente do algoritmo utilizado). Foi implementado um serviço de email móvel *energy-aware*, onde a política de adaptação leva em consideração o tipo de dado dos anexos, o estado da bateria, as condições de rede e a lista de algoritmos suportados pelo dispositivo destinatário. Foi testada a relação de compressão utilizando vários algoritmos de compressão. As taxas de compressão variam de acordo com o tipo do arquivo e do algoritmo de compressão. Foi escolhido o Nokia N810 como dispositivo experimental. Com testes em diferentes tipos de arquivos, percebeu-se que a compressão dos dados poderá obter economias de 10% a 60% dependendo do tipo de dado dos anexos e condições de rede, além de que a eficácia da compressão é a base para tomar a decisão da compressão.

Enfim, foi proposto um framework de adaptação de energia baseado em Proxy que utiliza compressão de dados com o objetivo de economizar energia em dispositivos móveis quando estes estão recebendo dados de uma rede sem fio. Para melhorar a eficiência da compressão o framework leva em consideração as características do dado, ambiente de

transmissão da rede, e informações de contexto do cliente. O framework mostrou uma economia de energia através do caso de estudo de serviço de email adaptado.

O trabalho de Rice e Hay (2010) resultou em um framework de medição que fornece com detalhes o consumo de energia de cada recurso e é projetado para desenvolver um entendimento de como alguns aspectos particulares de uma aplicação consomem energia. O estudo foi realizado nos dispositivos Android, G1 e Magic.

Detalhar o consumo de energia no dispositivo em independentes partes é um desafio, então o trabalho apresentou uma abordagem particular para coletar e analisar os dados. Explorou o custo de energia usando diferentes tipos de redes sem fio para demonstrar com detalhes como e quanto cada tipo de rede consome de energia nos dispositivos. A técnica desenvolvida, testada nos dispositivos Android, é aplicável a qualquer dispositivo móvel programável.

Um resultado interessante desse trabalho é que foi constatado que entre as redes 2G, 3G e WiFi, a rede WiFi é a que possui menor custo de energia para o dispositivo, seguida por 3G e então 2G. Esse tipo de análise realça a importância da consideração não só da velocidade de transmissão, mas também do tipo de rede em uso entre os parâmetros que definem o contexto do dispositivo. Dessa forma, trabalhos como o de Steinberg e Pasquale (2002) podem ser facilmente melhorados, pois os autores utilizam um modelo adaptativo simples para compressão, tanto para imagens (compressão com perda) quanto para texto (compressão sem perda) que considera somente a largura de banda do meio de transmissão para determinar se a compressão deve ser realizada.

Esse trabalho propõe uma nova técnica para decisão de quando um arquivo deve ser comprimido antes de sua transmissão em um ambiente sem fio. Ao contrário de alguns dos trabalhos anteriores, o modelo proposto foi testado com dispositivos reais, não sendo validado apenas com estimativas e simulações, além de ter sido implementado com base em estudos de quais os parâmetros que realmente influenciam na transferência de arquivos e principalmente no gasto de tempo e energia.

Capítulo 3

Referencial Teórico

3.1 Contexto do dispositivo

O modelo decisório desse trabalho será implementado como um serviço que será oferecido ao usuário. Nesse caso, serviços são definidos como uma maneira adaptada de acessar recursos ou funcionalidades remotas utilizando a infraestrutura de rede disponível. Todas as condições que serão analisadas para a definição da decisão do modelo definem o contexto do dispositivo móvel. Um fator relevante no desenvolvimento desses serviços é o uso dessas informações de contexto no qual o serviço é requisitado. O contexto pode ser dividido em duas partes: as características físicas, como capacidade do dispositivo, sua localização física, rede na qual está conectado no momento e características lógicas, como preferências do usuário relacionadas às aplicações que são utilizadas neste dispositivo (Salber et al., 1999). Essa informação é importante para que os serviços cientes do contexto possam ser utilizados de maneira adequada.

A informação do contexto deve permitir que o usuário saiba porque a aplicação está se comportando de determinada maneira e permite que os níveis de qualidade de um serviço possam ser atingidos (Dey e Newberger, 2009). Ao mesmo tempo, estes serviços podem utilizar os recursos de uma infraestrutura de computação em nuvem. A computação em nuvem é baseada em um paradigma no qual os recursos de rede são oferecidos como serviços com grande capacidade de escalabilidade. Ao oferecer um serviço, um nível de qualidade de serviço é previsto, sendo essa qualidade controlada pela infraestrutura servidora.

Consideraremos nesse trabalho como parâmetros do contexto as informações relativas ao tipo e velocidade da rede na qual está conectado o dispositivo, o tamanho do arquivo a ser baixado e uma estimativa da taxa de compressão desse arquivo.

Dentro desse ambiente, o serviço terá acesso a um servidor com os arquivos disponíveis para serem baixados. A comunicação entre o dispositivo e o servidor será através de requisições HTTP realizadas pelo serviço, que recebe como resposta um download do servidor. Entretanto, antes do serviço realizar a requisição HTTP, deve executar o modelo decisório, o que inclui coletar todas as informações do contexto, analisar a viabilidade da compressão e a partir da resposta do modelo requisitar o arquivo no servidor. Consideraremos que o servidor já possui os arquivos requisitados em sua forma normal e comprimidos.

3.2 Compressão adaptativa

Em geral, o formato comum de documentos na Web é o HTML e suas variantes (ASP, JSP, PHP, XML, DHTML e outros). Compactar um documento antes da sua transmissão através de um ambiente sem fio parece ser uma solução óbvia de adaptação se a intenção for apenas economizar a largura de banda. Porém a adaptação que será abordada nesse trabalho é a compressão de dados a serem enviados pelo canal de rede sem fio, diminuindo o tempo de transmissão, porém aumentando o tempo de processamento para compressão e descompressão. Esse tipo de adaptação introduz um estudo que avalia quando a compressão deve ocorrer.

A compressão dos arquivos com certeza diminui o tempo de transmissão, mas por outro lado, os arquivos compactados precisam ser comprimidos e descomprimidos, o que pode aumentar o tempo de resposta total e pode consumir mais energia do que as transmissões sem compressão. Suponha o seguinte cenário: um servidor Web acessado por diferentes clientes através de um canal sem fio. Como cada aparelho tem suas características próprias e estas características variam muito neste ambiente heterogêneo, a decisão de comprimir um arquivo não é uma tarefa simples e deve ser feita com cuidado. Para algumas aplicações e dispositivos a compressão pode ser útil, enquanto que o mesmo aplicativo em um dispositivo diferente poderia ter um melhor desempenho sem compressão. Tomando a decisão correta de comprimir ou não o arquivo pode-se aumentar o tempo de bateria nos dispositivos, diminuir o tempo de transmissão dos dados, e aumentar o desempenho geral do sistema.

O primeiro passo para avaliar se o arquivo deve ser comprimido é definir em quais cenários a compressão deve ser utilizada, ou seja, quando a compressão reduzir o tempo de resposta ou consumo de energia. O segundo passo é selecionar os parâmetros que interferem na escolha, como: largura de banda, tamanho do arquivo a ser transmitido, taxa de compressão, tipo do dispositivo onde a descompressão é feita, consumo de energia para

transmissão e processamento, entre outros. Em terceiro, todos esses parâmetros devem ser combinados para alcançar o objetivo descrito: economizar tempo e/ou energia (Couto et al., 2003).

O middleware que realiza essa adaptação tem que ser compatível com o sistema operacional (SO) usado no dispositivo. Cada um desses dispositivos possui um SO diferente, que gerencia os recursos do sistema e fornece uma interface entre o dispositivo e o usuário, possibilitando uma interação entre eles. Cada SO trabalha de uma forma diferente no gerenciamento de memória, processos, etc., e aborda recursos específicos de hardware de acordo com o dispositivo. O SO para dispositivos móveis, chamado Android, inicialmente desenvolvido pelo Google, é um exemplo de sistema que explora toda a evolução de recursos desenvolvidos e adaptados para esse tipo de dispositivo, principalmente a conexão à internet. Devido a essa característica do Android de explorar bastante a conexão com internet, este foi o sistema operacional escolhido para desenvolver um middleware que realizasse a compressão adaptativa dos dados a serem transmitidos pela rede sem fio.

3.3 Android

Antes de apresentar o modelo, ou os passos percorridos para alcançar o objetivo é preciso entender um pouco mais sobre a plataforma onde foi desenvolvida a implementação do trabalho.

3.3.1 Visão geral

A plataforma Android foi concebida inicialmente pelo Google. Atualmente, a plataforma está sendo mantida pelo Open Handset Alliance, que é um grupo formado por mais de 30 empresas (de tecnologias de dispositivos móveis, provedoras de serviços móveis, fabricantes, etc.) as quais se uniram para inovar e acelerar o desenvolvimento de aplicações e serviços, trazendo aos consumidores uma experiência mais rica em termos de recursos e menos custosa em termos financeiros para o mercado móvel. Pode-se dizer que a plataforma Android é a primeira plataforma móvel completa, aberta e livre, além de ter a característica peculiar de não ser dependente do hardware, possibilitando sua instalação em praticamente qualquer modelo de aparelho celular. Esta plataforma foi desenvolvida utilizando o sistema operacional Linux. Sendo assim, todas as características intrínsecas deste sistema foram incorporadas, como, por exemplo, o sistema de arquivos e o kernel.

Como o sistema operacional Android foi inicialmente desenvolvido pela empresa Google, a maioria dos seus recursos e aplicativos são direcionados a serviços disponibilizados pelo Google. Como exemplo, podemos citar Gmail, Gtalk, Google Maps, Google Agenda, e outros webservices que sejam considerados úteis pelo usuário. Além dos aplicativos que fazem uso dos serviços do google, alguns dispositivos Android já são vendidos com certos aplicativos instalados, como twitter, facebook, canais de notícias, etc. O Android ainda conta com vários desenvolvedores, que estão cada vez mais interessados em desenvolver aplicativos para Android, considerando que existe um grande investimento de empresas em Android contribuindo com a popularização e com o crescimento acelerado de dispositivos sendo comprados mundialmente. Milhares de aplicativos estão disponíveis no Android Market que seria um equivalente à conhecida AppStore, pertencente a Apple. Grande parte desses aplicativos são gratuitos, ou então tem preços muito baixos, contribuindo para que os usuários Android adquiram com maior facilidade os programas de sua preferência.

Todos esses aplicativos contribuem para que o Android seja cada vez mais usado por usuários que podem personalizar seu aparelho da forma que preferir. Porém, um detalhe importante a ser observado é que a maioria desses aplicativos utilizam a conexão à internet. Todos os exemplos citados são aplicativos que utilizam conexão HTTP para o acesso aos serviços e informações. O Android ainda possui o recurso de sincronização, que mantém todos os aplicativos sempre atualizados. Para isso é preciso que exista sempre uma conexão disponível, e assim o dispositivo utiliza essa conexão para cada aplicativo que esteja em execução solicitando sincronização. A utilização desse recurso gera uma grande concorrência no recurso de rede sem fio do dispositivo, pois várias são as aplicações que utilizam a conexão e requisitam informações na internet.

3.3.2 Arquitetura

A figura 3.1 mostra a arquitetura do Android, para facilitar a compreensão do funcionamento da plataforma. Podemos entender o Android como uma pilha de softwares. Cada camada da pilha agrupa vários programas que suportam funções específicas do sistema operacional.

A base da pilha é o kernel, que cuida do gerenciamento de memória, das configurações de segurança, do gerenciamento de energia e ainda possui vários drivers de hardware. O kernel é baseado no kernel Linux, e cada versão do android corresponde a uma versão do kernel Linux segundo a tabela 3.1.

O próximo nível da pilha é subdivido no grupo das bibliotecas (libraries) e o ambiente

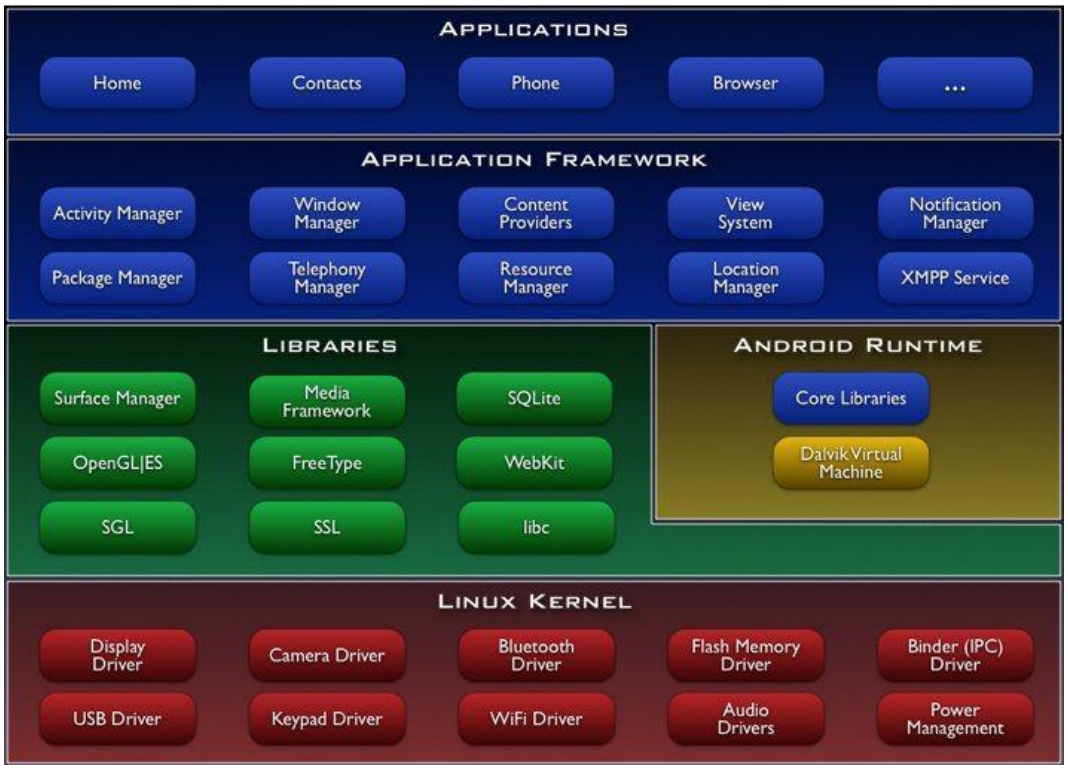

Figura 3.1: Arquitetura do Android Android Developers (2010)

Tabela 3.1: Versões de Kernel Linux no Android

Versão do Android	Versão do Kernel Linux
1.5	2.6.27
1.6	2.6.29
2.0	2.6.29
2.1	2.6.29
2.2	2.6.32

de execução (runtime) da plataforma Android, composto pelas bibliotecas padrão e pela máquina virtual denominada Dalvik (DVM). No primeiro grupo estão as bibliotecas escritas em C/C++, que são compostas por uma coleção de bibliotecas que são utilizadas pela plataforma Android. No que diz respeito ao ambiente de execução, a plataforma é composta pela máquina virtual Dalvik. Toda e qualquer aplicação em Android roda dentro de seu próprio processo, isto é, no contexto da sua instância de máquina virtual.

Esta VM foi escrita para que os dispositivos possam suportar múltiplas máquinas virtuais eficientemente, ou seja, Android usa as máquinas virtuais para rodar cada aplicação como seu próprio processo. Isso é importante por algumas razões. Primeiro, nenhuma aplicação é dependente de outra. Segundo, se uma aplicação pára, ela não afeta quaisquer outras aplicações rodando no dispositivo. Terceiro, isso simplifica o gerenciamento de memória. A camada seguinte na pilha é o framework de aplicação, onde estão localizados os componentes que permitirão com que novas estruturas sejam utilizadas para futuras aplicações, enfatizando a reutilização de código. No topo da pilha estão as aplicações em si. É onde são encontradas funções básicas do dispositivo, como cliente de e-mail, programa de SMS, calendário, mapas, navegador, gerenciador de contatos, e outras sendo todas essas aplicações escritas na linguagem Java.

3.3.3 Componentes de uma aplicação

A plataforma Android tem como característica uma alta taxa de resposta às requisições do usuário, além de fornecer uma API completa para a criação de aplicações cientes do contexto. Essa característica é refletida inclusive no estilo de programação dos dispositivos, uma vez que os desenvolvedores devem criar aplicativos que sejam responsivos à interação com o usuário, caso contrário, o sistema operacional finaliza a execução do aplicativo.

Cada aplicativo Android pode ser subdividido ainda mais em unidades funcionais distintas:

- Atividades: uma atividade apresenta uma interface visual focada nas atividades do usuário;
- Serviços: os serviços não têm uma interface visual do usuário, e executam em segundo plano por tempo indeterminado;
- Intents: respondem às solicitações de serviço de outra aplicação.

Vamos nos focar apenas nos conceitos de atividades e serviços, que foram os mais utilizados na aplicação criada. Atividades são os componentes de um aplicativo Android que estendem a classe base Activity e definem uma interface que consiste em uma visualização que responde a eventos. Se um aplicativo consiste em três janelas (por exemplo, uma janela de login, uma janela de visualização de texto e uma janela de visualização de arquivo), cada uma é geralmente representada por uma diferente classe Activity. O Android mantém uma pilha de histórico para cada execução de aplicativo a partir da

página inicial e é possível clicar no botão Voltar para rolar de volta por esse histórico de atividades. A atividade conta com vários métodos definidos de acordo a mudança de estado da atividade, onde cada estado significa uma parte do ciclo de vida de uma atividade mostrado na figura 3.2. A atividade possui essencialmente três estados: (1) executando, quando está em primeiro plano na tela; (2) pausada, se a atividade perde o foco, mas ainda é visível ao usuário; (3) parada, se está completamente escondida por outra atividade.

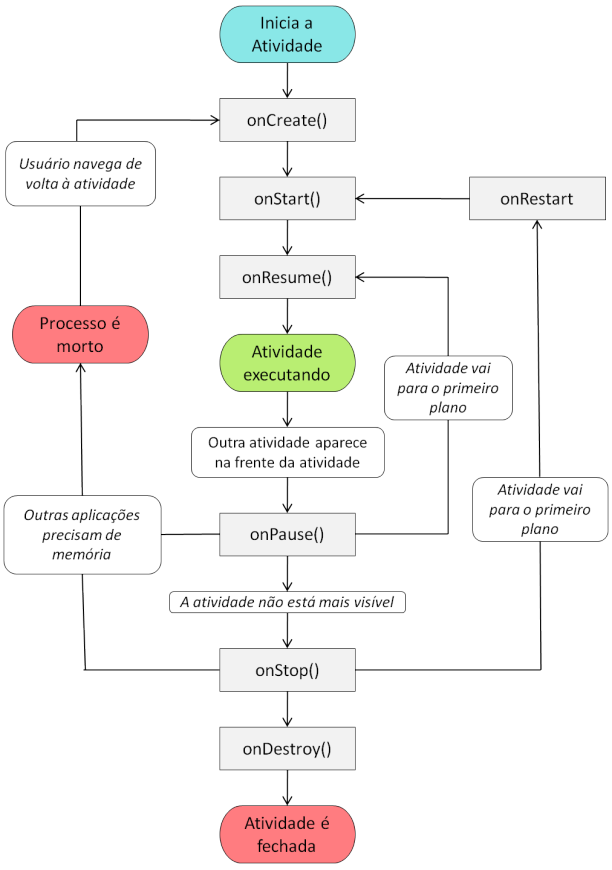

Figura 3.2: Ciclo de vida da Atividade (Android Developers, 2010)

Um serviço é um aplicativo executado em um nível baixo e sem um monitor ou interface gráfica. É geralmente um aplicativo que deve ser executado por muito tempo em segundo plano. É possível se conectar (*bind*) a um serviço já em execução ou iniciar o serviço se ele não estiver em execução, possibilitando o uso de chamadas de procedimento remoto como mostrado na figura 3.3. Em suma, o mecanismo funciona da seguinte forma: é declarada uma interface RPC que se pretende implementar através de um simples IDL

(linguagem de definição de interface). A partir dessa declaração, a ferramenta AIDL gera uma definição de interface Java que deve ser colocado à disposição de ambos processos, local e remoto.

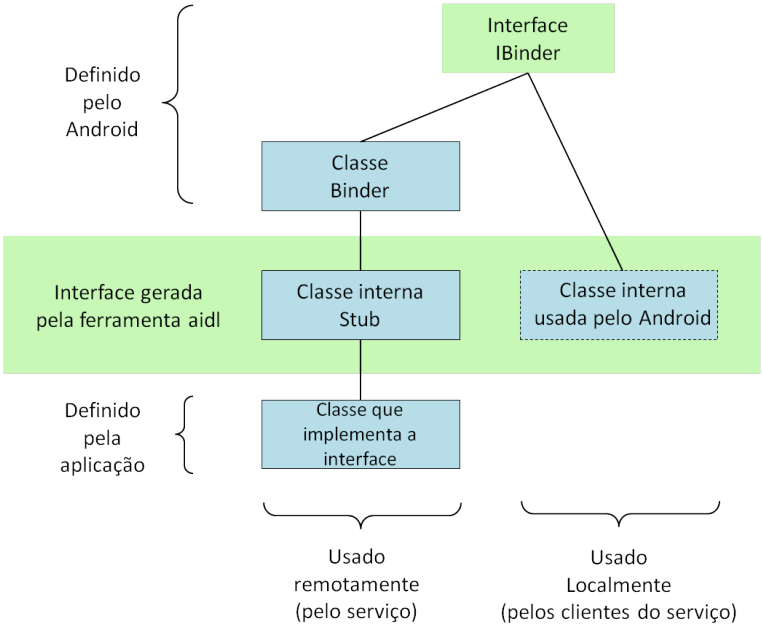

Figura 3.3: Chamadas de Procedimento Remoto (Android Developers, 2010)

Um exemplo de serviço é um programa reprodutor de mídia reproduzindo uma lista de músicas. Apesar de um aplicativo de reprodução de mídia apresentar uma interface que permite que os usuários definam suas listas de reprodução, o programa passa o controle para o serviço para realmente reproduzir as músicas da lista de reprodução fornecida. Como as atividades e os outros componentes, os serviços são executados na thread principal do processo.

Assim como a atividade, o serviço apresenta um ciclo de vida com alguns métodos que podem ser implementados pelo desenvolvedor. A figura 3.4 mostra um pouco do funcionamento desses métodos.

Por padrão, cada aplicativo é executado em seu próprio processo Linux, sendo todos independentes. O Android inicia o processo quando qualquer código do aplicativo precisa ser executado, e encerra o processo, quando não é mais preciso e os recursos do sistema são necessários por outras aplicações. Cada processo tem sua própria máquina virtual (DVM), por isso o código do aplicativo é executado de forma isolada a partir do código de todas as outras aplicações.

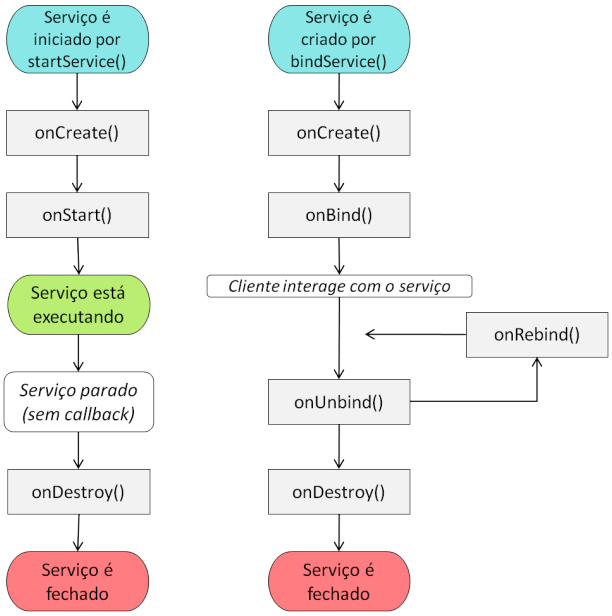

Figura 3.4: Ciclo de Vida do Serviço (Android Developers, 2010)

O processo onde um componente é executado é controlado pelo arquivo *manifest.xml*. Esse arquivo apresenta algumas informações essenciais sobre o aplicativo para o sistema Android. São informações que o sistema deve ter antes de executar qualquer código do aplicativo. Cada um dos componentes tem um atributo de processo que pode indicar o processo em que esse componente deve ser executado. Esses atributos podem ser definidos de modo que cada componente seja executado em seu próprio processo, ou de forma que alguns componentes compartilhem de um processo, enquanto outros não. Eles também podem ser configurados de forma que componentes de diferentes aplicações sejam executados no mesmo processo.

Todos os componentes são instanciados na thread principal do processo especificado e não são criadas threads separadas para cada instância. O Android pode decidir encerrar um processo em algum momento, quando houver pouca memória e esse recurso for exigido por outros processos que estão imediatamente atendendo ao usuário. Os componentes dos aplicativos em execução no processo são consequentemente destruídos. Um processo é reiniciado quando há algum trabalho a ser feito novamente por esses componentes.

Ao decidir quais processos devem terminar, o Android verifica a sua importância relativa para o usuário. Por exemplo, é mais fácil desligar um processo onde as atividades não estão mais visíveis na tela do que um processo com atividades visíveis. A decisão de encerrar um processo, portanto, depende do estado dos componentes em execução no

processo.

Mesmo que seja possível limitar uma aplicação a um único processo, provavelmente haverá momentos em que será necessário gerar uma thread para fazer algum trabalho. Como a interface do usuário deve ser sempre rápida para responder às ações do usuário, a thread que executa uma atividade não deve também executar as operações demoradas como downloads da rede. Qualquer coisa que não possa ser concluída rapidamente deve ser atribuída a um segmento diferente.

3.4 Compressão GZIP

Um dos primeiros passos para a realização desse trabalho foi verificar o algoritmo de compressão disponível na plataforma de desenvolvimento Android. Essa plataforma conta com métodos que implementam a tecnologia de compressão GZIP, criada por Jean-Loup Gailly e Mark Adler (Gailly e Adler, 2003), (Delorie, 2007). Essa funcionalidade está disponível no pacote java.util.zip, que como vimos está presente no Android. A classe GZIPInputStream deste pacote é usada para ler dados armazenados no formato GZIP, sendo capaz de realizar leitura e descompressão dos dados. Esse método foi escolhido por já ser implementado nas bibliotecas padrões do Java, além de ser considerado eficiente e confiável.

3.5 Redes neurais

A decisão a ser tomada pelo modelo é como um problema de classificação, onde um objeto precisa ser atribuido a uma classe predefinida, baseado em um número de atributos observados sobre aquele objeto. Nesse caso, os atributos são os parâmetros referentes ao contexto do dispositivo e as classes são baixar o arquivo comprimido, ou baixar o arquivo sem comprimir. Procedimentos de classificação estatística tradicionais, tais como análise discriminante são construídos com base na teoria da decisão bayesiana. Nesses procedimentos, um modelo probabilístico subjacente deve ser assumido de forma a calcular a probabilidade posterior sobre a decisão de classificação realizada. Uma grande limitação dos modelos estatísticos é que eles funcionam bem apenas quando as suposições subjacentes forem satisfeitas. A eficiência destes métodos depende das diversas hipóteses e condições nas quais os modelos são desenvolvidos. Os usuários devem ter um bom conhecimento das propriedades dos dados e recursos de modelo antes deles poderem ser aplicados com sucesso.

As redes neurais têm se mostrado uma importante ferramenta para a classificação, pois são uma alternativa promissora para vários métodos de classificação convencionais. A vantagem das redes neurais reside nos seguintes aspectos teóricos. Primeiro, as redes neurais são métodos auto-adaptativos orientados pelos dados, que podem ajustar-se aos dados sem qualquer especificação explícita de forma funcional ou de distribuição para o modelo subjacente. Segundo, redes neurais podem aproximar qualquer função com uma precisão arbitrária. Uma vez que qualquer procedimento de classificação procura uma relação funcional entre os membros do grupo e os atributos do objeto, a precisão da identificação dessa função básica é sem dúvida importante. Terceiro, as redes neurais são modelos não-lineares, o que as torna flexíveis em relação à modelagem do mundo real complexo. Finalmente, as redes neurais são capazes de estimar a probabilidade posterior, que fornece a base para o estabelecimento de regras de classificação e análise estatística (Zhang, 2000).

Com base nas vantagens oferecidas pelas redes neurais, o modelo implementado utilizou uma rede neural Multilayer Perceptron (MLP) como estrutura para tomar a decisão de compressão. Essa rede foi escolhida devido ao fato de que separa conjuntos que não são linearmente separáveis. O problema da decisão de compressão ou não depende de vários fatores, sendo assim, difícil de saber previamente se as classes são linearmente separáveis. Dessa forma, serão brevemente explicados alguns conceitos de redes neurais importantes para o entendimento da implementação do modelo, assim como para justificar o tipo da rede neural escolhida.

Segundo Haykin (2000), uma rede neural é um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro humano em dois aspectos:

1. O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.

2. Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Assim, uma Rede Neural Artificial (RNA) é uma forma de computação não algorítmica caracterizada por sistemas que, em algum nível, relembram a estrutura do cérebro humano. Por não ser baseada em regras ou programas, a computação neural se constitui uma alternativa à computação algorítmica convencional. As unidades de processamento simples de uma RNA, chamadas nodos ou neurônios, realizam o cálculo de determinadas

funções matemáticas (normalmente não lineares). Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais (Braga et al., 2000).

Os neurônios podem ser modelados como simples dispositivos de entrada/saída interligados em rede. A entrada é recebida dos neurônios que se encontram mais abaixo na cadeia de processamento e a saída é transmitida aos neurônios que estão mais acima na cadeia (Noriega, 2005). O trabalho de McCulloch e Pitts (1943) propõe um modelo de neurônio com n terminais de entrada $x_1, x_2, ..., x_n$ e apenas um terminal de saída y. Para emular o comportamento das sinapses, os terminais de entrada do neurônio têm pesos acoplados $w_1, w_2, ..., w_n$ cujos valores podem ser positivos ou negativos, dependendo das sinapses correspondentes serem de inibição ou excitação. Inicialmente o modelo de McCulloch e Pitts considerou que os neurônios seriam disparados quando recebessem impulsos que ultrapassassem o seu limiar de excitação, e isso seria simulado através do uso de funções de ativação. Posteriormente, simplificaram o modelo considerando que os neurônios em cada camada disparam sincronamente, isto é, todos são avaliados ao mesmo tempo. Uma limitação desse modelo é que as redes com apenas uma camada só conseguem implementar funções linearmente separáveis. A partir do modelo proposto por McCulloch e Pitts, foram derivados vários outros modelos com diferentes funções de ativação. Alguns exemplos de função de ativação são: função linear, função rampa, função degrau (step) e função sigmoidal.

O processo de aprendizagem é a característica mais importante das RNAs. Tem a função de modificar os pesos sinápticos da rede, ou seja, determinar a intensidade de conexões entre os neurônios, de uma forma ordenada para alcançar um objetivo de projeto desejado. Existem diversos algoritmos de aprendizado que realizam esse processo, cada qual com suas vantagens e desvantagens. Esses algoritmos basicamente se diferem na maneira pela qual o ajuste dos pesos é feito. Com isso, uma definição geral do que vem a ser aprendizagem pode ser expressa da seguinte forma (Mendel e McLaren, 1970):

"Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma continuada de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros."

Diversos métodos para treinamento de redes foram desenvolvidos, podendo ser agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não supervisionado. O aprendizado supervisionado é o mais comum e é chamado dessa forma porque a entrada e saída desejadas para a rede são fornecidas por um supervisor externo. O objetivo é ajustar os parâmetros da rede, de forma a encontrar uma ligação entre os

pares de entrada e saída fornecidos. A figura 3.5 ilustra o mecanismo de funcionamento do aprendizado supervisonado. O supervisor indica uma entrada para a rede, que calcula sua saída e compara com a saída desejada, recebendo a informação sobre o erro da resposta atual. Os pesos das conexões são ajustados para que minimizem o erro. Um dos exemplos mais conhecidos de algoritmo para aprendizado supervisionado é o algoritmo *backpropagation* (Rumelhart et al., 1988).

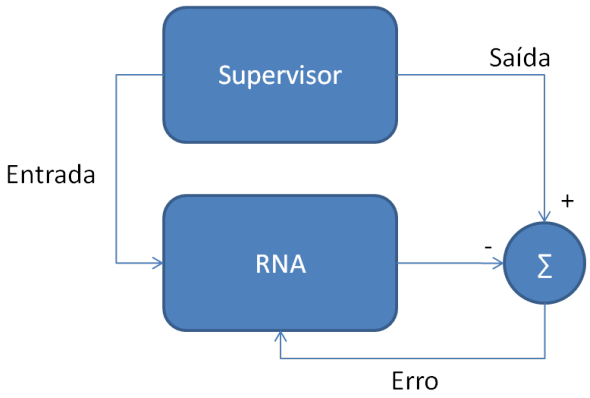

Figura 3.5: Aprendizado supervisionado

A definição da arquitetura de uma RNA é um parâmetro importante na sua concepção, uma vez que restringe o tipo de problema que pode ser tratado pela rede. Fazem parte da definição da arquitetura os seguintes parâmetros: número de camadas da rede, número de neurônios em cada camada, tipo de conexão entre os neurônios e topologia da rede. Como foi dito, redes com uma camada única de neurônios do modelo McCulloch e Pitts só conseguem resolver problemas linearmente separáveis. A solução de problemas não linearmente separáveis passa pelo uso de redes com uma ou mais camadas intermediárias, ou escondidas. Para treinar redes com mais de uma camada usamos o *backpropagation*. Afim de que este método possa ser utilizado, a função de ativação precisa ser contínua, diferenciável e, de preferência não decrescente. A função também deve informar os erros cometidos pela rede para as camadas anteriores com a maior precisão possível. Por isso, normalmente é usada a função sigmoidal, que é uma função semilinear, limitada e monotônica. É possível definir várias funções sigmoidais. Uma das funções sigmoidais mais importantes é a função logística, definida pela equação 3.1

$$y = \frac{1}{1 + \epsilon^{-x}} \tag{3.1}$$

Assim, existem as redes do tipo perceptron multicamadas, ou MLP (MultiLayer Per-

ceptron), que apresentam um poder computacional maior do que aquele apresentado pelas redes sem camadas intermediárias. A precisão obtida e a implementação de uma rede MLP dependem do número de nodos utilizados nas camadas intermediárias.

Capítulo 4

Implementação e testes

4.1 Implementação

O trecho de código 4.1 mostra uma estrutura usada para vincular a atividade e o serviço criados. Dessa maneira o serviço será vinculado à atividade assim que a mesma for criada, pois o código está no método *onCreate* da atividade. O comando *startService* inicializa o serviço, a partir de uma Intent, e o comando *bindService* vincula esse serviço à atividade. Assim que ocorre essa vinculação e também a desvinculação, podem ser executadas rotinas definidas pelo desenvolvedor a partir dos métodos *onServiceConnected* e *onServiceDisconnected* mostrados.

```
public class CompAdap extends Activity {

  private IServiceComp mService = null;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    startService(new Intent("com.service.compression.SERVICECOMP"));

    bindService(new Intent(IServiceComp.class.getName()), mConnection,
        Context.BIND_AUTO_CREATE);
  }

  private ServiceConnection mConnection = new ServiceConnection() {
    public void onServiceConnected(ComponentName className, IBinder service
        ) {
```

```
        mService = IServiceComp.Stub.asInterface(service);
        mService.connect(url, file);
    }

    public void onServiceDisconnected(ComponentName name) {
        mService = null;
    }
  };
}
```

Trecho de Código 4.1: Atividade se vinculando ao serviço

Trabalhando dessa forma, foi criada uma aplicação com uma atividade e um serviço, como mostrado na figura 4.1.

Figura 4.1: Integração - Atividade e Serviço

Essa aplicação simula um usuário navegando na internet e baixando arquivos aleatórios. Devido à característica do serviço de ser executado em segundo plano e à característica da atividade de ter que ser responsiva ao usuário, é trivial que em uma aplicação como essa, o download seja realizado no serviço e a atividade fique por conta apenas de interações com usuários. A atividade tem a função de coletar a URL que contém o endereço de um arquivo a ser baixado e acionar o serviço (*start*). Após o serviço ser iniciado, o mesmo é vinculado à atividade (*bind*) e recebe a URL coletada. Então

começa uma sequência de passos de execução do modelo. Em primeiro lugar, o serviço se conecta à URL através de HTTP. Em seguida coleta todas as informações necessárias para definir o contexto do dispositivo. A partir desses parâmetros, executa o modelo decisório que indica se deve baixar o arquivo comprimido ou não. Por fim, de acordo com a resposta do modelo, o serviço realiza efetivamente o download e realiza uma chamada de volta para a atividade (*callback*) indicando que o download foi realizado. Caso seja baixado o arquivo comprimido, antes de chamar a atividade novamente o serviço realiza a descompressão. O trecho de código 4.2 mostra uma das partes principais do código da aplicação.

```
private final Handler mHandler = new Handler() {
  @Override
  public void handleMessage(Message msg) {
    FileOutputStream saida = null;
    try {
      saida = new FileOutputStream("/sdcard/Testes_CS/file.txt");
    } catch (FileNotFoundException e1) {
      e1.printStackTrace();
    }
    PrintStream fileSaida = new PrintStream(saida);
    switch (msg.what) {
    case DOWNLOAD: {
      // Confere se existe rede
      ConnectivityManager cmConnectivityManager = (ConnectivityManager)
          getSystemService(CONNECTIVITY_SERVICE);
      NetworkInfo ni = cmConnectivityManager.getActiveNetworkInfo();
      if (ni != null) {
        try {
          // Conexao HTTP
          final HttpClient httpClient = new DefaultHttpClient();
          final HttpUriRequest request = new HttpGet(url);
          // Download do arquivo
          final HttpResponse resp = httpClient.execute(request);
          final int status = resp.getStatusLine().getStatusCode();
          if (status != HttpStatus.SC_OK) {
            Log.e("Service.mHandler", "HttpStatus != SC.OK");
          }

          BufferedReader in;
          if (baixarCompactado) {
            // Modelo decide baixar o arquivo COMPRIMIDO
            in = new BufferedReader(new InputStreamReader(
                new GZIPInputStream(resp.getEntity()
```

```
                    .getContent())));
            } else {
              // Modelo decide baixar o arquivo NORMAL
              in = new BufferedReader(new InputStreamReader(resp
                  .getEntity().getContent()));
            }

            String buffer = null;
            buffer = in.readLine();
            if (buffer != null)
              fileSaida.print(buffer);
            while ((buffer = in.readLine()) != null) {
              fileSaida.print("\n" + buffer);
              teste++;
            }
            fileSaida.close();

          } catch (ClientProtocolException e) {
            e.printStackTrace();
          } catch (FileNotFoundException e) {
            e.printStackTrace();
          } catch (IOException e) {
            e.printStackTrace();
          } catch (Exception e) {
            e.printStackTrace();
          }
        } else { // sem rede!
          fileSaida.print("WITHOUT NETWORK!!");
          Log.e("Service.mHandler.DOWNLOAD", "Without Network!");
        }

        final int N = mCallbacks.beginBroadcast();
        for (int i = 0; i < N; i++) {
          try {
            mCallbacks.getBroadcastItem(i).update();
          } catch (RemoteException e) {
            e.printStackTrace();
          }
        }
        mCallbacks.finishBroadcast();
      }
        break;
      default:
        super.handleMessage(msg);
```

```
        }
        Log.d("Service.Handler", "This function was executed! :D");
    }
};
```

Trecho de Código 4.2: Serviço realizando conexão HTTP

Com todos esses conceitos, o modelo decisório foi implementado a partir de uma rede neural MLP completamente conectada. Foram considerados quatro fatores:

1. Tamanho do arquivo;
2. Taxa de compressão;
3. Tipo de rede sem fio conectada;
4. Velocidade atual da rede sem fio;

A rede neural utilizada pode ser melhor visualizada na figura 4.2. Os quatro parâmetros do contexto do dispositivo formam a entrada. A camada intermediária possui 20 neurônios com função de ativação sigmoidal, já explicada anteriormente. A camada de saída possui dois neurônios, onde cada um representa uma classe: baixar o arquivo comprimido ou não comprimido.

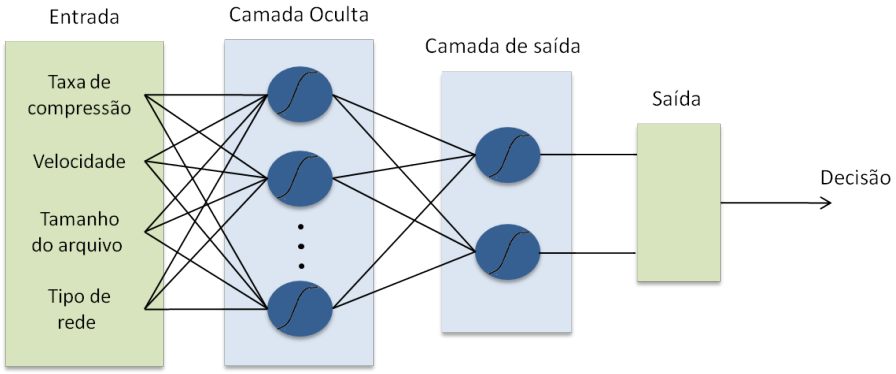

Figura 4.2: Rede Neural - Modelo Implementado

A partir de centenas de testes realizados com a aplicação descrita, foi possível montar uma base de treinamento para essa rede neural, onde foi aplicado o algoritmo *backpropagation*, que é um algoritmo de aprendizado supervisionado. Esse treinamento é comprovadamente eficiente na atualização dos pesos de cada conexão (Rumelhart et al., 1988). Uma vantagem da utilização da rede neural nesse caso foi a possibilidade de realizar um

treinamento antecipado que forneceu os valores dos pesos de cada conexão. Com esses pesos salvos, basta transferir a rede para o dispositivo e executar o cálculo de decisão com base nas funções de cada neurônio e pesos de cada conexão.

4.2 Experimentos

Com toda a estrutura explicada no capítulo 3, foi utilizado para experimentos um dispositivo com Android 2.1 desenvolvido pela empresa HTC, chamado Nexus. Esse dispositivo possui conexões 2G, 3G, Wifi 802.11 b/g e Bluetooth 2.0, sendo útil para futuros testes em diversificadas redes, considerando diferentes situações. Possui 512MB de memória RAM, um processador de 1GHz e uma bateria de 1400mAh, que, segundo sua especificação, dura mais de 290 horas em *standby*. Sua memória é expansível até 32GB, com cartão de memória.

Utilizando esse dispositivo, foi executada a aplicação descrita anteriormente em 4.1 e conduzida uma série de experimentos utilizando a rede Wifi. Nesses experimentos houve grande variação da velocidade de transmissão da rede, entre 1 e 72Mbps. Essa variação foi intencional, com o propósito de verificar a influência da velocidade de transmissão no tempo de download e gasto de bateria, fatores que são fundamentais para que o modelo decisório tome a decisão certa. Nesses testes foram baixados arquivos de texto (.txt) de 1, 5, 10, 500, 1000 e 2000Kb. Todos esses arquivos também foram comprimidos e disponibilizados no servidor, para que pudessem ser baixados também em sua forma comprimida, comparando-se os tempos de transmissão e gasto de bateria para concluir qual download seria mais vantajoso. Os arquivos comprimidos atingiram em média 90% de taxa de compressão, o que significa que arquivos de 1Mb passam a ter apenas 100Kb.

Todos os testes foram realizados também com o intuito de conhecer o comportamento do Android em situações não relatadas normalmente. Devemos considerar que esse sistema operacional está começando a se estabilizar e ainda há muito o que se conhecer dos softwares e dispositivos sendo lançados a cada dia.

Após centenas de testes e uma rigorosa análise dos dados obtidos, foi possível montar uma base de testes para uma rede neural, especificando suas entradas e saída esperada. As entradas consideradas são os valores de tamanho de arquivo, taxa de compressão, tipo de rede sem fio e velocidade da rede sem fio. A saída esperada é 1 se o download do arquivo comprimido é melhor e 0 se o download do arquivo não comprimido é melhor.

Com essa base de dados pronta, foi utilizada a toolbox para redes neurais do matlab (Neural Network Tollbox^{TM} 7) para testar a utilização da rede neural (Beale et al., 2010). A base foi dividida entre 70% para treinamento, 15% para validação e 15% para testes.

Os resultados foram satisfatórios, como mostrado nas matrizes de confusão da figura 4.3[1]. Com isso, percebemos que nos testes realizados, houve um acerto de 100% com a base de dados fornecida, ou seja, a rede classificou os dados, informando corretamente quando deve existir a compressão ou não.

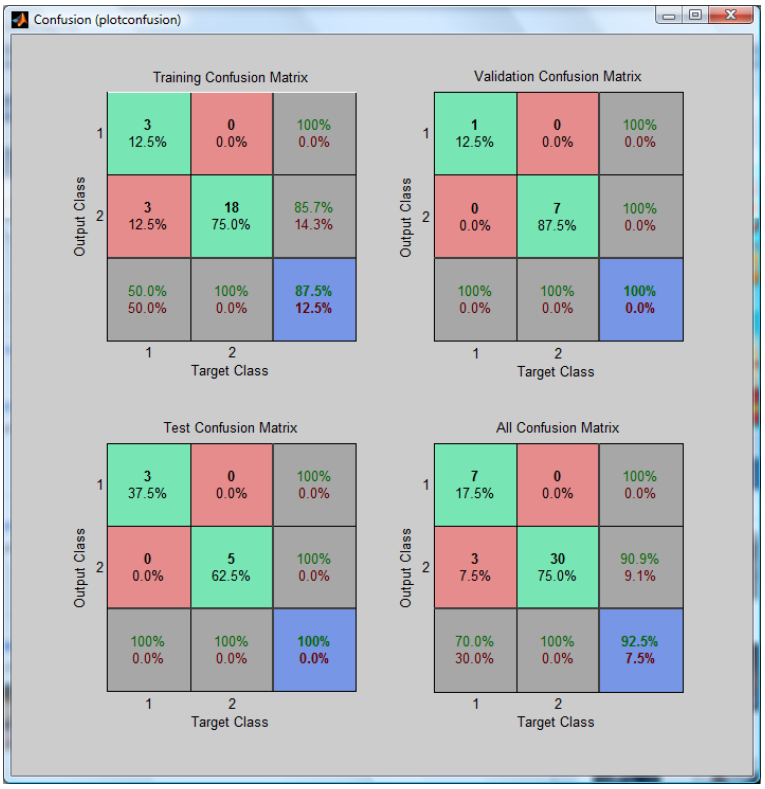

Figura 4.3: Matriz de Confusão

Como o Android permite várias aplicações rodando ao mesmo tempo, foram instaladas cinco aplicações no dispositivo que realizam a mesma tarefa, baixar um arquivo de algum servidor usando conexão HTTP. O arquivo a ser baixado foi o de tamanho 1Mb, também utilizado em todos os outros experimentos. As cinco instâncias foram executadas ao mesmo tempo, e todas elas baixavam 10 vezes o arquivo. Entre as cinco instâncias em execução, três delas baixavam o arquivo sem ser compactado e as outras duas baixavam o arquivo compactado, para depois descomprimir e salvar no cartão de memória. Esse experimento serviu para estressar o dispositivo, considerando que houve uma grande con-

[1] No campo da inteligência artificial uma matriz de confusão é uma ferramenta de visualização que se emprega em aprendizado supervisionado. Cada coluna da matriz representa o número de predições de cada classe, enquanto cada linha representa as instâncias na classe real. Um dos benefícios das matrizes de confusão é que facilitam ver se o sistema está confundindo duas classes.

corrência de recursos. Todas as instâncias precisavam da conexão o tempo todo, e duas delas precisavam de um maior tempo de processamento para descomprimir os arquivos. Com esse teste notou-se que as duas instâncias que baixavam os arquivos comprimidos acabaram bem antes das outras três. Comparando-se com os outros testes, é possível perceber que com várias instâncias rodando simultaneamente, o tempo de download é aproximadamente três vezes maior, como mostrado na tabela 4.1.

Tabela 4.1: Média dos tempos (em segundos) de download com arquivo de 1Mb

	Cinco instâncias simultâneas	Apenas uma instância
Não Comprimido	41,48	14,34
Comprimido	23,21	7,86

Capítulo 5

Conclusões e Trabalhos Futuros

Após a realização dos experimentos e análise dos dados, uma das primeiras considerações a ser feita é que usar arquivos de texto (.txt) e a compressão GZIP é uma situação em que a compressão se mostra realmente vantajosa, pois foram alcançadas taxas de compressão de em média 90%. Considerando essa informação e o trabalho de Rice e Hay (2010) (detalhado em 2), que afirma que o gasto com redes sem fio é maior que o gasto com compressão, conclui-se que para esses dispositivos, sempre que se economizar tempo na transmissão, também estará economizando bateria. Sendo assim, arquivos com alta taxa de compressão tendem à decisão de que devem ser comprimidos, pois dessa forma o tempo de transmissão será bem menor e o usuário perceberá esse menor tempo, assim como a economia no gasto de bateria.

A partir desse primeiro resultado, podemos perceber que existem trabalhos futuros a serem feitos, com arquivos de diferentes tipos. A partir da diversidade dos arquivos, serão obtidas diferentes taxas de compressão, fazendo com que a base de dados fique mais interessante. Além de diferentes tipos de arquivos, é preciso ainda realizar testes com outros tipos de rede, pois, como foi ressaltado, as redes 2G e 3G gastam mais energia que Wifi. É preciso realizar testes e análises para verificar o impacto desse gasto e tentar perceber quando é que esse fator será decisivo na decisão da compressão.

A rede neural se mostrou eficiente como estrutura de classificação para esse problema. Um outro trabalho futuro será implementar essa rede dentro do dispositivo. A intenção é que ocorra o treinamento antecipado, antes da instalação da aplicação, e assim sejam salvos os valores dos pesos das conexões para que quando o serviço seja acionado, precisará apenas executar a rede uma vez, calculando sua saída. Essa operação de execução da rede para o cálculo da saída é rápida e sua influência no tempo de resposta ao usuário e no gasto de bateria não chega a ser considerável. Já a operação de treinamento pode ser

custosa, devido às limitações de processamento de dispositivos móveis, sendo preferível então que seja realizada anteriormente.

O teste com cinco instâncias simultâneas nos faz concluir que a maneira como o Android trabalha, com diversas aplicações que fazem uso da conexão HTTP contribui para uma alta competição por recursos. Sendo assim, é viável que em alguns casos sejam baixados arquivos compactados, fazendo com que exista uma menor concorrência para o uso da rede. Mas também pode ser que em alguns casos seja preferível baixar o arquivo não compactado, considerando que outras aplicações em execução fazem mais uso de processador do que de conexão. Esse fator introduz mais um parâmetro no contexto do dispositivo, relativo ao número de aplicações sendo executadas no momento, que utilizam os mesmos recursos da aplicação de interesse do usuário. Quanto mais aplicações estiverem em execução, maior será o tempo de download. Assim, mais um trabalho futuro é analisar a influência de outras aplicações sobre a conexão com a internet, verificando o impacto causado sobre sua aplicação.

Dessa forma, esse trabalho contribui para o conhecimento do comportamento de dispositivos Android em diversas situações. Trabalhos que até hoje foram apenas simulados agora podem ser implementados e testados em dispositivos Android, em situações reais vividas pelo usuário. Com isso será possível analisar e comparar outros métodos de tomada de decisão afim de verificar qual é o mais eficiente e indicado para determinadas situações e dispositivos.

Referências Bibliográficas

Android Developers (2010). Android developers. `http://developer.android.com/index.html`.

Beale, M.; Hagan, M. e Demuth, H. (2010). Matlab neural network toolbox user's guide version 7.

Braga, A. P.; Carvalho, A. P. L. e Ludermir, T. B. (2000). *Redes Neurais Artificiais: teoria e aplicações*. Livros Técnicos e Científicos, 1 edição.

Couto, R. R.; Rabelo, R. A. e Loureiro, A. A. F. (2003). Compressão adaptativa de arquivos html em ambientes de comunicação sem fio. In *Proceedings of the XXI Simpósio Brasileiro de Redes de Computadores*, pp. 313–328.

Couto, R. R. P. (2003). Compressão adaptativa de arquivos html em ambientes de comunicação sem fio. Dissertação de mestrado, UFMG - Universidade Federal de Minas Gerais, PPGCC - Programa de Pós-graduação em Ciência da Computação.

Delorie, D. (2007). Gzip user's manual. `http://www.delorie.com/gnu/docs/gzip/gzip_toc.html`.

Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Computing*, 5:4–7.

Dey, A. K. e Newberger, A. (2009). Support for context-aware intelligibility and control. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pp. 859–868, New York, NY, USA. ACM.

Gailly, J. e Adler, M. (2003). The gzip home page. `http://www.gzip.org`.

Gupta, A. K. (2008). Challenges of mobile computing. In *2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008)*, Mandi Gobindgarh.

Haykin, S. (2000). *Redes Neurais: Princípios e Práticas*. Bookman, 2 edição.

IEEE (2010). IEEE - institute of electrical and electronics engineers - standards association. `http://standards.ieee.org/getieee802/802.11.html`.

Leichtenstern, K. e Andre, E. (2008). User-centred development of mobile interfaces to a pervasive computing environment. *International Conference on Advances in Computer-Human Interaction*, 0:114–119.

McCulloch, W. e Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5:115–133. 10.1007/BF02478259.

Mendel, J. M. e McLaren, R. W. (1970). *Adaptive, learning, and pattern recognition systems; theory and applications*. Academic Press, New York.

Noriega, L. (2005). Multilayer perceptron tutorial.

Rice, A. C. e Hay, S. (2010). Decomposing power measurements for mobile devices. In *PerCom*, pp. 70–78.

Rumelhart, D. E.; Hinton, G. E. e Williams, R. J. (1988). *Learning representations by back-propagating errors*, pp. 696–699. MIT Press, Cambridge, MA, USA.

Salber, D.; Dey, A. K. e Abowd, G. D. (1999). The context toolkit: Aiding the development of context-enabled applications. In *CHI*, pp. 434–441.

Steinberg, J. e Pasquale, J. (2002). A web middleware architecture for dynamic customization of content for wireless clients. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pp. 639–650, New York, NY, USA. ACM.

Xiao, Y.; Siekkinen, M. e Ylä-Jääski, A. (2010). Framework for energy-aware lossless compression in mobile services: the case of e-mail. In *IEEE International Conference on Communications*, pp. 1–6.

Zhang, G. P. (2000). Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(4):451–462.