FELIPE DE LIMA MILANI

DESENVOLVIMENTO DE APLICATIVO PARA A PLATAFORMA ANDROID PARA DISSEMINAÇÃO DE DADOS UTILIZANDO BLUETOOTH

LAVRAS - MG

2014

FELIPE DE LIMA MILANI

DESENVOLVIMENTO DE APLICATIVO PARA A PLATAFORMA ANDROID PARA DISSEMINAÇÃO DE DADOS UTILIZANDO BLUETOOTH

Monografia apresentada ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação.

Orientador

Prof.Tales Heimfarth

LAVRAS - MG

2014

FELIPE DE LIMA MILANI

APLICATIVO ANDROID PARA DISSEMINAÇÃO DE DADOS VIA BLUETOOTH

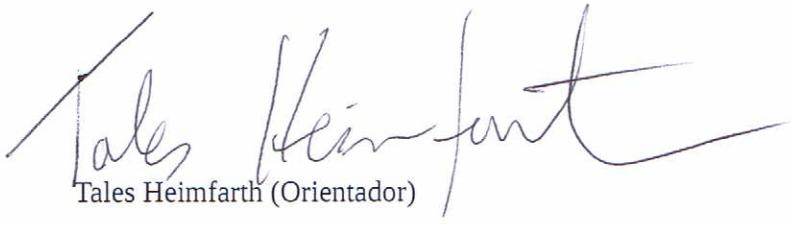
Monografia de graduação apresentada ao Colegiado do Curso de Bacharelado em Ciência da Computação, para obtenção do título de Bacharel.

APROVADA em 21 de julho de 2014.

Raphael Winckler de Bettio

João Carlos Giacomin

João Paulo de Araujo



Tales Heimfarth (Orientador)

LAVRAS-MG
2014

SUMÁRIO

LISTA DE FIGURAS

Resumo

Neste projeto será apresentado um aplicativo desenvolvido para a plataforma Android capaz de realizar a transferência de dados utilizando o bluetooth com arquitetura *Peer-to-peer*. O aplicativo é capaz de descobrir dispositivos semelhantes com localização próxima e criar uma rede entre dois usuários e, a partir daí, transferir dados entre estes de forma eficiente. O aplicativo conta com uma interface totalmente funcional para que o usuário não precise lidar diretamente com questões de baixo nível, como a configuração da rede e o envio de dados.

1 Introdução

Nos últimos anos a tecnologia permitiu que os celulares se tornassem pequenos e potentes computadores capazes de realizar tarefas muito mais complexas do que apenas fazer ligações utilizando as redes das operadoras de telefonia móvel. Atualmente eles são capazes de executar aplicações com as mais variadas funções e utilidades, uma delas é a possibilidade de gerenciar redes semelhantes às de computadores (Desktops e notebooks) convencionais para a troca de informações (HELGASON *et al.*, 2010).

Uma tendência mundial é o compartilhamento de arquivos dos mais variados tipos. Os usuários de dispositivos móveis querem fazê-lo da maneira mais fácil possível e, como o plano de dados de uma operadora é medido pela quantidade de dados recebidos, uma alternativa que não consuma essa cota é muito interessante (NORDSTRÖM; GUNNINGBERG; ROHNER, 2012).

Este trabalho é parte de um projeto maior que visa a utilização de um protocolo de roteamento, que utilize o rádio WiFi dos dispositivos móveis para estabelecer conexões simultâneas e criar uma rede ad-hoc oportunista automaticamente. A intenção é de que os aparelhos formem uma rede capaz de conectar vários dispositivos, onde cada dispositivo dentro do alcance de pelo menos um nó que faça parte da rede também possa se conectar. Ainda é possível que redes distintas, quando em contato, tornem-se uma só. A aplicação, na presença de uma rede, forneceria sua lista de interesses para todos os nós participantes, podendo obter o ítem desejado vindo de um dispositivo que nunca esteve dentro do seu alcance. Essa lista de interesse seria armazenada por terceiros para que, em conexões futuras, ele possa requisitar um dado e posteriormente entregá-lo ao requisitante original caso venha a encontrá-lo novamente. A Conexão entre dispositivos se dá de forma automática e *on the move*, ou seja, enquanto o usuário anda pela rua. A aplicação é capaz de reconhecer outros dispositivos rodando a aplicação, e então estabelecer conexão entre eles para a troca de dados.

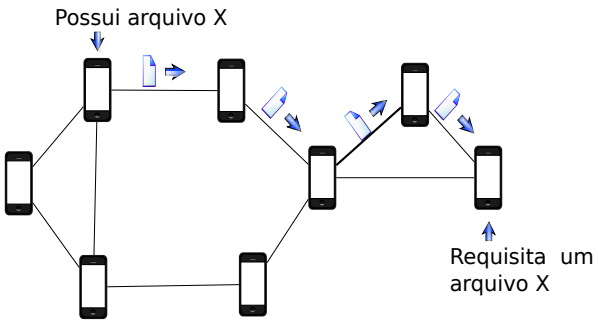


Figura 1: Rede criada pelo conceito do projeto principal.

Neste trabalho foi desenvolvido um protótipo. Algumas das funcionalidades foram modificadas para possibilitar o seu desenvolvimento. A tecnologia Bluetooth foi utilizada ao invés do WiFi e apenas uma conexão pode ser realizada por dispositivo. Além do tipo de conexão, também foi alterado o modo com que a aplicação lida com a lista e os dados, não é armazenado nenhum tipo de informação de conexões anteriores, ou seja, apenas os dois dispositivos conectados trocam conteúdo.

1.1 Motivação

Recentemente, as redes wiresless tem se tornado uma importante ferramenta no nosso cotidiano. Embora muita pesquisa tenha sido feita na área, novas aplicações das redes wireless continuam a surgir. O potencial máximo desse tipo de serviço ainda não foi alcançado (HSU; LIU; SEAH, 2011).

O uso de multimidia se espalhou dos PCs convencionais com internet para o bolso das pessoas através de celulares smartphones, tablets, entre outros. Atualmente existem dois modos de se compartilhar dados na internet: fazer download quando se está conectado ao computador via cabo, ou por meio de infraestruturas fixas como torres de celulares e pontos de acesso (HELGASON *et al.*, 2010).

Dispositivos móveis, tais como smartphones e tablets, estão tendo a oportunidade de acessar e compartilhar conteúdo enquanto estão se movendo livremente pelas ruas. A crescente dependência de serviços online está sobrecarregando as redes Wireless (3G), fazendo com que as operadoras tenham que descarregar um pouco desse tráfego em redes locais (WiFi) e até mesmo colocando limite nas taxas de transferência de dados com preços astronômicos (NORDSTRÖM; GUNNINGBERG; ROHNER, 2012).

1.2 Objetivos

Este trabalho tem como objetivo desenvolver uma aplicação para a plataforma Android capaz de abstrair as dificuldades de gerenciamento e configuração de uma rede Ad-hoc entre dispositivos com a intenção de compartilhar dados.

1.2.1 Objetivos Específicos

Os objetivos específicos relacionados a este trabalho são:

Desenvolver o *software*: Desenvolver um *software* capaz de conectar, buscar por arquivos desejados e realizar a transferência desses dados de forma oportunista, com um arquitetura *peer-to-peer*.

Desenvolver um sistema para a anotação e descoberta de dados: Elaborar um sistema de nomeação de dados (metadados) para auxiliar a busca de arquivos pela rede.

Interface do Usuário: Desenvolver uma interface que permita a interação com a aplicação.

1.3 Organização do Trabalho

Este trabalho encontra-se organizado em seis capítulos. O capítulo 1 apresenta uma introdução, a motivação e os objetivos. No capítulo 2 po-

dem ser encontradas as definições e bases teóricas para o entendimento do problema e também um sistema semelhante ao proposto neste trabalho. A metodologia de pesquisa para realização do trabalho encontra-se no capítulo 3. O capítulo 4 apresentará os conceitos utilizados no desenvolvimento do protótipo. O quinto capítulo irá apresentar e discutir os resultados obtidos com relação ao que foi proposto neste trabalho. E, por fim, no último capítulo será feita uma breve conclusão.

2 Referencial Teórico

Esta seção é dividida em 7 partes. Na primeira, que é composta do ítem 2.1, é apresentada a ferramenta utilizada para o desenvolvimento de aplicações para Android. Na segunda (ítens 2.2 a 2.5), são mostradas abordagens para estruturar redes entre dispositivos terminais. Na terceira parte (2.6 e 2.7), as duas soluções disponíveis no mercado para se criar redes entre dispositivos móveis são apresentadas. Na quarta parte (2.8), exibe dois dos sistemas para troca de conteúdo mais conhecidos. Na quinta parte (2.9), é discutido os conceitos sobre a formação e manutenção de uma rede com dispositivos móveis. A sexta parte (2.10), apresenta o problema de como encontrar o arquivo desejado dentre tantos outros armazenados. Por fim, a sétima e ultima seção, faz uma comparação com um sistema similar ao construído nesse trabalho.

Os estudos realizados nas seções 2.2 a 2.5 e 2.9 são justificados pelo fato de que este trabalho é parte de um projeto maior.

2.1 Android ADT

A ADT (Android Development Toolkit)[1] é um *plugin* para a IDE do eclipse que inclui o Android SDK (Software development kit) e permite o usuário criar aplicações para o sistema operacional Android.

ADT possibilita que novos projetos Android sejam criados rapidamente, incluindo interface de usuário(UI). Contendo ferramentas para *debug*, APIs necessárias, emulador do Sistema Android contendo várias configurações de dispositivos móveis (*hardware* e *software*) , além de tutoriais e amostras de códigos exemplificando o uso de cada uma das APIs.

É possível construir aplicações com as seguintes propriedades:

[1] Disponível em: http://developer.android.com/sdk/index.html

Handset layouts: A plataforma é adaptada tanto para dispositivos VGA maiores, gráficos 2D, bibliotecas gráficas 3D baseadas em OpenGL ES especificação 2.0 e os layouts mais tradicionais de smartphones.

Armazenamento: É utilizado SQLite para armazenamento de dados.

Mensagens: Tanto SMS como MMS são formas disponíveis de envio de mensagens.

Navegador: O navegador disponível no sistema é baseado no framework de Código aberto conhecido como WebKit.

Máquina Virtual Dalvik: Aplicações escritas em Java (linguagem de programação) são compiladas em bytecodes Dalvik e executadas usando a Dalvik virtual machine, que é uma máquina virtual especializada desenvolvida para uso em dispositivos móveis, o que permite que programas sejam distribuídos em formato binário (bytecode) e possam ser executados em qualquer dispositivo Android, independentemente do processador utilizado. Apesar de as aplicações Android serem escritas na linguagem Java, ela não é uma máquina virtual Java, já que não executa bytecode JVM. Atualmente, está disponível, em caráter experimental, uma nova máquina virtual chamada ART (Android RunTime) que promete otimizar o carregamento de aplicativos. O Android ART está disponível apenas no Android 4.4 KitKat e superiores.

Multimídia: O sistema suporta formatos de áudio e vídeo como: MPEG-4, H.264, MP3 , e Advanced Audio Coding|AAC.

Suporte Adicional de Hardware: O Android é totalmente capaz de fazer uso de câmeras de vídeo, tela sensível ao toque, GPS, acelerômetros e aceleração de gráficos 3D.

A codificação de aplicações Android se dá em 3 etapas. Primeiramente é preciso definir as permissões e o menor nível da API[2] do Android (Target API) que poderá utilizar a aplicação. No caso deste sistema foi utilizado a versão 2.0 que corresponde a API nível 5, que é o nível mínimo para a

[2]Cada nível de API corresponde a uma versão do sistema operacional Android

utilização do Bluetooth. Além de escolher a target API, o desenvolvedor tem que especificar quais permissões o *software* terá na utilização do sistema do Android. As permissões garantidas para esta aplicação foram 3:

Write External Storage: Permite escrever no cartão SD (armazenamento externo).

Bluetooth: Permite realizar transferência de dados e estabelecer conexão.

Bluetooth Admin: Necessário para varrer o espectro a procura de novos dispositivos e alterar as configurações do Bluetooth.

Todas essas configurações e muitas outras são feitas no arquivo Android-Manifest.xml, qualquer aplicação Android deve, obrigatoriamente, ter este arquivo.

O segundo passo é criar o *layout* da aplicação, ou interface de usuário, é por ela que o usuário irá interagir com o sistema. As interfaces do Android são criadas a partir de um arquivo xml que contém todos os objetos existentes na tela, cada tela corresponde a um arquivo xml e cada arquivo xml está ligado a uma classe java.

A terceira etapa é implementar as funcionalidades que irão fornecer conteúdo e reagir aos comandos do usuário.

2.2 Redes Tolerantes a Atraso (DTN)

Esta seção estuda o impacto de longas desconexões sobre o ciclo de vida de um dispositivo que pertence a uma rede móvel. Tais dispositivos estão sujeitos a entrar e sair do alcance da rede sem aviso, portanto uma abordagem sobre como realizar a conexão assim que possível e estratégias de troca de mensagens se tornam a melhor opção para o funcionamento do tipo de aplicação proposta neste projeto.

2.2.1 Definição

Uma rede DTN consiste de várias células de redes menores que se comunicam esporadicamente entre si. Em algumas aplicações estas conexões entre as células menores são agendadas e em outras são completamente imprevisíveis (PELUSI; PASSARELLA; CONTI, 2006).

Redes Tolerantes a Delay consistem em uma camada chamada *bundle*, que opera acima da camada de transporte. Um *Bundle* é uma unidade contendo blocos de dados que, juntos, carregam informação suficiente que signifique algo para a aplicação. O objetivo é entregar dados chamados *bundles* de um dispositivo para outro na presença de uma conexão oportunista usando diferentes protocolos de transporte assumindo que os nós podem armazenar e repassar os dados lidando com a interrupção de alguns links (HELGASON *et al.*, 2010).

Existem diferentes tipos de redes DTNs, dependendo da natureza de sua aplicação.

2.2.2 Características

As propriedades de uma DTN são muito diferentes das de uma rede de internet convencional, na qual algumas suposições estão implícitas, como a estabilidade da conexão, uma baixíssima taxa de perda de pacotes e o atraso de propagação razoavelmente baixo. As DTNs nem sempre satisfazem essas três propriedades e, às vezes, nenhuma delas, impossibilitando a utilização dos protocolos convencionais desenvolvidos para este tipo de rede (TCP e UDP) (ZHANG, 2006).

Para Jain, Fall e Patra (2004), este tipo de rede é conhecido por vivenciar frequentes desconexões de longa duração devido a mobilidade dos nós e, às vezes, nunca ter um caminho fim-a-fim conectando os dispositivos de origem e destino simultaneamente.

Para lidar com este problema de desconexão constante, uma abordagem simples é de complementar o protocolo *store-and-forward* para um do tipo *store-carry-forward* (SCF). Neste modelo, os dispositivos devem ser capazes de armazenar o dado por um tempo considerável, agindo como *data-mules*, pois um próximo salto pode não estar disponível de imediato. O problema em projetar tal protocolo reside na dificuldade de se encontrar a melhor opção para o próximo salto. O melhor hospedeiro para uma mensagem é aquele que tem maior chance de entregar a mensagem com sucesso (ZHANG, 2006).

O problema de roteamento em uma DTN tem muitas variáveis como a característica de topologia dinâmica e a demanda por tráfego. O completo conhecimento dessas variáveis facilitaria a computação de rotas ótimas. Entretanto, com um conhecimento parcial, a habilidade de computar rotas ótimas é dificultada e a performance é inferior (JAIN; FALL; PATRA, 2004).

De acordo com Zhang (2006), como existem vários tipos de DTNs, algumas com características únicas, novos protocolos devem ser desenvolvidos para satisfazer tais aplicações. Casos onde a trajetória dos dispostivos são conhecidas, se a capacidade de energia e armazenamento são pequenas ou em casos extremos onde a rota dos dispositivos são completamente aleatórias. Diferentes casos demandam diferente soluções.

O grafo de uma DTN consiste em um multigrafo direcionado, no qual mais de um link pode existir entre um par de nós. A razão para usar um multigrafo é simples: deve ser possível selecionar entre duas conexões distintas qual o caminho percorrer para trocar informações entre o mesmo par de nós (JAIN; FALL; PATRA, 2004).

2.3 Redes Oportunistas

A rede que será formada a partir desta aplicação não dependerá de nenhum tipo de infraestrutura. Este tipo de abordagem se justifica partindo do princípio de que todos os nós da rede são terminais e/ou intermediários.

2.3.1 Definição

Este tipo de conexão não necessita de infraestrutura para operar, os dispositivos se encontram em constante movimento. Os nós comunicam-se quando estão dentro da área de alcance um do outro. O nó pode descartar ou retransmitir os pacotes (no caso do nó retransmissor). Desta maneira, o pacote consegue ser enviado pela rede passando de nó em nó até chegar ao seu destinatário (ZHANG, 2006).

Nas redes oportunistas, os dispositivos móveis podem comunicar-se entre si, mesmo que nunca exista uma rota fim-a-fim conectando os dispositivos (PELUSI; PASSARELLA; CONTI, 2006).

2.3.2 Características

Muitos dos conceitos de Redes Oportunistas vem do estudo sobre DTNs(PELUSI; PASSARELLA; CONTI, 2006).

A rede Ad-hoc para dispositivos móveis é considerada uma tecnologia muito promissora. No entanto após anos de pesquisa na área, este produto ainda não foi incorporado ao consumo em massa. O principal motivo disso ter ocorrido é a falta de uma abordagem eficiente para uma rede ad hoc sem infraestrutura multi-hop (CONTI; GIORDANO, 2007).

Neste tipo de rede, os dispositivos aparecem de forma desordenada e sem aviso prévio, causando dificuldades na criação de uma topologia de rede bem estruturada, fazendo com que o roteamento convencional se torne pouco eficiente. O melhor caso é quando se conhece a topologia da rede criada e

assim pode-se criar um algoritmo de roteamento ótimo. Cada dispositivo está sujeito a sair da zona de alcance da rede e, talvez, retornar em algum momento, assim como nunca mais retornar (NGUYEN; GIORDANO, 2009).

Cada dispositivo integrante da rede age como se fosse um host e/ou intermediário e pode armazenar e repassar os dados entre os dispositivos, assim o alcance da rede aumenta com o aumento de nós pertencentes à rede (NGUYEN; GIORDANO, 2009).

2.4 Redes Peer-to-Peer

Em se tratando apenas de nós terminais, a utilização de um servidor para qualquer fim não se torna uma opção viável. Desta maneira, abordagens de conexões fim a fim são as mais indicadas para este tipo de problema.

2.4.1 Definição

Nas redes peer-to-peer, ou somente P2P, há uma dependência mínima (ou nula) de servidores com infraestrutura que permanecem sempre ligados, é uma rede onde cada nó (peer) se comporta como cliente e servidor simultaneamente, onde os nós se conectam diretamente entre si. Cada nó da rede é responsável por fornecer recursos para a rede, seja tráfego, processamento ou armazenamento. Os recursos são fornecidos e consumidos diretamente pelos nós existentes sem a necessidade de servidor para coordenar o processo (KUROSE, 2005).

2.4.2 Características

A ausência de um servidor centralizado e responsável por gerir todas as funcionalidades da rede pode ser sua característica mais marcante. Em uma rede P2P os nós são independentes e não possuem garantia de disponibilidade, o que faz com que os projetistas contornem esse problema utilizando a

técnica de replicação de dados, que nada mais é do que fazer cópias do conteúdo e armazenar em vários lugares diferentes, garantindo assim uma alta confiabilidade e reduzindo a probabilidade do usuário não conseguir acessar algum conteúdo.

A escalabilidade global é um dos princípios de um sistema P2P, uma vez que a união de nós é que permite a alta taxa de confiabilidade da rede, é necessário que a aplicação suporte o acesso de uma grande quantidade de usuários a uma enorme quantidade de objetos disponibilizados pela rede.

Outro fator importante para aplicações P2P é a privacidade dos usuários, que é obitida através da criação de uma rede virtual em nível de aplicação que controla o roteamento e interfere significantemente no desempenho da aplicação(KUROSE, 2005).

2.5 Conexões Ad-hoc-less

Apesar de focar em dispositivos móveis, uma futura implementação poderia utilizar de um sistema de conexão híbrido para potencializar o alcance da rede. Pensando por este lado é possível ampliar a utilidade do sistema e potencializar a sua funcionalidade.

De acordo com Trifunovic *et al.* (2011), Ad-hoc-less é uma forma de se criar uma rede sem fio local híbrida, onde um dos dispositivos assume, momentaneamente, o papel de um *Acces Point* (Ponto de acesso, AP), para que os outros possam estabelecer uma conexão. AP é um dispositivo dedicado exclusivamente ao reenvio de pacotes de terceiros, ou seja, o ponto em comum que liga todos os participantes da rede. Este modo também tem um consumo excessivo da bateria do dispositivo, causando prejuízo ao usuário que se encontra em tal estado, além de não poder requisitar e nem fornecer dados pela rede.

Esta arquitetura de rede oferece cinco variantes do serviço sendo que cada uma foca em um princípio, são elas:

1º Estático - Neste serviço, o dispositivo que estiver no modo AP fica impossibilitado de retornar ao modo de procura até que todos os dispositivos conectados a ele sejam desconectados por vontade do usuário.

2º Estação flexível - Indicado para escritórios, esta variante do serviço gera várias redes independentes que não podem comunicar-se entre si, no entando, a vantagem está no momento em que um dispositivo em movimento se encontra com outro AP rapidamente na medida que vai caminhando pelo ambiente.

3º AP flexível - Este modelo é uma variante do primeiro (Estático). A unica diferença é que neste modo o AP pode desconectar os nós nele conectados para que possa voltar ao modo de procura, e assim diminuir o gasto de energia do dispositivo.

4º Manual (AP) - Neste modo, o próprio usuário transforma seu dispositivo em um AP manualmente. O dispositivo permanece no estado AP até que haja uma intervenção manual do usuário para que este volte ao modo de procura.

5º APs fixos - Esta é, talvez, a abordagem mais óbvia, onde o AP é um dispositivo que dispõe de grandes quantidades de energia e capacidade de armazenamento, esse é o ponto positivo para que nao seja necessário que um dispositivo afim de trocar dados precise se tornar um AP.

2.6 Wifi Direct

O Wifi Direct é uma tecnologia desenvolvida pela Wifi Alliance, no intuito de conectar vários dispositivos sem a necessidade de um Ponto de Accesso (AP). Esta tecnologia permite que celulares, televisões, impressoras, etc formem uma rede capaz de compartilhar dados. Para isso é preciso que apenas um dos aparelhos possua a tecnologia do Wifi Direct, mesmo que sejam de fabricantes diferentes.

Os dispositivos com Wifi Direct se comunicam estabelencendo um grupo chamado *P2P Groups*, que são equivalentes às estruturas das redes convencionais. Apenas um dos dispositivos que fazem parte da rede assume as funcionalidades de um AP e é chamado de *P2P Group Owner (P2P GO)*, os demais dispositivos recebem o nome de *P2P clientes*. É importante ressaltar que estas funcionalidades não são fixadas, portanto o papel de cada dispositivo pode mudar com o tempo no grupo. Para criar um grupo, diferentes dispositivos entram em contato e então negociam qual papel cada dispositivo irá realizar. Assim que um dispositivo assume o papel de P2PGO, outros aparelhos podem se conectar da mesma maneira que uma rede Wifi convencional.

O Wifi peer-to-peer (p2p) do sistema Android 4.0 (API 14) ou superior permite que dispositivos, com o devido *hardware*, se conectem através da interface de rede sem fio. o Wifi p2p cumpre totalmente com os requisitos do ***Wifi Alliance Certified*** e portanto facilita a conexão entre vários dispositivos de diferentes fabricantes e funcionalidades. Esta API possui implementada as várias funcionalidades que são necessárias para a criação de uma aplicação que utilize o WiFi como conexão(WIFI...,).

2.7 Bluetooth

O Bluetooth é uma tecnologia de conexão sem fio de baixo custo energético e de curto alcance que opera na banda ISM 2.4 GHz, capaz de conectar até oito dispositivos ao mesmo tempo sem problemas de sincronização. O bluetooth é uma tecnologia muito difundida e está presente na maioria dos dispositivos móveis e até mesmo em aparelhos fixos.(HAARTSEN, 1998)

Os dispositivos Bluetooth organizam-se automaticamente para formar uma rede denominada **piconet**, onde até sete dispositivos podem participar. Um deles é eleito o "mestre". O mestre da piconet é quem define a sequência de saltos de canais de rádio. Esta sequência depende do relógio e do

endereço MAC do mestre, assim os "escravos"podem sintonizar na mesma sequência. Cada dispositivo Bluetooth pode participar de mais de um piconet.(PFüTZENREUTER,)

A distância nominal alcançada por esta tecnologia (Bluetooth 1.0) é de 10 metros quando o rádio usa uma potência de envio de 0dBm, podendo ser aumentada para até 100 metros com o uso de antenas externas, variando ainda mais as aplicações possíveis de serem construídas com essa tecnologia. A taxa de transferência alcançada chega a 1Mbit/segundo(HAARTSEN *et al.*, 1998). Nas versões mais atuais do bluetooth é possível chegar a uma taxa de até 24Mbits/segundo.

Na criação da tecnologia Bluetooth foi levado em consideração que a faixa de frequência de rádio na qual opera (2.4GHz ISM) é livre e, portanto, está sujeita a interferência de vários outros dispositivos que a utilizam. A imunidade a interferência pode ser obtida pela eliminação da interferência ou evitando-a. Entretanto, o tipo de interferência muda o tempo todo, fazendo com que o ato de evitar os ruídos seja mais interessante, desde que o sinal seja transmitido em pontos em que a frequência e/ou o tempo exibem um menor nível de ruído. Como a banda 2.4GHz fornece cerca de 80MHz de taxa de transmissão e a maioria dos sistemas *wireless* não conseguem alcançar todo esse potencial, existe uma grande probabilidade de que alguma faixa ofereça níveis de ruídos mais baixos, este sistema se chama Frequency-hopping spread spectrum (FHSS) (HAARTSEN, 2000).

2.8 Sistemas peer-to-peer

O estudo de protocolos de roteamento peer-to-peer se justifica por ter sido a abordagem escolhida na realização deste trabalho e pelo fato de que é necessário conhecer as soluções já apresentadas para poder escolher a que melhor satisfaz os requisitos desta aplicação.

2.8.1 Gnutella

O Gnutella é um dos protocolos P2P mais conhecidos, possui código aberto e foi desenvolvido em meados de 2000. É uma rede puramente P2P (opera sem um servidor central). Os clientes trocam dados entre si e as requisições são feitas ciclicamente pela rede. O processo de descoberta de nós e conteúdo são feitos através de 5 tipos de pacote de controle, são eles:

1. Ping
2. Pong
3. Query
4. Hit Query
5. Push

O primeiro é utilizado para descobrir quem são os nós vivos na rede, enquanto que o *pong* é a resposta ao *ping* de um vizinho encontrado. Um pacote *query* é enviado quando o usuário faz uma requisição para buscar conteúdo, e caso seja encontrado algum resultado um *hit query* é enviado de volta ao nó usuário que enviou a *query* e então um pacote do tipo *push* é enviado para ordenar a transferência do arquivo desejado, se mais de um *hit query* for recebido, então a aplicação pode realizar multiplas conexões com vários nós para obter um mesmo objeto, aumentando a largura da banda e resultando em maior taxa de transferência.

Uma vez que um nó entra na rede, este envia um pacote *ping* em *broadcast* para sinalizar que está vivo na rede. Como a rede Gnutella não possui confiabilidade devido aos nós que entram e saem da rede sem aviso, os nós que estão vivos na rede devem enviar periodicamente pacotes *ping* para saber quem ainda está disponível(RIPEANU, 2001).

2.8.2 BitTorrent

O BitTorrent é um protocolo de P2P muito utilizado para disseminação de dados. O conjunto de nós que participam da rede de transferência de um determinado arquivo se chama ***torrent***. Para realizar a transferência do arquivo entre os clientes, o arquivo é dividido em blocos de tamanhos iguais, geralmente 256Kbytes. Quando um nó entra neste torrent, ele não possui nenhum bloco do arquivo e, portanto deve começar realizando *download* de alguns blocos. Assim que um bloco é obtido por esse novo nó, ele pode ou não (dependendo da demanda do torrent) fazer *upload* desses blocos que já possui e dar continuidade à rede, dessa maneira, vai aumentando a disponibilidade dos blocos para os integrantes do torrent. Quando um nó termina o *download* de todos os blocos de um torrent, ele pode optar por deixar o torrent e parar de enviar blocos para outros nós, ou continuar enviando os blocos que possui. Um integrante do torrent também pode sair com apenas uma parte dos blocos do arquivo, e depois retornar para completá-lo.

Apesar de ser um sistema P2P, o BitTorrent necessita de apenas uma infraestrutura, o **rastreador**. Cada *torrent* precisa de um nó rastreador, este nó atua como um gerenciador de dispositivos no *torrent*. Quando um nó entra no *torrent*, ele se registra no rastreador e periodicamente informa o rastreador que ainda faz parte do *torrent*. Assim que um novo nó é registrado no *torrent* ele recebe do rastreador um subconjunto de IPs dos nós participantes e então o nó tenta estabelecer conexões TCP simultâneas com esses pares "vizinhos". A técnica utilizada para definir qual bloco de dado será enviado primeiro é denominada *rarest first* (o mais raro primeiro), a ideia é determinar qual bloco que o requisitante não possui e que é mais raro dentre seus vizinhos. Assim o *torrent* garante que a disponibilidade dos blocos permaceça constante(KUROSE, 2005).

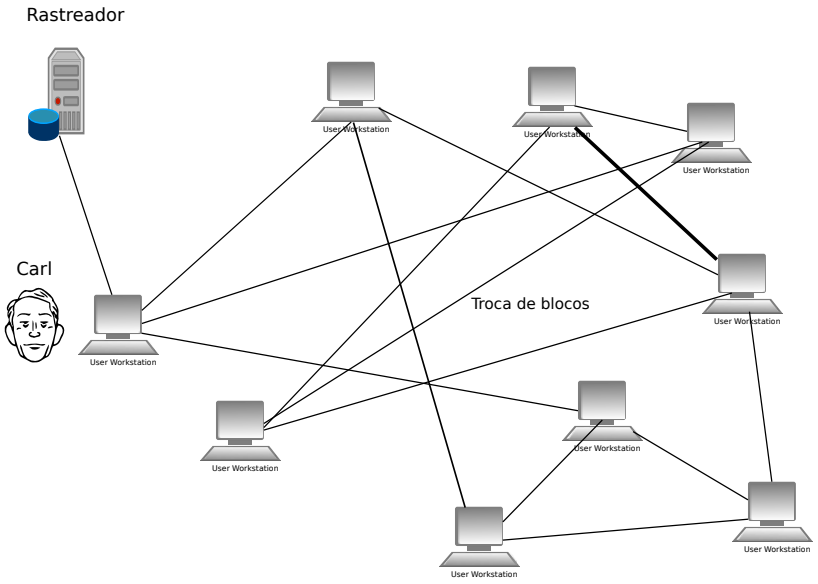


Figura 2: Distribuição de dados na rede do BitTorrent.

2.9 Sincronização e Descoberta de Vizinhos

Nesta seção será apresentado o problema de descobrir quais dispositivos estão passíveis de realizar conexão ao seu redor. Em uma rede móvel, é necessário que os dispositivos descubram quais outros dispositivos estão ao seu alcance.

Descobrir os possíveis pares ao qual você deve se comunicar é um processo que nunca acaba, principalmente pelo fato de que o projeto diz respeito a conexões que podem sofrer com a ausência de nós ao alcance por um período de tempo muito longo.

O módulo de descoberta descobre qual nó vizinho está rodando a aplicação ao enviar pacotes em *broadcast* de maneira que, quem estiver rodando a mesma aplicação e estiver dentro do alcance do rádio, irá responder. Esse pacote contém o endereço único do nó, possibilitando que seja estabelecida uma conexão entre ambos (HELGASON *et al.*, 2010).

Um sistem de configuração automática deve ser capaz de adicionar novos dispositivos à rede sem qualquer desconexão e sem qualquer tipo de interferência do usuário. Os dispositivos devem se reconhecer durante toda a vida útil da rede estabelecida, fazendo com que a rede tenha uma área abrangente cada vez maior (SUNDRAMOORTHY *et al.*, 2009).

Em um sistema de descoberta, cada dispositivo, podendo ser hardware ou software, deve receber uma identificação única. O nome é composto por uma sequência de bits capaz de diferenciar os vários dispositivos (SUNDRAMOORTHY *et al.*, 2009).

2.10 Descoberta e armazenamento de conteúdo

A descoberta e armazenamento de conteúdo é uma das funcionalidade mais críticas deste tipo de aplicação, se os dados corretos não forem retornados para os usuários, a aplicação perde grande parte da sua utilidade.

Quando se precisa de um método para localizar dados desejados em uma rede p2p, seja na forma de arquivos, tabelas, fotos, musicas, vídeos e etc, a solução mais simples é de procurar pelo nome do arquivo. Esta procura se torna obsoleta na maioria dos casos. Em se tratando de imagens feitas a partir de uma câmera fotográfica, o nome do arquivo é dado sequencialmente, portanto não trazem nenhuma informação sobre o conteúdo da foto (objetos, local, pessoas na foto, etc). Neste caso, uma abordagem mais elaborada tem que ser avaliada para recuperar os dados que o usuário realmente deseja. Para isso, associamos informações em forma de texto aos dados, chamados de metadados. Dessa maneira é possível retornar aos usuários um resultado com maior afinidade ao que está sendo requisitado.

Quando o problema é sobre retornar imagens sobre um termo pesquisado pelo usuário na internet, três técnicas são as mais comumente utilizadas:

Baseado em Contexto Neste método, é necessário possuir habilidades que os computadores não possuem, o que faz com que um humano seja necessário

para assistir o processo (análise de alto nível). A análise de baixo nível leva em conta as cores, textura e localização espacial dentre vários outras características do pixel para organizá-las de acordo com uma classificação requerida (GUDIVADA; RAGHAVAN, 1995).

Baseado em Texto Este método é utilizado pelo Google e pelo Yahoo para procurar através de textos que cercam a imagem, nome do arquivo da imagem, anotações, etc (WANG; LIU; CHIA, 2006).

Baseado em Ontologias Neste tipo classificação, os dados são separados em classes que vão se ramificando e representando cada vez mais especificamente o conhecimento sobre um determinado dado . As classes são criadas de acordo com relações previamente estabelecidas e então é utilizado um métodos de aprendizado de máquina para treinar e classificar um grupo de dados, cada um de acordo com suas características.

O armazenamento dos metadados é feito internamente, pois qualquer aplicação que utilize de uma conexão externa para requisitar/ enviar informações se tornam dependentes de uma rede com internet(JAN *et al.*, 2010). Uma vez que o foco deste projeto é evitar o uso da internet na obtenção de dados, não é viavel o uso desta para qualquer fim.

2.11 Haggle

O Haggle é um sistema de disseminação de dados semelhante ao proposto neste trabalho. Desenvolvido para várias plataformas, ele também tem a função de disseminar dados entre dispositivos móveis via redes oportunistas. O objetivo do projeto Haggle é de fornecer uma aplicação capaz de disseminar dados entre dispositivos móveis com um gasto de energia reduzido. Para diminuir o gasto de energia que este tipo de aplicação demanda, foi convencionado que os dados sejam transferidos respeitando um rank, onde esse rank é determinado pela quantidade de usuários que fazem a requisição do mesmo arquivo, ou seja, quanto mais pessoas procurando

pelo mesmo arquivo, maior a prioridade no rank de transferência. Os dispositivos que compõem a rede trocam interesses e então é calculado o rank de cada requisiçao feita pelos nós da rede. A partir daí começa a disseminação dos dados, sendo que o primeiro objeto a ser enviado é o primeiro da lista do rank. O sistema também leva em consideração as requisições feitas por outros dispositivos que foram previamente encontrados, fazendo com que nós que nunca estiveram próximos possam trocar informações de requisição(NORDSTRÖM; GUNNINGBERG; ROHNER, 2012).

Segundo Nordström, Gunningberg e Rohner (2012), o Haggle foi o primeiro projeto que propôs um sistema que determinasse a ordem de transferência dos dados.

O Haggle utiliza uma forma interessante e simples para identificar os dados a serem requisitados e transferidos pelos usuários. Primeiramente, o usuário tem que cadastrar o dado na aplicação, então este recebe um formato de dados exclusivo da aplicação, chamado de *data-object*. Nestes *data-objects* estão inclusos as metainformações que são necessárias para a identificação do dado (NORDSTRÖM; GUNNINGBERG; ROHNER, 2012).

O Haggle é um sistema muito mais complexo do que o proposto neste trabalho. Uma de suas características é a maneira como gerencia as transferências de dados na rede, a utilização de um protocolo de roteamento próprio, encapsulamento de dados para o manuseio da aplicação, um sistema de procura por objetos complexo e ainda conta com um sistema de segurança que verifica se os dados são confiáveis e se desejado, o usuário pode descartar os dados se achar que não o são.

Apesar de o Haggle ser um sistema desenvolvido para várias plataformas e ofereça maior segurança na transmissão de dados, ele não garante que os dados que o usuário requisita irão realmente ser disseminado devido a existência de uma ordem para enviar, pode ser que um dado nunca seja transmitido na rede.

3 Metodologia

Segundo Wainer (2007), o conhecimento em ciência da computação é obtido usando as seguintes grandes metodologias:

- Pesquisa Analítica
- Pesquisa Quantitativa
- Pesquisa Qualitativa
- Pesquisa Bibliográfica

A pesquisa desenvolvida classifica-se em grande parte como pesquisa qualitativa, pois não pretende mensurar uma variável de performance e sim fazer um estudo aprofundado sobre um sistema para troca de dados no cotidiano das pessoas.

Quanto aos seus objetivos, sua classificação é descritiva, pois são feitas observações, análises e registros a respeito do *software* desenvolvido.

3.1 Procediementos Metodológicos

- **Levantamento Bibliográfico**

  Nesta parte foi feito um estudo aprofundado sobre os requisitos para realizar tal trabalho. Foram pesquisados artigos, livros, revistas, etc. Estes artigos serviram de base para pensar nas possíveis soluções e abordagens na criação do protótipo.

- **Desenvolvimento**

  Nesta fase começou a implementação do protótipo proposto por este trabalho.

- **Avaliação**

  E, por fim, a avaliação do sistema obtido foi feita, o desempenho do bluetooth e do sistema de busca de dados foram testados de forma a validar a utilização destas abordagens.

4 Desenvolvimento do Aplicativo ShareDroid

Neste capítulo serão apresentadas as idéias, problemas e soluções encontradas durante o desenvolvimento do aplicativo.

Neste trabalho foi desenvolvido um protótipo que se conecta com apenas um dispositivo móvel através do bluetooth. Implementações mais complexas das funcionalidades do projeto estão sendo estudadas e testadas para a continuação do projeto.

O desenvolvimento do protótipo do aplicativo ShareDroid foi repartido em 4 módulos:

- Criação de uma lista de interesses
- Função para escolher o caminho da pasta de compartilhamento
- Um sistema para identificar os arquivos através de metadados
- Módulo de conexão via Bluetooth

Em cada módulo foi feito um levantamento dos requisitos necessários e então deu-se início a codificação e, posteriormente, aos teste referentes àquele módulo. Ao final de cada módulo uma nova funcionalidade era disponibilizada na aplicação, caracterizando como uma metodologia ágil de desenvolvimento de *software*.

4.1 ShareDroid

O ShareDroid é um aplicativo que tem como finalidade conectar dois dispositivos que rodam o sistema operacional Android, e realizar transferência de dados de forma oportunista utilizando uma arquitetura *peer-to-peer* para isso. O aplicativo é capaz de abstrair as dificuldades da criação e configuração da rede, também ocultando o próprio ato da transferência dos dados, porém avisando quando alguma operação for iniciada/concluída. A indentificação dos dados se dá por metainformações previamente estabelecidas, que

deverão ser fornecidas pelos usuários. O usuário que deseja algum dado deverá fornecer informações chave para que este seja encontrado em um outro dispositivo, também conectado à rede.

O Android é um sistema operacional desenvolvido pela Google para dispositivos móveis, tais como celulares e *tablets*. Para desenvolver aplicativos, é disponibilizada uma ferramenta chamada ADT (Android Development Toolkit) a qual consiste no ambiente de programação Eclipse e todas as APIs e ferramentas para *debug* necessárias na criação de aplicativos para esta plataforma.

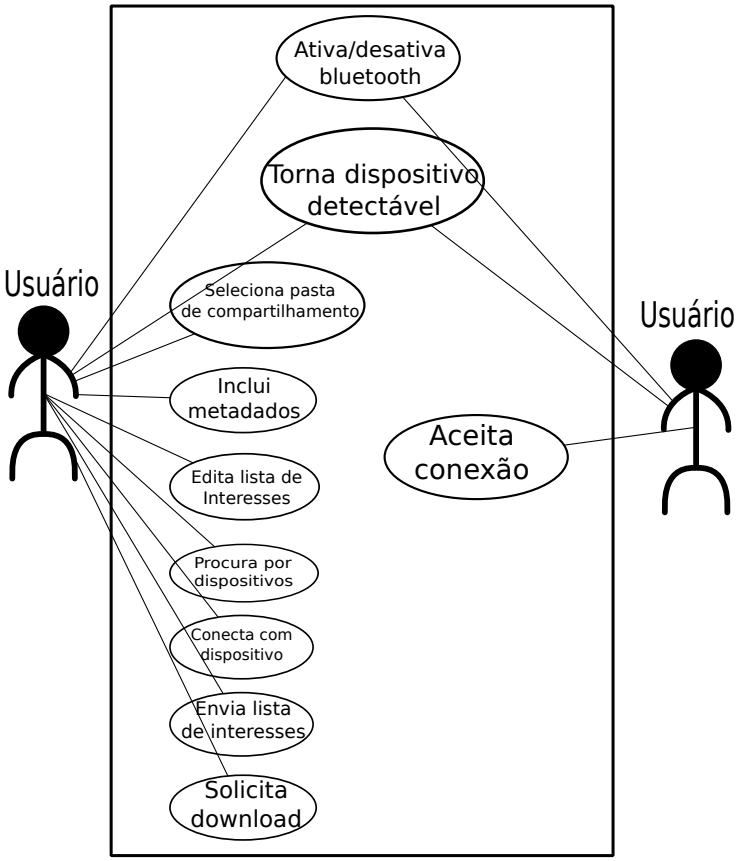


Figura 3: Casos de uso da aplicação ShareDroid

4.1.1 Lista de interesses

Para a criação desta funcionalidade, foi utilizado um sistema com banco de dados. O sistema Android possui total suporte às funcionalidades do SQLite. Os dados da lista são armazenados e removidos do banco de dados de acordo com a utilização do usuário. O banco é acessível pelo nome por toda a aplicação e apenas por ela. A lista que aparece na tela é carregada através de um *Array* fruto de uma query feita no banco de dados, quando algo é adicionado pelo usuário, primeiro ele é adicionado no banco para depois ser carregado para o *array* e então a lista é atualizada na tela do dispositivo. Para deletar um ítem ocorre o mesmo processo, o dado é removido do banco e depois a interface é atualizada.

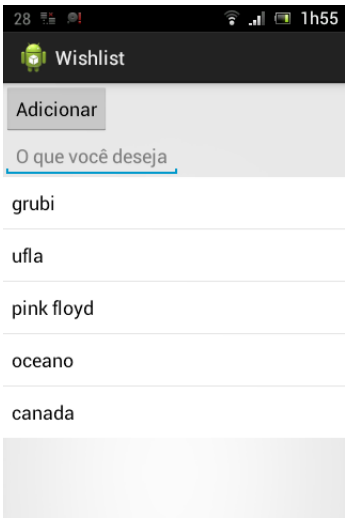


Figura 4: Lista de interesses

4.1.2 Escolha da Pasta de Compartilhamento

Para que o usuário possa escolher, foi desenvolvida uma atividade que lista os diretórios existentes à partir do cartão de memória. Quando uma pasta é selecionada, o nome da pasta é adicionado à variável do caminho

final e, se existir, seus sub-diretórios serão mostrados ao usuário. O caminho da pasta é salvo em uma variável no sistema para a utilização das funcionalidades de busca de conteúdo e também quando algum arquivo é baixado, o qual é posicionado na pasta escolhida. Também é possível criar sub-diretórios a partir desta atividade.

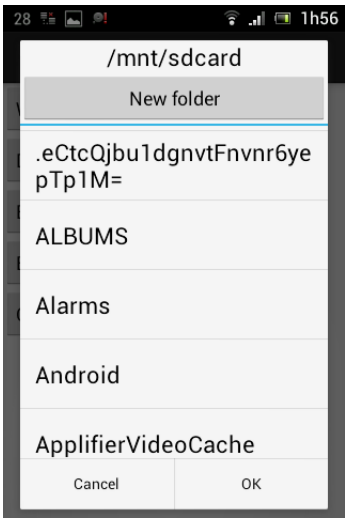


Figura 5: Escolha do diretório.

4.1.3 Identificação de arquivos por metadados

O método escolhido foi o modelo baseado em ontologias. Ele consiste de um sistema onde o próprio usuário adiciona informações sobre a imagem, ou arquivo, a ser compartilhada, sendo o nome do arquivo o único atributo essencial de qualquer objeto, ainda podendo ser adicionadas informações adicionais que o usuário desejar, através de campos auxiliares. Todas as informações são armazenadas em um arquivo XML armazenado internamente no dispositivo, onde será feita a busca para, posteriormente, retornar ao usuário um resultado com alguma precisão.

Primeiramente, para que um sistema de ontologias fosse criado, foi necessário estabelecer quais tipos de arquivos que um usuário pode requisitar e quais atributos são mais funcionais. Os quatro tipos mais comuns de arquivos são os arquivos de áudio, vídeo, imagem e executáveis. Após esse estudo, foram estabelecidas quais as informações que um arquivo deve possuir para ser facilmente encontrado. O tipo de informação varia de acordo com o tipo de arquivo escolhido. Para todos os arquivos, o nome do arquivo é um campo criado automaticamente no sistema.

Para o armazenamento de informações dos arquivos, foi utilizado um sistema com arquivo XML, o qual armazena as informações que o usuário fornece através da interface do aplicativo. O arquivo XML de metadados é criado assim que o usuário tenta cadastrar o primeiro arquivo, este arquivo XML é armazenado publicamente no cartão externo do dispositivo com o nome ShareDroid.xml

Para que um registro de um arquivo seja criado, é necessário que o usuário escolha o arquivo e tipo de arquivo na interface do usuário e então forneça as informações acerca do arquivo. Caso o usuário não forneça algum campo, ele será representado pela palavra null no arquivo XML. O usuário deve fornecer informações sobre o arquivo para que um registro de sua existência seja efetuado no sistema, portanto arquivos que não forem editados não poderão ser encontrados mesmo que estejam na pasta compartilhada.

Para criar tal funcionalidade, foi utilizada uma biblioteca externa chamada XStream[1]. Esta biblioteca é de facil manuseio e tem as funcionalidades de transformar objetos Java para XML e vice-versa.

A pesquisa realizada no arquivo XML se dá de forma que cada ítem da lista de interesses é procurado por toda a extensão do arquivo XML como uma substring de cada linha. Quando é encontrado um resultado positivo, é retornado o caminho absoluto do arquivo e enviado para o dispositivo requisitante.

[1]Esta biblioteca pode ser baixada no link: http://xstream.codehaus.org/

```
<XmlCreator.Music>
  <fileName>Música.mp3</fileName>
  <artistName>Artista</artistName>
  <songName>Nome da Música</songName>
  <album>Álbum</album>
  <year>2000</year>
</XmlCreator.Music>
<XmlCreator.Video>
  <fileName>Vídeo da formatura.avi</fileName>
  <title>formatura</title>
  <producer>Felipe Milani</producer>
  <year>2006</year>
  <genre>Festa</genre>
</XmlCreator.Video>
```

Figura 6: Exemplo de um arquivo XML para armazenamento de metadados.

4.1.4 Módulo de Conexão Bluetooth

A plataforma Android possui total suporte à tecnologia do Bluetooth, a qual permite que uma aplicação requisite uma lista com os aparelhos que ja foram pareados pelo menos uma vez, que se faça uma busca por dispositivos com o Bluetooth ativado e que ambos se conectem e transfiram dados entre si. Visando maximizar a taxa de transferência do protótipo, foi implementado que apenas uma conexão seja efetuada por vez, apesar de ser possível realizar multiplas conexões simultâneas.

Na aplicação é possível habilitar e desabilitar o Bluetooth, tornar-se visível para outros dispositivos, listar dispositivos previamente pareados e visíveis ao alcance da rede. Para que um dispositivo possa encontrar um ao qual nunca foi pareado previamente, é necessário primeiro tornar-se visível, caso contrario não será mostrado na lista de possíveis contatos. Todas essas operações são feitas utilizando a API Bluetooth disponível no ADT.

A partir do momento em que o usuário lista os dispositivos para efetuar conexão, a aplicação inicia uma *thread* que fica encarregada de aceitar conexões. Assim que algum dos usuários clica em um item na lista, é iniciada uma *thread* que tenta efetuar uma conexão com o dispositivo clicado. Se ambos nunca realizaram uma conexão, a aplicação irá perguntar ao dis-

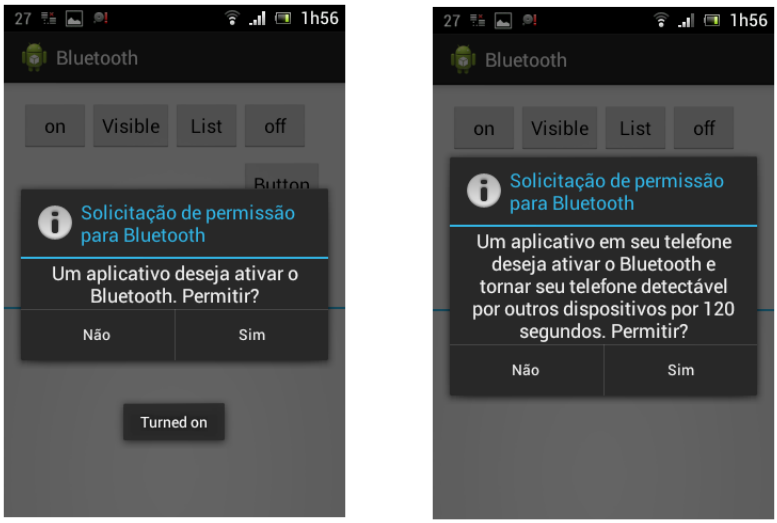


Figura 7: Dialogos para ativar e tornar o aparelho visível.

positivo que está recebendo a tentativa se realmente deseja conectar-se, se a resposta for sim, então ambos irão se conectar e esperar que a lista de interesses seja enviada por um deles (deve-se enviar manualmente a lista através de um botão). Quando os dispositivos se conectam com sucesso as *threads* responsáveis por aceitar conexões de ambos os dispositivo é finalizada e uma nova *thread* responsável pela leitura do socket de comunicação é iniciada. Se algum ítem na lista de interesse for encontrado, então será perguntado para o dispositivo requisitante se ele deseja realizar o *download* do arquivo encontrado.

O ShareDroid se comunica com três tipos de mensagens, além das utilizadas pela API Bluetooth para procurar dispositivos e estabelecer conexão. As mensagens são compostas de duas partes, o primeiro byte é o cabeçalho da mensagem, ele indica qual tipo de mensagem e o que deve ser feito com o resto da mensagem e a segunda parte é o conteúdo válido da mensagem.O primeiro tipo é a mensagem que envia o conteúdo da lista de interesses para o dispositivo pareado. Quando esta mensagem é recebida, uma operação de

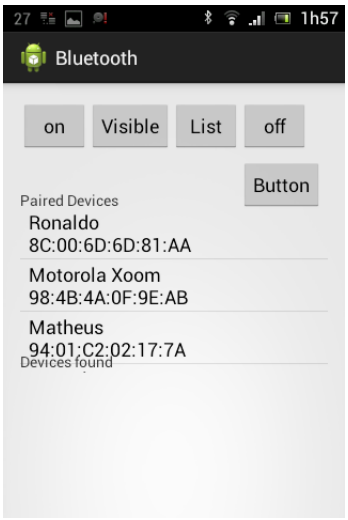


Figura 8: Lista de dispositivos disponíveis.

pesquisa no arquivo de metadados é acionada para cada ítem na lista, em caso de algum ítem combinar, o dispositivo envia uma mensagem, para cada ítem encontrado, contendo o nome do arquivo junto com a extensão, caso não seja encontrado nenhum resultado a aplicação não envia nada e procura pelo próximo ítem da lista . A terceira mensagem é o pedido de *download*, se o usuário deseja transferir o conteúdo encontrado, é enviada a resposta contendo o nome do arquivo desejado e então o arquivo é, finalmente, transferido.

4.1.5 Diagrama de Classes

O diagrama de classes exibe as classes que compõem o protótipo desenvolvido, o projeto é dividido em quatro funcionalidades distintas e principais iguais às que foram apresentados no ítem 4.2. Nele é possível observar que a classe MainActivity é o centro do diagrama. Esta classe é responsável por

mostrar a tela inicial do *software* e fornecer acesso a todas as outras funcionalidades do protótipo através dos métodos goto*[3].

As classes Imagem, Audio, Video e ProgramFile são herdeiras da classe Objeto, que representa a generalização dos tipos de arquivos comumente encontrados nos dispositivos e cada classe herdeira representa uma especificação de cada tipo, cada um com seus atributos específicos.

A classe Attr_Editor é responsável por fornecer ao usuário a inteface em que é possível escolher um arquivo e então editar seus atributos. Esta classe fornece um objeto do tipo Objeto com os atributos editados como parâmetro para a classe XmlCreator para que possa ser criado um registro de sua existência no arquivo XML.

A classe XmlCreator faz a função de manipular o arquivo de metadados. Ela transforma um objeto da classe Objeto em uma String no formato XML e também é capaz de constuir um objeto específico através de uma String com uma descrição no formato XML. É nesta classe que a busca por um determinado item da lista acontece, cada item da lista é passado como parâmetro para o método buscaObjPorAtributo que por sua vez retorna o caminho absoluto deste arquivo no sistema de arquivo dispositivo.

A classe Wishlist é responsável por manipular a lista de interesses do usuário, para isso é instanciado um objeto do tipo DataHelper, onde foram implementadas todas as operações de banco de dados, desde a sua criação até o fechamento.

A classe BluetoothHandler é responsável por manipular o *hardware* do Bluetooth, permitindo obter as informações sobre os dispositivos para fornecê-las para a classe BluetoothConnectionManager. Esta é responsável por realizar a conexão entre dispositivos e definir o que será feito com as informações recebidas através do socket. As três *threads* criadas a partir desta classe , AcceptThread, ConnectThread e ConnectedThread são responsáveis

[3]* é uma generalização para os nomes das classes possíveis

por aguardar uma requisição de conexão, criar o socket de comunicação e realizar a leitura e escrita nos sockets criados, respectivamente.

A classe DirectoryChooserActivity é responsável por apresentar ao usuário a interface da aplicação de escolha da pasta de compartilhamento, enquanto que a DirectoryChooserDialog é reponsável por todas as funcionalidades, como a criação de novas pastas, navegação entre os diretórios até o armazenamento do caminho escolhido pelo usuário.

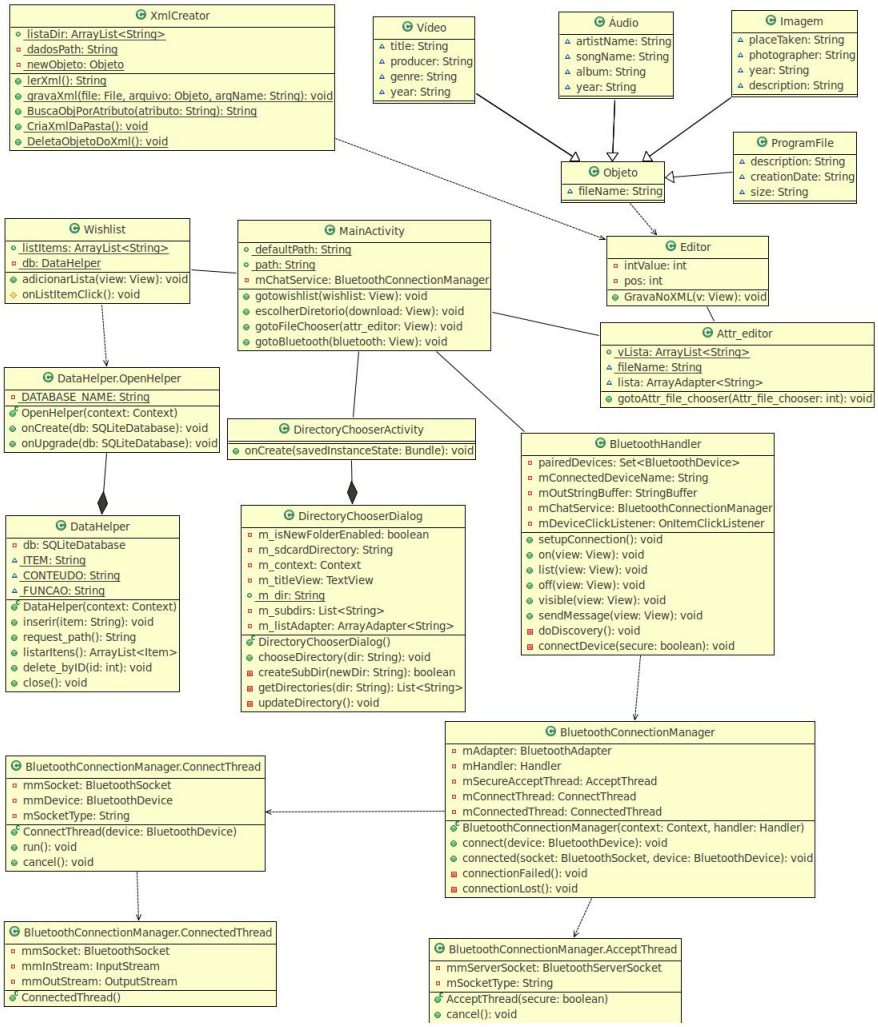


Figura 9: Diagrama de Classes simplificado do projeto ShareDroid.

5 Resultados

Neste capítulo será discutido os resultados apresentados pelos testes realizados com o aplicativo ShareDroid.

Testes foram realizados utilizando um smartphone *Sony Live! walkman* e um tablet *Motorola Xoom*, ambos com Android 4.0 Ice Cream Sandwich. Os testes realizados foram para verificar a precisão da busca de conteúdo realizada nos dispositivos baseado no tamanho da palavra pesquisada, e quantidade de metadados oferecidos pelo usuário do dispositivo em contato *versus* a quantidade de respostas possitivas obtidas e ainda foi aferido o consumo de energia durante a utilização da aplicação. A facilidade de navegação entre os menus foi testada por alguns usuários para entender se há dificuldade em utilizar o aplicativo.

5.1 Sistema de Busca de Conteúdo

Para este primeiro teste, foram introduzidos vários arquivos de diferentes tipos, obtidos a partir de *pendrives* e posicionados na pasta de compartilhamento, em ambos os dispositivos envolvidos. A lista de interesses foi povoada com 30 palavras com mais de 4 letras, sendo que 15 palavras combinavam com algum atributo de um dos arquivos, 15 palavras não combinavam com nenhum dos atributos. Foi feito um sorteio para saber qual palavra seria inserida em determinado dispositivo, totalizando 15 palavras para cada dispositivo. Os atributos dos arquivos foram editados de forma que representassem cada um da forma mais fiel possível. Após realizar a conexão e a pesquisa no dispositivo alheio, foi verificado que, em 8 dos casos, o atributo encontrado era condinzente com a palavra pesquisada e em 2, eles não eram referentes ao arquivo solicitado.

No segundo teste, os arquivos de dados e o XML foram mantidos, porém a lista de interesses foi modificada. A lista foi povoada com palavras de até 4 letras. Neste teste foi observado que, apesar do alto número de respostas

obtidas, os arquivos não correspondiam ao interesse pesquisado, ou seja, os arquivos encontrados não correspondiam ao real interesse da lista. Isso se deu devido ao fato de que o mecanismo de procura no arquivo XML fora implementado de forma que é pesquisado por uma substring do atributo, o que pode retornar resultados falsos quando a palavra pesquisada é pequena.

No terceiro teste desta funcionalidade, foram removidos todos os atributos dos arquivos, deixando apenas o nome do arquivo. A lista e os dados foram mantidos. Para este teste foi apurado que nenhum arquivo do tipo imagem foi encontrado, devido ao fato de que os usuários raramente alteram o nome do arquivo, atestando que a introdução de metadados para este tipo de arquivo é fundamental.

No gráfico abaixo, é mostrado o número de respostas obtidas quando uma única busca foi realizada, os arquivos que não foram encontrados não fazem parte do gráfico.

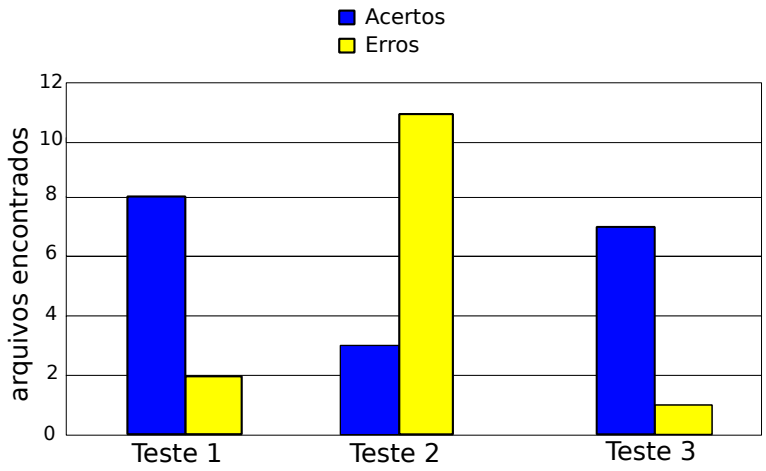


Figura 10: Gráfico do desempenho do sistema de busca do ShareDroid

5.2 Módulo de Conexão

Para esta funcionalidade, foram testados o tempo para estabelecer uma conexão, a velocidade de transferência e a agilidade na descoberta de novos dispositivos.

Uma das desvantagens do Bluetooth se mostrou na demora da varredura do meio para encontrar novos dispositivos, o processo é lento e custoso energeticamente. Porém, se a conexão for realizada com um aparelho ao qual ja tenha sido pareado, o processo é rápido, uma vez que dispensa a varredura do meio, e sem o custo excessivo de energia como no modo de descoberta. Em ambos os casos a conexão é estabelecida em menos de 5 segundos.

A taxa de transferência foi calculada pelo tempo em que um arquivo de 10MBytes demorou para ser transferido. Neste caso, a distância entre ambos os dispositivos era de 1 metro e demorou 220 segundos. Resultando em uma taxa de aproximadamente 50KB/s.

5.3 Interface do Usuário e Usabilidade

A interface gráfica do protótipo foi desenvolvida apenas para garantir que os testes das funcionalidade fossem executados, por isso apenas os botões e *layouts* básicos oferecidos pela ferramenta do SDK foram utilizados. Não foi feito nenhum tipo de avaliação sobre este quesito.

6 Conclusão e Trabalhos Futuros

Neste trabalho foi desenvolvido o ShareDroid, um aplicativo para disseminação de dados via bluetooth que utiliza uma arquitetura *peer-to-peer*. A aplicação permite ao usuário estabelecer conexão com dispositivos remotos e realizar transferência de dados via Bluetooth. É possível transferir qualquer tipo de dado desde que este tenha sido cadastrado na aplicação através da edição de metadados. O ShareDroid é um protótipo e muitas de

suas funcionalidades foram alteradas com o intuito de simplificar e viabilizar a implementação como um trabalho de conclusão de curso.

O aplicativo foi capaz de encontrar e transferir dados em dispositivos remotos com sucesso, o método de busca com metadados retornou resultados não desejados poucas vezes. A anotação de dados através de metadados é uma excelente opção, apesar de ser usuário-dependente. Ela se mostrou mais eficiente quando a única informação sobre o arquivo era seu nome. A tecnologia Bluetooth se mostrou eficiente no ato de transferir os dados, porém tendo em vista a demora em encontrar dispositivos disponíveis e o baixo alcance, é recomendado a utilização do WiFi no trabalho final.

Aplicações de rede que independem de infraestrutura se torna interessante para o usuário pelo fato de que não envolve uma estrutura central para organizar a rede entre os dispositivos, a qualquer momento e em qualquer local é possível estabelecer conexão e trocar conteúdo.

Como trabalho futuro, integrar um protocolo de roteamento que utilize o rádio WiFi, de preferência o desenvolvido pelo GRUBI, no ShareDroid. Outras funcionalidades também podem ser aprimoradas, como a busca e edição de metadados. Apenas a primeira modificação implicaria em um aumento de até 50m no alcance da rede acarretando em um aumento considerável na área de cobertura da rede do ShareDroid.

Referências

CONTI, M.; GIORDANO, S. Multihop ad hoc networking: The theory. *Communications Magazine, IEEE*, IEEE, v. 45, n. 4, p. 78–86, 2007.

GUDIVADA, V. N.; RAGHAVAN, V. V. Content based image retrieval systems. *Computer*, IEEE, v. 28, n. 9, p. 18–22, 1995.

HAARTSEN, J. Bluetooth-the universal radio interface for ad hoc, wireless connectivity. *Ericsson review*, v. 3, n. 1, p. 110–117, 1998.

HAARTSEN, J.; NAGHSHINEH, M.; INOUYE, J.; JOERESSEN, O. J.; ALLEN, W. Bluetooth: Vision, goals, and architecture. *ACM SIGMOBILE Mobile Computing and Communications Review*, ACM, v. 2, n. 4, p. 38–45, 1998.

HAARTSEN, J. C. The bluetooth radio system. *Personal Communications, IEEE*, IEEE, v. 7, n. 1, p. 28–36, 2000.

HELGASON, Ó.; YAVUZ, E.; KOUYOUMDJIEVA, S.; PAJEVIC, L.; KARLSSON, G. A mobile peer-to-peer system for opportunistic content-centric networking. In: ACM. *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*. [S.l.], 2010. p. 21–26.

HSU, C.; LIU, H.; SEAH, W. Opportunistic routing-a review and the challenges ahead. *Computer Networks*, Elsevier, 2011.

JAIN, S.; FALL, K.; PATRA, R. *Routing in a delay tolerant network*. [S.l.]: ACM, 2004.

JAN, S.; LI, M.; AL-SULTANY, G.; AL-RAWESHIDY, H. File annotation and sharing on low-end mobile devices. In: IEEE. *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*. [S.l.], 2010. v. 6, p. 2973–2977.

KUROSE, J. F. *Computer networking: a top-down approach featuring the Internet*. [S.l.]: Pearson Education India, 2005.

NGUYEN, H.; GIORDANO, S. Routing in opportunistic networks. *International Journal of Ambient Computing and Intelligence (IJACI)*, IGI Global, v. 1, n. 3, p. 19–38, 2009.

NORDSTRÖM, E.; GUNNINGBERG, P.; ROHNER, C. Haggle: Relevance-aware content sharing for mobile, devices using search. *URL http://haggle. googlecode. com*, 2012.

PELUSI, L.; PASSARELLA, A.; CONTI, M. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, IEEE, v. 44, n. 11, p. 134–141, 2006.

PFüTZENREUTER, E. *Piconets*: site. Acessado em 02/07/2014. Disponível em: <https://epx.com.br/artigos/bluetooth3.php>.

RIPEANU, M. Peer-to-peer architecture case study: Gnutella network. In: IEEE. *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*. [S.l.], 2001. p. 99–100.

SUNDRAMOORTHY, V.; HARTEL, P.; SCHOLTEN, H.; DARGIE, W. A taxonomy of service discovery systems. *Context-Aware Computing and Self-Managing Systems*, Chapman & Hall/CRC, v. 3, p. 43, 2009.

TRIFUNOVIC, S.; DISTL, B.; SCHATZMANN, D.; LEGENDRE, F. Wifi-opp: Ad-hoc-less opportunistic networking. In: ACM. *Proceedings of the 6th ACM workshop on Challenged networks*. [S.l.], 2011. p. 37–42.

WAINER, J. Métodos de pesquisa quantitativa e qualitativa para a ciência da computação. *Atualização em Informática. Org: Tomasz Kowaltowski; Karin Breitman. Rio de Janeiro: Ed. PUC-Rio*, 2007.

WANG, H.; LIU, S.; CHIA, L.-T. Does ontology help in image retrieval?: a comparison between keyword, text ontology and multi-modality ontology approaches. In: ACM. *Proceedings of the 14th annual ACM international conference on Multimedia*. [S.l.], 2006. p. 109–112.

WIFI Alliance: site. Acessado em 02/07/2014. Disponível em: <http://www.wi-fi.org/discover-wi-fi/wi-fi-direct>.

ZHANG, Z. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *Communications Surveys & Tutorials, IEEE*, IEEE, v. 8, n. 1, p. 24–37, 2006.