UNIVERSIDADE FEDERAL DE UBERLÃĆNDIA

Patrick Godinho

PheroCast App

Uberlândia, Brasil

2014

UNIVERSIDADE FEDERAL DE UBERLÃĆNDIA

Patrick Godinho

PheroCast App

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Lásaro Camargos

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2014

Patrick Godinho

PheroCast App

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 24 de novembro de 2012:

Lásaro Camargos
Orientador

Faina

Professor

Uberlândia, Brasil
2014

Dedico a meu pai Paulo Sergio de Jesus Oliveira que sempre foi um exemplo de perseverança na minha vida!

Agradecimentos

Agradeço a Deus, minha esposa e ao meu orientador pelo apoio e dedicação para que eu conseguisse chegar ao final deste trabalho.

“Alguma citação que ache conveniente? Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

”

Resumo

o resumo deve ressaltar o teste objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. teste teste

Palavras-chave: Manet, Android, Redes Oportunísticas

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Fig.	Area of the i^{th} component
456	Isto é um número
123	Isto é outro número
Zézão	este é o meu nome

Sumário

1 Introdução

1.1 Contextualização

O paradigma dos multiplos saltos (*multihop*) das redes mobile MANET (Mobile Ad hoc Networking) surgiu no campo da civilização nos anos 90 com a disponibilização das tecnologias wireless de prateleira capazes de fornecer conexões diretas entre os usuários de dispositivos, como por exemplo o Bluetooth(IEEE 802.15.1) para redes pessoais, e a 802.11 standards family para redes de alta velocidade. Especificamente esses padrões de rede sem fio permitiam comunicação direta entre os dispositivos de rede, na faixa de transmissão de suas interfaces sem fio, tornando a rede "single-hop ad hoc"uma realidade, isto é, "WLAN/WPAN infraestructureless"onde dispositivos comunicam sem a necessidade de nenhuma infraestrutura de rede.

Desde que surgiu, a MANET tem sido visto como um dos mais inovadores e desafiantes paradigmas de redes sem fio, e prometia tornar-se uma das principais tecnologias, cada vez mais presente no cotidiano de todos. As potencialidades desse paradigma de rede fez a rede ad-hoc uma opção atrativa para a construção das redes sem fio 4G, e portanto a MANET imediatamente ganhou impulso, e isto produziu enormes esforços de pesquisa na comunidade de redes móveis.

Ao passar dos anos a MANET sofreu com o pouco interesse e exploração das indústrias e entre os usuários[8]. Tal fato pode ser explicado por algumas razões tais como a alta complexidade da implementação, integração e experimentação; a falta de credibilidade nas simulações, e a baixa motivação socio-econômica para com essa tecnologia.

Impulssionados pelas lições aprendidas com a MANET, surgiram algumas Redes Móveis com a promessa de evitar os erros cometidos anteriormente seguindo uma abordagem de desenvolvimento mais pragmática, desenvolvendo orientado a aplicações, com redução de complexidade, com uma abordagem de pesquisa voltada ao cenário da aplicação, utilizando o modelo de simulação realista, a fim de basear o desenvolvimento do protocolo em estudos de simulações confiáveis e acreditáveis; e O desenvolver os testes de mesa reais com a participação de usuários, nos estágios iniciais do projeto desses novos paradigmas, a fim de colocar esses usuários envolvidos com o projeto da rede e testes de experimentação.(**??**)

Dentre esses paradimas de rede que surgiram destacaremos a Rede Oportunística a qual não considera a mobilidade dos nós um problema, e sim uma oportunidade de ser explorada,ou seja, nesse paradigma a mobilidade dos nós cria oportunidades de contato entre os mesmos, sendo assim, entender e modelar as propriedades da mobilidade humana,

bem como o tempo em que dois dispositivos ficam conectados, ou o tempo para que crie outra comunicação, é fundamental para caracterizar as restrições das comunicações oportunísitcas. asd Outra rede que terá destaque na fundamentação teórica deste trabalho são as Redes Veículares criadas para veículos que comunicam entre si explorando tecnologias sem fio. A VANET pode suportar uma plenitude de aplicações como de segurança veicular, propagação de tráfego, compartilhamento de entretenimento e dentre outras.

Estudar os rastros da movimentação humana, é o ponto de partida para entender as propriedades da mobilidade, com o objetivo de fornecer as características temporais da mobilidade dos dispositivos/usuários. Experiências que são também usadas para derivar modelos de mobilidade, que são a base para simulações de rede oportunistas realistas. Além de estudos que tentam acoplar os resultados com os aspectos sociais dos donos dos dispositivos. (**??**)

Esse trabalho tem como objetivo desenvolver uma aplicativo Android para a coleta de dados sobre a mobilidade de usuários de smartphones, o qual coletará pontos de redes sem fio que seus dispositivos encontrarem durante seu dia-a-dia, ou seja, deixará registrado que passou em determinado ponto de rede.Após a coleta, armazenará nas nuvens para uma possível análise realista da mobilidade do usuário.

O aplicativo foi nomeado PheroCast App, por ser um trabalho impulssionado tambem para a cooperação para a pesquisa do PheroCast, algoritmos para prever a posição futura de nós móveis em MANET, permitindo que as infra-estruturas se adaptem de forma proactiva. Algoritmos estes denominados PheroCast, nome dado pela inspiração ao fenômeno que acontece na comunidade de formigas, em que a troca de informações acontece através da deposição e detecção de feromônios no ambiente. (**??**)

Após coleta dos dados, iremos contribuir com a *Crawdad*, um projeto de pesquisa em redes sem fio, a qual demandam logs e dados capturados através de redes *wireless*, para o estudo de como usuários reais, aplicações e dispositivos usam redes reais sob condições reais. Nossos dados cooperarão para enteder problemas, possiveis soluções e avaliar novas aplicações e serviços.

2 Fundamentação Teórica

2.1 Redes Móveis

2.1.1 MANET

O paradigma dos multiplos saltos (*multihop*) das redes mobile MANET (Mobile Ad hoc Networking) surgiu no campo da civilização nos anos 90 com a disponibilização das tecnologias wireless de prateleira capazes de fornecer conexões diretas entre os usuários de dispositivos, como por exemplo o Bluetooth(IEEE 802.15.1) para redes pessoais, e a 802.11 standards family para redes de alta velocidade. Especificamente esses padrões de rede sem fio permitiam comunicação direta entre os dispositivos de rede, na faixa de transmissão de suas interfaces sem fio, tornando a rede "single-hop ad hoc"uma realidade, isto é, WLAN/WPAN infraestructureless"onde dispositivos comunicam sem a necessidade de nenhuma infraestrutura de rede. O paradigma multihop foi então idealizado para tornar possível a comunicação entre 2 nós de rede, sem a necessidade de desenvolver nenhuma infraestrutura de rede onipresente. Nos anos 90, temos assistido a utilização do paradigma "multihop"nas MANETs (Redes ad-hoc móveis) onde usuários próximos (fisicamente) comunicam diretamente entre si (explorando as interfaces de redes sem fio de seus dispositivos no modo ad-hoc) não apenas para compartilhar seus dados, mas tambem para retransmitir o trafico de outros pontos de rede que não podem comunicar diretamente, operando assim como os roteadores fazem na internet. Por essa razão. na MANET, dispositivos dos usuários cooperativamente podem fornecer serviço de internet, geralmente fornecida pela rede de infra-estrutura (por exemplo, roteadores, switches, servidores). Desde que surgiu, a MANET tem sido visto como um dos mais inovadores e desafiantes paradigmas de redes sem fio, e prometia tornar-se uma das principais tecnologias, cada vez mais presente no cotidiano de todos. As potencialidades desse paradigma de rede fez a rede ad-hoc uma opção atrativa para a construção das redes sem fio 4G, e portanto a MANET imediatamente ganhou impulso, e isto produziu enormes esforços de pesquisa na comunidade de redes móveis. O modelo INTERNET foi a central para a MANET Internet Engineering Task Force (IETF) grupo de trabalho, o qual, herdou os protocolos TCP/IP stack layering, assumindo a visão do MANET centrada em IP. see “Mobile Ad Hoc Networks (MANETs)” by J. P. Macker and M. S. Scott Corson in [1]. As pesquisas da comunidade MANET tem sido focadas no que chamamos de de "pure general purpose MANETs"(objetivo geral puro), onde o "puro"indica que não há infrastrutura para implementar as funcionalidades de rede, e nenhuma autoridade é responsável para gerenciar e controlar a rede.

2.1.2 Redes Oportunísticas

Redes Oportunisticas é uma das mais interessantes evoluções do paradigma de rede baseada em saltos (multihop). De fato, enquanto MANET representa uma abordagem para esconder a mobilidade dos nós construindo um estável caminho fim a fim como na Internet, redes oportunísticas não consideram a mobilidade dos nós um problema, e sim uma oportunidade de ser explorada. Nas Redes Oportunísticas, a mobilidade dos nós cria oportunidades de contato entre os mesmos, as quais podem ser usadas por exemplo para conectar partes de redes que são de outra forma desconectada. Especificamente, de acordo com esse paradigma, nós podem carregar fisicamente dados em seu buffer enquanto eles se movimentam na rede, até que eles entrem em contato com um adequado ad-hoc. Assim, ao contrário da MANET, um nó mantem o armazenamento do dado enquanto não há um proximo bom salto "hop". Isso implica que, com o paradigma oportunistico, um dado pode ser entregue de uma origem para um destino mesmo se não existir um caminho fim-a-fim entre eles, explorando a sequência de gráficos gerados pela movimentação dos nós. [33,34]. Esse é um paradigma relativamente jovem, e a pesquisa da rede oportuniística ainda está ocorrendo. Entretanto, pode se argumentar que seu impacto ainda será provado. No entanto, dado que podemos considerar a rede ad-hoc veicular (VANET), uma dos mais avançados e concretos desenvolvimentos do paradigma das redes oportunísticas, podemos afirmar que esse paradigma já tem um papel significativo na área de redes de computadores. Além da VANET, outros cenários motivam a rede oportunística, discutido em [7]. Redes Oportunísticas, rede oportunísticas parecem muito adequadas para comunicações em ambientes difusos onde o ambiente está saturado por dispositivos (com tecnologias sem fio de curto alcance) que podem se auto-organizar em uma rede de interações locais entre os usuários. Nesses cenários, a rede é geralmente dividida em ilhas desconectadas, o que pode ser interligados explorando a mobilidade dos nós. Isso implica em uma mudança da comunicação baseada em pacotes, para comunicação baseada em mensagens, trazendo novas oportunidades para o desengo do protocolo das aplicações.

Rede Oportunistica se afasta da abordagem orientada a Internet utilizada na MANET, pois não prevê um caminho fim-a-fim da origem para o destino. Isso elimina o grande esforço de esconder a mobilidade dos nós, tal fato que causou a complexidade alta nas MANETs. Na Rede Oportunística, o paradigma "multihop"se torna mais eficiente, pelo fato de não necessitar manter caminhos e tabelas de rotas), e a mobilidade dos humanos torna uma oportunidade de comunicação.

Três principais direções tem caracterizado a pesqusia em rede oportunística, modelos de mobilidade, protocolos de roteamento e disseminação de dados.

Entender e modelar as propriedades da mobilidade humana, é fundamental para caracterizar as restrições das comunicações oportunísitcas, e desenhar as eficientes e práticas estratégias de compartilhamento[35,36]. Os principais elementos que caracterizam a

mobilidade humana é o "contact time", que se dá pela duração entre dois dispositivos; e o "inter-contact time"(ICT) que se dá pela duração entre dois contatos consecutivos entre dispositivos. Em particular, a caracterização da ICT tem gerado um excelente debate na comunidade científica as quais diferentes grupos de pesquisa têm reportado diferentes resultados variando entre várias funções.

2.1.3 VANET

A Rede Veicular AdHoc (VANET) é uma rede "multihop"adhoc criada para veículos que comunicam entre si explorando tecnologias sem fio (tipicamente) pertencentes a família 802.11. Essa é uma especialização do paradigma "multihop"adhoc bem mtivado pelo valor sócio-econômico dos avançados Sistemas de Transporte Inteligente (ITS) destinados a reduzir os congestionamentos, o número de acidentes, dentre outros. De fato, VANET pode suportar uma plenitude de aplicações incluindo as que envolvem segurança no tráfego (alertas de obstáculos na estrada, disseminação de mensagens de segurança), informações de tráfego e serviços de entretenimento/informações(jogos, streaming de multimedia, dentre outros). Um outro exemplo é um carro envolvido em um acidente pode explorar a possibilidade de se comunicar diretamente com outros veículos para informar os veículos próximos da situação perigosa.[44]

Sistemas ITS avançados necessitam tanto de comunicações vehicle-to-roadside (V2R) tanto como as vehicle-to-vehicle (V2V). Nas comunicações V2R um veículo tipicamente explora tecnologias sem fio baseada em infra-estrutura, como redes de celular, WiMAX e Wi-Fi, para comunicar com um ponto de estação ou acesso da estrada. No entanto, as unidades da estrada não são densas o suficiente para garantir a cobertura de rede exigido pelas aplicações e, portanto, são adotadas as comunicações V2V para estender a conectividade de rede/cobertura e garantir melhor desempenho da rede.

Comunicações V2V são baseadas no paradigma de rede multihop ad hoc "puro"como no MANET. Especificamente, de acordo com esse paradigma, os veículos na estrada se auto organizam dinamicamente na VANET explorando suas interfaces de comunicação sem fio. No entanto, o alto nível de mobilidade e a possibilidade do cenário de escassez de rede causadas pela baixa intensidade do tráfico, torna ineficiente o paradigma de comunicação utilizado na MANET "store-and-forward"utilizado na MANET, e força para a adoção do mais flexível, pragmático e robusto "store-carry-and-forward"paradigma, adotado pelas redes oportunísticas.

Além disso, sempre que possível as comunicações V2V exploram as comunicações V2R para se tornarem mais robustas e reduzindo algumas fragilidades e vulnerabilidades de uma comunicação "infrastructureless"pura.

O Alto valor socio-econômico das aplicações veiculares impulssionaram as organi-

zações internacionais de normalização para elaborar especificações técnicas a serem adotadas pela indústria de veículos. Entre eles, vale a pena lembrar a família padrão IEEE 1609 para acesso wireless em Ambiente Veicular(WAVE), que foi desenvolvido com base no padrão IEEE 802.11p. Normalmente, o consumo de energia não é um problema para esta rede como baterias para veículos são continuamente recarregado.

Comunicações V2R exploram tecnologias bem estabelecidas (Wi-Fi, WiMAX, ZigBee, etc) que operam baseados em infra-estrutura. Sua adoção em redes veiculares requer uma análise cuidadosa de seu desempenho ao operar em ambientes altamente dinâmicos, onde o tempo de conexão entre o veículo e a unidade na estrada pode ser curto e/ou vários veículos estão conectados à mesma unidade de beira de estrada (questão escalabilidade).

O campo de pesquisa em V2V herdaram resultados obtidos em MANET relacionados aos protocolos de roteamento/compartilhamento multihop ad hoc, os quais tem sido melhorado e modificado para adaptar-los as peculiaridades da área veicular. A tarefa de roteamento é um desafio na VANET, devida à alta mobilidade dos veículos que são conectados entre si intermitentemente. Entretanto, em VANET, a mobilidade dos pontos de redes (veiculos) são restritos pelas caracteristicas da estrada, e pelos outros veiculos que movimentam ao longo da estrada. Uma atenção especial está sendo reservada para o desenvolvimento dos otimizados "one-to-all"protocolos de roteamento assim como várias aplicações desenvolvidas para VANET usam comunicações broadcasting (geocasting). Broadcast ou geocast são tambem os serviços basicos de comunicação para disseminação de conteudo na VANET. Devida as alternadas condições de conectividade, o paradigma oportunístico aplicado a rede veicular tem gerado recentemente uma grande quantidade de literaturas, principalmente em protocolos de roteamento e disseminação de dados nas redes veiculares.

Tanto simulação quanto experimentos tem papéis importantes na formulação e avaliação das soluções desenvolvidas para redes veiculares. Nas simulações das redes veiculares, bastante atenção tem sido dedicado para desenvolver modelos realistas de estradas e de mobilidade dos veículos através do estudo e aprofundamento de literaturas desenvolvidas na área de sistemas de transporte, por exemplo modelos de como os carros se movem ao longo de uma estrada, tendo em conta a sua velocidade, a distância entre eles, sinais de trânsito, o traçado viário, etc

Um aspecto importante para se considerar está relacionado à simulação dos canais sem fio entre veículos e de/para unidades na estrada levando em conta a forma como o sinal de rádio propaga neste ambiente.

2.2 Desenvolvimento para Smartphones

Com o grande progresso da produção e comercialização de smartphones e tablets (e.g., Applet Ipad, Apple Iphone, Galaxy S4), fica evidente a quantidade de pessoas que foram alcançadas pelos dispositivos móeis, em especial pelos smartphones. Nota-se que existe, de fato, uma grande migração da computação tradicional, baseada em computadores pessoais, para uma nova era, muitas vezes chamadas de computação ubíqua.

A idéia desse novo paradigma, em geral, é expor um novo mundo, onde a computação está totalmente inserida e acoplada no cotidiano da vida das pessoas, principalmente a partir do uso de dispositivos que comunicam-se de forma transparente, possibilitando a troca de informações em qualquer momento e em qualquer lugar.

Levando isto em conta, é notável e fundamental, o desenvolvimento de novos sistemas operacionais e aplicações que atendam essa área em um crescente avanço e evolução. Tais aplicações que se diferenciam dos presentes na computação tradicional (i.e.. aplicações desktop), como por exemplo a substituição dos meios de entrada de dados, que se dava por teclado e mouse, por algo mais intuitivo como a utilização de telas sensíveis ao toque.

O mercado corporativo também está crescendo muito, e diversas empresas estão buscando incorporar aplicações móveis a seu dia-a-dia para agilizar seus negócios e integrar as aplicações móveis com seus sistemas de back-end. Empresas obviamente visam lucro e mais lucro, e aos celulares e smartphones podem ocupar um importante espaço em um mundo onde a palavra "mobilidade"está cada vez mais conhecida.[unijui]

Desta forma aplicações que executam em um celular podem estar literalmente conectadas e online, sincronizando informações diretamente de um servidor confiável da empresa. Hoje em dia diversos bancos oferecem serviços aos seus usuários, onde é possível pagar suas contas e visualizar o extrato de sua conta corrente diretamente de um celular.

As empresas e os desenvolvedores buscam uma plataforma moderna e ágil para o desenvolvimento de aplicações para auxiliar em seus negócios e lucros. Já os usuários comuns buscam um celular ocm um visual elegante e moderno, de fácil navegaão e uma infinidade de recursos.

Para acompanhar essa evolução da tecnologia e satisfazer os usuários, os fabricantes e operadoras de celulares, as empresas e os desenvolvedores, existe uma grande corrida estrelada pelas maiores empresas do mundo em tecnologia móvel para competir por esse nicho do mercado.(**??**)

2.2.1 Android

O Android é uma plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional Linux, uma interface visual rica, diversas aplicações já instaladas e ainda um ambiente de desenvolvimento bastante poderoso, inovador e flexível. Outro ponto importante é a possibilidade de utilizar a linguagem Java para desenvolver aplicações usufruindo de todos os recursos que a poderosa linguagem fornece. Entretanto possui suportepara o desenvolvimento de aplicações nativas, escritas em C e C++, através do android NDK. (**??**)

A primeira plataforma para aplicações móveis completamente livre e de código aberto (open-source) foi o Android, o que representa uma grande vantagem para sua evolução, uma vez que diversos programadores do mundo poderão contribuir para melhorar a plataforma. Tais características, cooperaram para a escolha pelo Android a nossa plataforma de desenvolvimento do aplicativo deste trabalho.

Com a variedade de recursos do Android, seria fácil confundi-lo com um sistema operacional desktop. O Android é um ambiente em camadas baseado em kernel Linux e que inclui funções ricas. O subsistema da UI inclui:

- Janelas
- Visualizações
- Widgets para a exibição de elementos comuns como caixas de edição, listas e listas suspensas

O Android inclui um navegador incorporável baseado em WebKit, o mesmo mecanismo navegador de software livre equipando o navegador Mobile Safari do iPhone.

O Android ostenta uma rica lista de opções de conectividade, incluindo WiFi, Bluetooth e dados wireless através de uma conexão celular (por exemplo, GPRS, EDGE e 3G). Uma técnica popular em aplicativos Android é estabelecer um link com o Google Maps para exibir um endereço diretamente em um aplicativo. O suporte para serviços baseados em locais (como GPS) e acelerômetros também está disponível na pilha de software Android, embora nem todos os dispositivos Android sejam equipados com o hardware necessário. Existe também suporte para câmera.

Historicamente, duas áreas onde aplicações móveis lutaram para acompanhar suas contrapartes de desktop são gráfico/mídia e métodos de armazenamento de dados. O Android aborda o desafio dos gráficos com suporte integrado para gráficos em 2-D e 3-D, incluindo a biblioteca OpenGL. O peso do armazenamento de dados é amenizado porque a plataforma Android inclui o banco de dados SQLite de software livre popular. A Figura 1 mostra uma visualização simplificada das camadas do software Android.

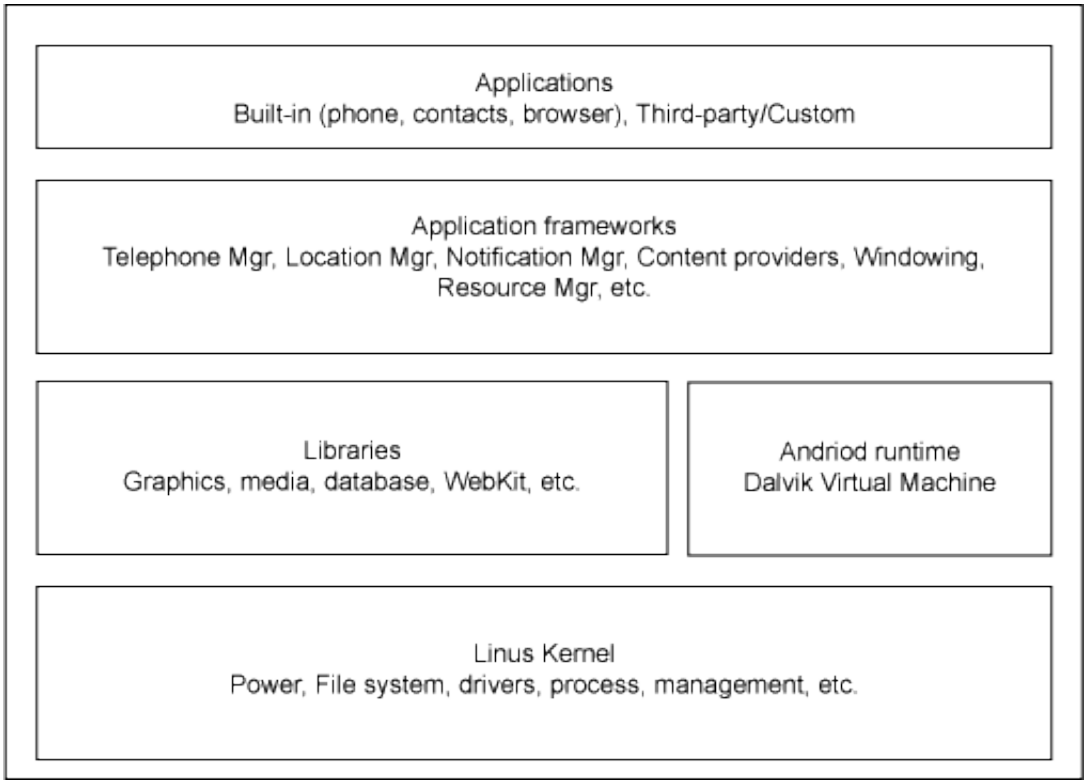

Figura 1 – Camadas do Software Android

Conforme mencionado, o Android é executado sobre um kernel Linux. Os aplicativos Android são gravados na linguagem de programação Java e são executados em uma máquina virtual (VM). É importante observar que a VM não é uma JVM, como você pode esperar, mas é uma Dalvik Virtual Machine, uma tecnologia de software livre. Cada aplicativo Android é executado em uma instância da Dalvik VM, que, por sua vez, reside em um processo gerenciado por kernel Linux, conforme mostrado abaixo.

Um aplicativo Android consiste em uma ou mais das classificações a seguir:

Atividades Um aplicativo que possui uma UI visível é implementado com uma atividade. Quando um usuário seleciona um aplicativo da tela inicial ou de um ativador de aplicativo, uma atividade é iniciada.

Serviços Um serviço deve ser utilizado para qualquer aplicativo que precise persistir por um longo período de tempo, como um monitor de rede ou um aplicativo de verificação de atualização.

Provedores de conteúdo Você pode pensar em provedores de conteúdo como um servidor de banco de dados. O trabalho de um provedor de conteúdo é gerenciar o acesso aos dados que persistem, como um banco de dados SQLite. Se seu aplicativo for muito simples, você não precisa necessariamente criar um provedor de conteúdo. Se estiver construindo um aplicativo maior, ou um que disponibilize dados para várias atividades ou aplicativos, um provedor de conteúdo será o meio de você acessar seus dados.

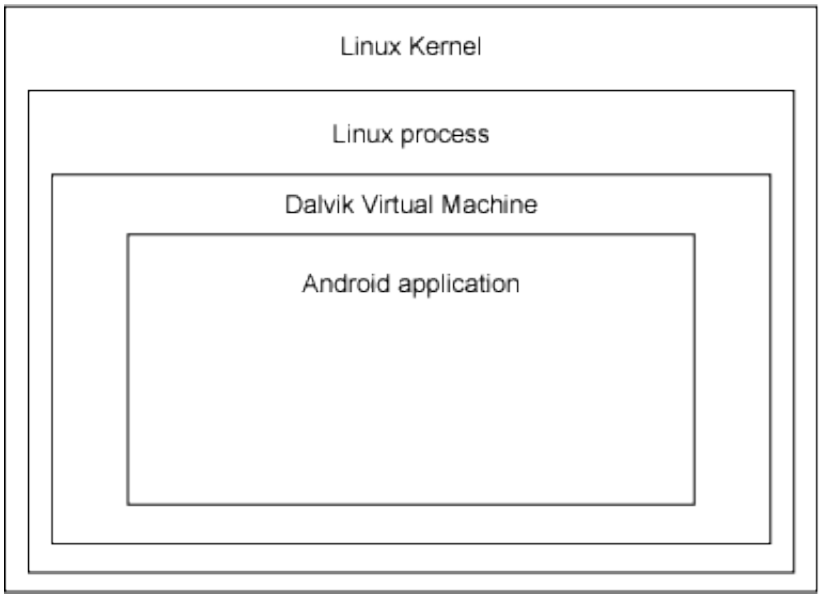

Figura 2 – Dalvik VM

Receptores de transmissão Um aplicativo Android pode ser ativado para processar um elemento de dados ou para responder a um evento, como o recebimento de uma mensagem de texto. Um aplicativo Android, junto com um arquivo chamado AndroidManifest.xml, é implementado em um dispositivo. O AndroidManifest.xml contém as informações de configuração necessárias para você instalá-lo corretamente no dispositivo.

Ele inclui os nomes de classes necessários e os tipos de eventos que o aplicativo está pronto para processar, além das permissões necessárias que o aplicativo precisa para execução. Por exemplo, se um aplicativo exigir acesso à rede — para fazer o download de um arquivo, por exemplo — essa permissão deve ser declarada explicitamente no arquivo de manifesto. Muitos aplicativos podem ter essa permissão específica ativada. Tal segurança declarativa ajuda a reduzir a probabilidade de um aplicativo perigoso causar danos em seu dispositivo.(**??**)

3 Desenvolvimento

Este trabalho foi desenvolvido na linguagem Java orientado a objetos , utilizando o Android SDK integrado com a IDE Eclipse.

Este trabalho se divide em três partes: 1- Execução periódica; 2- BroadCastReceiver - Coleta dos dados;3- Armazenamento na nuvem

3.1 Execução periódica

Para coletar os dados, o aplicativo acorda a cada um minuto e uma thread é executada, a qual verifica se o serviço de WIFI está ligado no aparelho. Se sim, a thread registra um *BroadCast Receiver* responsável por receber a lista de pontos de rede que estão ao alcance do dispositivo. Após receber a lista, é cancelado o registro do *BroadCast Receiver*. Caso o serviço de Wifi do celular esteja desligado, é feito o mesmo procedimento descrito acima, mas ligando o Wifi no começo da operação, e desligando no final, assim como está desenhado o diagrama abaixo.

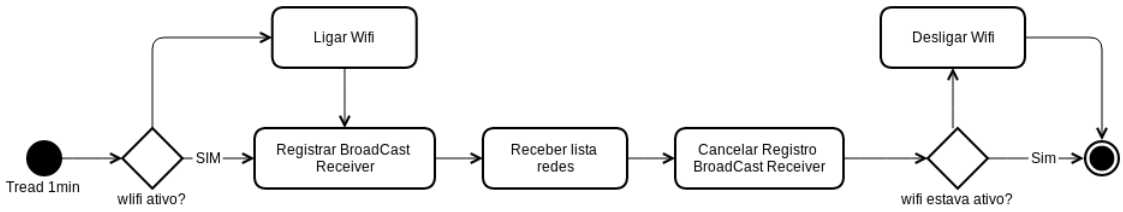

Figura 3 – Diagrama de estado de execução da thread

3.2 BroadCastReceiver - Coleta dos dados

No momento em que a lista de pontos de rede é recebida pelo BroadCastReceiver, para cada registro da lista é criado um objeto da classe NetworkPoint a qual possui os atributos da rede. Esses atributos são definidos pela classe nativa do Android *ScanResult*, sendo eles:

- BSSID (MAC address)
- SSID (Nome da rede)
- capabilities (Características gerais)

- frequency (Frequência)
- level (Nível de distância)

Além desses atributos, também é guardado a hora em que o dispositivo passou pela rede.

Após criar o objeto, o mesmo é gravado no banco de dados local através das funções da classe NetworkPointDAO.

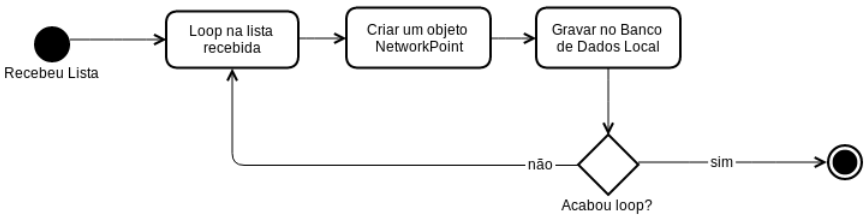

Figura 4 – Diagrama de estado de execução do BroadCast Receiver

3.3 Descarregando na nuvem

No aplicativo, registramos um filtro a fim de sempre que o dispositivo conectar-se a internet pela tecnologia Wifi, um método que escrevemos para descarregar os dados armazenados na nuvem seja executado. O armazenamento na nuvem foi feito através do mecanismo de formulário da plataforma Google Docs, onde fazemos requisições HTTP, como se estivéssemos respondendo ao formulário. Assim as respostas do mesmo são guardadas em uma planilha, a qual estão armazenados os campos do objeto mais o identificador do usuário, que no caso escolhemos o e-mail registrado no aparelho.

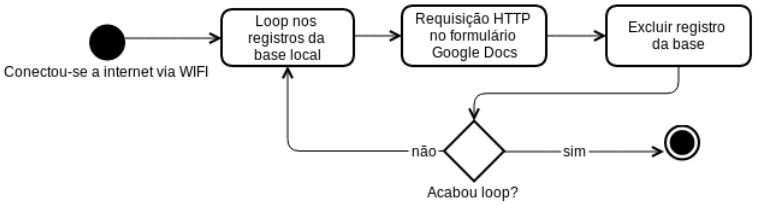

Figura 5 – Diagrama de estado de execução do armazenamento na nuvem

4 Resultados

Conclusão e Trabalhos Futuros

No decorrer do desenvolvimento deste trabalho, surgiram algumas idéias que foram consideradas como trabalhos futuros serem complementados. Por exemplo, implementar a anonimização dos dados, ou seja, para que cada usuário que cooperou com a coleta do log, não seja identificado. Tal requisito poderia ser implementado com o uso da teconologia GUID , a qual cria um identificador anônimo para cada dispositivo.

Concluímos no desenvolvimento do nosso trabalho que a escolha pela plataforma do Google Docs, apesar de ser uma maneira prática de armazenamento de dados, apresenta problemas de escabilidade dependendo da quantidade de dados a serem armazenados, pelo fato da exibição dos dados se dar através de planilhas. Uma sugestão para trabalhos futuros seria a implementação de um serviço próprio para captação e armazenamento de dados, como por exemplo aplicações WEB otimizadas para o serviço de armazenamento.

Para a expansão da utilização desse coletor de dados, o qual é o objetivo central do nosso trabalho, um ótimo trabalho futuro seria implementar o aplicativo para outras plataformas, tais como IOS, Windows Phone, BlackBerry, dentre outros.

Referências

Google Android. Novatec, 2013. Citado 2 vezes nas páginas 17 e 18.

Nenhuma citação no texto.

Frank Ableson. Introdução ao desenvolvimento do android, 2009. Citado na página 20.

P.R. Coelho, E. Fynn, L.F. Faina, R. Pasquini, and L. Camargos. Node position forecast in manet with pherocast. In *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, pages 73–80, May 2014. Citado na página 12.

M. Conti and S. Giordano. Mobile ad hoc networking: milestones, challenges, and new research directions. *Communications Magazine, IEEE*, 52(1):85–96, January 2014. Citado na página 11.

Anexos

ANEXO A – Código fonte

A.1 MainActivity.java

```
1  package ufu.tcc.patrick.pherocast;
2
3  import android.annotation.SuppressLint;
4  import android.content.BroadcastReceiver;
5  import android.content.Context;
6  import android.content.Intent;
7  import android.content.IntentFilter;
8  import android.net.wifi.ScanResult;
9  import android.net.wifi.WifiManager;
10 import android.os.Bundle;
11 import android.os.Handler;
12 import android.support.v7.app.ActionBarActivity;
13 import android.widget.TextView;
14 import android.widget.Toast;
15
16 import java.io.PrintStream;
17 import java.text.SimpleDateFormat;
18 import java.util.ArrayList;
19 import java.util.Date;
20 import java.util.List;
21 import java.util.Timer;
22 import java.util.TimerTask;
23
24 public class MainActivity extends ActionBarActivity {
25     final Handler handler = new Handler();
26     boolean scanInitiated;
27     TimerTask scanTask;
28     Timer t = new Timer();
29     TextView text;
30     WifiManager wifi;
31     boolean wifiOn;
32     WifiScanReceiver wifiReciever;
33     String[] wifis;
34     List<ScanResult> localList = new ArrayList<ScanResult>();
35
```

```
36      private String pegarHoraAtual() {
37          return new SimpleDateFormat("dd/MM/yyyy hh:mm:ss").format
                (new Date());
38      }
39
40      public void getWifiState() {
41          System.out.println("passou aqui " + wifi.isWifiEnabled())
                ;
42          if (!wifi.isWifiEnabled()) {
43              wifiOn = false;
44              Toast.makeText(getApplicationContext(),
45                      "Wifi desativado, estamos ativando...", 1).
                           show();
46              wifi.setWifiEnabled(true);
47              return;
48          }
49          wifiOn = true;
50      }
51
52      public void onBackPressed() {
53          moveTaskToBack(true);
54      }
55
56      protected void onCreate(Bundle paramBundle) {
57          super.onCreate(paramBundle);
58          setContentView(2130903064);
59          wifi = ((WifiManager) getSystemService("wifi"));
60          wifiReciever = new WifiScanReceiver();
61          scanTask = new TimerTask() {
62              public void run() {
63                  handler.post(new Runnable() {
64                      public void run() {
65                          getWifiState();
66                          registerReceiver(wifiReciever, new
                                IntentFilter(
67                                      "android.net.wifi.SCAN_RESULTS"))
                                        ;
68                          scanInitiated = true;
69                          wifi.startScan();
70                      }
71                  });
72              }
```

```
73          };
74          t.schedule(scanTask, 300L, 60000L);
75      }
76
77      class WifiScanReceiver extends BroadcastReceiver {
78          WifiScanReceiver() {
79          }
80
81          @SuppressLint({ "UseValueOf", "NewApi" })
82          public void onReceive(Context paramContext, Intent
                paramIntent) {
83
84              NetworkPointDAO localNetworkPointDAO =
                    NetworkPointDAO
85                          .getInstance(getBaseContext());
86              if (scanInitiated) {
87                  localList = wifi.getScanResults();
88                  wifis = new String[localList.size()];
89
90              }
91              for (int i = 0;; i++) {
92                  if (i >= localList.size()) {
93                      scanInitiated = false;
94                      System.out.println(wifiOn);
95                      if (!wifiOn)
96                          wifi.setWifiEnabled(false);
97                      unregisterReceiver(this);
98                      return;
99                  }
100                 wifis[i] = (((ScanResult) localList.get(i)).SSID
                        + ", "
101                             + ((ScanResult) localList.get(i)).
                                frequency + ", "
102                             + ((ScanResult) localList.get(i)).level +
                                ", " + ((ScanResult) localList
103                             .get(i)).BSSID);
104                 localNetworkPointDAO
105                         .salvar(new NetworkPoint(
106                                 ((ScanResult) localList.get(i)).
                                    BSSID,
107                                 ((ScanResult) localList.get(i)).
                                    SSID,
```

```
108                                     ((ScanResult) localList.get(i)).
                                           capabilities,
109                                     ((ScanResult) localList.get(i)).
                                           frequency,
110                                     ((ScanResult) localList.get(i)).
                                           level,
111                                     pegarHoraAtual()));
112                 }
113             }
114         }
115 }
```

A.2 HttpRequest.java

```
1  package ufu.tcc.patrick.pherocast;
2
3
4  import java.io.IOException;
5  import java.io.InputStream;
6  import java.io.UnsupportedEncodingException;
7  import java.net.HttpURLConnection;
8  import java.net.URL;
9  import java.net.URLConnection;
10 import java.net.URLEncoder;
11
12 import org.apache.http.HttpResponse;
13 import org.apache.http.client.methods.HttpGet;
14 import org.apache.http.client.methods.HttpPost;
15 import org.apache.http.client.params.ClientPNames;
16 import org.apache.http.client.params.CookiePolicy;
17 import org.apache.http.entity.StringEntity;
18 import org.apache.http.impl.client.DefaultHttpClient;
19 import org.apache.http.params.BasicHttpParams;
20 import org.apache.http.params.HttpConnectionParams;
21 import org.apache.http.params.HttpParams;
22 import org.apache.http.protocol.BasicHttpContext;
23 import org.apache.http.protocol.HttpContext;
24 import org.apache.http.util.EntityUtils;
25 import org.json.JSONObject;
26
27 import android.util.Log;
28
```

```
29 /*
30  * This helper class was created by StackOverflow user: MattC
       http://stackoverflow.com/users/21126/mattc
31  * IT was posted as an Answer to this question: http://
       stackoverflow.com/questions/2253061/secure-http-post-in-
       android
32  */
33
34 public class HttpRequest{
35
36     DefaultHttpClient httpClient;
37     HttpContext localContext;
38     private String ret;
39
40     HttpResponse response = null;
41     HttpPost httpPost = null;
42     HttpGet httpGet = null;
43
44     public HttpRequest(){
45         HttpParams myParams = new BasicHttpParams();
46
47         HttpConnectionParams.setConnectionTimeout(myParams,
               10000);
48         HttpConnectionParams.setSoTimeout(myParams, 10000);
49         httpClient = new DefaultHttpClient(myParams);
50         localContext = new BasicHttpContext();
51     }
52
53     public void clearCookies() {
54         httpClient.getCookieStore().clear();
55     }
56
57     public void abort() {
58         try {
59             if (httpClient != null) {
60                 System.out.println("Abort.");
61                 httpPost.abort();
62             }
63         } catch (Exception e) {
64             System.out.println("Your App Name Here" + e);
65         }
66     }
```

```
67
68      public String sendPost(String url, String data) {
69          return sendPost(url, data, null);
70      }
71
72      public String sendJSONPost(String url, JSONObject data) {
73          return sendPost(url, data.toString(), "application/json")
                ;
74      }
75
76      public String sendPost(String url, String data, String
           contentType) {
77          ret = null;
78
79          httpClient.getParams().setParameter(ClientPNames.
               COOKIE_POLICY, CookiePolicy.RFC_2109);
80
81          httpPost = new HttpPost(url);
82          response = null;
83
84          StringEntity tmp = null;
85
86          Log.d("Your App Name Here", "Setting httpPost headers");
87
88          httpPost.setHeader("User-Agent", "SET YOUR USER AGENT
               STRING HERE");
89          httpPost.setHeader("Accept", "text/html,application/xml,
               application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,
               image/png,*;q=0.5");
90
91          if (contentType != null) {
92              httpPost.setHeader("Content-Type", contentType);
93          } else {
94              httpPost.setHeader("Content-Type", "application/x-www
                   -form-urlencoded");
95          }
96
97          try {
98              tmp = new StringEntity(data,"UTF-8");
99          } catch (UnsupportedEncodingException e) {
100             Log.e("Your App Name Here", "HttpUtils :
                   UnsupportedEncodingException : "+e);
```

```
101         }
102 
103         httpPost.setEntity(tmp);
104 
105         Log.d("Your App Name Here", url + "?" + data);
106 
107         try {
108             response = httpClient.execute(httpPost,localContext);
109 
110             if (response != null) {
111                 ret = EntityUtils.toString(response.getEntity());
112             }
113         } catch (Exception e) {
114             Log.e("Your App Name Here", "HttpUtils: " + e);
115         }
116 
117         Log.d("Your App Name Here", "Returning value:" + ret);
118 
119         return ret;
120     }
121 
122     public String sendGet(String url) {
123         httpGet = new HttpGet(url);
124 
125         try {
126             response = httpClient.execute(httpGet);
127         } catch (Exception e) {
128             Log.e("Your App Name Here", e.getMessage());
129         }
130 
131         //int status = response.getStatusLine().getStatusCode();
132 
133         // we assume that the response body contains the error
               message
134         try {
135             ret = EntityUtils.toString(response.getEntity());
136         } catch (IOException e) {
137             Log.e("Your App Name Here", e.getMessage());
138         }
139 
140         return ret;
141     }
```

```
142
143     public InputStream getHttpStream(String urlString) throws
            IOException {
144         InputStream in = null;
145         int response = -1;
146
147         URL url = new URL(urlString);
148         URLConnection conn = url.openConnection();
149
150         if (!(conn instanceof HttpURLConnection))
151             throw new IOException("Not an HTTP connection");
152
153         try{
154             HttpURLConnection httpConn = (HttpURLConnection) conn
                    ;
155             httpConn.setAllowUserInteraction(false);
156             httpConn.setInstanceFollowRedirects(true);
157             httpConn.setRequestMethod("GET");
158             httpConn.connect();
159
160             response = httpConn.getResponseCode();
161
162             if (response == HttpURLConnection.HTTP_OK) {
163                 in = httpConn.getInputStream();
164             }
165         } catch (Exception e) {
166             throw new IOException("Error connecting");
167         } // end try-catch
168
169         return in;
170     }
171
172
173     public void postData() {
174         String fullUrl = "https://docs.google.com/forms/d/1
                AYvV0gFgB1hBuoRKnMsXy1LyF8-Ce8VAshAtho6Z08s/
                formResponse";
175         HttpRequest mReq = new HttpRequest();
176         String col1 = "Hello";
177         String col2 = "World";
178
179         String data = "entry.1680144410=" + URLEncoder.encode(
```

```
            col1) + "&" +
180                         "entry.1558298396=" + URLEncoder.encode(
                               col2);
181         String response = mReq.sendPost(fullUrl, data);
182         Log.i("DocsUpload", response);
183     }
184 }
```

A.3 NetworkChangeReceiver.java

```
1  package ufu.tcc.patrick.pherocast;
2
3  import android.content.BroadcastReceiver;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.net.ConnectivityManager;
7  import android.net.NetworkInfo;
8  import android.util.Log;
9  import java.io.PrintStream;
10 import java.net.URLEncoder;
11 import java.util.ArrayList;
12 import java.util.Iterator;
13
14 public class NetworkChangeReceiver extends BroadcastReceiver
15 {
16   public void enviarParaDocs(Context paramContext)
17   {
18     NetworkPointDAO localNetworkPointDAO = NetworkPointDAO.
          getInstance(paramContext);
19     ArrayList localArrayList = (ArrayList)localNetworkPointDAO.
          recuperarTodos();
20     HttpRequest localHttpRequest = new HttpRequest();
21     Iterator localIterator = localArrayList.iterator();
22     while (true)
23     {
24       if (!localIterator.hasNext())
25         return;
26       NetworkPoint localNetworkPoint = (NetworkPoint)
            localIterator.next();
27       String str1 = localNetworkPoint.getSsid();
28       String str2 = localNetworkPoint.getBssid();
29       String str3 = localNetworkPoint.getCapabilities();
```

```
30        String str4 = String.valueOf(localNetworkPoint.getFrequency
              ());
31        String str5 = String.valueOf(localNetworkPoint.getLevel());
32        String str6 = localNetworkPoint.getData();
33        String str7 = UserEmailFetcher.getEmail(paramContext);
34        localHttpRequest.sendPost("https://docs.google.com/forms/d
              /1G_dkyvwug--i_We7qAaA3QV-Xw_plTBJeKdElW22S4w/
              formResponse", "entry_2059700=" + URLEncoder.encode(str1)
               + "&" + "entry_1828317397=" + URLEncoder.encode(str2) +
              "&" + "entry_2146852893=" + URLEncoder.encode(str3) + "&"
               + "entry_312023197=" + URLEncoder.encode(str4) + "&" + "
              entry_644637792=" + URLEncoder.encode(str5) + "&" + "
              entry_604910793=" + URLEncoder.encode(str6) + "&" + "
              entry_612999935= " + URLEncoder.encode(str7));
35        localNetworkPointDAO.deletar(localNetworkPoint);
36      }
37    }
38
39    public void onReceive(final Context paramContext, Intent
          paramIntent)
40    {
41      ConnectivityManager localConnectivityManager = (
            ConnectivityManager)paramContext.getSystemService("
            connectivity");
42      NetworkInfo localNetworkInfo1 = localConnectivityManager.
            getNetworkInfo(1);
43      NetworkInfo localNetworkInfo2 = localConnectivityManager.
            getNetworkInfo(0);
44      if ((localNetworkInfo1.isAvailable()) || (localNetworkInfo2.
            isConnected()));
45      try
46      {
47        new Thread(new Runnable()
48        {
49          public void run()
50          {
51            enviarParaDocs(paramContext);
52          }
53        }).start();
54        return;
55      }
56      catch (Exception localException)
```

```
57      {
58        Log.d("Netowk Available ", localException.getMessage());
59      }
60    }
61  }
```

A.4 NetworkPoint.java

```
1   package ufu.tcc.patrick.pherocast;
2
3   public class NetworkPoint
4   {
5     private String bssid;
6     private String capabilities;
7     private String data;
8     private int frequency;
9     private int level;
10    private String ssid;
11    private long timestamp;
12
13    public NetworkPoint()
14    {
15    }
16
17    public NetworkPoint(String paramString1, String paramString2,
          String paramString3, int paramInt1, int paramInt2, String
          paramString4)
18    {
19      this.bssid = paramString1;
20      this.ssid = paramString2;
21      this.capabilities = paramString3;
22      this.frequency = paramInt1;
23      this.level = paramInt2;
24      this.data = paramString4;
25    }
26
27    public String getBssid()
28    {
29      return this.bssid;
30    }
31
32    public String getCapabilities()
```

```
33  {
34    return this.capabilities;
35  }
36
37  public String getData()
38  {
39    return this.data;
40  }
41
42  public int getFrequency()
43  {
44    return this.frequency;
45  }
46
47  public int getLevel()
48  {
49    return this.level;
50  }
51
52  public String getSsid()
53  {
54    return this.ssid;
55  }
56
57  public long getTimestamp()
58  {
59    return this.timestamp;
60  }
61
62  public void setBssid(String paramString)
63  {
64    this.bssid = paramString;
65  }
66
67  public void setCapabilities(String paramString)
68  {
69    this.capabilities = paramString;
70  }
71
72  public void setData(String paramString)
73  {
74    this.data = paramString;
```

```
75    }
76
77    public void setFrequency(int paramInt)
78    {
79      this.frequency = paramInt;
80    }
81
82    public void setLevel(int paramInt)
83    {
84      this.level = paramInt;
85    }
86
87    public void setSsid(String paramString)
88    {
89      this.ssid = paramString;
90    }
91
92    public void setTimestamp(long paramLong)
93    {
94      this.timestamp = paramLong;
95    }
96  }
```

A.5 NetworkPointDAO.java

```
1  package ufu.tcc.patrick.pherocast;
2
3  import android.content.ContentValues;
4  import android.content.Context;
5  import android.database.Cursor;
6  import android.database.sqlite.SQLiteDatabase;
7  import java.io.PrintStream;
8  import java.util.ArrayList;
9  import java.util.List;
10
11 public class NetworkPointDAO
12 {
13   public static final String COLUNA_BSSID = "bssid";
14   public static final String COLUNA_CAPABILITIES = "capabilities"
       ;
15   public static final String COLUNA_DATA = "data_adicao";
16   public static final String COLUNA_FREQUENCIA = "frequencia";
```

```
17  public static final String COLUNA_LEVEL = "level";
18  public static final String COLUNA_SSID = "ssid";
19  public static final String COLUNA_TIMESTAMP = "timestamp";
20  public static final String NOME_TABELA = "network_point";
21  public static final String SCRIPT_CRIACAO_TABELA_NETWORK_POINT
       = "CREATE TABLE network_point(bssid TEXT, ssid TEXT,
       capabilities TEXT, frequencia INTEGER, level INTEGER,
       timestamp LONG, data_adicao STRING )";
22  public static final String SCRIPT_DELECAO_TABELA = "DROP TABLE
       IF EXISTS network_point";
23  private static NetworkPointDAO instance;
24  private SQLiteDatabase dataBase = null;
25
26  private NetworkPointDAO(Context paramContext)
27  {
28    this.dataBase = PersistenceHelper.getInstance(paramContext).
         getWritableDatabase();
29  }
30
31  private List<NetworkPoint> construirNetworkPorCursor(Cursor
       paramCursor)
32  {
33    ArrayList localArrayList = new ArrayList();
34    if (paramCursor == null)
35      return localArrayList;
36    try
37    {
38      if (paramCursor.moveToFirst())
39      {
40        boolean bool;
41        do
42        {
43          int i = paramCursor.getColumnIndex("ssid");
44          int j = paramCursor.getColumnIndex("bssid");
45          int k = paramCursor.getColumnIndex("capabilities");
46          int m = paramCursor.getColumnIndex("frequencia");
47          int n = paramCursor.getColumnIndex("level");
48          paramCursor.getColumnIndex("timestamp");
49          int i1 = paramCursor.getColumnIndex("data_adicao");
50          String str1 = paramCursor.getString(i);
51          String str2 = paramCursor.getString(j);
52          String str3 = paramCursor.getString(k);
```

```
53            int i2 = paramCursor.getInt(n);
54            localArrayList.add(new NetworkPoint(str2, str1, str3,
                 paramCursor.getInt(m), i2, paramCursor.getString(i1))
                 );
55            bool = paramCursor.moveToNext();
56          }
57          while (bool);
58        }
59        return localArrayList;
60      }
61      finally
62      {
63        paramCursor.close();
64      }
65      //throw localObject;
66    }
67
68    private ContentValues gerarContentValues(NetworkPoint
         paramNetworkPoint)
69    {
70      ContentValues localContentValues = new ContentValues();
71      localContentValues.put("bssid", paramNetworkPoint.getBssid())
           ;
72      localContentValues.put("ssid", paramNetworkPoint.getSsid());
73      localContentValues.put("capabilities", paramNetworkPoint.
           getCapabilities());
74      localContentValues.put("frequencia", Integer.valueOf(
           paramNetworkPoint.getFrequency()));
75      localContentValues.put("level", Integer.valueOf(
           paramNetworkPoint.getLevel()));
76      localContentValues.put("data_adicao", paramNetworkPoint.
           getData());
77      return localContentValues;
78    }
79
80    public static NetworkPointDAO getInstance(Context paramContext)
81    {
82      if (instance == null)
83        instance = new NetworkPointDAO(paramContext);
84      return instance;
85    }
86
```

```
87  public void deletar(NetworkPoint paramNetworkPoint)
88  {
89    String[] arrayOfString = new String[1];
90    arrayOfString[0] = String.valueOf(paramNetworkPoint.getBssid
        ());
91    this.dataBase.delete("network_point", "bssid =  ?",
        arrayOfString);
92  }
93
94  public void editar(NetworkPoint paramNetworkPoint)
95  {
96    ContentValues localContentValues = gerarContentValues(
        paramNetworkPoint);
97    String[] arrayOfString = new String[1];
98    arrayOfString[0] = String.valueOf(paramNetworkPoint.getBssid
        ());
99    this.dataBase.update("network_point", localContentValues, "
        bssid = ?", arrayOfString);
100 }
101
102 public void fecharConexao()
103 {
104   if ((this.dataBase != null) && (this.dataBase.isOpen()))
105     this.dataBase.close();
106 }
107
108 public int getQuantidade()
109 {
110   return this.dataBase.rawQuery("select * from network_point",
        null).getCount();
111 }
112
113 public List<NetworkPoint> recuperarTodos()
114 {
115   return construirNetworkPorCursor(this.dataBase.rawQuery("
        SELECT * FROM network_point", null));
116 }
117
118 public void salvar(NetworkPoint paramNetworkPoint)
119 {
120   ContentValues localContentValues = gerarContentValues(
        paramNetworkPoint);
```

```
121     this.dataBase.insert("network_point", null,
            localContentValues);
122     System.out.println(paramNetworkPoint.getData() + "AUHUHAHUU")
            ;
123   }
124
125   public void truncarTabela()
126   {
127     this.dataBase.execSQL("DELETE FROM network_point;");
128   }
129 }
```

A.6 PersistenceHelper.java

```
1  package ufu.tcc.patrick.pherocast;
2
3  import android.content.Context;
4  import android.database.sqlite.SQLiteDatabase;
5  import android.database.sqlite.SQLiteOpenHelper;
6
7  public class PersistenceHelper extends SQLiteOpenHelper
8  {
9    public static final String NOME_BANCO = "wificollector";
10   public static final int VERSAO = 1;
11   private static PersistenceHelper instance;
12
13   private PersistenceHelper(Context paramContext)
14   {
15     super(paramContext, "wificollector", null, 1);
16   }
17
18   public static PersistenceHelper getInstance(Context
         paramContext)
19   {
20     if (instance == null)
21       instance = new PersistenceHelper(paramContext);
22     return instance;
23   }
24
25   public void onCreate(SQLiteDatabase paramSQLiteDatabase)
26   {
27     paramSQLiteDatabase.execSQL("CREATE TABLE network_point(bssid
```

```
        TEXT, ssid TEXT, capabilities TEXT, frequencia INTEGER,
        level INTEGER, timestamp LONG, data_adicao STRING )");
28    }
29
30    public void onUpgrade(SQLiteDatabase paramSQLiteDatabase, int
         paramInt1, int paramInt2)
31    {
32      paramSQLiteDatabase.execSQL("DROP TABLE IF EXISTS
           network_point");
33      onCreate(paramSQLiteDatabase);
34    }
35  }
```

A.7 UserEmailFetcher.java

```
1   package ufu.tcc.patrick.pherocast;
2
3   import android.accounts.Account;
4   import android.accounts.AccountManager;
5   import android.content.Context;
6   import android.util.Log;
7
8   public class UserEmailFetcher
9   {
10    private static Account getAccount(AccountManager
         paramAccountManager)
11    {
12      Account[] arrayOfAccount = paramAccountManager.
           getAccountsByType("com.google");
13      if (arrayOfAccount.length > 0)
14        return arrayOfAccount[0];
15      return null;
16    }
17
18    static String getEmail(Context paramContext)
19    {
20      Account localAccount = getAccount(AccountManager.get(
           paramContext));
21      if (localAccount == null)
22        return null;
23      Log.w("TESTE DE EMAIL", localAccount.name);
24      return localAccount.name;
```

```
25     }
26 }
```