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: Lasaro 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:

Lasaro Camargos
Orientador

Professor

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 Lasaro Camargos pelo apoio e dedicação para que eu conseguisse chegar ao final deste trabalho.

Resumo

Palavras-chave: Manet, Android, Redes Oportunísticas

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Contextualização

1.1.1 MANET

MANET é a abreviação de Mobile Ad-Hoc Networks, a qual se define pelas redes em que nós se movimentam livremente, comunicam entre si e entre o meio em que está, sem a necessidade de nenhuma infraestrutura de rede. Tais ações permitem que várias aplicações sejam desenvolvidas com objetivo por exemplo da troca de informações entre os nós ou até o compartilhamento de serviços, como por exemplo a funcionalidade que temos hoje em dia de rotear a internet do celular com dispositivos próximos.

Tambem definida por MANET pode ser conceituada como um sistema autonomo de servidores moveis (que tambem servem de roteadores), os quais unidos formam uma rede de comunicacao representada por um grafo.

Os nós em uma MANET se movem arbitrariamente, e essa é a grande característica da rede, mas também o principal ofensor na qualidade da comunicação disponibilizada nesse ambiente, pois a arquitetura entre eles deve sempre estar se adaptando para os novos meios em que estão inseridos.

Desde que surgiu, a MANET tem sido vista como uma das mais desafiantes abordagens de redes móveis tanto pela promessa de crescimento de dispositivos móveis, quanto pela sua complexidade, o que impulsionou a pesquisa por esse paradigma. Apartir das pesqusias intensas na MANET, surgiram outras redes móveis baseadas em sua proposta inicial. Como por exemplo a VANET que será introduzida na próxima seção.

1.1.2 VANET

1.1.2.1 Conceito

VANET são redes móveis MANET onde os nós são os veículos e até a estrada, onde há comunicação apenas entre os automóveis (IVC - Inter Vehicle Communication), ou entre os mesmos e a estrada (RVC - Road Vehicle Communication).

Assim como na MANET, a Vehicular Ad-Hoc Network também permite que várias aplicações com vários objetivos sejam desenvolvidos, bem como alertar obstáculos na estrada, broadcast de informações de segurança, compartilhar informações de tráfego ou até solicitar socorro em algum acidente.

As simualções e experimentos tem tido um papel importante no desenvolvimento

das soluções para redes veiculares. A atenção maior tem sido dada no desenvolvimento de modelos realistas de estrada e principalmente no estudo da mobilidade dos veículos. Por exemplo, modelos de como os carros se movem ao longo do trajeto, levando em consideração sua velocidade, sinais de transito, dentre outros fatores.

1.1.2.2 Problema

O grande desafio das Redes Ad-Hoc Veiculares é o roteamento entre os nos, devida a alta taxa de mobilidade dos veículos que são conectados entre si intermitentemente, dificultando a entrega das mensagens entre os ou estrada, pois frequentemente um nó muda seu endereço, ficando assim diferente de qual se identificou.

Alem do problema citado acima, existe um agravante relacionado a velocidade de conexão entre os , os quais se comunicam muito rápido, levando em consideração que os mesmos estejam em uma via rápida.

1.1.3 Predição de localização

1.1.3.1 Motivo

A predição da localidade de um nó, o qual pode ser um automóvel, uma pessoa com um dispositivo móvel, é algo bastante motivador a se desenvolver pelo fato de trazer otimizações da comunicação das redes MANET e suas derivadas, através da gestão proativa. (**??**)

Por exemplo, caso um nó mande uma requisição para outro e logo após mude de localização, o responsável pela resposta será proativo o bastante para saber em qual posição o destinatário estará, resolvendo assim o problema da mobilidade, o qual é o principal desafio da VANET.

A predição de localização também é alvo de escopo de várias aplicações com o objetivo por exemplo de prever e otimizar tráfego em cidades inteligentes, ou até prever potenciais clientes para um passeio ecológico.

1.1.3.2 PheroCast

1.1.3.2.1 Conceito

PheroCast é o nome que se deu à abordagem de previsão da localização futura de um nó em uma rede ad-hoc móvel. Foi nomeado assim pela semelhança com o fenomeno que acontece na colonia das formigas, o qual para alertar problemas, ou marcar o caminho, por onde a formiga passar, ela deixará um rastro, formando assim um caminho.

Nessa abordagem, o algoritmo de previsão é desenvolvido para ser processado nos próprios nós móveis. Assim, a cada intervalo de tempo definido, o nó registra um rastro

de onde está. Daí vem a semelhança com o "Pheronomium"deixado no rastro por onde a formiga passa.

Assim, com o histórico dos "traces"de um determinado nó na forma de um grafo, é possível determinar uma probabilidade de onde ele estará em determinado horário apartir de um certo lugar.

1.1.3.2.2 Resultados

Segundo (**??**) foi feita uma avaliação do algoritmo PheroCast usando os caminhos dos onibus de Seattle Metro Transit, que vale a pena ressaltar, não foram extraídos para a abordagem nem com o intervalo de tempo ideal. Foi descoberto que a abordagem obteve bons resultados em relação à predição das futuras posicões dos onibus. A avaliação obteve 77,8% de resultados positivos

1.1.3.2.3 Problema

O grande problema do PheroCast é a sua avaliação limitada, ou seja, é necessário, para uma abordagem com essas características, uma avaliação extensa com dados reais, que ssimulam de verdade a mobilidade humana.

Atualmente não existem "traces"reais, e sim módulos de mobilidade sintéticos, os quais não são realistas. Daí vem a proposta deste trabalho, o qual irá cooperar para avaliação confiável do PheroCast, coletando dados de mobilidade humana reais, através de seus dispositivos móveis.

1.2 Proposta

A proposta deste trabalho é desenvolver um sistema que rode em dispositivos móveis, para capturar "traces"reais de mobilidade humana. Para tal, o sistema irá coletar em um intervalo de tempo, todas as redes wireless, bem como suas informações, ao redor do dispositivo móvel do usuário e armazenar em um servidor para que os dados sejam compilados e utilizados para avaliação do PheroCast.

O objetivo é que o sistema seja utilizado por mais de um usuário para que assim tenhamos grande diversidade de histórico de traces.

2 Referencial Teórico

2.1 PheroCast

O PheroCast, conjunto de algoritmos para prever a localização de um nó em alguma rede móvel, foi desenvolvido por um grupo de pesquisa da Universidade Federal de Uberlândia, o The Distributed Systems and Networks Research Group, criado em 2013, constituído de doutores, mestres e alunos da Faculdade de Computação. citar grupo.

Nosso objetivo nessa seção não é de descrever o algoritmo bem como seus passos, cálculos e funções, e sim, explicar o conceito do algoritmo e seus principais componentes. Tais assuntos foram utilizados para o desenvolvimento deste trabalho.

Para entedermos a abordagem do Pherocast, que consiste na previsão da posição dos nós de uma rede móvel baseada no conceitos dos feromônios, vamos conceituar e exemplificar alguns componentes utilizados pelo algoritmo.

Como as formigas passam a vida em contato com o solo, em suas caminhadas, elas deixam um rastro de feromônio que pode ser seguida por ouras formigas. Trazendo para o nosso contexto, um nó em uma rede móvel pode sinalizar os pontos em que passou, e assim construir um caminho que percorreu de um ponto a outro. Podemos também afirmar com esse conceito que um nó está em uma localização X se X é o sinal mais perto de sua posição.

Cada marca, ou log deixado pelo nó contém a informação da localização e a hora em que passou por ele, e a direção tornando assim análises como a velocidade do nó, ou a grandeza da localização. Por exemplo, caso um nó sinalize que está em uma localização em vários intervalos de tempo consecutivos, podemos concluir que essa localização, ou área de alcance é grande, ou que o nó está em baixa velocidade. Por outro lado, caso uma localização é pouco marcada, o nó pode estar em uma alta velocidade ou a área é pequena, onde pouco "feromônio"foi liberado.

No PheroCast esses sinais que contém a localização e a hora, são convertidos em Phero Trails, um dígrafo que contém como vértices os locais visitados pelos nós, e a direção em que estão (norte, sul, leste, oeste) ligados por arestas de forma em que represente a movimentação do nó, e assim seu caminho o qual possui uma localização inicial e uma final.

Com os Phero Trails definidos, temos a capacidade de utilizar um dos algoritmos do PheroCast e traçar o que chamamos de Phero Maps, dígrafos que contém além da localização e a direção, a quantidade de vezes que o nó passou por ali, ligados por arestas

que indicam a próxima localização que o nó passou, e assim por diante. Assim conseguimos concluir que para chegar em uma localização final, o nó passou por diversos caminhos diferentes, também definimos as possíveis rotas entre um determinado ponto e outro.

Dado um Phero Map definido, temos todas as variáveis que o algoritmo de previsão de uma futura localização do nó, descrita no Phero Cast necessita. Tais elas são: localização, direção, quantidade de vezes que passou pela localização.

O produto final desse trabalho, tem como objetivo produzir dados reais de localização, tratando os feromônios como registros de rede wifi,registrados pelo dispositivo móvel em um determinado intervalo de tempo, e armazenando em um local para ser reproduzidas em mapas e diagramas, como aborda o Phero Cast.

2.2 Android

Para o desenvolvimento desse trabalho, utilizamos o sistema operacional Android, baseado em Linux que opera em celulares, netbooks, tablets, dentre outros dispositivos. O Android nos fornece um robusto framework de aplicação que nos permite implementar aplicações para dispositivos móveis utilizando Java.

Tal framework disponibiliza alguns componentes, os quais alguns deles foram utilizados na construção desse trabalho. Iremos detalhar cada componente a seguir:

http://developer.android.com/guide/components/fundamentals.html

2.2.1 Activities

O componente Activity representa uma única tela com interface para o usuário. Por exemplo, imaginemos um aplicativo de lista de compras, iremos ter uma Activity para exibir a lista de compras, outra activity para adicionar um novo produto na lista de compras, e outra para exibir um relatório de quanto gastamos no mês. Uma Activity é implementada no código do aplicativo extendendo a classe Activity do Android.

2.2.2 Services

Service é o componente que roda em background no Android, podendo processar longas operações ou processar algum processo remoto. Um service não é responsável por fornecer interface para o usuário. Para entendermos melhor o funcionamento do Service, podemos exemplificar com o processo de tocar música enquanto se navega em outro aplicativo. Um Service é implementado no código do aplicativo extendendo a classe Service do Android.

2.2.3 Content Providers

Os Content Providers são parte importantíssima da arquitetura de um sistema android. É responsabilidade deles prover às aplicações o conteúdo que elas precisam para funcionar, ou seja, os dados.

As aplicações poderiam muito bem acessar diretamente um banco de dados, por exemplo. Porém, é uma boa prática tornar o modo como os dados são gravados transparente à aplicação.

Além disso, essa técnica permite a criação de Shared Content Providers, que são providers "públicos" que podem ser acessados por várias aplicações. Por exemplo, existe o content provider de SMS/MMS que permite a qualquer aplicação ler as mensagens recebidas por um telefone celular.

2.2.4 Broadcast receivers

O Broadcast Receiver é o componente responsável por receber as mensagens broadcast do Sistema Operacional, por exemplo anúncios de que a tela foi desligada, ou a bateria está com pouca carga, ou uma foto foi tirada, ou algo na rede foi alterado. Esse componente, não fornece telas para o usuário, mas é possível criar uma barra de notificação através dele.

2.2.5 Arquivo Manifest

Para que o Android consiga iniciar um componente da aplicação, o sistema deve saber que esse componente existe através do arquivo AndroidManifest.xml. A aplicação desenvolvida deve conter as declarações de todos os componentes nesse arquivo, o qual está sempre localizado na pasta raíz do projeto.

3 Proposta

3.1 Arquitetura

A arquitetura proposta consiste em duas partes, clente e servidor. O cliente é responsável pela coleta dos dados, como posição atual, lista de SSID visíveis, e quaisquer outros dados que se deseje capturar. Oportunisticamente o cliente enviará os dados ao servidor. O servidor deverá consumir os dados enviados pelo dispositivo através de uma interface REST implementada e armazenar em um banco de dados, como mostra na Figura 1. A seguir detalhamos cada um das partes.

Figura 1 – Arquitetura Proposta

3.1.1 Cliente

Do lado do cliente desenvolveremos um sistema para rodar em dispositivos com o sistema operacional Android, utilizando a linguagem Java.

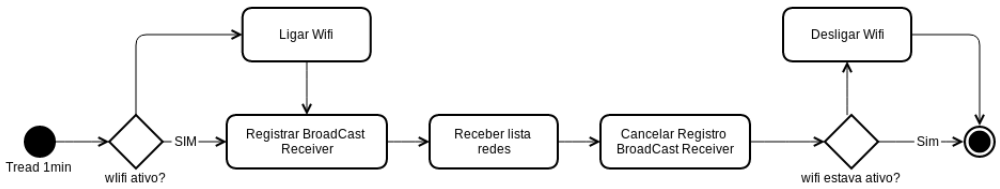

Figura 2 – Processo de captura dos dados pelo aplicativo.

O aplicativo irá utilizar os serviços de Wifi do aparelho para realizar a coleta dos pontos de rede que estão ao redor naquele determinado momento. Após coleta, o sistema irá armazenar em base de dados local do dispositivo, para isso utilizaremos o banco de dados SQLite. Tal procedimento é ilustrado na Figura 2.

Para realizar o envio ao servidor, o aplicativo será alertado quando o dispositivo se conectar a uma rede WiFi com rede Internet, assim irá enviar os dados armazenados em sua base local para o servidor de aplicação na nuvem e apagar os mesmos.

Caso o serviço de WiFi esteja desligado, o aplicativo deverá ser responsável por após ligar o Wifi e coletar os dados, desligar o mesmo, voltando assim para o estado inicial antes da coleta. Fazendo com que assim, não haja alteração no funcionamento do aparelho.

3.1.2 Servidor

Para o servidor, propomos utilizar algum servidor na nuvem de alta escalabilidade e alta disponibilidade, por se tratar de inserções contínuas todo o tempo.

O servidor deverá armazenar as informações virão do cliente, como:

- SSID
- BSSID
- Recursos
- Frequencia
- Level
- Horário
- Identificador do Usuário do Dispositivo

3.1.3 Limpeza dos dados

O sistema deverá tratar os dados do cliente com anonimidade, ou seja deverá haver algoritmos de segurança que tornem os dados do cliente anonimos, para preservar a identidade do usuário.

4 Desenvolvimento

4.1 Aplicativo

4.2 Google Docs

4.3 Limpeza dos dados

5 Resultados

5.1 Dados Coletados

5.2 Análise

6 Conclusão

Referências

Nenhuma citação no texto.

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 2 vezes nas páginas 11 e 12.

Anexos

ANEXO A – Código fonte

A.1 MainActivity.java

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

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

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

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

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  }
```