Redução do Consumo de Energia para Apps baseados em Geolocalização

Leonardo da Silva Oliveira
dev.loliveira@gmail.com
PESC/COPPETEC

Priscila da Silva Oliveira
pri22.de@gmail.com
UNESA

Resumo: Smartphones vêm se tornando dispositivos indispensáveis no cotidiano contemporâneo. Isso se dá pela rotina multitarefa das pessoas, graças ao crescente poder de processamento, armazenamento e principalmente intensa variedade de aplicativos disponíveis em lojas online, tais como as Play Store e AppleStore. Dentre as variadas categorias de aplicativos que podem ser encontrados nessas lojas, uma vem adquirindo atenção especial: os aplicativos baseados em geolocalização. Apesar da importância inerente aos dados geográficos obtidos em tempo real, esse tipo de aplicação gera preocupações com consumo de energia, em função do uso constante dos sensores de GPS e Wifi. De modo a reduzir o consumo de energia demandado pelos sensores de Wifi e GPS, a presente pesquisa adotou as estratégias do atraso de rede e do uso de filtros de tempo e distância, respectivamente. Ao utilizar as técnicas citadas foi possível alcançar o objetivo proposto pela pesquisa e obter um ganho significativo de economia de energia por parte da aplicação.

Palavras Chave: Geolocalização - Smartphone - Apps - Google-maps -

1. INTRODUÇÃO

Smartphones vêm se tornando dispositivos indispensáveis no cotidiano contemporâneo. Isso se dá pela rotina multitarefas das pessoas, graças ao crescente poder de processamento, armazenamento e principalmente intensa variadade de aplicativos disponíveis em lojas online, tais como as Play Store e AppleStore.

Dentre as principais categorias de aplicativos disponíveis, uma vem adquirindo atenção especial: os aplicativos baseados em geolocalização. Tal conceito possui vital importância em relação ao eficaz gerenciamento de dados geográficos para o processo de tomada de decisão gerencial e a forma como os mesmos podem ser associados a diversos outros tipos de informações, tais como financeiras, comportamentais e de saúde pública. No entanto, para que os aplicativos possam estar cientes da localização do aparelho, eles necessitam de informações captadas por sensores específicos, como o GPS ou Wifi. De modo a conseguir maior precisão nas informações obtidas, há a necessidade de se manter um canal de comunicação constante utilizando estes sensores.

No entanto, o uso não gerenciado desses sensores pode consumir toda a bateria do aparelho em menos de 06 horas (RAMOS *et al*, 2012). Portanto torna-se necessário a adoção de medidas que visem a redução do consumo por parte dos aplicativos enquanto minimiza o impacto da experiência do usuário.

Nesse contexto, a presente pesquisa visa apresentar meios de realizar o uso consciente de tais sensores de modo a minimizar a taxa de consumo de bateria enquanto preserva a experiência do usuário. Tendo, assim, como objetivo principal, a proposta de criação de uma ferramenta capaz de reduzir o consumo de bateria para aplicativo de geolocalização em tempo real definindo intervalos regulares para o uso dos sensores de GPS e Wifi e atingir, assim, as seguintes metas:

- ✓ Desenvolver uma classe-base para o gerenciamento do uso dos sensores, utilizando os princípios de orientação a objetos (OO);
- ✓ Definir intervalos regulares para o uso desses sensores;
- ✓ Estipular regras, como tempo e distância, para definir a taxa de atualização dos dados de GPS e
- ✓ Obter o fator de economia do uso de bateria a partir da implementação da ferramenta aos cenários de uso da mesma.

Para tanto, realizou-se um estudo bibliográfico prévio dos principais conceitos que norteiam a pesquisa, seguido pela criação do aplicativo “FriendsLocator”, para a aplicação prática do caso, cuja a utilização é baseada no uso de Wifi e GPS. E, por meio do uso da metodologia “Atraso” e da aplicação das melhores práticas do Google para redução de consumo de energia em aplicativos, tornou-se possível a obtenção do fator de economia da bateria após a implementação da melhoria proposta pela presente pesquisa.

2. METODOLOGIA

Para atingir os objetivos propostos faz-se necessário o desenvolvimento dos tópicos a seguir:

a. Desenvolver um estudo bibliográfico acerca dos conceitos de Geolocalização e redução do consumo do Wifi para serviços recorrentes;
b. Utilizar a linguagem de programação Java para desenvolver o aplicativo baseado em geolocalização: "*Friends Locator*";
c. Utilizar o ambiente de desenvolvimento *Android Studio* para codificar a aplicação;
d. Realizar testes unitários para garantir a qualidade e validação do software desenvolvido;
e. Utilizar a técnica do "atraso" para reduzir o consumo do Wifi e as melhores práticas do Google para reduzir o consumo do GPS;
f. Criar dois cenários: i) o cenário no qual o smatphone não utiliza as técnicas supracitadas e o ii) o segundo cenário, no qual a aplicação é utilizada juntatamente com as técnicas supracitadas;
g. Obter o fator econômico do consumo de energia do aplicativo, utilizando o aplicativo *PowerTutor*.

3. REFERENCIAL TEÓRICO

Para subsidiar melhor entendimento da pesquisa, construiu-se um referencial teórico dos principais conceitos norteadores da mesma: O conceito de Geolocalização e a redução do consumo de Wifi e GPS para serviços recorrentes.

3.1. O CONCEITO DE GEOLOCALIZAÇÃO

Trata-se do processamento da informação do local onde o usuário de um GPS (global position system) se encontra. O GPS baseia-se em sistema de coordenadas de latitude e longitude, provendo informações sobre o usuário do dispositivo em questão sobre sua localização, com base em dados providos via satélite. O uso inicial do GPS era basicamente militar, tornando-se, hoje em dia, fator popularizado entre massas.

Geolocalização, nos tempos atuais, passa a ser, portanto, uma ferramenta de gestão com diferencial competitivo. Tão evidentemente importante que relevantes estudos sobre o conceito podem ser citados nos trabalhos de (LUTHCKE *et al,* 2005), (ANISETTI *et al,* 2008), (KING, 2009), (COHEN *et al*, 2010), (STUTCHBURY *et al*, 2009) e (TSAI *et al*, 2010).

Vale ressaltar que o princípio de geolocalização em tempo real gera problemas relacionados devido sua constante demanda por energia. Dentre os sensores capazes de calcular a localização do aparelho, o GPS é o que mais consome energia ao longo do tempo. (XIA e*t al,* 2011) apresentam em seu trabalho, acerca da importância da minimização do uso de energia, que é possível obter a redução desse consumo tanto do Wifi quanto do GPS através de um trabalho colaborativo entre os dois sensores. Essa abordagem é conhecida como A*ssisted-GPS* (A-GPS).

A seção seguinte apresentará a estratégia de economia do consumo de energia do Wifi.

3.2. REDUÇÃO DO CONSUMO DO WIFI PARA SERVIÇOS RECORRENTES

O aplicativo "*FriendsLocator*" faz uso de diversos serviços de rede para estabelecer comunicação constante com a nuvem responsável pelos dados dos usuários. Esses serviços

variam desde a checagem de convites pendentes até a obtenção atualizada da localização dos contatos do usuário. Dada sua natureza recorrente, torna-se necessário a implementação de uma restrição de uso para reduzir a taxa de consumo de energia pelo sensor de Wifi.

A técnica utilizada para alcançar esse objetivo é conhecida como "atraso". (BALASUBRAMANIAN *et al,* 2009) realizaram um trabalho detalhado sobre essa abordagem. Os autores segmentam os softwares que fazem uso de serviços de rede em função de sua tolerância ao atraso. O objetivo dessa estratégia é reduzir a quantidade de energia consumida pelo sensor ao realizar a transição do estado "*idlle"* para o estado "em uso", e vice-versa. Para isso, todos os serviços de rede utilizados pela aplicação são mantidos em um *pool* de serviços temporários, agendados para serem executados paralelamente.

A sessão 4.6 apresenta um exemplo de implementação para essa estratégia.

A seção seguinte apresenta a app *FriendsLocato*r, bem como a construção dos algoritmos que implementam as técnicas supracitadas.

4. A APLICAÇÃO

Para implementação do caso prático foi criado o aplicativo *"FriendsLocator*", baseado em Geolocalização.

FriendsLocator é um aplicativo desenvolvido para smartphone baseado em geolocalização em tempo real de seus usuários. As informações pertinentes aos mesmos, como os seus contatos e as suas respectivas localizações, serão mantidas e fornecidas por uma nuvem *(cloud computing*).

A comunicação do aplicativo e a nuvem será realizada utilizando a tecnologia JSON pois, além de possuir uma organização estruturada de dados em formato amigável ao olho humano, há diversas bibliotecas capazes de gerar objetos de suas respectivas linguagens de programação utilizando dados em formato JSON como entrada. Esse tipo de comunicação permite a criação de uma abstração não apenas do sistema operacional mas também do dispositivo utilizado pelo usuário.

4.1. A API DE COMUNICAÇÃO

De modo a fornecer os serviços, a *cloud* deverá prover uma API de comunicação consistente. Segundo a organização 3scale (2012), API é uma interface a um componente de software que pode ser invocado à distancia através de uma rede, utilizando padrões de comunicação.

A seguir serão apresentada a lista de funcionalidades básicas que a nuvem deverá oferecer em sua API:

a) Mecanismos de autenticação e criação de contas;

b) Importar e remover contatos e

c) Mecanismos para fornecer e atualizar a localização de usuários.

4.2. CONSUMO DE ENERGIA VS EXPERIÊNCIA DO USUÁRIO

Prover a localização de um smartphone de maneira constante e precisa, com baixo consumo de energia, é um grande desafio. Dada a natureza desses aplicativos, há a necessidade de se manter um canal de comunicação constante entre os sensores de GPS e Wifi. Porém, caso não venha a ser implementado um controle para o uso desses sensores, o aplicativo irá consumir rapidamente a bateria do aparelho, tornando-o assim inviável para uso.

Existem algumas estratégias que podem ser adotadas para minimizar o consumo energético tanto pelo GPS quanto pelo Wifi. As sessões a seguir apresentam algumas das técnicas que podem ser adotadas para reduzir o consumo de energia por esses sensores enquanto minimizam o impacto da experiência do usuário.

4.3. CONTROLANDO O USO DO GPS

O Android oferece em sua *software development kit* (SDK) oficial mecanismos para restringir o uso do GPS de modo a reduzir sua demanda por energia. Antes de se fazer uso do sensor é possível especificar filtros que definam o intervalo entre as atualizações da localização. Um desses filtros é o tempo, que determina que o sensor somente deverá realizar o cálculo da localização uma vez a cada *n* milissegundos. Outro filtro importante que pode ser especificado é a distância mínima, indicanndo que o provedor da localização somente enviará para o aplicativo as atualizações quando a diferença entre a localização corrente e a medida anteriormente for superior a distância especificada (em metros).

Uma forma de implementação, como apresentado no algoritmo 1, de uma classe capaz de obter atualizações do sensor de GPS enquando implementa mecanismos de preservar o consumo de bateria, encontra-se a seguir.

Algoritmo 1: Classe responsável pela redução do consumo do GPS

```
public class DeviceLocation implements LocationListener {
   private final int      TIME_FREQ = 1 * 1000;    // 1 segundo
   private final int      DIST_FREQ = 3;           // 3 metros
   private final int      LC_TIME_FREQ = 5 * 1000; // 5 segundos
   private final int      LC_DIST_FREQ = 6;        // 6 metros
   private boolean        lowerConsuption = false;

 public DeviceLocation(Context context) {
      this.context = context;
   }

   @Override
   public void onLocationChanged(Location location) {
      /* Callback chamado a cada atualização da localização */
   }

   @Override
   public void onStatusChanged(String provider, int status, Bundle extras) {
      /* Atualização do status do provedor de serviço */
   }

   protected LocationManager getLocationManager(Context context) {
      /* Deve retornar um objeto da classe LocationManager */
   }

   protected LocationProvider getProvider(LocationManager locationManager) {
      /* Deve retornar um provedor de serviço de localização */
   }

   public void stopLowerConsumption() {
```

```
    lowerConsuption = false;
    restart();
}

public void switchToLowerConsumption() {
    lowerConsuption = true;
    restart();
}

public boolean isLowerConsumption() {
    return lowerConsuption;
}

public void start() {
    /* Taxa de consumo de energia será verificada nesse método.
     * Caso a instância dessa classe esteja programada para
     * operar em baixo consumo de energia, então serão
     * utilizadas as variáveis LC_TIME_FREQ, LC_DIST_FREQ.
     */
    LocationManager locationManager;
    locationManager = getLocationManager(context);
    if(isLowerConsuption())
      locationManager.requestLocationUpdates(
          provider.getName(),
          LC_TIME_FREQ, LC_DIST_FREQ,
          this);
    else
      locationManager.requestLocationUpdates(
          provider.getName(),
          TIME_FREQ, DIST_FREQ,
          this);
}

public void stop() {
    locationManager.removeUpdates(this);
}

public void restart() {
    /* Método utilizado para reiniciar o serviço. Útil quando o
     * objeto precisa alterar seu modo de operação de consumo.
     */
    stop();
    start();
}
```

Os métodos mais relevantes para o contexto do presente trabalho são: *stopLowerConsumption*, *switchToLowerConsumption* e *isLowerConsumption*. Como seus nomes sugerem, suas responsabilidades estão ligadas diretamente com a taxa de uso GPS. Suas implementações fazem uso de cinco variáveis vitais para a lógica: (LC_)TIME_FREQ, (LC_)DIST_FREQ, lowerConsumption.

As variáveis inteiras, TIME_FREQ, DIST_FREQ, indicam as regras de tempo e distância mínimas entre as atualizações, respectivamente. Suas variantes, LC_TIME_FREQ e LC_DIST_FREQ, possuem as mesmas responsabilidades porém, representam o modo de operação de baixo consumo de energia. Seus valores devem ser alterados para refletir o contexto de cada aplicação. Para um aplicativo de localização em tempo real, caracterizado como sensíveis ao atraso, julgou-se que 5 segundos e 6 metros fossem valores ideais para a redução do consumo de energia ao mesmo tempo em que era preservada a experiência do usuário.

O método *start,* como o nome sugere, é responsável por iniciar o sensor de GPS utilizando os devidos filtros de tempo e distância baseado no modo de operação da classe.

4.4. CONTROLANDO O USO DO WIFI

De modo a reduzir os ciclos de transição de estados do Wifi e consequentemente sua demanda por energia a longo prazo, decidiu-se utilizar a estratégia do "atraso". As sessões seguintes irão demonstrar trechos de códigos que representam uma forma de implementação para essa abordagem.

Foi construída uma classe abstrata responsável por realizar as requisições de rede. Essa classe atua como um intermediário para realizar o controle de acesso ao sensor e restringir seu uso. O algoritmo 2, a seguir, apresenta a classe implementada.

Algoritmo 2: Classe responsável pelo controle da taxa de uso do Wifi

```
public abstract class NetworkScheduler implements Runnable {
   private static int              THRESHOLD = 10000;
   private long                    t0 = 0;

   public void schedule(final Runnable r) {
      final long scheduleAt;
      long currentTime = System.currentTimeMillis();

      if(t0 == 0) {
         t0 = currentTime;
         scheduleAt = THRESHOLD;
      } else if(currentTime - t0 < THRESHOLD) {
         scheduleAt = THRESHOLD - (currentTime - t0);
      } else {
         t0 = currentTime;
         scheduleAt = THRESHOLD;
      }

      Thread thread = new Thread() {
         @Override
         public void run() {
            try {
               sleep(scheduleAt);
               r.run();
            } catch(InterruptedException iE) {
               }
         }
      };
      thread.start();
   }
}
```

É possível notar que a sua implementação está fazendo uso da estratégia do "atraso" para minimizar a quantidade de vezes que o sensor precisará ser ligado/desligado entre as transmissões. Foi estipulado um intervalo de 10 segundos de limite (*threshold)*. Isso significa que ao realizar uma transferência utilizando o Wifi, essa requisição irá aguardar 10 segundos antes de utilizar o sensor, e toda requisição realizada nesse intervalo será agendada para ser executada paralelamente à primeira requisição.

Ao final da implementação do método "*schedule*" é possível notar a criação de uma thread. Por padrão, para evitar que serviços de rede causem danos à experiência do usuário, o S.O Android não permite que os mesmos possam ser executados na *thread* principal. Essa regra é válida para os aplicativos que estejam utilizando a versão da SDK *Honeycomb* ou superior.

Caso o desenvolvedor tente executar os serviços de rede na thread principal, será lançada a exceção *"NetworkOnMainThreadException"*. Para evitar esse problema, a chamada no método "*run"*, implementado pelas subclasses, é realizada dentro de uma nova thread. Para garantir a consistência de que os serviços irão executar dentro de sua respectiva faixa de tempo, foi utilizada a função "*sleep*". Essa função pausa o fluxo de execução na thread em que foi chamada por *n* milissegundos.

A seguir, no algoritmo 3, será apresentado um exemplo de como uma subclasse pode ser implementada para se beneficiar da estratégia do atraso.

Algoritmo 3: Subclasse responsável pela extensão das características da NetworkScheduler

```
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;

public class DownloadPage extends NetworkScheduler {
   private String              pageUrl;

   public DownloadPage(String pageUrl) {
      this.pageUrl = pageUrl;
   }

   public void schedule() {
      super.schedule(this);
   }

   /*
    * Esse método será chamado pela superclasse quando chegar o momento de
    * de ativar a Wifi
    */
   public void run() {
      HttpClient client = new DefaultHttpClient();
      HttpResponse response = client.execute( new HttpGet( this.pageUrl ) );
      /* Download da página concluído... */
   }
}
```

O método mais importante das subclasses de "*NetworkScheduler"* é o método "*run"*, pois é nele no qual as requisições de rede serão realizadas. No exemplo apresentado no algoritmo 3 é possível notar que será realizado o download de uma página web utilizando o protocolo de transferência de hipertexto (HTTTP).

Normalmente, após a conclusão da execução de um serviço de rede, é preciso enviar os dados baixados para outro segmento da aplicação. Uma maneira de implementar essa "ponte" de comunicação entre diferentes partes da aplicação, que podem estar executando em diferentes *threads*, é utilizando a classe *Handler*, como demonstrado no algoritmo 4. Nesse são apresentados duas classes que podem se comunicar utilizando um *Handler* como intermediário. Lembrando que o objetivo é implementar, de maneira simples, uma forma de viabilizar a troca de dados entre duas classes que podem estar executando em diferentes *threads* na aplicação.

Algoritmo 4: Classe responsável pela recepção do resultado da operação

```
/* Implementação da classe HandlerReceiver */
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

public class HandlerReceiver implements Handler.Callback {
   private int         a;
   private int         b;

   public HandlerReceiver(int a, int b) {
      this.a = a;
      this.b = b;
   }
   @Override
   public boolean handleMessage(Message message) {
      /*
       * Cálculo foi realizado pela classe HandlerSender e enviado
       * para HandlerReceiver através do método sendMessage
       */
      Bundle bundle = message.getData();
      int result = bundle.getInt( "result" );
      Log.d("HandlerReceiver", String.format("%d + %d = %d", a, b, result));
   }
}

/* Implementação da classe HandlerSender */
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;

public class HandlerSender {
   private int a;
   private int b;
   private Handler receiver;

   public HandlerSender(int a, int b, Handler receiver) {
      this.a = a;
      this.b = b;
      this.receiver = receiver;
   }

   public void run() {
      /*  Irá somar dois números inteiros e enviar o resultado
       * para o Handler destinatário */
      Message message = new Message();
      Bundle bundle = new Bundle();

      bundle.putInt("result", a+b);
      message.setData( bundle );
      receiver.sendMessage( message );
   }
}
```

5. CONLUSÕES E RESULTADOS

Foi utilizado o aplicativo *PowerTutor* para obter os resultados do consumo de energia da aplicação em função do tempo. A Tabela 1 apresenta o consumo de energia por parte da aplicação nos dois cenários propostos.

Tabela 1: Resultados de Testes Unitários para verificação de economia de consumo de bateria

Cenário 1 - Uso Normal da Bateria		Cenário 2 - Uso Com Economia de bateria	
T (min)	CONSUMO (J)	T (min)	CONSUMO (J)
1	0.997	1	0.406
2	2	2	0.788
3	2.7	3	1.4
4	4	4	1.7
5	5	5	2
6	5.9	6	2.3
7	6.9	7	2.5
8	7.9	8	2.7
9	8.8	9	3
10	9.8	10	3.2
11	11.4	11	5.8
12	13.5	12	5.9
13	14.7	13	6.1
14	16.2	14	6.3
15	17.5	15	6.7
16	18.6	16	6.9
17	20.1	17	7.1
18	21.5	18	7.4
19	22.6	19	7.6
20	23.7	20	7.9

Com base nos dados obtidos na tabela supracitada, é possível notar o ganho considerável na economia de energia na aplicação junto ao cenário econômico 2, também representada pela análise gráfica a seguir, como na figura 1.

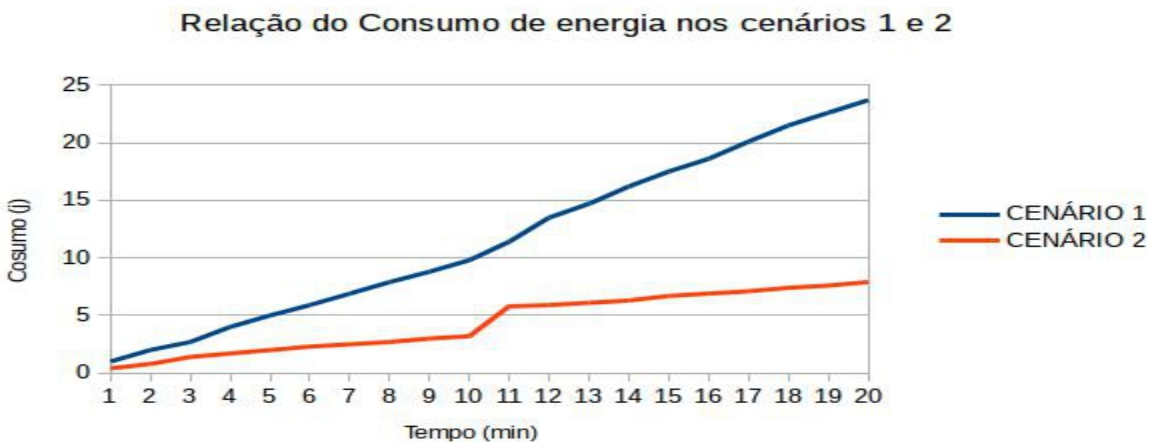

Figura 1: Representação gráfica da relação de consumo de energia da bateria nos cenários 1 e 2

Durante o teste realizado, com duração de 20 minutos de uso da bateria do smartphone, foram encontradas as médias do consumo para os cenários (1), consumo normal e (2), consumo econômico. Para isso foram geradas duas versões distintas da app, uma sem a adoção dessas estratégias de melhoria e a outra contendo a implementação das técnicas do atraso para Wifi e dos filtros para uso do GPS, respectivamente. O fator de economia médio pode ser encontrado na tabela 2, a seguir:

Tabela 2: Relação de economia de consumo de bateria

Utilização Média da Bateria (mW)	
Consumo Normal	Consumo Econômico
27	5

Portanto, a adoção de estratégias e boas praticas sobre o uso do Wifi e GPS para o desenvolvimento de aplicativos para smartphone pode viabilizar melhorias no tocante à economia do consumo de bateria, aumentando a satisfação do usuário do aparelho ao reduzir a necessidade de recargas do mesmo e possibilitando o aumento da vida útil da bateria.

6. REFERÊNCIAS

ANISETTI, M. et al. Advanced localization of mobile terminal in cellular network. In International Journal of communications, Network and system sciences. Vol.1, Nr.1: June 6, 2008.

BALASUBRAMANIAN, N., BALASUBRAMANIAN, A., VENKATARAMANI, A., Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications, 2009.

COHEN, M.B. et al. Geolocation of terrestrial gamma-ray flash source lightning. In GEOPHYSICAL RESEARCH LETTERS, VOL. 37, 2010.

COMISSÃO EUROPÉIA, Opinion 13/2011 on Geolocation services on smart mobile devices, Disponível em: http://ec.europa.eu/justice/policies/privacy/docs/wpdocs/2011/wp185_en.pdf.

HANDLER, Disponível em: https://developer.android.com/reference/android/os/Handler.html, Acessado em: 05 de Maio de 2016.

KING, K. Geolocation and Federalism on the internet: Cutting internet gambling's Gordian Knot (July 14,2009). Columbia Science and Technology Law Review, Vol. XI, 2010.

LOCATIONMANAGER, Disponível em: https://developer.android.com/reference/android/location/LocationManager.html, Acessado em: 08 de Maio de 2016.

LUTHCKE, S.B. et al. Reduction of ICESat systematic geolocation errors and the impact on ice sheet elevation chance detection. Disponivel em: http://onlinelibrary.wiley.com/doi/10.1029/2005GL023689/full, Acessado em : 03 de fevereiro de 2016.

NETWORKONMAINTHREADEXCEPTION, Disponível em: https://developer.android.com/reference/android/os/NetworkOnMainThreadException.html Acessado em: 05 de Maio de 2016.

POWERTUTOR, Disponível em: https://play.google.com/store/apps/details?id=edu.umich.PowerTutor, Acessado em: 14 de Maio de 2016.

RAMOS., H *et al*, Energy Efficient GPS Sensing with Cloud Offloading, 2012.

STUTCHBURY, B.J.M et al. Tracking Long-distance songbird migration by using geolocators. Disponivel: http://science.sciencemag.org/content/323/5916/896. Acessado em 04 de janeiro de 2016.

TSAI, J. et al. Location-sharing Technologies: Privacy risks and controls, Carnegie Mellon University, USA, 2009.

XIA, F., ZHANG, WEI., DING, F., HAO, F., A-GPS Assisted Wifi Access Point Discovery on Mobile Devices for Energy Saving, 2011