ABSTRACT
In recent years, the market of location-aware applications has become increasingly larger. However, one of the main problems is still partially unsolved: battery consumption. The low operating time of the batteries hampers the opportunities for collecting users’ location data in order to provide a variety of services, ranging from events suggestion to sport tracking. This article proposes a way to diminish the battery consumption by reducing the communication between the clients and the server. The approach is to model the trajectory of the users, so that the clients communicate a new position only when necessary: namely, when the predicted position excessively deviates from the actual one. In order to demonstrate the idea, a prototype has been implemented. This paper evaluates the performance of the model-based technique and it compares the proposed approach with the approach commonly used for location tracking in mobile devices.

Categories and Subject Descriptors
C.2.4 [Computer-Subject Descriptors]: Distributed Systems; C.4 [Performance of System]: Design Studies

Keywords
Location-aware applications; location-tracking; energy efficiency; trajectory modeling

1. INTRODUCTION
Location-aware applications are becoming increasingly important in the market of mobile devices. On one hand, it is common that mobile devices are equipped with positioning sensors, such as Global Positioning System (GPS). On the other hand, developers are widely exploiting location in order to improve their applications. Some examples include context-aware advertisements, popular events suggestions, navigation, and sport tracking [17][1][8]. It is also important to mention applications that require a continuous location tracking, such as road traffic applications [6][23].

These systems are usually composed of two distinct components. On one side are the mobile devices, responsible for collecting users’ positions. On the other side is the cloud system, where the logic of the application is executed.

Crucial to these cases is the communication between the two components. The common way for the cloud to know the user position is that the users periodically send their own location to the cloud. The location is typically coming from GPS sensor. The more frequently the users send the location to the cloud, the more accurate knowledge the cloud has of the location of the users. However, the higher the frequency of communication, the higher is the load both for the mobile devices and for the network resources. A frequent communication is especially harmful for the mobile device battery [2].

Battery consumption, in fact, remains one of the main problems of location-aware applications, although smartphone hardware is getting progressively more energy efficient. There are two main problems that quickly consume the battery power of mobile devices. First, the continuous utilization of the sensors providing the location. Second, the communication with the server, which is needed in order to provide a service to the user. A simple solution would be to force the application to retrieve the location less frequently, however the inaccuracy of this approach presents a crucial drawback, making it unfeasible in some cases.

The possibility of limiting the activity of the location sensors has been extensively investigated. Earlier solutions in these cases propose the use of different sensors, such as the accelerometer and the compass, in order to diminish the activity of the GPS sensor [25][20]. Another approach is about understanding the users’ mobility patterns to predict their trajectories. This leads to a reduced need for sampling the location continuously [26][12]. However, there is a lack of research into the strategy of minimizing the communication.

The trajectory tracking approach used in this work is known as dead reckoning (DR), a technique extensively studied [24]. However, studies are still needed regarding the application of DR in mobile cloud computing.

This article proposes a system that utilizes dead reckoning for tracking the location of the users. The system collects past positions to build a model of the user’s trajectory. The model is shared between the cloud system and the mobile device and it is updated only if the actual position excessively deviates from the prediction. Via this model, both the cloud and the mobile device can estimate a position at...
any time. If the mobile device detects that the estimated position deviates from the current position (received from the GPS sensor), it notifies the cloud. This approach allows the cloud to maintain an accurate knowledge of the users' locations, reducing the communication. On one hand, it reduces the battery consumption of the mobile devices. On the other hand, it reduces the resources utilization of the network providers.

The main contributions of this paper are:

- An effective way to reduce the communication, saving resources and energy in position-tracking cloud applications.
- A working open-source prototype, demonstrating the proposed approach.
- An evaluation of the performance of the prototype in terms of accuracy of the predicted locations, exchanged messages and energy consumption. The evaluation includes a comparison between the proposed approach and the commonly used technique used in mobile applications.

## 2. RELATED WORK

Dead reckoning has been extensively studied in different fields, such as communication in distributed virtual environment and robotics [21, 16].

A remarkable work related to efficient trajectory tracking focuses on optimization of moving object databases [19]. The article compares and improves algorithms for trajectory simplification to store in a database. The authors propose a solution with two components: a communication protocol and simplification algorithm. The communication protocol utilizes linear dead reckoning, which is the same algorithm proposed in this article. The simplification of the trajectory is achieved by using Douglas-Peucker algorithm.

Other studies utilize Douglas-Peucker algorithm [15]. The authors propose a novel way to efficiently track a mobile entity. The proposed system uses Douglas-Peucker algorithm for trajectory simplification. The simplification reduces the size the data that has to be sent, reducing the communication. Since Douglas-Peucker algorithm is not used for prediction, the need of the communication with the server is recognized via other sensors, such as compass or accelerometer.

Researchers argue that dead reckoning is not suitable for mobile cloud computing and they propose algorithms for energy-efficient trajectory tracking by using a variety of sensors [9, 14]. The idea is to limit the activity of GPS sensor and sample data only when a change in direction is detected by the compass or accelerometer. In addition, the proposed system is able to automatically adjust the trajectory error. The authors argue that dead reckoning performance are lower than the proposed solution.

Different studies propose energy-efficient position tracking approaches that do not involve a cloud system. Namely the users’ location is not sent through the network, but it is only recorded in the mobile device. In this scenario the communication is not a problem, and the goal of the researchers is to lower the number of requests to the location sensors.

One of the most promising studies investigates how to use different sensors and a machine-learning model to predict the user position and limit the queries to the location sensor [25]. The idea is to retrieve a new location from the sensor only when the trajectory changes.

Other studies propose more complex models to predict the users’ trajectory. An extensive research [19] shows that human trajectories are highly dependent on habits and time. Exploiting this outcome, it is possible to develop accurate models of the trajectories.

Assuming that the GPS sensor is one of the most energy-consuming sensors, researchers investigate the possibilities to automatically change the source of position [12]. This approach leads to decrease the usage of energy-consuming sensors, such as GPS, and utilizes other sensors, such as WiFi or GPRS. The downside of this approach, however, is that the accuracy decreases.

Studies show that position tracking is critical in some situations such as vehicle to pedestrian collision avoidance [8]. In this case given the high mobility of the clients, it is necessary to heavily utilize 3G or LTE connections, which are particularly battery consuming. The proposed approach is to dynamically change the interval of communication according to the context of pedestrian, switching from high frequency only when high accuracy is needed.

Other research shows that efficient-tracking is possible also using different communication technology, such as SMS [18]. The authors focus on a novel system to track the user position by communicating with SMS. However, this approach has some limitations: the system can track only one user at the time and it relies on SMS, which may result in more expenses for the user.

Different articles provide solutions for real-time tracking of the user position using 802.11 wireless network [20, 11]. Even if using wireless network as communication technology is useful if all entities are connected, it is a unacceptable limitation in most of the contexts. An interesting use case of real-time tracking is to prevent car theft and monitor car parameters [23].

## 3. PROPOSED SOLUTION

### 3.1 Overview

The proposed model-based approach is composed of two different components: a mobile client, which is responsible for collecting the positions of the users, and the server at the cloud, which is periodically building a predictive model of users’ trajectory.

The first interaction between the two components is a bootstrap phase: the client sends some position readings to the server, which builds a model of the users’ trajectory. The bootstrap phase lasts about 5 seconds. Afterwards, the server sends the model to the client. This step is crucial for the system: using the model the server predicts new positions, while the client periodically checks its accuracy by comparing the predicted position with the new position obtained from the sensors.

In particular, the client continuously queries the location sensors and checks the new location and the location predicted by the models. If the two positions are similar, there is no need for communication. Conversely, if the two positions are different, the client sends the updated location to the server, which then calculates the updated model.

Every time the client retrieves a new location, it checks the distance between the predicted and the actual position. If the distance is excessively large, it requests a model up-
date. The cloud system updates the model and sends it to the client. It is important to note that the communication occurs only in this case. In other words, if the model always predicts users’ position correctly, the communication occurs only at the bootstrap phase.

3.2 Model

The system utilizes a linear dead reckoning. Dead reckoning is performed in two steps: first, given two coordinates they are projected on a plane, using WGS8 conversion. Second, the line passing between two points and the speed are calculated. As a result the system can predict the position of the users at a given time. Algorithm 1 shows how the speed and the direction are computed, while Algorithm 2 illustrates how the next positions are predicted.

Algorithm 1 Linear Dead Reckoning
1: Input: \( < lat_1, long_1, t_1 >, < lat_2, long_2, t_2 > \)
2: \( x_1, y_1 \leftarrow \text{toWGS8}(lat_1, lon_1) \)
3: \( x_2, y_2 \leftarrow \text{toWGS8}(lat_2, lon_2) \)
4: \( \Delta t \leftarrow t_2 - t_1 \)
5: \( \Delta s \leftarrow \text{euclidean-distance}(x_1, y_1, x_2, y_2) \)
6: \( \text{speed} \leftarrow \frac{\Delta s}{\Delta t} \)
7: \( \text{angle} \leftarrow \arctan(2(y_2 - y_1, x_2 - x_1)) \)
8: return \( < \text{angle}, \text{speed} > \)

Algorithm 2 Prediction of the user location at a certain time
1: Input: \( \text{speed}, \text{angle}, t_{\text{new}}, t \) (timestamp of last update), \( x \) (x of last update), \( y \) (y of last update)
2: \( \Delta t \leftarrow t_{\text{new}} - t_{\text{last}} \)
3: \( \Delta s \leftarrow \text{speed} \times \Delta t \)
4: \( x_{\text{new}} \leftarrow x + \Delta s \cos(\text{angle}) \)
5: \( y_{\text{new}} \leftarrow y + \Delta s \sin(\text{angle}) \)
6: \( \text{lat}, \text{long} \leftarrow \text{toLatLon}(x_{\text{new}}, y_{\text{new}}) \)
7: return \( < \text{lat}, \text{long} > \)

The reason why to use this model is that human trajectories are usually regular. If the trajectories are straight lines with constant velocity, the results of the model-based approach are considerably good.

4. EXPERIMENT

The prototype is implemented entirely using web technologies: the server is implemented using node.js and the data is stored using Redis. The client is implemented using HTML5 and Javascript. The communication is based on web-sockets.

The technologies are chosen in order to implement a scalable server and cross-platform clients. The code is open-source and available for further research [3]. The device used in the experiments is the smartphone LG G2, running Android 4.4.2.

The goals of the experiments are to understand the accuracy and the efficiency of the prototype. We propose an analysis based on three parameters: the average distance of the model predictions from the GPS data, the rate of exchanged messages, and the average power consumption. The number of exchanged messages measures the amount of communication necessary in order to keep updated the knowledge of the cloud system about the users’ trajectory.

In order to demonstrate that reducing the communication has a crucial role in savings energy, we compare the energy consumption of three types of applications: a network-heavy application, a GPS-heavy application, and an hybrid application. The reason of this additional experiment is that studies related to energy-efficient tracking mentions both approaches as best practice. Some works suggest that the communication is one of the main cause of battery consumption, others that it is better to limit the utilization of GPS during the experiments. They differ mainly because of velocity variance: during the trip the bus stops, accelerates, and turns much more frequently and rapidly than the user walking in the campus, for this reason the bus experiment is more challenging from the model point of view.

Concerning the prototype, we performed two experiments. The first experiment takes place on the campus of Aalto University. The distance of the full path is 2 km and the experiment last about 24.6 minutes. The second experiment evaluates the two approaches recording data on a bus trip of 23 minutes covering a distance of about 10 km.

Table 1 shows some additional information about the experiments. They differ mainly because of velocity variance: during the trip the bus stops, accelerates, and turns much more frequently and rapidly than the user walking in the campus, for this reason the bus experiment is more challenging from the model point of view.

A crucial role for the performance of the model is the deviation limit. The deviation limit is the maximum distance between the position predicted by the model and the current position retrieved by the GPS sensor. If the distance is higher than the deviation limit, a new model is needed and communication occurs.

Using a small deviation limit, the accuracy increases together with the amount of communication. Conversely, having a large deviation limit will decrease the communication, but the accuracy will be negatively affected. In the experiments, we study different values of deviation limit in order to better understand how it influences the performance.

Each experiment compares two types of data: one using the proposed solution (i.e. the model-based approach), and one using the commonly used approach, which is to communicate with the cloud system every time a new position is available.

4.1 Accuracy

We first look at the accuracy of the model-based approach when changing the deviation limit parameter’s value. We compute the accuracy as average distance between the GPS data and the predictions of the model. It is important to remark that the deviation limit as a strong impact on accuracy measure: the more the deviation limit is large, the more the errors are visible.

Figure 1 depicts the average errors measured with respect to the GPS measurement during the experiments. In all the experiments, the error is limited and it is comparable with the GPS accuracy, which is about 3 meters [4]. The errors during the bus experiment are higher mainly due to frequent velocity changes.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Walk</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average speed</td>
<td>1.4 m/s</td>
<td>7.9 m/s</td>
</tr>
<tr>
<td>Standard deviation of speed</td>
<td>0.2 m/s</td>
<td>6.6 m/s</td>
</tr>
<tr>
<td>Minimum speed</td>
<td>0.25 m/s</td>
<td>0.25 m/s</td>
</tr>
<tr>
<td>Maximum speed</td>
<td>1.75 m/s</td>
<td>22.3 ms/s</td>
</tr>
<tr>
<td>Time of the experiments</td>
<td>24.7 min</td>
<td>22.9 min</td>
</tr>
<tr>
<td>Distance</td>
<td>2.1 km</td>
<td>10.9 km</td>
</tr>
</tbody>
</table>

Table 1: Additional information about the experiments.
4.2 Exchanged messages

One of the goals of the experiment is to study whether the proposed solution reduces the need of communication between the client and the server. The metric used in the experiment is the number of exchanged messages.

Figure 2 shows a comparison between the proposed solution using different deviation limits and the commonly used approach. The model-based strategy saves from to 90.2% up to 98.4% of the messages in the walking experiment, and from 61.1% up to 81.4% of the messages in the bus experiment.

4.3 Energy consumption

During the experiment, we recorded the data about the communication and the size of the exchanged packets. We simulate the network traffic and measure the energy consumption with a power monitor. The device used for this experiment is a Samsung Galaxy S4.

Table 2 shows the average power consumption using 3G and LTE networks using four deviation limits. The measurements has been recorded with the screen turned on. The average power consumption without communication is 580 mW.

It is important to remember that the energy measurements focus only on the power consumed by communicating the position via mobile network. The GPS sensor is not considered in the measurements because it is independent from the chosen approach.

4.4 GPS vs Network

The last experiment is designed to validate the assumption that reducing the communication will increase the battery operating time of a smartphone.

The experiments records the energy consumption of three applications: a network-heavy application, a GPS-heavy application, and an hybrid application. The network traffic is generated by sending package to a server, while GPS-heavy application simply retrieve a new position at high rate.

The utilized tools are the same of the previous experiments. Table 3 reports the results. An important remark is to notice that GPS consumes less energy than network. Therefore, limiting the communication has a stronger impact than limiting GPS in energy saving.

5. DISCUSSION

5.1 Deviation limit

The value of deviation limit has a crucial role on the performance. This threshold represents the maximum distance between the predicted position and the GPS position. If the distance is higher than the deviation limit, a new model is needed. The larger is the deviation limit, the larger is the average error. However the communication rate is progressively reduced. The choice of the right value for the deviation limit depends on the requirements of the application: if the accuracy is a critical parameter, then the deviation limit should be as small as possible. Conversely, if the main goal is energy-efficiency a large deviation limit has to be chosen.

5.2 Average Power consumption vs Number of messages

Table 2 lists the average power consumption using 3G and LTE networks. The experiments show that using LTE network consumes more power than 3G network; however, the

https://www.msoon.com/LabEquipment/PowerMonitor/
savings are comparable between the two network technologies.

The power consumption is lower when the message savings is higher, however an interesting result is to compare the traffic saving and the average power consumption saving. Figure 3 shows the savings in the walking and bus experiments. Even if the network traffic savings is close to 98%, the average power consumption is reduced only between 42% and 38% in the walking experiment, and about 22.5% and 22.7% in the bus experiment (using 3G and LTE respectively).

Although the energy saving is remarkable, it is low compared to the network traffic reduction. In order to investigate further, we measure the power savings in a university-owned network, which utilizes Discontinuous communication (DRX). Studies show that DRX enhances the energy efficiency [10].

Two experiment has been simulated: the walking experiment and the bus experiment, using 30 meters as deviation limit. In the first case the average power is 650 mW, while in the second case 660mW.

Looking at the results comparing the normal approach with the model-based approach, in the walking experiments with DRX enabled the energy savings are 50%, 11% more than the experiment with commercial LTE network. In the bus experiment the savings are about 49%, 27% more then with commercial LTE network.

6. CONCLUSION AND FUTURE WORK

The experiments show that the proposed solution performs as good as the commonly used approach in developing mobile application. However, the model-based approach saves a significant amount of communication and leads up to 41% lower average power consumption.

A possible improvement is to extend the model using more points and build a more complex curve instead of using simple linear dead reckoning. Another future work would be to apply the same method used to estimate the velocity to estimate acceleration. Then, it would be possible to exploit new information to accurately predict the trajectory of the users.

Another improvement of the model would be to estimate users’ trajectories using a model that considers the time and the habits of people [13] or history of past locations [25]. A different approach in improving the accuracy of the system is to trigger a model update using different sensors such as the accelerometer and the compass. These sensors have a negligible impact on the battery life and they may identify more precisely when to send a new position and update the model [25].

Acknowledgements

This work was partly supported by the Academy of Finland, grant number 278207, and by the Finnish Funding Agency for Technology and Innovation, grant number 200/31/2013.

7. REFERENCES


