Adaptive Query Algorithm for Location Oriented Applications

Daniel-Octavian Rizea, Dan-Ștefan Tudose, Alexandru-Corneliu Olteanu and Nicolae Țăpuș
Computer Science and Engineering Department
University Politehnica of Bucharest
Bucharest, Romania
daniel.rizea@cti.pub.ro, dan.tudose@cs.pub.ro, alexandru.olteanu@cs.pub.ro, nicolae.tapus@cs.pub.ro

***Abstract*—The Internet of Things and Ubiquitous Computing are becoming more and more popular in the IT world. There are increasingly more heterogeneous systems of interconnected devices, from general purpose smartphones to highly specialized embedded devices. An important constraint of mobile devices is the limited battery capacity. Users will stop using devices that have to be recharged frequently and that are not reliable. For both small sensors and mobile devices a significant energy is consumed with data communication. We develop a system, consisting of a smartphone and an embedded device designed to measure air quality. This paper presents an Adaptive Query Algorithm that can improve the energy consumption for Bluetooth data transfers between the smartphone and the embedded device. Our tests show that considerate querying can lead to power consumption 20 times smaller than in the case of a permanent connection.**

***Keywords*— smartphones, power consumption, communication, location oriented.**

I. Introduction

With the growth of projects and systems that rely on the Ubiquitous Computing paradigm, several challenges and research opportunities arise. The tendency to make smaller devices, which communicate with each other by WiFi, Bluetooth and other emerging technologies, is faced with a veritable power consumption barrier. In mobile devices more than a third of the power consumption is spent on communication tasks and in small sensors more than a half is wasted on communication [1]. Devices can be made smaller and smaller, but battery cells usually cannot.

We developed a system consisting of a smartphone and an embedded device designed to measure air quality. The communication between the mobile phone and the sensor device needs to be effective and power efficient in order to have a minimum effect on the phone's recharge cycles. One approach to this issue, presented in this paper, is to reduce the power consumption of the device using an adaptive query algorithm to minimize the energy used in data transfers. The algorithm gives great advantages to location-aware projects by reducing data communication when it is not needed.

This work has been funded by the European Commission under grant agreement FP7-ICT-258280 TWISNet project.

II. Background and Related Work

Analyzing power consumption is an interesting research topic, as shown by various projects that can be found in the literature. Caroll and Heiser [1] design multiple micro-benchmarks to associate the power costs to modules of a mobile system. They try to determine the power consumption of different parts like CPU, GSM and WiFi. Zhang et al. [2] describe a power module construction technique that they use to characterize 3G, GSM, WiFi, CPU and screen, and to introduce PowerTutor, an Android application that can use these models for power consumption estimation on any device. We use such tools to estimate the power consumption of different components, but we found little solutions on estimating power consumption from the Bluetooth radio.

Balasubramanian et al. [3] show that 3G, GSM and WiFi incur a high tail energy overhead. Pering et al. [4] describe methods to reduce the power consumption by switching between Bluetooth and WiFi. The Bluetooth module has a power consumption as much as 10 times lower than WiFi. WiFi is intended for high-bandwidth and 100 meters coverage while Bluetooth is designed for low-bandwidth and a coverage of 10 meters. The authors also describe that power consumption in idle mode compared to active mode is 4 times smaller for WiFi and 6 to 10 times smaller for Bluetooth. We find this significant as it justifies the need of advanced algorithms that power down these interfaces when communication is not necessary.

Various other papers try to determine algorithms and technologies for reducing energy consumption, by modeling optimum power as a Nash equilibrium [5], by employing back-off methods for synchronization [6], or introducing an active-sleep duty cycle [7]. We propose an adaptive algorithm for reducing the polling rate in location-aware applications, leveraging the fact that new queries are not necessary if the location has not changed much.

III. System Design

The system consists of an Android application interconnected with an embedded device, built by our team, designed to measure air quality. The app on the smartphone needs to query the sensor device using Bluetooth in order to

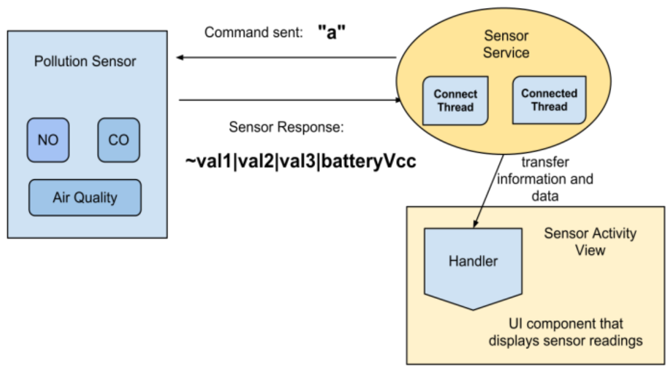

Fig. 1. General architecture and communication mechanism

collect pollution data and display it intuitively to the user on a map, as shown in Fig.1.

The communication between the mobile phone and the sensor device needs to be effective and power efficient in order to have a minimum effect on the phone's recharge cycles. One approach to minimize power consumption is to switch on the Bluetooth adapter only when a query must be done. We will show the effectiveness of this method in the next section.

IV. THE ADAPTIVE QUERY ALGORITHM

We propose an Adaptive Algorithm that changes the update time interval based on the previous user location and the previous update time. This approach is suitable for location-aware applications by reducing data communication when it is not needed.

For example, let us consider that a user remains in a location for about 1 hour. In a normal approach 60 queries will be made for the device (assuming a 1 minute period between queries). In our adaptive approach the second time the application wakes up it will compare the last location with the previous one and will increase the query interval by doubling the previous time interval. Only 6 sensor queries will be executed. Let's assume now that the service battery usage for about 1 hour and 60 requests is about 4%, then our algorithm will have a total of 0.4% battery usage over 1 hour. The improvement is significant and is needed in order to make the application popular among users.

It is important to note that the query time interval, the cool down time and other parameters can vary according to the application's purpose and needs. The variations of these parameters influence the total power consumption of the Bluetooth module. We will give a formal presentation of this adaptive algorithm, tuned for our needs in the PollutionTrack project.

A. *System Characteristics*

PollutionTrack is a personal pollution monitoring system. It can be set to only record pollution information. In this case, the system is in **passive mode** and the user is not informed when the pollution levels are increasing. This mode can be used when the user does not want to be actively informed of pollution danger levels and only wants to collect air pollution data. In this case, the update period, cool down time and wakeup parameters can be relaxed. In this mode the location will have the biggest impact on query intervals and pollution data values will have a small contribution in setting the update periods.

The **active mode** of the system is used when the user wants to be actively informed whenever specific pollution levels are crossed. In this case the adaptive algorithm is more aggressive and will have a smaller wakeup time interval. It is important for the user to be notified when a pollution level is crossed. For this to work, the system will make more queries whenever an increasing pattern of pollution values are detected. This being the case, the query period of the sensor will be heavily influenced by the pollution information and not so much by location changes.

B. *The algorithm*

The update time interval is dynamically calculated and uses the following formula:

$$T_u = T_b * (c_{ld} + c_{dd}) \qquad (1)$$

, where:

- T_u: update time
- T_b: update time base
- c_{ld}: location dependent coefficient
- c_{dd}: data dependent coefficient

```
function location_dependent(){
   if ( mode is passive )
        1: if ( current_location == new_location )
               cld = α * cld
           else
               cld = 1/ α * cld

        2: if( cld > max_cld [ passive mode ] )
               max_cld [ passive mode ] = cld

        3: if( cld < min_cld [ passive mode ] )
               max_cld [ passive mode ] = cld

   else  // mode is active
        1: if ( current_location == new_location )
               cld = 3 * β * cld
           else
               cld = 2 / 3 * β * cld

        2: if( cld > max_cld [ active mode ] )
               max_cld [ active mode ] = cld

        3: if( cld < min_cld [ active mode ] )
               max_cld [ active mode ] = cld

   return cld
}
```

Fig. 2. Pseudocode for the function that computes the location dependent coefficient

The location dependent coefficient and the data dependent coefficient are computed separately and their values follow the observations made regarding the system characteristics regarding the passive and active mode.

The function depicted in Fig.2 computes the location dependent coefficient, based on the current and previous location. For passive and active modes the behavior is in principle the same, only some parameters vary. If the location stays the same, then the time between queries is increased. This means the smartphone will issue fewer queries and less power will be consumed. If the location changes then the time between queries is reduced. This will enable the sensor to collect fresh data. This behavior is based on the fact that pollution information does not change radically in a small period of time and in the same location. The active mode of the algorithm is used for dangerous environments where pollution levels can grow rapidly in the same location and over a few minutes.

For the data dependent function, depicted in Fig.3, we hold a queue with the last n recorded values. Based on these and the current value from the sensor, we determine how fast the pollution data changes. We compute the change rate with:

$$r = \sum (v - D_i) / v \qquad (2)$$

, where:

- r: change rate
- n: window size
- v: current value
- D: array of size n with recently collected data

The size of the window should vary according to different needs. In our experiments we use a size of 5 values, as this should cover timeframes of more than one minute, given the sampling period which, in our case is 15 seconds.

If the change rate is positive, there is a growth in the values, so the queries should be made more often, at a rate directly proportional with the growth rate. If it is negative, the queries should occur less often. Also, please note how in passive mode the effect of the change rate is twice the effect in active mode.

```
function data_dependent(){
     array D[n]
     **compute change rate** according to (2)

     if (mode is passive)
          **cdd = 2 * λ / change_rate**
     else  // mode is active
          **cdd = λ / change_rate**

     return cdd
}
```

Fig. 3. Pseudocode for the function that computes the data dependent coefficient

V. Tests And Results

Communication is a critical aspect when dealing with client-server architectures, especially considering performance constraints.

A. *Experimental Setup*

The tests were run on a HTC Nexus One, running Android 2.2. Some relevant hardware characteristics are its Qualcomm Snapdragon CPU at 1GHz and the Bluetooth module with Bluetooth 2.0 + EDR. For testing, we have used both the sensor device and a Java computer application that we developed, which behaves exactly as the pollution sensor, having the same Bluetooth communication protocol. With this program we can emulate the behavior of the sensor and vary the sensors data to test the data dependent function.

Each test involves running the application for 300 seconds. Most of the performance tests were run while remaining in the same location, as this is the maximum efficiency case for the algorithm. We study three query strategies: Permanent Connection, Adaptive Algorithm and Fixed Time Interval Query. In the former, the smartphone establishes a permanent connection with the sensor device. This is demanding in terms of energy consumption from the CPU and also from the Bluetooth module. For Fixed Time Interval Query the CPU and Bluetooth module are put into connection state, where power consumption is the biggest, only when a data query is needed. The Adaptive Algorithm is the most efficient. It has the characteristics of the Fixed Time Interval but also adapts the query periods to the environment and the user's location, saving as much energy as possible without impacting the usefulness of the application.

$$\alpha + \beta = \chi. \qquad (1) \qquad (1)$$

B. *CPU power consumption*

The power consumption has been recorded using the PowerTutor app [2]. Fig.4 shows the power consumption on the CPU for the three query strategies. We then estimate the energy consumption over 300 seconds, as presented in Table I.

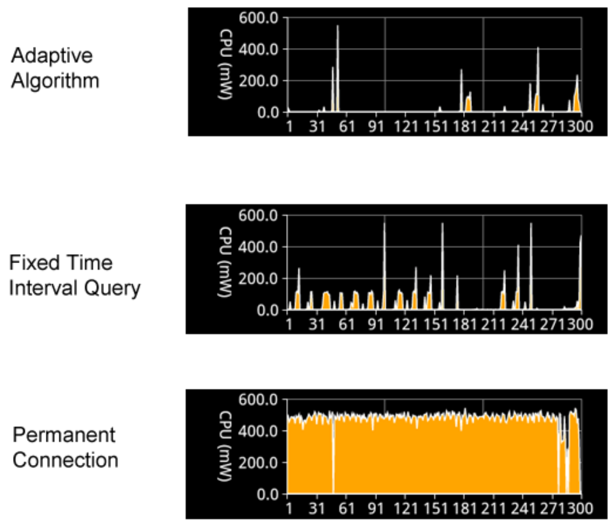

Fig. 4. CPU energy consumption

TABLE I. ESTIMATED ENERGY CONSUMPTION ON CPU FOR THE THREE QUERY STRATEGIES.

	Test 1 Energy (mJ)	Test 2 Energy (mJ)	Test 3 Energy (mJ)	Average Energy (mJ)
Permanent Connection	136800	133800	137800	136133
Fixed Time Query Interval	5524	5237	5635	5465
Adaptive Algorithm	1340	1226	1364	1310

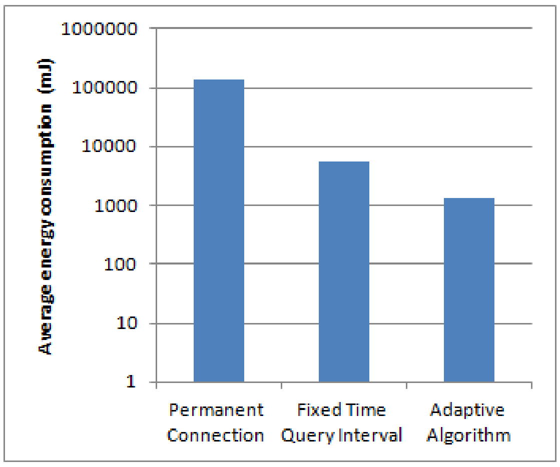

Fig. 5. Estimated values for energy consumption on CPU

From the results we can see that the Adaptive Algorithm approach is almost 5 times better than the Fixed Time Query Interval and 100 times better than the permanent connection approach. The significant power saving is justified because in 300 seconds, the duration of the tests, we had 20 fixed queries at every 15 seconds in Fixed Time Query Interval mode and only 4 queries in the Adaptive mode (the location was not modified) which makes the Adaptive Algorithm mode to have 5 times less queries than Fixed Time interval.

C. *Data transfers*

Data transfers are performed, as described in Section III, on a query basis: the smartphone asks for data from the sensor device, which in turn responds with a single message. Thus, for a single exchange, we have 2 bytes sent from the smartphone and 21 bytes received. We compute the data transfers for a period of 300 seconds.

As an energy consumption model, we use the following formula:

$$E_{total} = P_{conn} * T_{conn} + P_{trans} * T_{trans} + P_{on} * T_{on} \quad (3)$$

Based on this formula we can calculate the estimated power consumption for the three query strategies that we used. For all the three cases we have a T_{trans} time that is the same for

TABLE II. QUANTITY OF TRANSFERRED DATA AND ESTIMATED ENERGY CONSUMPTION ON BLUETOOTH IN THE THREE QUERY STRATEGIES: THE SIZE OF THE TRANSFERS IS COMPUTED GIVEN THE MESSAGE STRUCTURE FROM SECTION III

	Data transferred (bytes)	Energy(mJ)
Permanent Connection	21 x 20 = 420	28740
Fixed Time Query Interval	21 x 20 = 420	14180
Adaptive Algorithm	21 x 4 = 84	6228

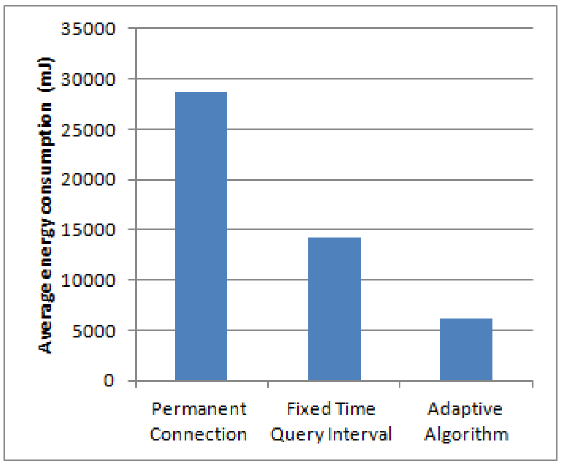

Fig. 6. Estimated values for energy consumption on Bluetooth

Permanent Connection and Fixed Time Query Interval but is smaller for Adaptive Algorithm. In the case of Permanent Connection we have no T_{on} time because the Bluetooth is always connected. For Fixed Query Interval we have a T_{conn} time when the Bluetooth is in connected state. After the data is transmitted the Bluetooth connection is terminated, saving energy. In the Adaptive Algorithm the Bluetooth connection time varies, adapting the query time based on location and environment data.

We can also approximate the benefits of the algorithm concerning energy consumption. The measurements done in [8] show that a Bluetooth module in the on state (with no connection) takes 15mW (P_{on}), in the connected idle state takes 65mW (P_{conn}) and in transmission 432mW (P_{trans}). In both the Fixed Time Query Interval and Permanent Connection we intend to read values from the sensor at every 15 seconds.

T_{trans} is less for the Adaptive Algorithm than in the other two cases because the adaptive query algorithm varies the query interval. In our 300 second test we have 20 requests at intervals of 15 seconds. The transmission time takes approximately 1 second per query. T_{conn} = 300 seconds and T_{trans} is 15 seconds. T_{on} = 285. With the help of the adaptive algorithm, the number of queries goes down to 4 queries in 300 seconds.

VI. Future Work

We plan to further improve the algorithm. We could use the inertial movement unit or scarce GPS readings to determine velocity and use that to alter the query period and conserve battery energy. Regression and other machine learning techniques can be used to learn of possible events from the collected data and alter the interrogation period. We can also include other factors that depend on the purpose of the project. For example, in our pollution detection project the query interval can be lowered when the data collected shows that the user is entering in a potentially dangerous environment. Data would be collected at a higher pace in order to warn the user as soon as the environment conditions become hazardous.

We also need to improve the accuracy in expressing power consumption for communication. We consider changes in the experimental setup, which would allow us to take precise readings of the energy spent on communication.

VII. Conclusions

We have studied power consumption in a system consisting of a smartphone and an embedded device designed to measure air quality.

The system has two functioning modes: passive and aggressive. These two modes influence how the adaptive algorithm performs. In passive mode, the location factor has a greater impact on the query interval. In aggressive mode the recorded data varies the parameters used in the algorithm.

We have done measurements to compare power consumption dedicated to CPU usage in three cases: using a permanent connection, using a fixed time query interval and using our adaptive algorithm. We find that our algorithm is 5 times more efficient than the Fixed Time Query Interval strategy, which coincides with the improvement in the number of queries.

We extend the estimation on energy saving for Bluetooth communication, using energy models from the literature. We arrive to the conclusion that, reducing the transmission time and the connected idle time, the Adaptive Algorithm is approximately twice more efficient than the Fixed Time Query Interval strategy, which in turn is approximately twice more efficient than the Permanent Connection.

The adaptive algorithm is suited for location-aware applications, leveraging the fact that new queries are not necessary if the location has not changed much. We find that considerate querying can lead to energy consumption 20 times smaller than in the case of a permanent connection. We plan to improve the algorithm in our further studies, by determining velocity and using machine learning techniques in order to alter the query period.

Acknowledgement

This work has been funded by the European Commission under grant agreement FP7-ICT-258280 TWISNet project.

References

[1] Carroll, Aaron, and Gernot Heiser. "An analysis of power consumption in a smartphone." Proceedings of the 2010 USENIX conference on USENIX annual technical conference. USENIX Association, 2010.

[2] Zhang, Lide, et al. "Accurate online power estimation and automatic battery behavior based power model generation for smartphones." Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. ACM, 2010.

[3] Balasubramanian, Niranjan, Aruna Balasubramanian, and Arun Venkataramani. "Energy consumption in mobile phones: a measurement study and implications for network applications." Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009.

[4] Pering, Trevor, et al. "Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces." Proceedings of the 4th international conference on Mobile systems, applications and services. ACM, 2006.

[5] Yu, Wei, George Ginis, and John M. Cioffi. "An adaptive multiuser power control algorithm for VDSL." Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE. Vol. 1. IEEE, 2001.

[6]] Agarwal, Anant, and Mathews Cherian. Adaptive backoff synchronization techniques. Vol. 17. No. 3. ACM, 1989.

[7] Van Dam, Tijs, and Koen Langendoen. "An adaptive energy-efficient MAC protocol for wireless sensor networks." Proceedings of the 1st international conference on Embedded networked sensor systems. ACM, 2003.

[8] Perrucci, G. P., F. H. P. Fitzek, and J. Widmer. "Survey on energy consumption entities on the smartphone platform." Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd. IEEE, 2011.

[9] Frank Fitzek et al., “Bluetooth: consumption for different states”, Technical Report, Retrieved 21.12.2012, Available online: http://mobiledevices.kom.aau.dk/research/energy_measurements_on_mobile_phones/results/data_communication/bluetooth_different_status/