Context-Aware Mobile Power Management Using Fuzzy Inference as a Service

Mohammad Moghimi, Jagannathan Venkatesh, Piero Zappi and Tajana Rosing

University of California, San Diego, Computer Science and Engineering,
9500 Gilman Drive, La Jolla, CA, 92037. USA.
{mmoghimi,jvenkatesh, pzappi,tajana}@ucsd.edu

Abstract. As smartphones become ubiquitous, their energy consumption remains one of the most important issues. Mobile devices operate in a dynamically changing context, and their embedded sensors can be used to extract the relevant context needed for resource optimization. In this paper, we present a context-aware power management system implemented as a widely-applicable middleware application. Fuzzy inference is used to represent a high-level description of context, which is provided as a service. We test our approach using actual periodic and streaming applications on a mobile phone. Our results show energy reduction of 13-50% for periodic applications, and 18-36% for streaming applications.

Keywords: Context Awareness; Energy Efficiency; Mobile Systems; Fuzzy Inference.

1 Introduction

Mobile devices such as smartphones have become ubiquitous. As a consequence, they operate in a dynamically changing context, both in terms of user needs and computational requirements. Context has been defined as "any information that can be used to characterize the situation of an entity", and in mobile devices, it is obtained from sensors, communication media, or locally available data [15]. Context awareness in mobile devices has been proven useful when developing user interfaces [15][12], proactive service provisioning [4][5], or efficient resource management [10]. The concept has been explored in several comprehensive studies [2][8][21] over the past decade, as new mobile technologies emerge. In general, previous work on context aware systems focuses either on novel technique for context recognition, or on efficient management of context knowledge. However, one area that has been largely overlooked is leveraging context recognition methods to provide adaptive power management as a service for mobile applications.

The rationale for power management in mobile devices stems from their growth in computational complexity and power consumption with each process generation. However, as mobile battery technology has not grown at the same rate, device lifetimes are decreasing [19]. Subsequent work [3] analyzes power consumption in Android-based smartphones during common tasks, noting the significant contribution of GPRS and Wi-Fi radios, graphics processors, and the CPU. Consequently, substantial research has been devoted to leveraging context to reduce power consumption or manage battery use. In [1] the authors aim to optimize the high energy overhead in mobile networking by leveraging characteristics of typical mobile applications, scheduling transfers in batches to reduce the overhead of each transfer, and aggressively prefetching data for applications that can benefit from it. Similarly, the work in [20] curbs the energy consumption of data transmission over Wi-Fi or cellular networks by exploiting the complementary energy traces of each type of connection. The application opts to use either Wi-Fi or cellular data transmission by determining the lower-energy operation based on the current context. The authors demonstrate an improvement in battery lifetime by up to 42%. The work presented in [19] uses energy traces from applications and location information to determine when and where phones can be charged, and how much remaining energy is available for critical applications. Consequently, the energy manager allocates resources for non-critical applications based on the time to the next predicted charging opportunity. Musolesi et al [17] investigates the high energy consumption of continuous location sensing applications which need to communicate GPS data with a backend server. Using short-term prediction of movement based on context, the application duty-cycles communication with the backend, demonstrating up to 60% communication savings with limited impact on accuracy. Similarly, [13] and [24] leverage context to determine how often to retrieve GPS location, opting instead to use the adaptive frequency or other means of determining location for energy savings.

The related work above opts to exploit context awareness either in a limited fashion, or for a specific application. However, mobile platforms such as Android, Windows Phone, and iOS run many applications at a time. Competition among running applications for access to sensor data in order to independently retrieve similar context information is both inefficient and redundant. A better approach is an intermediate, extensible layer which can retrieve input from the many sensors and sources of raw context available on phones. In turn, it would consolidate the input into higher-level, filtered, and processed context that is queryable and usable by all applications.

The advantage of this approach is the elimination of redundancy in each application, which would otherwise need to independently process context data, as well as removing the constraint of sharing sensor access among applications. Previous work approached the concept of context as a service for goals other than power management. For example, the context manager designed in [15] retrieves raw context information from sensors, offering high-level context aimed at providing an adaptive UI. Similarly, [11] designs and implements a context delivery system for Symbian devices. The end-to-end implementation provides context as a middleware service, abstracting raw sensor input into usable context via fuzzy inference. The resulting context is made available to applications as a query, via subscription to specific updates, or through even higher abstracted services, which perform post-processing using simple Bayesian networks for more complex context. The system exemplifies the type of context management we aim to leverage in our work. We aim to extend this approach further by providing context-aware actions to applications in a power-aware, adaptive user experience.

Typical context aware systems use decision tree-based methods in their inference block [10] [20]. Decision trees are a generalization of IF-THEN-ELSE decisions, whose outputs represent the actions to be taken. Rules can be as simple as a timeout from the last screen interaction to more complex rules combining the interaction of data from multiple sensors. This approach establishes crisp thresholds on context variables, which can cause extreme or unintentional responses to complex input. In contrast, fuzzy inference handles context in a more natural way. Since mobile system context is derived in part from environmental behavior, it is inherently fuzzy: described using degrees of confidence rather than absolutes [15]. Additionally, as context recognition involves a degree of uncertainty due to sensor inaccuracies or the variability in defining a context, the use of exact thresholds is not always reasonable. As such, fuzzy logic variables have a truth value or a degree of confidence ranging from 0 and 1, which is a more natural expression of typical context information. Fuzzy inference provides an additional advantage: the aggregation of different rules based on confidence values expressed in a continuous manner, which is difficult when using a decision tree or discrete implementation. As higher abstractions of context can determined by the correlation of variables with different, overlapping confidence factors, fuzzy logic provides a natural means to aggregate raw input data into appropriate context. Due to its advantages, fuzzy logic has been used extensively for context detection in mobile devices as well as other embedded systems. Mäntyjärvi et al. [11][15] design and implement a context manager based on fuzzy inference to provide adaptive UI management for Symbian devices. Ciaramella et al. [4] develop an application selection service using fuzzy logic that incorporates location, time, and perceived task (based on applications running and preset situations). The contexts are provided as a query to a backend service which returns the most recommended application to achieve the perceived task. Martinez et al. [16] propose a fuzzy logic system for energy management in hybrid vehicles, referencing the ability of fuzzy systems to outperform their counterparts in embedded domains, as well as their usefulness for energy management. Context variables represent battery state-of-charge, supercapacitor state, terrain, and driving conditions, and use human experience to determine which power subsystem should propel the vehicle at a given time.

In this paper, we leverage such context managers to provide adaptive power management for mobile phones. We design and implement a low complexity, extensible framework that uses fuzzy inference of power-related context variables to mitigate a significant problem in mobile devices: energy consumption. Applications expose needed context parameters to the power manager and register rules for adaptive behavior. In turn, the power manager tunes applications parameters based on the rules and the available context. We test our power manager on two applications that are representative of typical mobile tasks. The first represents applications that run in the background, periodically retrieving data from a remote sever (e.g. RSS feed, email). We define a cost function that models the trade-off between energy consumption of the system and its performance (measured as delay in presenting the user desired information) with which we can compare our approach to decision tree systems. We also develop and test a representative of streaming applications (e.g. audio and video streaming), which require continuous, lossy data retrieval from a backend server. We define a threshold for audible bitrates, and measure power savings on the device under context-aware conditions in comparison to non-adaptive conditions.

The rest of the paper is organized as follows. Section 2 describes the proposed power management method. Section 3 outlines the applications developed, the experiments performed, and their results. Section 4 concludes with possible future directions.

2 System Design

In this section, we present an overview of the system, including the implementation of fuzzy logic and exploring the major components of the power management system.

2.1 Fuzzy Inference

Fuzzy logic is a generalization of set theory, which, instead of binary membership, uses functions that assign every input a value between 0 and 1. The outputs, or context variables, can be used to provide a high-level abstraction of low level data. For example, time contexts such as *TimeOfDay* = {*Morning, Work Hours, Evening, Night*} represent a high-level abstraction of the low-level input of system time. Figure 1 depicts the member functions that are composed to form the fuzzy logic context for *TimeOfDay*.

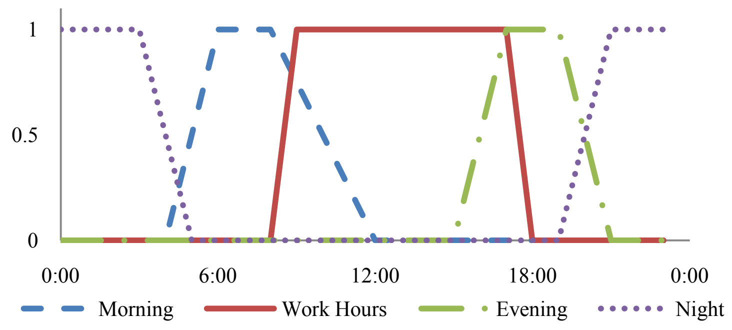

Fig. 1. Fuzzy membership functions for *TimeOfDay*. In our experiments, the values for *Morning*, *Work Hours*, *Evening*, and *Night*, are determined statically, though this can be determined through machine learning models and other methods as well.

Client applications can take advantage of the more gradual transitions in fuzzy logic to develop more gradual control of system behavior. In the above example, *Night* is determined by the period in which night is detected with absolute confidence ($x \leq 3 \cup x > 21$) as well as a gradual threshold for when night begins ($19 < x \leq 21$) and ends ($3 < x \leq 5$).

$$\textbf{Night} = \begin{cases} \mathbf{1}, & \boldsymbol{x} \leq \mathbf{3} \cup \boldsymbol{x} > 21 \\ \frac{5}{2} - \frac{1}{2}\boldsymbol{x}, & \mathbf{3} < x \leq 5 \\ \frac{1}{2}\boldsymbol{x} - \frac{19}{2}, & \mathbf{19} < x \leq 21 \\ \mathbf{0}, & \mathbf{5} < x \leq 19 \end{cases} \tag{1}$$

This example illustrates a fundamental advantage of fuzzy logic for use with context: there is not necessarily a pure definition of *Night*, so instead of attempting to force such an absolute definition, we instead define the boundary around night with a growing confidence.

Defining the optimal thresholds for each member of *TimeOfDay* is highly dependent on the behavior of the user. While in this case, these boundaries are defined statically, the fuzzy context management systems explored in Section 1 defined thresholds statically or through learned behavior.

2.2 Context-aware Power Manager

The power management system we develop is designed to be flexible and extensible, providing context-aware, adaptive behavior, to applications. This approach strikes a balance between individual applications retrieving sensor information independently [2], and a central context management system that registers updates with every application that requests it [11]. In both cases, the burden of continually adapting to context changes is placed on the applications themselves. Additionally, in the former case, the Context Detection Block (CDB) can become a bottleneck for applications attempting to retrieve context, and inefficient if they try to retrieve the same context information. For example, two applications that need the same context variable (i.e. the time of day, or the ambient noise level) must separately be updated every time the context changes. Instead, we place a power manager in between the application and the CDB, which serves a dual purpose. First, it aggregates power-relevant context variables from both the sensors directly, and from a CDB, which provides higher abstractions of raw context. This removes the inefficiency of applications retrieving the same context. Secondly, instead of performing actions independently, applications register rules for updating context-dependent input data with the power manager.

This allows similar applications to adapt to context with the same variables. The resulting system, composed of sensor input, CDB, and the power manager, forms a flexible framework, able to use any context detection block, including those specified in Section 1. Figure 2 shows the architecture of the framework.

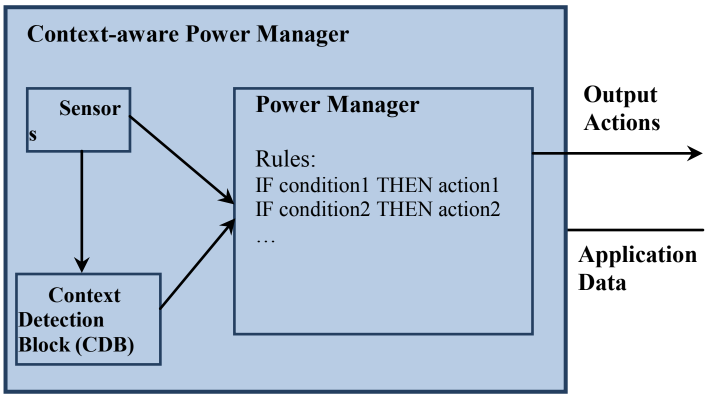

Fig. 2. Layout of the *Context-aware Power Manager*, showing *sensors* for raw data, a *context-detection block* for processed data, and the *power manager* with sample rules

The power manager, unique to our system, retrieves context information and outputs a set of application-specific actions. Context information is either derived directly from input sensors (for simple contexts such as time or date), or as high-level abstractions in the context detection block (for processed context such as relative location, noise level, or user activity level). The output actions are application-specific variables, registered by the respective applications. The power manager tunes these variables based on static rules to provide context-aware energy-efficient behavior.

For example, let us consider a periodic application whose rate of download (*UpdateRate*) can be modified by the power manager based on the time of day. The rules governing the update can simply use numerical (via system time), or a more fuzzy representation, such as the *TimeOfDay* variable discussed above (provided by the CDB). Such a rule can be paired with the output action, i.e. *UpdateRate*={10min, 30min, 60min, 120min}. A rule connecting *TimeOfDay* with *UpdateRate*, can be specified in the power manager in the format "IF context THEN output_action". In the case of *TimeOfDay*, we can establish the update interval of a periodic application based on the intuition that users do not check an application for updates at night:

IF *TimeOfDay* == "night", **THEN** *UpdateRate*= "1hr-update"

This further illustrates the advantage of registering rules in the power manager: different periodic applications that desire to use the same action can all register to use UpdateRate to determine their output action, reducing their query rate to once per hour.

The system, implemented for Android API Version 10, is designed to be versatile and extensible. The power manager can retrieve new raw (from sensors) or abstracted (from the context detection block) context variables, extend and re-aggregate existing context variables, and establish new or extend rules for output actions.

Our system accomplishes all such goals with limited interaction with the established platform. Each application provides two descriptor files to the power manager: a fuzzy control logic (FCL) file and a properties file. The FCL file [9] exposes context-sensitive application parameters that can be tuned, and establishes the rules for adapting to the context. More specifically, the FCL file lists the context variables needed, the output actions modified, the fuzzy sets that compose both, and the piecewise linear membership functions which translate input context to the output actions. The properties file is used to store updates to the context-dependent variables for an application. Figure 3 shows the overview of the system and the interaction between the components.

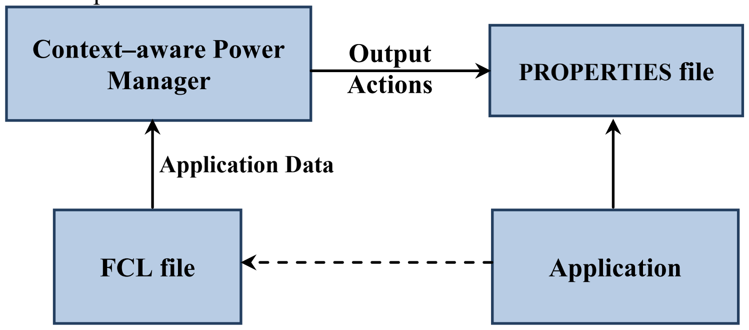

Fig. 3. Software component overview, consisting of the *Context-aware Power Manager* (see **Fig. 2**), the *FCL file* that provide application data, and the *PROPERTIES file*, to which output actions are published, and which is read by the *application*.

When a context-aware application is developed, it registers with the power manager by exposing its FCL and properties files. The power manager parses the FCL file, registering any new context inputs and action outputs, and integrates the rules needed to translate the context into the member functions. When the power manager is running, it provides the defuzzified output variables, which determine the application's adaptations, to the properties file. For example, in the case of streaming media applications, context would be used to determine the bitrate of the incoming stream, which strongly affects the energy consumed by the device. The power manager would output the new bitrate for the application to the properties file, which then changes the quality of the incoming stream as desired.

3 Experiments and Results

3.1 Experimental Setup

We implemented our framework on both a T-Mobile G1 development phone and a Nexus One phone. Power measurements for the running applications are provided via the PowerTutor application. PowerTutor estimates power consumption based on a power model customized for each phone's hardware, and has been demonstrated to have less than 2.5% long-term error [23]. Our experiments consisted of two separate classes of applications: *periodic downloading applications*, which query and retrieve data from a backend at specific intervals, and *streaming applications*, which need to constantly retrieve data over the duration of the stream.

3.2 Periodically Downloading Applications

Periodically downloading applications have risen in popularity, and now consist of a significant portion of mobile apps [5]. We model our test applications after common examples of such apps: RSS feed-following, Twitter, and e-mail retrieval, implemented in RSSApp, TwitterApp, and EmailApp, respectively. Each application is implemented as a background service without a graphical user interface (GUI). Communication to the backend composes a majority of the energy requirement for such applications, and consequently, our goal with each of the model applications is to reduce the amount of energy spent on connectivity. The work in [1] analyzed the energy cost of such network-dependent applications, identifying tail energy, caused by mobile data connections' retention of high-energy states even after completion of data retrieval, to be a significant contributor to energy use. To compensate, the authors implemented batch-processing of periodic applications to avoid recurring overheads, and a similar approach was undertaken in [14]. We approached the problem in a similar manner, with the added advantage of the context framework.

Observed user interaction with periodic applications shows a variation over the course of the day, requiring more frequent updates in the morning, as the user is brought up to date on the events of the recent past, and less frequent updates throughout the day, culminating in low access needs at night. Additionally, we leverage the lower energy requirements of Wi-Fi over GSM/GPRS connectivity.

We establish *DayType* = {Weekday, Weekend}, *TimeOfDay* = {Morning, Noon, Evening, Night}, and *WifiAccess* = {Wi-Fi, No Wi-Fi} as context variables, and *TimeStep* = {Short, Medium, Long} and *levelOfInfo* = {0, 1, 2} as the output actions. *TimeStep* refers to the update frequency of the application, *levelOfInfo* refers to the depth of data needed by the application (0 = title only, 1 = title and text, and 2 = title, text, and attachments), and *WifiAccess* = represents the access to Wi-Fi connectivity. The fuzzy context rules are described in **Error! Reference source not found.** and **Error! Reference source not found.**.

Table 1. *TimeStep* membership sets for *DayType* and *TimeOfDay*.

	Morning	**Noon**	**Evening**	**Night**
Weekday	Short	Short	Medium	Long
Weekend	Medium	Medium	Medium	Long

Table 2. *LevelOfInfo* membership sets for *WifiAccess* and *TimeOfDay*.

	Morning	**Noon**	**Evening**	**Night**
Wi-Fi	2	1	1	0
No Wi-Fi	2	2	2	0

The defuzzification of *TimeStep* is obtained by the center of gravity (cogs) for the piecewise functions that comprise Short, Medium, and Long, each weighted by the level of confidence in the particular function. This is defined in Equation 2:

$$\mathbf{f} = \mathbf{cogs}(\mathbf{C_{short}} * \mathbf{f_{short}} \cup \mathbf{C_{med}} * \mathbf{f_{med}} \cup \mathbf{C_{long}} * \mathbf{f_{long}}) \,. \qquad (2)$$

where Short, Medium, and Long denote the piecewise functions that compose each set, and C_{short}, C_{med}, and C_{long} representing the confidence for each set, respectively. The output, f is the numerical output representing the period for updating the application.

The experiments for the RSSApp and TwitterApp use feed traces from mashable.com, a technology news feed, and the EmailApp used a Gmail account to retrieve email messages. User interaction is simulated on three user models: Heavy User, Medium User, and Light User, which are derived from models presented by Zhang et al. in [23]. The delay time is measured from the time of the request to the time of the response.

Error! Reference source not found. summarizes the actual power consumption and the delay for different adaptation techniques: fuzzy inference, a simple decision-tree model, and a fixed update rate representing no adaptation. The decision tree model uses crisp thresholds for each state, defining the action to take as a list of conditionals.

Table 3. Comparison of adaptive techniques for periodic downloading applications

	Fuzzy Inference		**Decision Tree**		**Fixed**	
	Avg. Power (W)	***Delay (s)***	***Avg. Power (W)***	***Delay (s)***	***Avg. Power (W)***	***Delay (s)***
RSS App	0.345	9.94±3.0	0.390	10.80±2.6	0.658	6.34±2.3
Twitter App	0.289	7.33±2.4	0.330	9.40±1.9	0.780	5.9±1.0
Email App	0.510	8.11±0.4	0.500	14.00±2.4	0.800	10±1.1

The energy consumption in all three periodic applications is significantly reduced due to context-aware adjustment of both the frequency of the data retrieved and the level of data downloaded. However, longer *TimeStep* values and lower *levelOfInfo* increase the chance that users experience a delay in retrieving backend data or the level of data they require. Fuzzy logic demonstrates, on average, a 49% improvement in average application power reduction over no adaptation, although with 14% increase in retrieval delay. Additionally, fuzzy logic demonstrates, on average, 12% improvement in power reduction for the RSSApp and the TwitterApp, and a reduction in delay of 14.5%. For the EmailApp, the decision tree method performed slightly better in terms of power consumption (2%), though with 42% increase in delay. An analysis of the results shows that the decision tree's crisp boundaries allow it to spend more time in the longest *TimeStep*, whereas the smoother transitions of fuzzy inference result in more frequent intermediate *TimeSteps*. However, the slight power saving comes at a high delay cost. While the delay cost is a qualitative measure, fuzzy inference still provides an improvement over the decision tree model, with a significant improvement in delay.

We then evaluate the tradeoff between the time-delay needed to retrieve the necessary data and the energy cost for retrieving data. We test the three user models: light, medium, and heavy, where each model indicates the frequency of application usage based on time and day. In order to represent the usage pattern of a mobile user, we can expect a fixed average rate of use, with the average number varying between light, medium, and heavy. As each device usage instance is independent, one representation of usage patterns is a Poisson distribution, which has previously been published in [7]. Figure 3 depicts the tradeoff between power usage and the average delay that a user faces to get updated data for the different usage patterns. The average delay stabilizes near 5 seconds as usage increases, regardless of the increase in

UpdateTime. The result helps illustrate two ways to optimize the functionality of the power manager: utilizing learning algorithms to limit thresholds, and statically analyzing an application so that appropriate limits may be set to minimize power consumption.

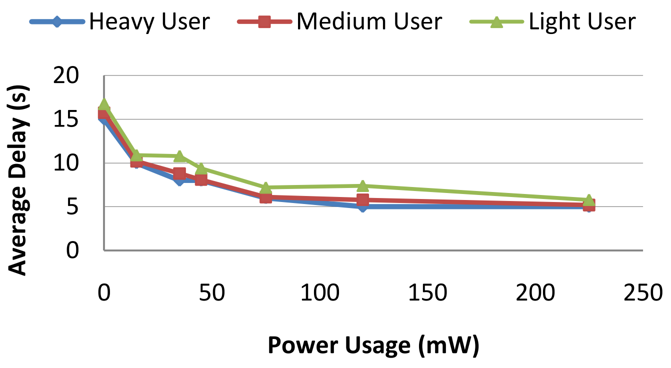

Fig. 4. The delay vs. power usage for periodically downloading applications.

3.3 Streaming Applications

Streaming applications are common in the mobile world, with the prevalence of services like Pandora and Google Music. We model our test application after a streaming music service, allowing streaming playback of a remote playlist. Similar to periodic applications, data retrieval with the backend service composes a majority of the energy consumption for streaming applications, although real-time playback precludes the use of batch information retrieval. Instead, we leveraged available context to vary the bitrate of data recovery, reducing the amount of data consumed by the running application over its lifetime.

In designing the context awareness of the application, we take advantage of the notion that a quiet environment enables reducing the quality of the streaming music, as the user is less likely to increase the volume and become exposed to distortion. Conversely, in when the environment is noisy, a higher quality stream mitigates distortion at higher volumes. Similarly, we perform a simple activity analysis from accelerometer readings. If the user is stationary, the quality of the stream is likely more important, as the music is a primary activity. If the user is walking or running, however, he or she is most likely listening to music as a secondary activity, and listening to music is a secondary activity.

As such, we define *AmbientNoise* = {*Quiet*, *Moderate*, *Loud*} and *Activity* = {*Stationary*, *Walking*, *Running*} as context variables, and *Bitrate* = {*Very Low*, *Low*, *Medium*, *High*, and *Very High*} as the output actions. The fuzzy context rules are described in the table below.

Table 4. *Bitrate* member sets for *AmbientNoise* and *Activity*.

	Stationary	**Walking**	**Running**
Quiet	Medium	Low	Very Low
Moderate	High	Medium	Low

Loud	Very High	High	Medium

The experiments used accelerometer and microphone traces from a 1-hour period of activity which involved all three *Activity* types and all three *AmbientNoise* levels. Accelerometer amplitudes and directions were converted to abstracted activity contexts using a variation of the algorithms presented in [22], simplified to reflect only walking, running, and remaining stationary. Microphone input was translated to quiet, moderate, and loud using a training set to capture minimum and maximum volume levels, and dividing the range into three equal sets. A separate program is used to stitch together a variable-bitrate file based on the traces and Equation 3 below. The resulting file is then streamed to the phone for the experiment. The defuzzification of the output settings is defined by the linear combination of confidence of settings associated with each member:

$$\begin{aligned} f = f_{med} * \frac{(C_{stat} + C_{quiet}) + (C_{walk} + C_{mod}) + (C_{run} + C_{loud})}{6} \\ + \mathbf{f_{high}} * \frac{(\mathbf{C_{stat}} + \mathbf{C_{mod}}) + (\mathbf{C_{walk}} + \mathbf{C_{loud}})}{\mathbf{4}} \\ + \mathbf{f_{low}} * \frac{(\mathbf{C_{walk}} + \mathbf{C_{quiet}}) + (\mathbf{C_{run}} + \mathbf{C_{mod}})}{\mathbf{4}} \\ + \mathbf{f_{v.low}} * \frac{(\mathbf{C_{run}}+\mathbf{C_{quiet}})}{\mathbf{2}} + \mathbf{f_{v.high}} * \frac{(\mathbf{C_{stat}}+\mathbf{C_{loud}})}{\mathbf{2}} . \end{aligned} \qquad \textbf{(3)}$$

where the $f_{v.low} = 128\text{kbps}$, $f_{low} = 256\text{kbps}$, $f_{med} = 384\text{kbps}$, $f_{high} = 512\text{kbps}$, and $f_{v.high} = 640\text{kbps}$. The selected frequencies for each state were determined by linear growth from the lowest to the highest available frequencies {128,192,…,640}, and the resulting frequency is discretized to the closest member in the frequency range.

Table 5. Comparison of adaptive techniques for streaming applications.

	Fuzzy Inference		**Decision Tree**		**Fixed**	
	Avg. Power (W)	***Median bitrate (kbps)***	***Avg. Power (W)***	***Median bitrate (kbps)***	***Avg. Power (W)***	***Median bitrate (kbps)***
Streaming App	0.689	256	0.844	256	0.983	512

We again perform a comparison of fuzzy inference with both a decision tree model and a fixed (non-adaptive) model. The results in Table 5 show marked improvement, with fuzzy inference providing, on average, 29.9% improvement over a non-adaptive application and 18.4% improvement over a decision tree model. The improvement over the decision tree model is most striking, and is attributed to the more natural, intermediate transitions enabled by confidence values between states. The discrete sets of the decision tree force sudden, large transitions which do not take advantage of potential intermediate transitions between, for example, Stationary+Moderate and Walking+Quiet. As such, the decision tree spends a longer time in the higher-bitrate state and consumes on average more power. Figure 5 displays the results of one of the experiments, and helps illustrate the gradual vs. stark transitions between the fuzzy inference and decision tree methods for a subset of the streaming experiment. For example, the raw sensor data shows a steady increase in ambient noise between minutes 4 and 10. Both the decision tree and the fuzzy inference models react to this change by increasing the bitrate accordingly. However, the sharp, early increase in the decision tree model contrasts with the more gradual change of fuzzy logic for the same period. Additionally, at time t=5, the crisp threshold of the decision tree model jumps to 512kb, implying the need for a High bitrate. However, fuzzy inference demonstrates that there is actually not a need for a High bitrate due to there not being enough confidence for that state, choosing instead to gradually increase with increasing ambient noise until leveling off at 448kbps. Similarly, the decision tree model increases to 640kbps at t=11, though fuzzy inference demonstrates that a lowered confidence in the Very High state should actually limit the value to 576kbps. Both these cases exemplify the advantage of fuzzy logic in adapting to context and, more importantly, its ability to better reflect the more graduated changes that actually occur in the surrounding context.

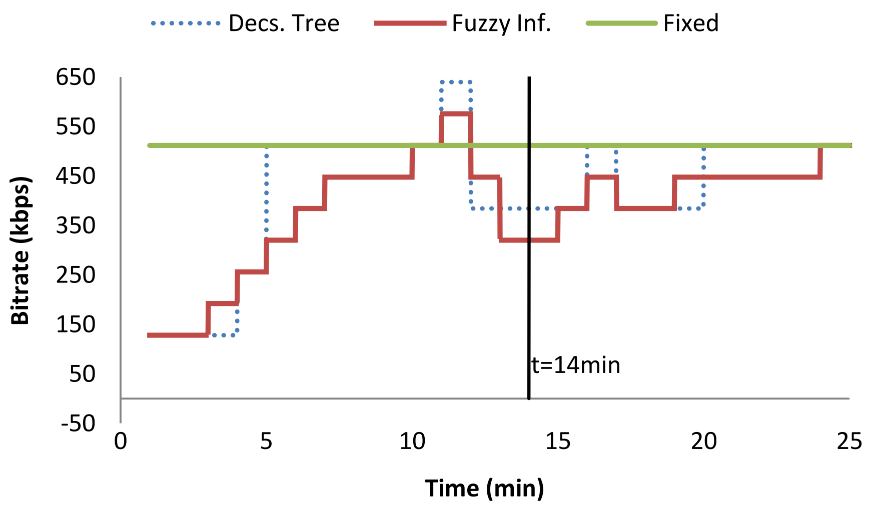

Fig. 5. Experimental bitrate results for Streaming Applications. At t=14, the decision tree model was streaming at 384kbps, the fuzzy logic model was streaming at 320kbps, compared with a fixed stream at 512kbps. Time t=4 to t=10 demonstrates the difference in response for decision tree compared to fuzzy logic in adapting to gradually increasing ambient noise levels.

4 Conclusion

In this paper, we introduce the idea of using context information to minimize energy usage of characteristic applications on mobile devices. We explore the background research in context awareness, and illustrate the advantages fuzzy inference of context provides to both mobile applications and in energy efficiency. We then propose and implement a context-aware power manager on mobile phones leveraging both raw and abstract context detection, and fuzzy inference to adapt typical mobile applications' behavior to current context. The proposed technique has been compared to similar method and showed 10 to 50% percent lower power consumption, as well as up to 18% improvement over traditional decision-tree models. Future work includes expanding the power manager with additional context variables and applications, as well as using learning techniques to dynamically determine higher-level context, as opposed to fixing them with static values. This has been utilized in related work for determining context [13][18], and allows abstract context variables to be adapted to the user.

References

1. N. Balasubramanian, A. Balasubramanian, Aruna and Venkataramani, "Energy Consumption in Mobile Phones: A measurement Study and Implications for Network Applications," in Proc. of 9th conf. on Internet Measurement, p.280-293, 2009.
2. C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D. Riboni, "A survey of context modelling and reasoning techniques," In Pervasive and Mobile Computing, Volume 6, Issue 2, p. 161-180, 2010
3. A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone". In Proc. USENIX, p. 21-21, 2010.
4. A. Ciaramella, M. Cimino, B. Lazzerini and F. Marcelloni, "Situation-Aware Mobile Service Recommendation with Fuzzy Logic and Semantic Web," In Proc. 9th Int. Conf. on Intelligent Systems Design and Applications, p.1037-1042, 2009.
5. G. Chen and D. Kotz, " Solar: An Open Platform for Context-Aware Mobile Application," In 1st Int. Conf. on Pervasive Computing, p.41-47, 2002.
6. H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan and D. Estrin, "Diversity in smartphone usage," In Proc. 8th Int. Conf. on Mobile systems, applications, and services, p.179-194, 2010.
7. S. Gunduz, M. Tamer Ozsu, "A Poisson Model for User Accesses to Web Pages," In ISCIS, p. 332-339, 2003.
8. J. Hong, E. Suh, S. Kim, "Context-aware systems: A literature review and classification," In Expert Systems with Applications, Volume 36, Issue 4, p.8509-8522, 2009.
9. International Electrotechnical Commission. "IEC 1131- Programmable Controllers Part 7 - Fuzzy Control Programming," 1997.
10. K. Kim, A. Min, D. Gupta, P. Mohapatra and J. Singh, "Improving Energy Efficiency of Wi-Fi Sensing on Smartphones," in Proc. of IEEE Int. Conf. on Computer Communications, 2011.
11. P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen and E. Malm, "Managing Context Information in Mobile Devices," In IEEE Pervasive Computing, vol. 2, no. 3, p.42-51, 2003
12. T. Lemlouma, N. Layaida, "Context-aware adaptation for mobile devices," In Proc. of IEEE Int. Conf. Mobile on Data Management, p. 106- 111, 2004.
13. K. Lin, A. Kansal, D. Lymberopoulos and F. Zhao, "Energy-Accuracy Trade-off for Continuous Mobile Device Location," In Proc. 8th Int. Conf. on Mobile Systems Applications, and Services, p.285-297, 2010.
14. M. Mahesh and M. Calder, "Batch Scheduling of Recurrent Applications for Energy Savings on Mobile Phones," In Sensor Mesh and Ad Hoc Communications and Networks, p.1-3, 2010.

15. J. Mäntyjärvi and T. Seppänen, "Adapting applications in handheld devices using fuzzy context information," In Interacting with Computers, vol. 15, no. 4, p.521-538, 2003.
16. J. Martínez, R.. John, D. Hissel, M. Péra, "A survey-based type-2 fuzzy logic system for energy management in hybrid electrical vehicles," In Information Sciences, vol. 190, no. 1, p.192-207, 2012.
17. M. Musolesi, M. Piraccini, K. Fodor, A. Corradi and A. Campbell, "Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones," In LNCS, vol. 6030, Pervasive Computing, P. 355-372, 2010.
18. J. Paek, J. Kim, and R. Govindan, " Energy-Efficient Rate-Adaptive GPS-based Positioning for Smartphones," In Proc. 8th Int. Conf. on Mobile Systems Applications, and Services, p.299-314, 2010
19. N. Ravi, J. Scott, H. Lu, L. Iftode, "Context-aware Battery Management for Mobile Phones," In Pervasive Computing and Communications, p.224-233, 2008.
20. A. Rahmati and L. Zhong, " Context-for-Wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer," In Proc. 8th Int. Conf. on Mobile Systems Applications, and Services, p.165-178, 2007.
21. A. Shye, B. Scholbrock and G. Memik, "Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures," In Proc. 42nd Int. Sym. on Microarchitecture, p.168-178, 2009.
22. G. Weiss, J. Kwapisz, S. Moore. "Activity Recognition using Cell Phone Accelerometers," In ACM SIGKDD Explorations, p. 74-82, 2010.
23. L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. Dick, Z. Mao and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," In Proc. 8th Int. Conf. on Hardware/software codesign and system synthesis, p.105-114, 2010.
24. Z. Zhuang, K. Kim, and J. Singh, "Improving Energy Efficiency of Location Sensing on Smartphones," In Proc. 8th Int. Conf. on Mobile Systems Applications, and Services, p.315-330, 2010.