ErdOS: Achieving Energy Savings in Mobile OS

Narseo Vallina-Rodriguez
Computer Lab
University of Cambridge
JJ Thomson Avenue
Cambridge, CB3 0FD
United Kingdom
Narseo.Vallina-Rodriguez@cl.cam.ac.uk

Jon Crowcroft
Computer Lab
University of Cambridge
JJ Thomson Avenue
Cambridge, CB3 0FD
United Kingdom
Jon.Crowcroft@cl.cam.ac.uk

ABSTRACT

The integration of multiple hardware components available in current smartphones improves their functionality but reduces their battery life to few hours of operation. Despite the positive improvements achieved by hardware and operating system vendors to make mobile platforms more energy eff cient at various levels, we believe that an eff cient power management in mobile devices is compromised by strict layering of the system caused by complex mobile business models that mitigates against cross-layering optimisations. However, there is a lot of room for improvement in the operating system. This paper presents ErdOS, a user-centered energy-aware operating system that extends the battery life of mobile handsets by managing resources proactively and by exploiting opportunistic access to resources in nearby devices using social connections between users.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Distributed Systems*

General Terms

Design

Keywords

Mobile computing, operating systems, energy-awareness, resources management, resources sharing, opportunistic computing

1. INTRODUCTION

Current mobile platforms integrate sensors such as GPS, several types of wireless interfaces, a giga-hertz range multicore CPU and a touchscreen. This trend bootstrapped the birth of rich mobile applications that, despite improving the usability of the device, can become energy sinks depending on the way users' interact with their handsets. In fact, the state of the art of lithium-ion batteries clearly indicates that capacity will still be constrained by design parameters such as battery size and weight for years to come.

MobiArch'11, June 28, 2011, Bethesda, Maryland, USA.

Both hardware and operating system manufacturers found positive and interesting approaches to extend the battery life of mobile handsets at different hardware and software levels. However, their efforts are limited by the strict layering of the system that makes diff cult to exploit cross-layer optimisations that might otherwise be fairly straightforward. As an example, and unlike with laptops, the operating system does not get direct access to information about aspects of the handset hardware such as telephony and radio hardware power consumption. The reason behind this limitation is a complex business ecosystem in which multiple players (e.g. cellular network providers, content providers, cloud service providers, hardware manufacturers and operating system vendors) compete to retain their share of the mobile business. New open platforms like Android and Nokia's Maemo offer new opportunities for improvement.

This paper presents ErdOS, a mobile operating system that exploits user-centered optimisations to extend the battery life of mobile handsets. We believe that one of the reasons behind mobile energy ineff ciency is that current operating systems do not naturally control access to energy-consuming resources to applications taking into account the patterns of users interactions with their handsets. Two techniques that ErdOS integrates to solve this problem are:

- A **proactive resources management system** that predicts the future resources demands and status based on the users' habits and preferences.
- **Opportunistic access to computing resources available in nearby devices** using local wireless interfaces and information about users' social networks (which can be obtained from online services, email accounts and address book) to provide access control policies.

The system attempts to make optimal use of all the resources available in the environment in a distributed fashion taking into account the situation, the users' preferences and both local and remote available resources. Our earlier work on understanding the impact of mobile users on resources demand [14] and the results we obtained by simulations (which are detailed later), clearly indicate that it is possible to achieve energy savings with those techniques without truncating user experience. However, a system like ErdOS opens new technical and research challenges such as fair scheduling algorithms for distributed resources in dynamic scenarios, energy-aware access control policies for sharing resources, adequate inter-process communication (IPC) mechanisms for accessing a diverse range of remote resources and f nally, non computational intense techniques for monitoring and forecasting resources demands and state. Nevertheless, we allow the user to decide whether to enable or not the automatic features of ErdOS. Some

users may prefer to get feedback from the system about future energy limitations or resources unavailability to adapt the way they interact with their devices rather than enabling automatic resources management in order to extend the battery life.

2. MOTIVATION

Past work reported in the literature proposed a re-examination of some aspects of operating systems design and implementation from an energy efficiency perspective, rather than the more traditional target of maximising performance [13]. Researchers have recognized the mismatch between the original design assumptions underlying the resource management mechanisms of operating systems and applications' behavior [4]. As we reported in our previous work [14], that incongruity is even more dramatic in mobile devices where simultaneous use of the diverse hardware systems in a modern multitasking smartphone arising from personal usage patterns can limit many handsets to just a few hours of operation.

In practice, a modern mobile operating system will attempt to extend the battery lifetime of the handset by making selective use of the available resources. However, experience shows that this is not efficient. Choices are most often implemented through the use of static policies and by standby power states, automatic control of the screen backlight, and actively switching particular subsystems (such as networking interfaces) on or off as demand dictates. They do not take account of the dynamism of the users' interaction with applications and the importance of context on resources availability (e.g. location impact on network coverage). As a result, it is quite possible for a power-hungry application to drastically shorten the operating time of the handset.

There are daily situations in which mobile applications can drain the battery due to inadequate management of resources. A common activity such as synchronising the email client is a good example. Performing this kind of action can be energy-expensive since it requires waking up the CPU and the radio interface when the handset is in idle mode followed by a DNS request and a connection with the email server. Such action happens regularly even in situations when its execution does not improve the usability of the system (e.g. at night when the user is sleeping) and in scenarios where it may well be known that there is no network coverage from previous experiences.

On the other hand, previous experiments on human mobility and social interaction using Bluetooth scans indicate that there are many opportunities for establishing opportunistic connections between devices [2]. Table 1 shows the power consumption of several embedded hardware devices in modern smartphones [1]. A mobile device can clearly save energy by accessing a resource like GPS remotely from a nearby handset rather than accessing the local GPS receiver (although the device sharing the resource sacrifices itself in the short-term in order to share its resources to others). Moreover, it is possible to achieve benefits in temporal terms. Performing a Bluetooth scan and connecting with a nearby device takes 11.5 seconds on average while retrieving the first position on the GPS receiver can take from 4 seconds to the order of minutes depending on the availability of the orbital data for the GPS satellites. The following section shows that it is possible to save up to 11% of energy in an extremely adversarial scenario such as a natural park in the Arctic Circle by sharing GPS reads with nearby devices. In this case of scenario, optimising energy consumption of mobile phones is paramount due to the limited access to power sources. Actually, the nodes of this case study present a high mobility (they are hiking) and the chances of encountering other devices are extremely low. Nevertheless, those results clearly indicate that it is possible to achieve significant energy savings in other scenarios (such as public places or urban and rural locations) with more stationary nodes and more chances of establishing opportunistic connections with nearby devices.

Energy consumption per hardware module		
Bluetooth	Near (30 cm)	36.0 mW
	Far (10 m)	44.9 mW
WiFi	Idle	8mW
	Full Capacity	720 mW
GSM	Idle	58mW
	Full Capacity	620 mW
GPS		143.1 mW

Table 1: Detailed power consumption on a modern smartphone (Openmoko Neo Freerunner)

All this type of evidence motivates our work on ErdOS; a mobile OS that exploits contextual information to manage resources proactively by learning from the usage patterns of the mobile user (or as it has been said, provides feedback to the users about their energy consumption) and also capable of providing transparent access to remote resources without impacting the user experience with mobile applications. We claim that an operating system that supports these features can also bootstrap many new collaborative applications that require accessing resources that span over several mobile devices such as collaborative sensing [8] or distributed computing [9].

3. PRELIMINARY RESULTS

In addition to our prior work on understanding users' interaction with mobile phone resources [14], we run a simulation to demonstrate the potential energy benefits of sharing GPS readings with nearby devices [1] (one of the use cases explained in the previous section). We used the dataset collected during one of the hikes that were part of ExtremeCom'09 workshop in Padjelanta National Park (Sweden) [15].

This adversarial environment where the hike took place lacks of any infrastructure such as power access points and network coverage. Padjelanta National Park is a completely desert land in the Arctic Circle in which retrieving location and saving energy in the mobile device is essential due to the lack of charging opportunities. The dataset contains Bluetooth scans and GPS readings of 17 conference attendees during a day. During this period, the devices successfully accessed 2832 times their GPS receivers and performed 3533 bluetooth scans. The participants were 11% of their time co-located [2] with at least another device during 151 seconds on average. However, since it was very unlikely to have them co-located during a long period because of the nodes' dynamism and social interaction between participants, the savings of accessing remote resources in this simulation are lower than the ones that can be obtained in other scenarios.

Figure 1 shows the estimated average energy savings and the system fairness achieved per device for different time thresholds (i.e. the maximum allowed time elapsed between the actions of reading the GPS receiver on a device and transmitting it over Bluetooth to a nearby one) compared to the case in which each node accesses the GPS receiver locally . We define fairness as the ratio between

[1]The simulation does not consider accessing remote GPS receivers on demand

[2]That is in the range of someone's Bluetooth radio. (10m. Aprox)

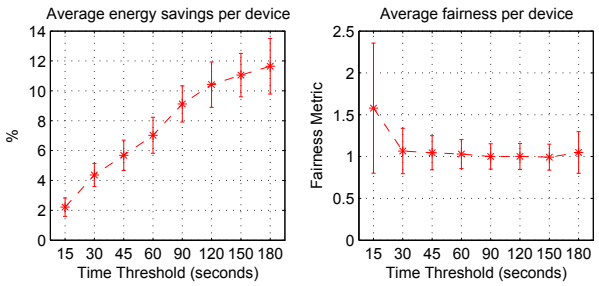

Figure 1: Relative average energy savings *(left)* and fairness *(right)* over all the subjects (95 % confidence interval) by enabling access to nearby GPS data from neighbouring devices if available over Bluetooth. The x axis shows the maximum allowed age of the remote information.

the number of times each device shares a resource with others and the number of times the device accesses a remote one. Values near 1 indicate that the system behaves more fairly since a device is accessing remote resources and sharing its local ones in the same ratio. On the other hand, values lower than 1 mean that the device behaves as a free-rider while higher ones indicate that the device is more altruistic. ErdOS aims to keep this value near 1 in the long term across any pair of devices sharing resources. As we can see, as the allowed age of the remote GPS data increases, the energy savings are higher because it makes available older information from more devices. However, the drawback of allowing older information is that it can increase the error in the location reading up to 100 m because of the distance walked by the node between the time the information was read from the GPS receiver and the time it was transmitted from the device sharing the resource to a nearby device.

It is important to highlight that this scenario is a very adversarial case from a social perspective. The users were hiking on a very isolated area and they had few social encounters with each other because they tended to cluster depending on their hiking speed and even on the conversation they had at the moment. In any case, the energy savings in such scenario indicate that we can achieve more promising results (we estimate that the energy savings can be up to 40% in the case of GPS sharing) in other social situations such as pubs, restaurants or even stadiums in which the social encounters (and therefore the energy savings) will be much higher and co-located for longer.

4. INTRODUCING ERDOS

This section describes the initial design considerations of ErdOS as an Android OS extension. We chose Android OS because it is a well documented open source platform with good development tools. Actually, we aim ErdOS to offer support to existing applications (they will be able to access remote resources transparently) so we can compare the performance and limitations of commercial applications running on both ErdOS and Android.

4.1 User-centered Resources Management.

Energy demands on mobile phones are caused by hardware components that might operate concurrently responding to applications executions, users' actions generated by social actions (e.g. an incoming email or phone call) and users' habits (e.g. charging the handset at home) [14]. This consideration makes possible profiling both resources state and applications' resources demands based on users' **activities** that can be inferred from contextual information.

The concept of activity is inspired by the real-time operating system Rialto [7]. In this case, it is defined as a logical set of application operations whose resources usage can be grouped together. However, an activity in ErdOS is defined as a set of specific locations and times in which a particular set of applications and resources are executed, accessed or modified. In fact, allocating and managing resources on mobile devices proactively is radically different to the traditional philosophy of breaking down the energy consumption of each application and hardware component to application activities [6] or threads [5]. Moreover, this approach is more flexible and efficient than algorithmic and rule-based resources management despite the implicit energy and computational overhead caused by monitoring and profiling both resources and user habits.

ErdOS learns from users' behaviour and infers their *activities* using time-based clustering algorithms with contextual information gathered from energy-passive information already available in the handset (e.g. base station identifiers can provide location information without incurring an additional energy-cost). ErdOS does not necessarily need to know the semantic meaning of activities but it must be capable of differentiating between stationary situations at defined locations and times and transitions between them. For example, ErdOS must identify when the user is at their workplace, at home or commuting between those places and also account for the resources state and demands at those scenarios. Stationary activities can be easily inferred when the user is subscribed to a reduced number of base stations for a long time while actions that involve mobility can be inferred from changes in the base station subscription.

ErdOS includes a context-aware fine-grained resources monitoring tool to know the resources usage and state both at the applications and system level. The applications executed, and therefore the resources and energy demands, might be different at each one of those activities as well as changes on the state of the resources like when the device is plugged-in to a power source. However, those actions are highly predictable [14]. ErdOS can forecast those situations and the resources demands of the users there, but it must be able to deal with uncertainty in order to identify new activities and changes in the users' patterns of usage (e.g. when a new application is installed in the device or the user stays at a new location). ErdOS classifies users' activities in a hierarchical tree: they span from high level activities known as **users' activities** (e.g. commuting to work) to second level ones like social interactions (e.g. performing a phone call), applications executed and users' actions (e.g. charging the handset).

Indeed, allocating resources to applications proactively based on contextual information and users' habits can save important amounts of energy. As there is no significant anonymous resource consumption, it is possible to forecast when a specific resource will be demanded by an application and when a resource will be available or not due to environmental constrains. All this information is gathered by intercepting and mapping all the system calls and context changes that can happen on the device. For instance, if the OS knows that at a given location the cellular network has a poor signal strength (and consequently, a higher energy consumption), ErdOS

can perform any of the following three actions depending on the user preferences: f rst, it can intercept and forward the system call to a remote device that might have this resource available (as we will describe in the following section); second, it can block the request to the resource from the application (in case accessing to the resource is very unlikely); or third, it can simply send feedback to the user about the possibility of not being able to access this resource and leave the f nal decision in their hands. Executing any of these actions will not cause any disturbance to the user experience and it will allow saving energy that might be otherwise wasted by the OS turning-on resources that might not be able to provide the desired service.

However, there is a clear trade-off between resolution and computational (and consequently, energy) overhead regarding f negrained monitoring of the resources, applications, users' context and users' interaction patterns with the mobile device. This issue requires further research which is currently being done. A periodic monitoring of the resources has been implemented in our previous work described in [14] and takes a maximum power of 60 mW when sampling (mostly consumed by the CPU). Nevertheless, we believe that an event-based monitoring tool can be more energy eff cient as there are long periods of time in which the phone is totally idle [12]. This information (combined with contextual information) is used as inputs by ErdOS' forecasting algorithms to identify future resources' state and demands from a user-perspective. The algorithm output will trigger the appropriate energy-saving policy based on the already mentioned user-def ned preferences. However, as performing forecasting algorithms can be resource-demanding, off-loading this computation to the cloud (or even to nearby devices with enough resources if available) in order to reduce the energy consumption in the local handset is obviously a future research line. In fact, frameworks like Maui already demonstrated the energy benef ts of migrating computation to wall-powered machines in the cloud [3].

4.2 Accessing Remote Devices and Security

To support access to remote devices, ErdOS provides a robust decentralised naming system, a local wireless API for discovery of nearby resources, and a set of Inter-Process Communication (IPC) mechanisms adapted to different classes of resources. The ErdOS name system is an extension of Haggle's one [11] and it includes the users' social network graph (with the different ways of identifying an user like email and Bluetooth MAC addresses), the granted permissions to the available resources at each contact, the public keys to provide security and the set of IPC mechanisms required to access remote resources.

The values in Table 1 indicate that Bluetooth is (at the moment) the most suitable wireless interface for sharing resources opportunistically since it is highly deployed, it allows scanning nearby devices and it presents a relatively low power consumption compared to 802.11 standards. ErdOS uses a wide range of IPC mechanisms appropriate to the type of resource being shared. As an example, sensors such as GPS can be accessed by registering an interest in it by doing a RPC to the remote phone service once it is known that such a resource will be required by applications. The node installs an event handler on the remote device (i.e. a callback routine) and then receives multicast messages which clone the GPS message data from the remote phone using a common wireless interface to the nodes that are interested in this information. Other sensors and resources like networking interfaces and CPU require totally different approaches. Since a cluster of Erdos users are typically engaged in local communication, we do not need the full complexity of the Internet Protocol suite, and we can employ

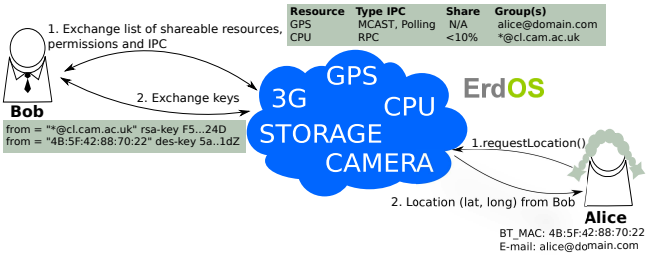

Figure 2: An scenario in which Bob joins a cluster of ErdOS devices and Alice performs a location.

extreme examples of cross-layer optimisation in IPC support. For example, given group keys (discussed later), location information can be securely broadcast on any radio currently in use. Indeed, in extreme we can simply piggyback many useful data including latest GPS readings, to the beacons employed in the base MAC protocol of many wireless standards. Only recipients possessing the right key can decrypt the associated data.

ErdOS is based on the hypothesis that existing social connections between users (based on their strength) can enable opportunistic resources sharing between users securely. However, since a common concern with *ad hoc* sharing resources between mobile devices is that of unauthorised access control by unknown neighbours. We propose leveraging the integration of *a priori* knowledge of the social group from the online social network, to provide strong identity (e.g. based on self-certifying cryptographically assigned identif ers derived from online social network names or e-mail names), and group keys for encryption of data for inter-process communication (whether RPC, unicast or multicast messaging, for example as with piggybacked personal identifying data to MAC beacons, mentioned in the previous section). In fact, since membership of friendship sets changes relatively slowly compared with actual physical encounters and use of shared resource, the cost of establishing these identif ers and keys is amortised over many actual uses, and can be carried out when in contact with infrastructure (cloud services) making use of computational resources there to derive the necessary keys from time to time (e.g. when re-keying a group after membership change). Nevertheless, it might be necessary to enable cooperation between unknown users and between users with weak social links. We are currently conducting research in this area in order to understand the potential trade-offs between optimising energy consumption and keeping users' privacy.

Figure 2 illustrates two examples of the multiple situations that could happen when at least two devices are part of an ErdOS cluster of resources. As we can see, when Bob arrives to an ErdOS cluster (or encounters a single ErdOS device), it exchanges the list of resources that he can share to each individual node that is part of the cluster (based on the user permissions) and the respective public keys. This exchange can be triggered by resources demand forecasts, application demands or by changes of context like arriving to new location. All the information exchanged during the handshake period is added to the name graph of the nearby nodes so they know what kind of resources Bob can provide to them. As an example, Bob granted Alice permission to access his GPS information. In case of any of Alice's applications requests that specif c resource, she will try to connect with Bob or any other device able to provide her an specif c resource, trying to maintain the fairness metric around one with each user in order to keep a fair collaboration between nodes in the long term. As fair allocation dictates that one battery and resources must not be drained and used in prefer-

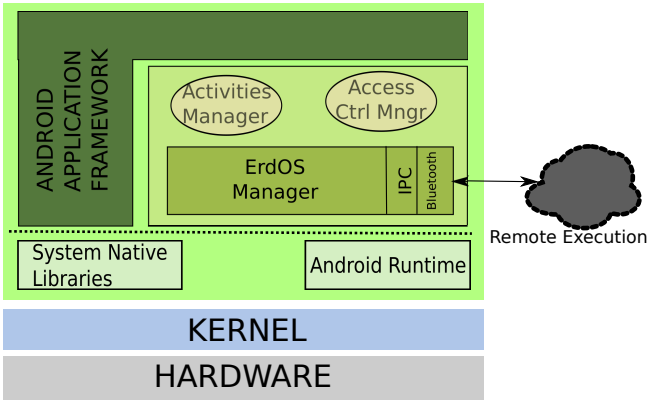

Figure 3: ErdOS architecture as an Android OS extension

ence to others, access control is def ned based on the availability of enough local resources to satisfy future own needs. As a consequence, devices can block access to other users that have a fairness metric above an specif c threshold. This approach does not make privacy a very relevant issue since users will share their resources with known nodes and groups that they def ne. Moreover the fact that only neighbouring nodes can access someone's resources limits considerably the impact of potential attacks.

4.3 Architecture

Most of the modif cations required to enable the features required by ErdOS can be done at the user space. However, we also need to modify Android's framework to intercept the system calls to monitor resources usage by applications and changes in contextual data, while also enabling communication between Android components and ErdOS ones:

- Activities Manager. Monitors and prof les *users' activities*, resources usage and their state by polling them and by intercepting system calls and changes in contextual information from the Android Runtime and the Android Application Framework. It predicts the resources demands in the device using machine learning techniques and triggers events to perform actions like discovering nearby devices, switching off hardware components, accessing nearby resources or providing feedback to the user.
- Access Control Manager. Contains the name graph with the users' identif ers, the social network graph, the access control policies, the public keys and metadata about resources' availability in remote devices with the appropriate IPC mechanisms required to access remote resources.
- ErdOS Manager. It is the central component that is responsible for managing local resources based on forecasts of their state and the applications' demands. It also discovers nearby devices and decides how, in which handset and when to access an specif c resource demanded by applications. It advertises and grants access to local resources required by nearby devices.

5. RELATED WORK

Researchers have been claiming for long that energy is a key resource for operating systems to manage [13]. As far as we know and despite the indications about achievable energy savings in mobile devices with an OS like ErdOS, there has not been any attempt on trying to build a system capable of both sharing and proactively manage resources for this purpose. Pre-eminent works on energy-aware operating systems are ECOSystem [16], Odyssey/PowerScope [5] and Quanto [6]. Those projects aim to alleviate the energy problem by enabling the OS to allocate energy to particular processes or by leveraging the collaboration between applications and operating system to enable applications' adaptivity to resources. However, since these techniques were mainly designed for laptops or embedded systems, their performance is compromised when they are applied to mobile devices with complex user interaction patterns. Cinder [10] is one of the few attempts mad to build an energy-aware mobile OS. Its design is inspired on EcoSystem and Quanto. However, Cinder's approach is not the most suitable for mobile handsets since it does not take into account the dynamics and interdependencies already mentioned. The sampling and energy prof ling techniques, the fact that it uses an inaccurate indicator such as the battery discharging rate to measure the energy consumption, also combined with a rigid and non-adaptive management, leads to ineff ciency in systems in which more than 90 processes can run simultaneously. These are some of the problems that ErdOS aims to tackle by managing resources and applications based on historical information about their state and their usage. This approach is customised for each users' habits and preferences, taking advantage of resources that can be available in co-located devices as in collaborative systems like Crowd Computing [9] and Darwin Phones [8].

6. CONCLUSIONS AND FUTURE WORK

We have presented the motivation and design considerations behind ErdOS, a user-centered and energy-aware OS that manages resources pro-actively aided by contextual information while it allows access to remote resources in co-located devices opportunistically both to save energy and to improve the usability of the device. ErdOS exploits social networks information and provides optimisations of IPC mechanisms to provide access control to nearby devices. We believe that this positioning paper made a strong claim showing the potential benef ts of ErdOS as well as the interesting research questions that it opens. The preliminary results obtained by simulations on a adversary scenario where battery charging opportunities are limited, clearly demonstrate that it is possible to reduce energy consumption by sharing resources opportunistically.

An ErdOS prototype is currently being implemented as an extension of the Android OS. Consequently, this initial design supports the execution of existing Android applications. The system implementation will be followed by a detailed performance analysis to accurately evaluate its benef ts and limitations such as its computational and energy overhead (mainly caused by prof ling and forecasting resources usage and state), the potential security issues, fairness and scalability. We will demonstrate the individual benef ts and trade-offs in different use cases such as sharing cellular networks, sharing sensors like GPS and also process migration to nearby devices and also to the cloud. Our initial hypothesis claims that social connections between users can be used to encourage them to share their computing resources securely opportunistically with nearby devices. Consequently, we have excluded support for incentive mechanisms to enable collaboration between unknown users (a possibility can be by building pricing models for resources that can have an inherent monetary cost like cellular networks). We believe that if users notice a personal benef t in terms of energy savings and usability by sharing their resources in the long-term with subjects they personally know and trust, there is no need to implement potentially complex or costly incentive schemes to enforce cooperation (e.g. a strategy proof, unforgeable, decentralised

currency or even just a bittorrent style tit-for-tat scheme, which would not address frequent disconnection or short term asymmetry of use). In fact, incentive protocols present an implicit energy and computation overhead caused by the additional data transmissions and computation they require. Nevertheless, in case those features are required after evaluating users' concerns about sharing resources with that initial prototype, they will be implemented as additional ErdOS modules.

7. REFERENCES

[1] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *USENIX Annual Technical Conference*, 2010.

[2] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 2007.

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code off oad. In *ACM MobiSys*, 2010.

[4] C. S. Ellis. The case for higher-level power management. In *HotOS*, 1999.

[5] J. Flinn and M. Satyanarayanan. PowerScope: A Tool for Prof ling the Energy Usage of Mobile Applications. In *WMCSA*, 1999.

[6] R. Fonseca, P. Dutta, P. Levis, I. Stoica, C. Berkeley, and C. Stanford. Quanto: Tracking energy in networked embedded systems. In *OSDI*, 2008.

[7] M. B. Jones, D. L. McCulley, A. Forin, P. J. Leach, D. Roşu, and D. L. Roberts. An overview of the Rialto real-time architecture. In *ACM SIGOPS*, 1996.

[8] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *ACM MobiSys*, 2010.

[9] D. Murray, E. Yoneki, J. Crowcroft, and S. Hand. Crowd computing. In *ACM MobiHeld*, 2010.

[10] S. M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich. Apprehending joule thieves with Cinder. In *ACM MobiHeld*, 2009.

[11] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: a Networking Architecture Designed Around Mobile Users. In *WONS*, 2006.

[12] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. Livelab: measuring wireless networks and smartphone users in the f eld. *ACM SIGMETRICS*, 2011.

[13] A. Vahdat, A. Lebeck, and C. S. Ellis. Every joule is precious: the case for revisiting operating system design for energy eff ciency. In *ACM SIGOPS*, 2000.

[14] N. Vallina-Rodriguez, P. Hui, J. Crowcroft, and A. Rice. Exhausting battery statistics: understanding the energy demands on mobile handsets. In *ACM MobiHeld*, 2010.

[15] E. Yoneki and F. Abdesslem. Finding a Data Blackhole in Bluetooth Scanning. In *ExtremeCom*, 2009.

[16] H. Zeng, C. Ellis, A. R. Lebeck, and A. Vahdat. ECOSystem: Managing energy as a f rst class operating system resource. *ACM ASPLOS*, 2002.