Agent Survivability Through Power Awareness

Maxim Peysakhov[α], Andrew Mroczkowski[α], Leonardo Urbano[α], Jacob Warren[α],
Vincent A. Cicirello[β], William Regli[α] and Moshe Kam[α]
[α] Drexel University, Department of Computer Science, Philadelphia PA 19104
[β] The Richard Stockton College, Computer Science and Information Systems, Pomona, NJ 08240

Abstract

Wireless mobile ad-hoc networks (MANETS) of personal digital assistants (PDAs) is a growing field of study due to potential applications in environments that lack a functioning communications infrastructure. Mobile computing platforms such as PDAs and Tablet PCs offer software designers great challenges to the design of survivable software applications. One of these challenges pertains to limited power resources. Processor loads, large amounts of wireless network activity, and input/output (I/O operations can strain the limited battery life inherent with such mobile platforms. In this paper we demonstrate how power awareness of software agents can increase system robustness and survivability. This paper presents first steps in a direction that can derive large payoffs for decentralized coordination and collaboration on power-constrained computing platforms.

1. Introduction

Wireless mobile ad hoc networks (MANETS) of personal digital assistants (PDAs) is a growing field of study due to potential applications in environments that lack functioning communications infrastructure. An example of such environments is the military battlefield, resulting in significant interest in MANETs by the United States Army (e.g., US Army Land Warrior program [7]). Since the availability of a MANET requires that the PDAs be continually powered, it is critical that the battery profile and power consumption be optimized.

Power consumption from the batteries of PDAs in MANETs affects application run-time and performance, wireless connectivity and device portability. Computationally intensive programs tend to consume more battery power than those which are less computationally intensive [10]. The effective range and quality of wireless connectivity will degrade as battery levels drop [6]. At the present state of battery technology, small increments in stored energy (i.e., bigger batteries) tend to come at large costs in weight [4, 24].

Battery life is among the most serious constraints in actual deployed MANETs, such as the Commander's Digital Assistant of the United States Army Land Warrior program [7]. While there is some literature on PDA battery depletion under varying processor loads [10], little has been reported on the discharge characteristics of PDA batteries under common processor and hardware loads found in typical MANETs. In MANETs, it is often desired that PDAs perform mathematically intensive cryptographic algorithms to secure transmissions between units [8] and that they employ the most powerful wireless ethernet adapters to communicate at the maximum possible range. Often, these PDAs have accessories plugged into them such as global positioning system (GPS) receivers, short-range Bluetooth wireless adapters, headphones and speakers for audio adapters and digital cameras.

MANET connectivity and overall stability is significantly related to battery power. In PDA MANETs that employ ad hoc routing algorithms, wireless signal strength between nearest neighbors is sometimes used to compute optimal routes between nodes. As a consequence, sudden Byzantine network failures can result from low battery power of even a single node. Detecting low battery and executing contingencies based on such warnings can improve MANET stability (i.e., continuous connectivity).

The classical definition of agent as entity capable of perceiving and affecting its environment is given by Russell and Norvig [19]. In a traditional wired network, agents may need to perceive the actions of system users and other agents. They may have to take into account organizational, economical and legal constraints placed on their choices and other application level information. However, agents deployed over a MANET may also need to pay closer attention to the low level details of their executing environment. As discussed elsewhere by the authors [5, 18], software agents operating on MANETs can benefit from low level

0-7803-9447-X/05/$20.00

knowledge of networking state information. Another critical and severely limited resource in mobile computing environments is power. In this paper, we advocate that giving agents detailed knowledge of the state of their power supply can significantly improve the survivability of the system, increase its deployment time and make it more beneficial to the end user of the system.

Most of the modern operating systems designed for battery powered computing devices are power aware to some degree. Most of them are capable of warning the user that the battery level is approaching critical and/or gracefully shutting down the system to avoid the loss of any local data. In this paper, we consider much higher levels of power awareness, where agents are given information about current state, and power consumption predictions of local as well as remote hosts and can consider such actions as migrating to another host, delegating tasks to agents on other hosts or refusing to accept the tasks that will drain power levels below some critical threshold. Embodied agents (agents implemented in hardware) can have even more options. They can position themselves stationary to serve as sensors, network routers, information repositories, or for other tasks that are not as power demanding as tasks requiring physical motion [20]. Power should also be taken into consideration in the mission planning phase (i.e., if several agents are capable of performing a potentially hazardous task, everything else being equal the agent with the lower power reserves should perform the task since it has lower utility for the system). Although the general approach is equally applicable to hardware and software agents, the current paper will concentrate only on software agents.

The remainder of this paper is organized as follows: Section 2 presents work in the areas of power aware computing and networking as well as several other related fields. Section 3 formalizes the problem. Section 4 provides experimental results. Finally, in Section 5 the paper concludes with a discussion of future work.

2. Related Work

Bhardwaj et al define a system as perfectly power-aware if its energy dissipation in a scenario s is no greater than that of a system designed to execute scenario s as efficiently as possible [3]. From this, they define the power-awareness of a system relative to the perfectly power-aware system. Power-awareness is an important concern to the design of energy efficient systems. For example, researchers have considered power-awareness for field programmable gate arrays (FPGA) [11], embedded systems [14], power-aware scheduling [16], communications systems [21], memory page allocation [12], low-power CMOS VLSI [25], among others.

Most relevant to our work is power-awareness for MANET environments. Much of what has been done in this area is focused around power-aware routing. The network interface of the handheld devices commonly a part of a MANET is the greatest consumer of energy. There are several actions one can take to conserve energy in MANET environments. For example, a few relatively simple actions can include reducing the brightness of the backlight or switching to a sleep mode. However, a more sophisticated option is to optimize the network protocol design—e.g., optimize the network protocol to minimize unnecessary retransmissions, as well as to minimize the transmission of network state updates. This strategy is referred to as power-aware routing. Power-aware routing adds information about power to the route updates and avoids using nodes with low battery power to route data. Network information (e.g., routes and topology information) are shared among the nodes of the MANET only when necessary. Some examples of power-aware routing protocols include [17, 2, 22, 15, 13, 1]. For example, the COMPOW protocol [17] selects a common power level for the nodes of the network that is the minimum power level at which the network is fully connected. Reducing transmissions power in this way increases network capacity, improves throughput, and decreases interference.

3. Problem Formulation

3.1. Formal Model

The problem can be formalized as follows. Consider a set of renewable resources $R = \{r_1, r_2, \ldots r_m\}$ and a set of non-renewable resources $Q = \{q_1, q_2, \ldots q_p\}$. An example of a renewable resource is a CPU required for computations or a network interface required to transmit data. Each renewable resource r_i has a capacity $C(r_i)$ indicating how much of the resource is available at any given moment in time. The capacity of a renewable resource (for the remainder of this paper) is assumed constant across the problem instance. For a network interface this capacity might be the available bandwidth. An example of a non-renewable resource might be the power level of the battery of a handheld device (if we assume that once a battery is drained the device no longer functions in our scenario). Each non-renewable resource q_i has a capacity $C(q_i)$. For a non-renewable resource, the capacity changes over time. For example, a battery's power level will decay over time depending on the activity of the handheld device.

Now, consider a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$. Each task t_i requires one of $|A_{i,j}|$ alternative sets of resources. That is, there may be more than one way of accomplishing a task t_i. For example, in our MANET environment, we

may have a task that involves performing some computations. One alternative might be to perform those computations on device d_1 requiring d_1's CPU and battery. A second alternative might be to transmit the problem to d_2 and perform those computations there, requiring d_1's network interface and battery and d_2's CPU and battery. Let's define the j-th alternative resource set for task t_i as $A_{i,j} = \{r_{i,j,1}, r_{i,j,2}, \ldots, r_{i,j,k}, q_{i,j,1}, q_{i,j,2}, \ldots, q_{i,j,l}\}$. Let $|r_{i,j,k}|$ indicate the quantity of the renewable resource $r_{i,j,k}$ required by task t_i for alternative resource allocation $A_{i,j}$. And similarly, let $|q_{i,j,l}|$ indicate the quantity of the non-renewable resource $q_{i,j,l}$ required by task t_i for alternative resource allocation $A_{i,j}$

Tasks can be executed simultaneously provided that there exists an assignment of alternative resource sets to the set of tasks, $\{(t_1, A_{1,j_1}), (t_2, A_{2,j_2}), \ldots, (t_n, A_{n,j_n})\}$, such that the following constraints are satisfied. For each resource $r \in R$ and each non-renewable resource $q \in Q$, $\sum_{r_{i,j_i,k}=r} |r_{i,j_i,k}| \leq C(r)$ and $\sum_{q_{i,j_i,k}=q} |q_{i,j_i,k}| \leq C(q)$. Furthermore, upon the execution of a task t_i, the capacities of all non-renewable resources required by the task are decayed $C(q) = C(q) - \sum_{q_{i,j_i,k}=q} |q_{i,j_i,k}|$.

Given this formulation, one can envision different types of optimization objectives in this resource allocation problem that are resource-aware (especially in terms of non-renewable resources). We are particularly interested in power-aware scenarios where we focus on power-related non-renewable resources such as the power level of the batteries of handheld computing devices. For example, one might consider the objective of minimizing the global power consumption of the system in performing the set of tasks (or maximizing the remaining power):

$$\max \sum_{q \in Q} C(q), \tag{1}$$

where the $q \in Q$ are the batteries of the handheld devices. The drawback to such an optimization objective is that it does not consider any potential relative importance of the devices. For example, we can consider a scenario where only a single device is "alive" at the completion of the scenario with a power level $C(q_1)$. But perhaps it would have been preferable if instead a different but more critical handheld was still alive at the end such that $C(q_2) < C(q_1)$. Perhaps the network interface of the handheld with battery q_1 has malfunctioned and the device is useless for communications purposes and further consider that the users of these two devices are trapped together and need to call for help (e.g., they may be firefighters who have become trapped when a part of a corridor has collapsed). Therefore, it would be a more desirable result in this situation if the device with functioning communications resources had sufficient power level to operate.

Let us instead develop a device-aware objective function. Define a set of handheld devices as $H = \{h_1, h_2, \ldots, h_j\}$. Each $h \in H$ is comprised of a set of renewable resources and non-renewable resources. For simplicity, lets consider a single non-renewable resource for each (i.e., its battery) and refer to that non-renewable resource as $q(h_j)$. Each device is assigned a value that indicates the level of importance for the availability of that device: $V(h_j)$. This value can increase or decrease over time. For example, perhaps a user has acquired critical information for the success of the mission that is now stored on her handheld—this should increase the value of the survival of this device. Or perhaps, the network interface of a particular device has malfunctioned and it can no longer usefully serve as a communications router—the value of this device should be decreased. Now, let us define a survivability optimization objective as:

$$\max \sum_{h \in H, C(q(h)) > \tau} V(h), \tag{2}$$

where τ is the minimum required remaining capacity of the battery of device h for h to be considered "available". That is, if the battery level drops below τ, information, or services, on that device can no longer be accessed.

In order to optimize this objective function, one must choose from among the alternative resource allocations for each of the tasks that both maximizes the values $V(h)$ as well as minimizes power consumption, particularly for high-valued devices h.

3.2. Data Acquisition

In performing a study of the potential benefits that can be obtained from instilling in software agents the ability to be power-aware, it was necessary to begin by studying the power profiles for different battery configurations on handheld devices.

A series of experiments was conducted with four different loads on iPaqs with two different battery configurations designed to stress the PDA in different ways. Each test is repeated three times to establish an average and eliminate outliers. We chose to use one year old heavily used internal and backpack batteries for the "low power" configuration and an upgraded backpack battery with the PGF battery and the external PR2 battery as our "enhanced battery" configurations.

In all of the experiments in this section (unless stated otherwise) the LCD screen brightness was set to maximum and a Cisco Aironet wireless ethernet card was plugged into the backpack.

3.2.1. Benchmarking tools In order to emulate different load types, we installed the MiBench benchmark suite [9] on several test iPaqs and measured the battery life while the iPaq performed benchmark tests. The MiBench suite has

over 2 dozen benchmarking programs written to create different loads on the ARM processor by executing a variety of tasks, including memory loading and storing, input/output operations, conditional and unconditional branching and floating point and integer math, to name a few (see Figure 1 in [9]). The MiBench suite was used in [10] to measure program power consumption of the iPaq 3670 under a variety of benchmark loads.

In our experiment, the four loads considered are:

Load 1 - CPU. Intense number-crunching using the "basicmath" and "qsort" benchmarks (two of the most CPU intensive benchmarks in the MiBench suite [9]). Basicmath performs mathematical calculations such as cubic function solving, integer square root, and angle conversions from degrees to radians. The input data is a fixed set of constants included with MiBench. Qsort sorts a large array of strings into ascending order using the well known quick sort algorithm. The data is a set of three-tuples representing points of data.

Load 2 - Network. Intense network card activity using a *ping flood*. A *ping flood* sends messages at a rate equal to the greater of that which sends a message whenever the previous has returned or one hundred times a second. This emulates high data throughput typical in MANETs.

Load 3 - Input/Output. Input-output functions using the "tiff2rgba" benchmark. This benchmark performs numerous load/store operations with the built-in flash memory on the iPaq by converting a color image in the TIFF format into an RGB color formatted TIFF image. Out of all of the benchmarks available in the MiBench suite, load/store operations account for approximately 80% of the total instructions for the tiff2rgba benchmark.

Load 4 - Best Case. No loads (i.e., idle).

3.2.2. Procedure The iPaq's internal stock battery voltage (measured in millivolts) was logged and timestamped every five seconds. Voltage is sampled only from the internal stock battery of the iPaq since any additional batteries will deplete before the stock battery depletes. Therefore, the stock battery voltage is the best measure of remaining power.

Prior to each trial, the iPaq charges through the backpack via the wall transformer until both the iPaq and expansion pack read 100%. Then, depending upon the test, a benchmark from the MiBench suite is started and the LCD backlight set to maximum brightness. Finally, the unit is unplugged from the charger and data logging begins. All data is logged from the time the unit is disconnected from the charger until the battery expires and the unit's PMU shuts the iPaq off.

3.2.3. Results

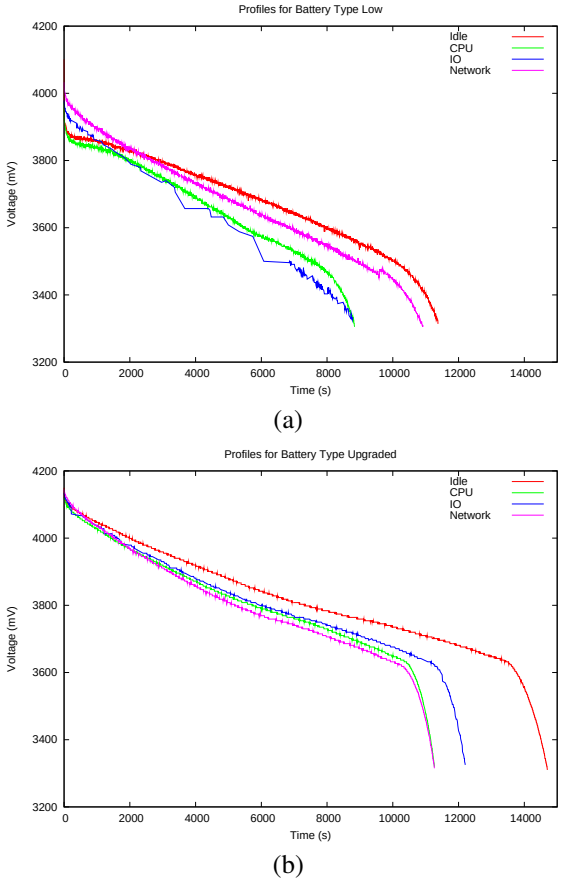

Figure 1. Discharge profiles for (a) the low type battery under different loads; and (b) the upgraded battery under different loads.

Low power vs. Enhanced power configuration Figure 1 shows the voltage of the iPaq stock battery (measured in mV) while discharging in the low and upgraded configurations, respectively, under a variety of loads to better illustrate the effects of each load on power consumption. In each of the four loads, improved battery life due to the PGF is noticeable but clearly not the 300% improvement that is claimed by King Rex Technologies or that is suggested by the fact that the PGF battery has triple the capacity of the standard stock battery. See Table 1 for up-time details.

4. Empirical Validation

4.1. Motivating Example

We chose a video surveillance as a main driving application behind our simulated experiments. There are a number of hosts engaged in collecting image data, analyzing the images and storing them. Each of the hosts is equipped with a

	Idle	CPU	I/O	Net
Low	3.1hrs	2.2hrs	2.5hrs	2.8hrs
Upgraded	3.8hrs	3.1hrs	3.3hrs	3.1hrs

Table 1. Summary of approximate iPaq battery life with two (2) different battery configurations under Idle, CPU, I/O, and Network loads.

motion detector that has a probability $\frac{1}{4}$ to trigger the host's camera. Roughly half of the images are discarded immediately. The other half are input to a CPU intensive image analyzer. After the analysis another 50% of the images are eliminated. The remainder of the images can be stored anywhere on the network. Each of the actions puts the host into an appropriate state according to Figure 2(a) with corresponding discharge profile. The state corresponding to the saving of the image is a compound state. In this state agents are given a choice to store the image locally with idle discharge penalty or to forward the image to the another place on the network with the network discharge penalty. The power cost for each operation is computed from linear approximations of the data described in Section 3.2. If the voltage reaches a certain level (3500mV in most experiments) the device is considered to be out of operation. It can no longer take, process, or store images, nor is it able to serve as a networked host. Images on these devices are considered irretrievable.

The characteristics that the systems are compared on are the number of hosts in operation and the number of retrievable images in the system.

4.2. System Setup

4.2.1. Base-line Experiment To establish a comparison base line we performed a set of experiments with power unaware agents. A power unaware agent takes a picture every time it senses a stimuli (approximately every 4 iterations or seconds). It always chooses to process the image locally and to store it locally if space is available. If no storage space is left, the host transmits the image for storage to any host that has storage space.

In all of the experiments we used 6 hosts with low and 6 hosts with upgraded batteries. Hosts are also given different amounts of storage space: 64 MB—corresponding to the internal on-board memory of the device, 256MB—external compact flash (CF) card, and 5120MB—integrated hard drive (HD). The host composition for the experiment is given in Table 2.

4.2.2. Power-Aware Experiment One of the few differences between power aware and unaware agents is that the former has knowledge of the current state of the battery for every host on the network. Armed with this knowledge they

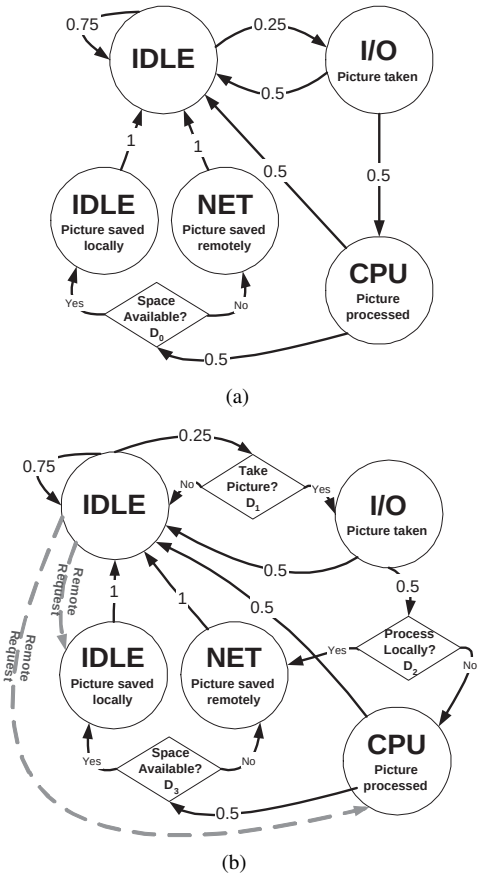

Figure 2. A state chart for simulated agents: power unaware (a) and power aware (b)

	64MB	256MB	5120MB
Low	3	2	1
Upgraded	3	2	1

Table 2. Experiment hardware composition.

can make intelligent decisions on actions they are about to perform.

The first decision for power aware agent to make (Node "D_1" on the diagram in Figure 2(b)) is to decide if it should take a picture when the alarm is triggered. In these experiments the agent with the lowest battery level assumes it is has the greatest danger of running out of power first and refrains from taking the picture, thus saving some energy. By following this rule, the hope is to discharge all the hosts at uniform rates, keeping them all active on the network for as long as possible.

Second decision point "D_2" comes after the image is taken. Each agent makes a decision to either process the image locally or transmit it elsewhere for processing and storage. The image is processed locally only on the host with the highest battery charge if and only if doing so will not affect its rank as the highest powered host. Otherwise the image is transmitted out for processing. The image is transmitted to the host with the highest level of power if processing this image on that host and then sending it someplace else for storage will preserve its power rank. Otherwise the image is sent to the host with available storage space with the highest level of remaining power.

Final decision point is "D_3" At this point the agent on the host can decide to either store the image locally or transmit it elsewhere for storage. The agent always stores it locally if space is available. Otherwise the image is shipped to the host with he highest remaining battery power and available hard drive space. The idea behind this rule is to store the images on the hosts that will stay active the longest.

If the decision is made to transmit the image to another host for storage and/or processing, that host has to honor the request. The rest of the operation of the power aware agents is identical to the behavior of unaware agents.

4.3. Experiment Analysis

A set of 100 simulations were conducted for power aware and power unaware agents. The observations were made at each iteration of the simulator. The values were averaged across all of the experiments and curves for both types of agents superimposed for easy comparison (Figure 3). The performance of power unaware agents is plotted with dark blue color while the performance of power aware agents is plotted in light green color. Dashed lines represent the standard deviation of the observed parameter. We analyze the systems' performance separately in terms of number of available hosts (Figure 3(a)), images that are accessible (3(b)) and total images processed (3(c)) in the following subsections. The performance of power aware and unaware agents are compared for each category.

4.3.1. Number of Hosts in Operation The plot in Figure 3(a) represents the average number of hosts that are in operation for any given moment in the simulation. The dashed line at the bottom of the plot represents a standard deviation and remains at zero for most of the run indicating high consistency across runs of the experiment. Short spikes in deviation correspond to brief periods of time where the subsets of hosts were running out of battery power. The average time before all hosts are out of operation is 13496.59 for power unaware and 14134.55 for power aware agents. As one can see power awareness does not extend the life of the hosts significantly (only by about 5%). If we assume that one simulator time unit is equivalent to one second, then power aware agents extend the time until all units are dead only by approximately 10.6 minutes on average. This is understandable since actions that increase battery power are not available to the agents. The only thing they can do is to redistribute the load between existing hosts to achieve the task more efficiently.

4.3.2. Number of Accessible Images The behavior of the system is more significant in terms of accessible images (Figure 3(b)). For the power unaware system, the number of images grows linearly until all low powered hosts run out of battery power at roughly the same time. At this moment about half of the images are lost. Linear growth continues for a while, until terminated when the second batch of hosts run out of power. The curve peaks at 4858 accessible images at iteration 9150. The performance of the power unaware system is plotted with a blue line in Figure 3(b). The standard deviation for the power unaware agents is plotted with lighter dashed blue line and remains near zero during the entire run, except when a significant number of hosts run out of battery power at the same time.

Power aware agents also exhibit a near-linear growth in the number of accessible images but at a slower rate. The reason for this is twofold.

1. some agents decide not to take a picture to prolong the life of their hosts; and
2. agents spend more time transmitting images from one host to another, thus having less time to take an image (images are not taken unless the host is otherwise idle).

However, despite slower growth and higher variance, the power aware agents do not suffer *any* loss of data with the loss of half of the hosts (the low power hosts). The reason is that all of the images are saved on the hosts with higher energy reserves by the power aware agents' heuristics. The power aware agents result in a peak of 4941 accessible images (approximately 2% more accessible images) at simulator iteration 13919. That is, if we again assume that simulator iterations are equivalent to seconds, the power aware agents allow slightly more images to be accessible for a time period of over an hour and twenty minutes longer than the power unaware agents. More data is available to the system users for a longer period of time.

4.3.3. Number of Processed Pictures The plot in Figure 3(c) shows the total number of images processed by the system. The power aware system (green line) processes slightly less images than the power unaware system (blue line) for much of the run, due to the higher overhead and the ability of agents to choose to not take a picture. However, by the end of the run the power aware system not only catches up but actually processes more images than the power unaware system. The power aware system not only keeps im-

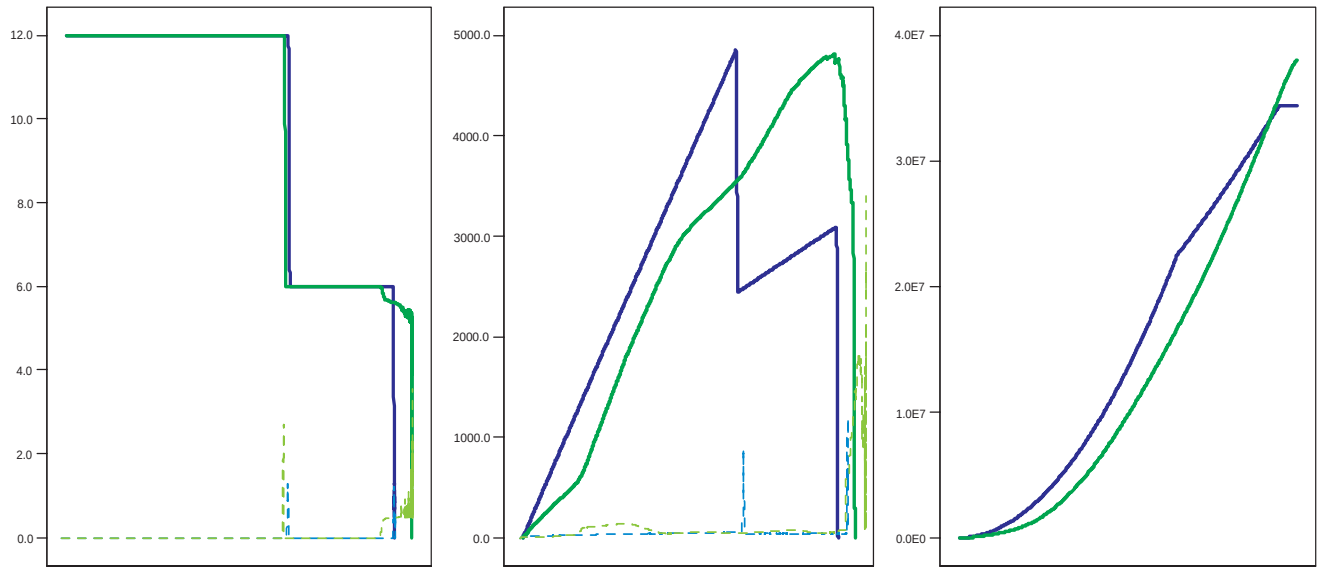

Figure 3. Average number and standard deviation of available hosts (a) and images (b). Cumulative sum of pictures taken during the run (c).

ages available on the live hosts longer, but also takes more images due to its extended run time.

5. Future Work and Conclusions

In this paper we demonstrated how power awareness of software agents can increase system robustness and survivability. This paper has presented first steps in a direction that can derive large payoffs for decentralized coordination and collaboration on power-constrained computing platforms.

In the future we are planning several extensions to this work:

- We are planning to use more accurate approximations of the battery discharge profiles.
- For now, agents were given a priori knowledge of the state of the battery and about the expected discharge profiles. In the future we are planning to employ one of the AI based classifying techniques to recognize the type of power supply and use this knowledge to select the appropriate profile.
- Currently power aware agents behave according to the set of rules defined by the designers at compile time. In the future we are planning to outfit the agents with a planner or reasoner so they can find beneficial behaviors independently during runtime.
- Experiments mentioned in this paper are performed on the static network with no routing. In reality, however, mobile devices are often composed of a MANET and necessitate routing. Taking the ad hoc nature of the networks into account will allow us to test more realistic scenarios.
- We are planning to implement the power aware agents and test them in a real life scenario utilizing the SWAT [23] and PA-UWNT [5] testing environments.

References

[1] S. Banerjee and A. Misra. Minimum energy paths for reliable communication in multi-hop wireless networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 146–156. ACM Press, 2002.

[2] P. Bergamo, A. Giovanardi, A. Travasoni, D. Maniezzo, G. Mazzini, and M. Zorzi. Distributed power control for energy efficient routing in ad hoc networks. *Wireless Networks*, 10(1):29–42, 2004.

[3] M. Bhardwaj, R. Min, and A. Chandrakasan. Power-aware systems. In *Proc. 34th Asilomar Conf. Signals, Systems and Computers*, pages 1695–1701, November 2000.

[4] T. Burd and R. Brodersen. Processor design for portable systems. *Journal of VLSI Signal Processing, 13(2/3).*, pages 203–222., 1996.

[5] V. Cicirello, M. Peysakhov, G. Anderson, G. Naik, K. Tsang, W. Regli, and M. Kam. Designing dependable agent systems for mobile wireless networks. *IEEE Intelligent Sys-*

tems, 19(5):23–29, September/October 2004. Special Issue on Dependable Agent Systems.

[6] T. ElBatt, S. V. Krishnamurthy, D. Connors, and S. K. Dao. Power management for throughput enhancement in wireless ad-hoc networks. In *Proceedings of IEEE International Conference on Communications.*, pages 1506–1513., June 2000.

[7] S. Erwin. "army to upgrade land warrior system with blueforce tracker.". *"National Defense Magazine, Arlington, Virginia."*, February 2004.

[8] G. Anderson et. al. A secure wireless agent-based testbed. In *Proceedings of the Second IEEE International Information Assurance Workshop (IWIA 2004)*, April 2004.

[9] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. *IEEE 4th Annual Workshop on Workload Characterization, Austin, TX.*, Dec. 2001.

[10] C. Krintz, Y. Wen, and R. Wolski. Predicting program power consumption. Last accessed 3/17/04., July 2002.

[11] J. Lamoureux and S. J. E. Wilton. On the interaction between power-aware fpga cad algorithms. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 701. IEEE Computer Society, 2003.

[12] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis. Power aware page allocation. *SIGPLAN Not.*, 35(11):105–116, 2000.

[13] Q. Li, J. Aslam, and D. Rus. Online power-aware routing in wireless ad-hoc networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 97–107. ACM Press, 2001.

[14] J. Liu, P. H. Chou, N. Bagherzadeh, and F. Kurdahi. Power-aware scheduling under timing constraints for mission-critical embedded systems. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 840–845. ACM Press, 2001.

[15] M. Maleki, K. Dantu, and M. Pedram. Power-aware source routing protocol for mobile ad hoc networks. In *ISLPED '02: Proceedings of the 2002 international symposium on Low power electronics and design*, pages 72–75. ACM Press, 2002.

[16] P. Mejia-Alvarez, E. Levner, and D. Mossé. Adaptive scheduling server for power-aware real-time tasks. *Trans. on Embedded Computing Sys.*, 3(2):284–306, 2004.

[17] S. Narayanaswamy, V. Kawadia, R. Sreenivas, and P. Kumar. Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the compow. In *European Wireless Conference*, 2002.

[18] M. Peysakhov, D. Artz, E. Sultanik, and W. C. Regli. Network awareness for mobile agents on ad hoc networks. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, July 2004.

[19] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 2002.

[20] J. Sauter, B. Matthews, A. Yinger, and V. Parunak. Design and evaluation of pheronome algorithms for swarms of autonomous surveillance vehicles. In *Proceedings of the Second Annual SWARM Conference*, June 2004.

[21] C. Schurgers, V. Raghunathan, and M. B. Srivastava. Power management for energy-aware communication systems. *Trans. on Embedded Computing Sys.*, 2(3):431–447, 2003.

[22] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190. ACM Press, 1998.

[23] E. Sultanik, D. Artz, G. Anderson, M. Kam, W. Regli, M. Peysakhov, J. Sevy, N. Belov, N. Morizio, and A. Mroczkowski. Secure mobile agents on ad hoc wireless networks. In *The Fifteenth Innovative Applications of Artificial Intelligence Conference*, pages 129–36. American Association for Artificial Intelligence, Aug 2003.

[24] S. Udani and J. Smith. Power management in mobile computing. *Technical report, University of Pennsylvania.*, MS-CIS-9826., 1998.

[25] K. won Choi and A. Chatterjee. Udsm (ultra-deep submicron)-aware post-layout power optimization for ultra low-power cmos vlsi. In *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, pages 72–77. ACM Press, 2003.