Energy consumption models for ad-hoc mobile terminals

Emmanuel Lochin[1] Anne Fladenmuller[1] Jean-Yves Moulin[1] Serge Fdida[1]
[1] LIP6 – Université Pierre et Marie Curie
4, place Jussieu
75252 Paris Cedex 05 - France
{emmanuel.lochin, anne.fladenmuller, jean-yves.moulin, serge.fdida}@lip6.fr

***Abstract*— This paper describes a set of experiments based on ACPI BIOS measurements which evaluate the energy consumption of an IEEE802.11 wireless network interface. Based on our ACPI measurements, two models of energy consumption are presented in section VI: (1) an analytical model for network simulator, (2) an empirical model for testbed implementation. The aim of these studies is to provide an environnement both for simulation and implementation. This work is the first step to propose a new power-aware routing protocol for mobile ad-hoc networks.**

I. Introduction

A MANET [1] network [2] consists of a set of mobile hosts/routers connected together with wireless links and working as a network with no dedicated routing infrastructure nor centralized administration. Besides obvious military applications, ad-hoc networks may be used in any environment requiring a rapidly deployable wireless infrastructure. In MANET, a mobile host is required to forward packets between nodes whether or not it is the destination of the communication. Such an approach has an impact on node energy consumption as packet transmissions consume battery power. Classical routing protocols use metrics such as number of hops or delays between routers but in an ad-hoc network context, it would be interesting to consider the remaining battery power of nodes. This should prevent from using all battery resources of a node when an alternate route can be found to spread the forwarding load more fairly within the network. We thus expect to get a better behavior when some nodes have low battery resources. The scope of this paper stands as a preliminary study to perform a battery aware routing algorithm. Based on measurements we evaluate the feasibility of getting information about battery consumption and we then present a mathematical model to predict the remaining energy consumption of a mobile host. This model is based on the measurements given by ACPI BIOS evaluation.

[1] Mobile Ad-hoc NETwork : IETF Working Group

II. Related work

Many empirical studies precisely determine the energy consumption of mobiles and propose mathematical models of this consumption [6], [4], [3]. In these articles, it is clearly shown that energy consumption and transmission bandwidth are not synonymous as it is necessary to consider not only the cost of transmitting a packet but also of receiving and even discarding it. Figure 2 shows that a network interface is a significant source of energy consumption for a mobile host, but the energy consumed by an interface depends on its operating mode.

Accurate measurements are made with specific electronic equipments able to measure energy consumption in $mWatts$ [6], [9]. These measurements identify the energy consumption of a mobile host depending on the operating mode of its wireless interface. These measurements nevertheless are not easy to reproduce as such, for ad-hoc networks, as they require particular measurement tools. The aim of our research is to make efficient measurements to estimate the power-level of an host. This may not be as accurate as the previous studies but this approach doesn't require any special type of equipments. We thus propose to use ACPI management. An exhaustive presentation of ACPI BIOS is presented in III.

III. ACPI and Power Management

[8] presents the implementation of ACPI specifications and its goals. A brief description extract from this article is explained below.

The Advanced Configuration and Power Interface (ACPI) specification was introduced by Intel, Toshiba

and Phoenix (joined later by Compaq and Microsoft). It defines the hardware and software interfaces that enable operating system directed power management system to control the power of various motherboard components. It was developed to establish common interfaces for various hardware devices enabling operating systems to direct power management of both devices and entire system. ACPI evolves from the existing collection of power management BIOS code, Advanced Power Management (APM) application programming interfaces, tables and so on into well-defined power management and configuration interface specification. ACPI provides the means for an orderly transition from existing legacy hardware to ACPI hardware, and it allows for both ACPI and legacy mechanisms to exist in a single machine and to be used as needed.

A. ACPI Specification and Structure

ACPI defines the hardware and software interface and the data structures that exist at this level. The specification describes the interfaces between components, the contents of the ACPI System Description Tables, and the related semantics of the other ACPI components. The ACPI System Description Tables, is the heart of the ACPI implementation and the role of ACPI system firmware is primarily to supply the ACPI tables to the layers above it. ACPI is merely a software-hardware specification. More details about ACPI specifications and the table it exposes can be found in ACPI specifications [1].

B. Goals

The primary goals of ACPI are:

1) The interfaces should be applicable and suitable to all classes of computers including desktops, mobile workstations and server machines.
2) ACPI compliant hardware devices should provide an uniform interface. They implement 4 processor states, 4 different global states and 5 different sleep states (S0 - S5). For example, S5 is soft off state, S4 is sleeping state. S4 consumes more power than S5, but takes less time to wakeup from sleep mode.
3) Machine implementers have the freedom to implement a wide range of solutions, while still maintaining fun os support.
4) Implementation of power management at operating system level win makes it practical for the OS and application to be aware of power consumption.

In traditional systems, the hardware-software interface also known as BIOS uses the power interface provided by the hardware to switch modes. The defacto standards has been the Advanced Power Management system or APM introduced by Intel and Microsoft. The user specifies timeout values for various devices such as disks and network interfaces. The BIOS switches to power saving modes when the device is inactive until it hits the timeout. For example, a hard disk is made to spin down to save power, the network interfaces are switched to sleep state, memory is moved to power down state, processors are switched to lower frequency and/or voltage etc.

With this traditional style of power management, BIOS does a significant amount of work. The policies implemented in the ROM BIOS are rather too simple, as it is complicated to write BIOS code. Resides that, the policies are fixed, as the code that implements various policies are burned into ROM.

ACPI is a substitute for APM at the hardware-software level. However ACPI, unlike APM, does not deal with power management. Instead it exposes the tables present in various hardware devices to the operating systems. ACPI lays the responsibility of power management on the operating system as it is in the best position to make judgments relating to power management. In effect, ACPI is a standard that encourages hardware vendors to provide a standardized interface that is ACPI compliant, and operating systems to export the tables that ACPI exposes to the applications. Much of the work is now done at the OS level and rather than at the hardware or BIOS level.

Implementing ACPI at the OS level is much easier. Various policies can be implemented for devices present in the system. It encourages hardware vendors to just provide the interface for switching power states, and not actually do any power savings. This means that, we now have an unification of power management algorithms in the OS which will reduce conflicts between hardware, firmware and will enhance reliability. Power management at the OS level also allows us to implement complicated policies, allowing the users to turn the knob between conservative and aggressive power saving modes. Advantages of ACPI include:

1) Moving power management functionality into the OS makes it available on every machine on which the OS is installed. The level of functionality (power savings, and so on) varies from machine to machine, but users and applications will see the same power interface and semantics on all ACPI compliant machines.

2) Power management is not restricted at the BIOS and the hardware level. With ACPI and a suitable model, applications can tune themselves and conserve power.
3) There is much less state information for the BIOS to retain and manage because the OS manages it. This leads to a simpler implementation at the BIOS level.
4) Power management algorithms are unified in the OS, yielding much better integration between the OS and the hardware.
5) OS can deal with power management of devices dynamically, as the interface provides dynamic registering or loading and unloading of devices.

C. How we use it ?

Thanks to the FreeBSD `sysctl` command, we are able to get battery state information.

```
hw.acpi.power_button_state: S2
hw.acpi.sleep_button_state: S1
hw.acpi.lid_switch_state: S2
hw.acpi.standby_state: S1
hw.acpi.suspend_state: S3
hw.acpi.sleep_delay: 0
hw.acpi.s4bios: 1
hw.acpi.verbose: 0
hw.acpi.disable_on_poweroff: 1
hw.acpi.thermal.min_runtime: 0
hw.acpi.thermal.polling_rate: 30
hw.acpi.thermal.tz0.temperature: 3307
hw.acpi.thermal.tz0.active: -1
hw.acpi.thermal.tz0.thermal_flags: 0
hw.acpi.thermal.tz0._PSV: -1
hw.acpi.thermal.tz0._HOT: -1
hw.acpi.thermal.tz0._CRT: 3712
hw.acpi.thermal.tz0._ACx: -1 -1 -1 -1 -1
-1 -1 -1 -1 -1
hw.acpi.acline: 0
hw.acpi.battery.life: 99
hw.acpi.battery.time: 256
hw.acpi.battery.state: 1
hw.acpi.battery.units: 1
hw.acpi.battery.info_expire: 5
```

- `hw.acpi.battery.life`: % battery
- `hw.acpi.battery.time`: estimation of remaining time in minute
- `hw.acpi.battery.state`: 1 for out-load, 2 in-load
- `hw.acpi.battery.units`: number of battery unit in the mobile
- `hw.acpi.battery.info_expire`: refresh rate in second

IV. Experimental background

In order to measure energy consumption of mobile hosts, we have used a test-bed of 3 mobiles like presented in 1.

- a transmitter : A
- a forwarder: B

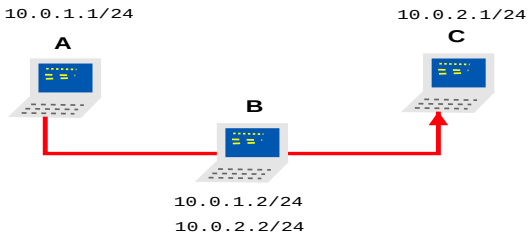

Fig. 1. Testbed

- a receiver : C

Each mobile is composed of a DELL© Pentium III 800Mhz with 256Mo RAM equiped with CISCO© 350 Wireless card. Each machine was placed roughly 20cm apart. Care was taken that each mobile host was configured in the same manner and running an identical set of processes, since the larger the load on the processor the larger its energy consumption[10]. Thus by running an identical set of process, only the consumption related to the wireless card is a variable in our experiments.

Although for our experiments, mobility is not the core of our study we have chosen to be as close as possible as real operating conditions. Therefore we have chosen to emulate an ad-hoc routing protocol by setting the route between host A and C as presented in figure 1. Host B has two IPs address thanks to IP aliasing and is a forwarder between host A and host C. So, mobile C is reachable by mobile A through B. The traffic generated for our tests is obtained with a traffic-generator called BENCH [7]. We transmit TCP traffic of 1024 bytes packets from host A to host C.

V. Experimental measurements and results

A. Energy consumption states

The network interface of a mobile host is able to take four states :

1) transmit : for transmitting data,
2) receive : for receiving data,
3) idle mode : in this state, the interface is able to transmit or receive data, this is the default mode,
4) sleep mode : extremely low power consumption, the interface can neither transmit nor receive until it is woken up.

[5] shows that transmission is more consuming in terms of battery than reception. Through experimental results, this paper shows that the consumption of a Lucent IEEE 802.11 card is approximately 25% higher in receive mode than in idle mode and approximately 63% higher in transmit mode than idle mode. These results appear to be similar with different types of

card. In an ad-hoc network, hosts will hear all transmissions in their neighborhood, whether or not they are part of the communication. It may thus be interesting to define a passive receive mode, which would not require a handling at OSI layer 3. The distinction between idle and passive receive mode would be interesting to make but special equipments, such as the ones used [5] would be needed to differentiate these two states. As for our experiments we base only our measurements on the ACPI BIOS information it is not possible to make such a distinction. For our definition of a low consumption state we will thus assimilate the idle and passive receive mode.

The aim of our measurements is to determine the energy consumption of a forwarding host, as we want to identify the battery discharge due to data forwarding. For this purpose we only need to measure the discharge of forwarding nodes and two consumption states are required:

- **high consumption state** when the host is in forwarding mode (receive and transmit mode),
- **low consumption state** when the host does not forward packets (passive receive or idle mode).

This first definition of two consumption states may appear simple but information returned by BIOS ACPI gives the real perception a mobile node has of its own energy consumption.

B. Hypothesis

In this section we will present the parameters defining our experimental scenarios.

1) Size of packets: All our traffic is composed of 1024 bytes packets. Others experiments have already shown the importance of packets size on a ad-hoc network and its influence on energy consumption. Then we have chosen to simplify our measurements and not take into account this variable.

2) Signal strength: When the receiving signal is too low, the transmitter can often detect it and if necessary decide to decrease its transmission speed to permit a better reception. It may thus affect slightly the energy required to receive data. Nevertheless we have decided to neglect this effect and thus to simplify our studies by not taking into account signal strength variation. This would be also very difficult to quantify the signal strength effect without specific equipments.

3) Measurements presentation: Each figure below has an x-axis which represents the time in seconds and a y-axis which gives the percentage of battery load except for figure 9 where we have swapped these axes to simplify the comprehension.

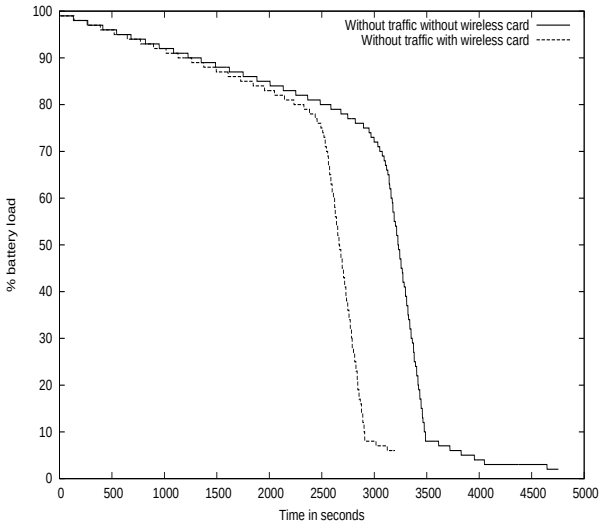

Fig. 2. Comparison of battery discharge with or without wireless card

C. Threshold characterization

ACPI BIOS doesn't give a linear estimation of battery discharge. Figure 2 shows that about 75% of battery charge, the mobile host accuses a rapid discharge. From now on, we call *battery threshold* the threshold corresponding to a rapid decrease on energy consumption estimation. One can note that this threshold differs from each battery used, but it seems that this threshold is approximately comprised between 70% and 80%.

D. Battery discharge of a forwarding mobile host

In order to confirm the effect of packet forwarding on battery resources we have generated a traffic from host A to host C. Host B was acting as a router for this traffic. As shown on figure 3, when B forwards a $2Mbit/s$ traffic, its battery level decreases more rapidly than when it forwards data at $0.5Mbit/s$. Note that for these experiments, we have not restricted our transmission to a given volume of data but in both cases we have transmitted until the battery was fully discharged. The maximum data rate chosen for our experiments ($2Mbit/s$) is close to the maximum TCP throughput that can be reached with a node receiving and transmitting simultaneously.

From these measurements we can see that the battery discharge is faster at $2Mbit/s$ than at $0.5Mbit/s$ but it is not possible to conclude whether this is due to the volume of forwarded data or to the transmission rate.

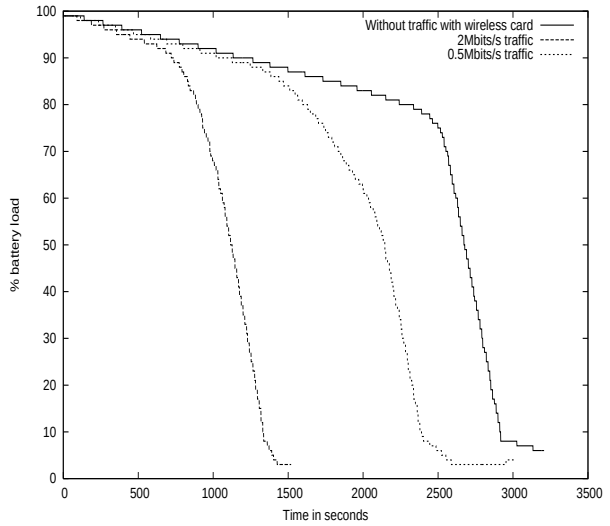

Fig. 3. Comparison of battery discharge with different throughput

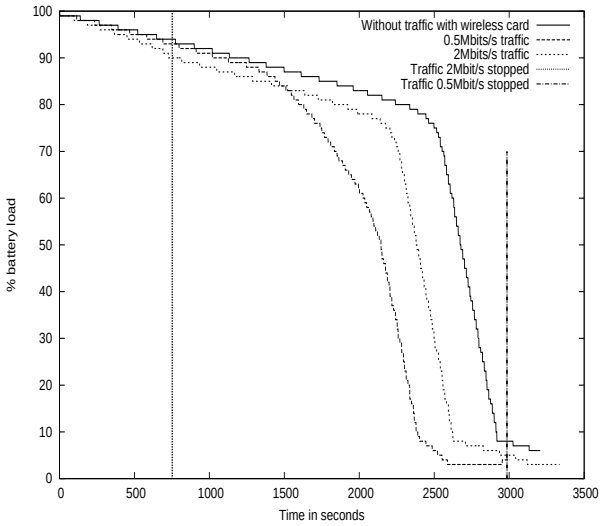

Fig. 4. Comparison of battery discharge with a fixed number of transmitted packets

E. With constant throughput and a fixed number of messages

In this section we want to compare the impact of volume of data transmitted versus data throughput to determine which one as a greater effect on battery discharge. We have used the same operating conditions as for the tests presented in section V-D. We have simply fixed the number of packets transitting through node B. As shown on figure 4, energy consumption becomes more important when we transmit 200000 packets at $0.5Mbit/s$ than at $2Mbit/s$. This shows that a node with little battery power left should either stop transmitting or transmit as fast as possible the transitting packets. This can be explained by the fact a wireless card consumes more energy when its components are active. Thus for a given amount of data the faster the node transmits, the shorter it will stay active. The transmission speed doesn't affect as much the battery consumption at least to a certain extend.

The results also clearly show that at the end of the transfer of the 200000 packets at $2Mbit/s$, the energy curve matches the pattern of the curve in idle mode and so decreases its energy consumption less quickly than before.

Other measurements like the ones presented in figure 5 confirm the importance of the couple *(throughput, number of transmitted messages)* in the evaluation of the mobile host energy consumption.

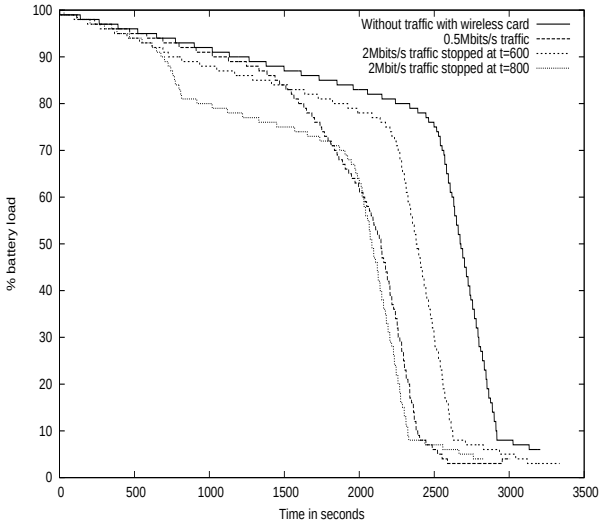

Fig. 5. Variation of battery discharge depending on the forwarded traffic

VI. CALCUL AND MODEL FOR ENERGY CONSUMPTION EVALUATION

A. Mathematical model

Depending on the mode we are in, high or low consumption mode [2], the discharge curve can be defined by an exponential or a linear function. In low consumption mode, the curve remains linear until the discharge threshold is reached. At this stage, the discharge accelerates and becomes similar to the one obtained in a high consumption mode. The value of this discharge threshold [3] may differ from one battery to another and can also vary during a battery lifetime. In our tests, its value remained quite stable and the curve in low discharge mode (no forwarding) changes its de-

[2]The 2 modes are defined in V-A.

[3]See section V-C for details.

crease slope approximately when the battery reached 80% of its remaining capacity.

The consumption mode has also an impact on the slope inclination as explained in the previous sections. As a mobile host transmits data, it reaches faster its threshold than when it is in low consumption mode.

In order to evaluate the remaining lifetime of a battery we are presenting a model of a battery consumption curve. In low consumption mode, This model is decomposed in two parts one before reaching the threshold one after. In high consumtion mode only the exponential formula is needed.

- high consumption state :

$$y = \%_{battery-init} - e^{\lambda * x} \quad (1)$$

  λ is function of the throughput
- low consumption state (before the threshold):

$$y = \%_{battery-current} - (\epsilon * x) \quad (2)$$

  In these formula, ϵ corresponds to the discharge slope in low consumption mode. Note that the ϵ value is fixed for a given battery. Its value may vary in time depending on how old the battery is, but for our experiments and in low consumption mode ϵ remained equal to 0.009.

In Figures 6 and 7 we supperpose the curves our mathematical model and the plots obtained in figure 3, 4 and 5, in order to validate these equations. The values of λ and ϵ are found by empirical approximations.

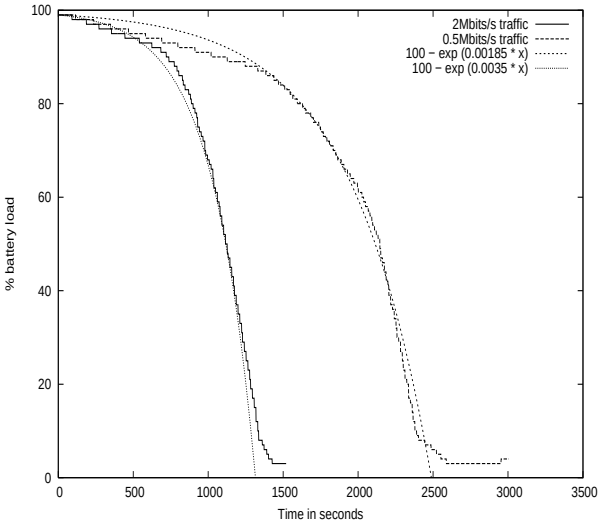

Fig. 6. Validation of our full discharge model in high consumption mode

On figure 7 we can see that the $2Mbit/s$ traffic curve is properly defined with our equations. From:

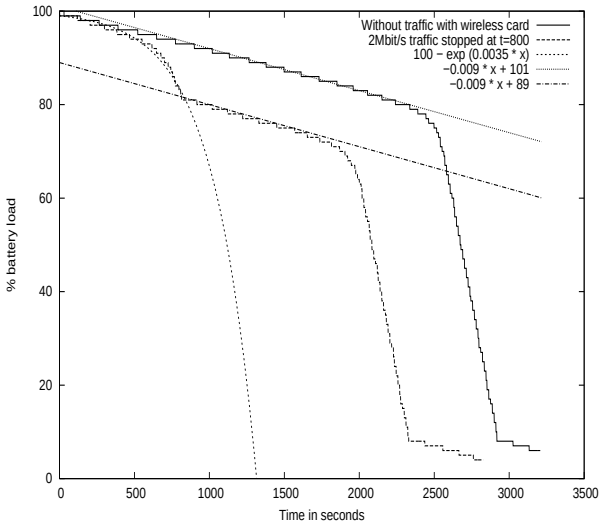

Fig. 7. Validation of our discharge model in high and low consumption mode

$$\begin{cases} t = [0, 800] : 100 - e^{0.0035 * x} \\ t = [801, 1800] : (-0.09 * x) + 89 \end{cases} \quad (3)$$

After $t \approx 1800$ we reach the threshold which indicates we are now in high consumption mode and the remaining lifetime of the battery is low. We have not drawn the mathematical model for this part of the curve on the figure.

B. Empirical model

Equations given above are interesting to use with a network simulator, but for a real testbed, they require to know the initial battery level at the start of the measurements and the values of λ and ϵ. All these parameters depends on the battery used and on the throughput for λ only. Such parameters may be somehow difficult to get as for instance it is restrictive to assume each battery will be fully charged at the start of the algorithm. Thus for our context of power saving routing protocol, we can not assume all these parameters can be obtained. We have thus defined an simpler algorithm which automatically determines whether a node increases or reduces its battery consumption thanks to the results obtained with the ACPI measurements.

Figure 8 presents a simple manner to calculate the fluctuations of energy consumption without knowing the current battery state nor the consumption mode[4]. Each segment $p1$ and $p2$ on figure 8 represents the laps of time a node stays at the same battery load. Each time the decrease represents 1% of the battery discharge, we measure the length of the p segment.

[4]high or low

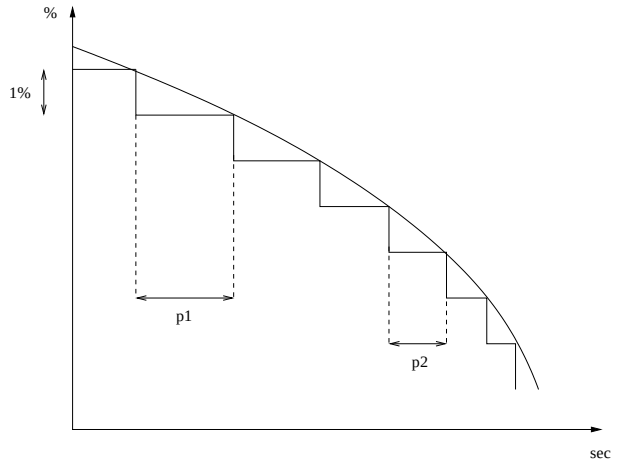

Fig. 8. Calcul method of the curve slope

Thanks to this simple algorithm below, we are able to determine if host is reducing or increasing its battery consumption by looking at the p value.

```
1   low_consumption_p =  current_p = 0
    while
    {
            measured_p =  current_p
5           if (current_p < low_consumption_p)
                    state = high_consumption
            else
                    low_consumption_p = current_p
                    state = low_consumption
10          return(state,measured_p)
    }
```

This algorithm compares the current segment length (time during which the node remains at the same battery level) with the previous segment length in order to find the larger one. By simple dichotomy, we can deduce whether we are in higher or lower consumption mode. The larger stored segment identifies the lowest consumption mode for our experiments.

We have implemented this algorithm on mobile host and compare to the energy consumption given by ACPI BIOS informations. Results are presented in figure 9. We could show that after start where the algorithm determine the initial state, evaluation of low or high consumption mode is correct. This method has the advantage that it does not take care about energy consumption of receiving or emitting messages. We are focused only on the energy consumption and are able to give the state of our host.

VII. Conclusion and future work

In a concern of spreading more fairly the forwarding load among nodes it is important to define a battery-aware ad-hoc routing protocol. The aim of such an approach is to prevent from fully discharging mobile terminals batteries for communications they

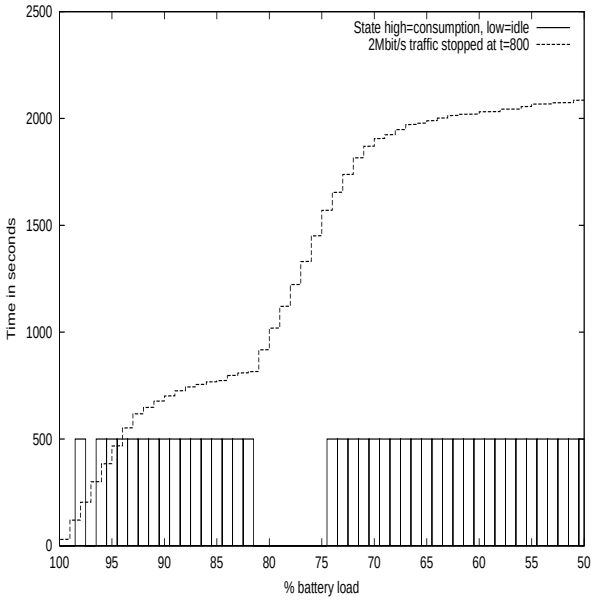

Fig. 9. Evaluation of battery state

are not part of. A battery aware routing algorithm would take into account the current battery level of nodes to determine the route in the network, with the lower battery consumption cost.

Such an approach would only be feasible if one can evaluate in real time the remaining battery level of mobile terminals. Many experimental results have been presented which give accurate measures of a battery discharge, but they require dedicated equipments to obtain these measurements.

The contribution of our work in this paper is to present experimental results which give rough, but significative, information about a battery discharge. We propose two models to evaluate the remaining lifetime of a battery. The first one is an analytical model which can be used with a network simulator and a second an empirical one to be used for real testbed.

We have shown that the discharge pattern can be classified as high and low consumption mode depending on both the amount of forwarded traffic and the remaining lifetime of the battery. We present an easy way to implement this model which can later be incorporated in many energy aware routing algorithm. This implementation can be done with the ACPI BIOS tools which can be currently found on many laptops.

References

[1] Compaq and Intel and Microsoft and Phoenix and Toshiba. ACPI : Advanced configuration and Power Interface, 2001.

[2] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Peformance Issues and Evaluation Considérations. RFC 2501, IETF, January 1999.

[3] J. Ebert, B. Burns, and A. Wolisz. A trace-based approach for determining the energy consumption of a WLAN network interface . In *In Proceedings of European Wireless*, pages 230–236, Florence, Italy, February 2002.

[4] Laura Marie Feeney. An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. *Journal of Mobile Networks and Applications*, 3(6):239–249, 2001.

[5] Laura Marie Feeney. A QoS Aware Power Save Protocol for Wireless Ad Hoc Networks. In *In Proceedings of Med-Hoc-Net*, Sardegna, Italy, September 2002.

[6] Laura Marie Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *In Proceedings of IEEE Infocom*, Anchorage, AK, US, June 2001.

[7] Vincent Roca. BENCH: a Network Performance Evaluation Environment. http://www.inrialpes.fr/planete/people/roca/.

[8] Saisanthosh Balakrishnan and Jyothir Ramanan. Power-Aware Operating Systems using ACPI. Technical Report CS 736 project, University of Wisconsin, Computer Science Department.

[9] Anmol Sheth and Richard Han. An Implementation of Transmit Power Control in 802.11b Wireless Networks. Technical report CU-CS-934-02, Department of Computer Science, University of Colorado, Boulder , August 2002.

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