

# Battery-Efficient Architecture for an 802.11 MAC Processor <sup>\*</sup>

†Kanishka Lahiri

‡Anand Raghunathan

†Sujit Dey

†Dept. of ECE, UC San Diego, La Jolla, CA

‡C & C Research Labs, NEC USA, Princeton, NJ

*Abstract*—Rapid growth in the complexity of wireless devices, communication protocols, and applications, combined with slow improvements in battery technologies, have created a “battery gap” that is only projected to increase with advances in wireless communication technologies and applications. Conventional approaches to bridging this gap exploit low-power network protocols and handset architectures. However, it is now well known that minimizing total energy or average power drawn from a battery does not necessarily lead to maximizing battery life, calling for new battery-driven approaches to protocol and hardware design.

In this paper, we present a battery-efficient architecture for an 802.11 MAC processor, which incorporates a new battery-driven approach to power management. The MAC processor employs a novel on-chip bus architecture that is capable of regulating the profile of the current drawn by the system, enabling battery discharge at high efficiencies. The proposed battery friendly MAC processor architecture enables significant increases in battery capacity and lifetime, while minimizing performance impacts. Further, the developed architecture provides mechanisms that allow for trade-offs between battery life and performance, and can be configured to adapt the power management techniques based on the network traffic characteristics.

## I. INTRODUCTION

As wireless handsets evolve towards higher data rates and richer functionality (*e.g.*, W-LAN devices, 3-G handsets), energy demands are expected to grow at extremely rapid rates. Projections of the growth of energy requirements in hand-held devices alongside those of battery capacity illustrate an increasing “battery gap”, indicating that battery technology will impose increasingly severe constraints on the design of such systems. In order to reduce this gap, and at the same time facilitate development of complex wireless applications, a crucial need arises for wireless devices to make the most efficient use possible of limited battery resources.

Most known approaches to improving battery life of mobile devices aim at reducing the total energy consumed by the system. While such approaches are often successful in extending battery life to some extent, it is well understood that minimizing total energy consumption does not necessarily lead to maximizing battery lifetime [1]- [6]. The reason behind this is that batteries, being non-ideal sources of energy, often deliver significantly less (or more) energy than the manufacturer’s rated capacity. This departure of the delivered energy from the specified capacity depends strongly on the profile of the current drawn by the load system. Hence, battery life extension methods that ignore the characteristics of a system’s current consumption profile (*e.g.*, those that minimize total energy consumption, or average power dissipation) fail to maximally extract the energy stored in a battery.

The focus of this paper is on battery driven power management techniques and architectures for wireless devices. In particular, we describe a battery-efficient design of a MAC processor that uses new power management techniques to achieve significant improvements in battery capacity and lifetime.

## A. Paper Overview and Contributions

The two main contributions made in this work are:

- An architecture for MAC layer processing that incorporates battery driven power management aimed at increasing battery life by tailoring the system current profile to ensure efficient discharge of the battery.
- A novel bus architecture for the MAC processor that implements the battery driven power management technique.

We demonstrate mechanisms in the architecture that (i) allow higher layer protocols, or application, to select a desired region in the battery/performance trade-off space, and (ii) allow the architecture to adapt to different types of network traffic.

In Section II, we provide relevant background, describing important characteristics of contemporary battery technologies. In addition, since we use the on-chip bus architecture to implement battery driven power management, we describe some concepts related to the design of system-level bus architectures. In Section III we present a high performance architecture that implements the MAC processor. Section IV describes the application of battery driven power management to the design of the MAC processor. Finally, in Section V, we describe experiments that investigate the ability of the MAC processor to effectively trade off performance for battery efficiency; and its configurability for different types of network traffic, and varying user/application requirements.

## B. Related Work

Recognizing the problem of energy consumption in a mobile environment, there has been considerable activity in developing energy/battery-efficient communication protocols for wireless networks, *e.g.*, [7]- [11]. These efforts address battery/energy efficiency by developing new protocols, modifying existing ones, or through power management techniques applied at the network level. These approaches typically do not consider the effect of the architecture of each individual node, or the implementation of the protocol, on battery efficiency. In this work, we are concerned with developing a battery-efficient implementation of a specific protocol, namely a MAC processor for an 802.11 based device. A large body of work deals with methodologies for low power design [12] and dynamic power management [13] of electronic systems. However, these techniques aim at reducing average power dissipation, or minimizing energy consumption of the entire system, and hence, as pointed out earlier, do not necessarily lead to the most battery-efficient design. Recent work on accurate and efficient modeling of electro-chemical phenomena in Lithium-ion batteries is paving the way for developing new, battery-efficient design techniques [2]- [5]. In our work, we use such battery models to help drive the design of battery-efficient architectures.

## II. BACKGROUND

In this section, we first describe characteristics of typical batteries used in mobile devices. In addition, since our battery driven

<sup>\*</sup> This work was supported by the National Science Foundation, grant 99-12414

power management technique is implemented in the MAC processor's bus architecture, we briefly describe concepts and terminology associated with on-chip bus architectures.

#### A. Battery Discharge Characteristics

The capacity of a battery is typically expressed in terms of *standard capacity (mAh)* — the amount of energy that can be obtained from the battery when it is discharged at a specific constant current called the *rated current*. However, the energy obtained from a battery during discharge may be different from this value (greater or lesser), and is expressed as the *actual capacity*. Batteries also have a *theoretical capacity*, which the actual capacity can never exceed. In the rest of this paper, the term battery capacity will refer to the actual capacity of the battery. The *rate capacity effect* is an electro-chemical phenomena that significantly influences the actual capacity of a battery, capturing the dependency between the magnitude of the discharge current and the actual capacity. For higher rates of discharge (large current values), the discharge efficiency of the battery is lower, resulting in reduced capacity and lifetime. In addition, idle times contribute to a *recovery effect*, potentially increasing the capacity and lifetime of the battery. Both these effects have been observed in the case of popular battery technologies, including Lithium-ion batteries [2], [14]. While both the rate capacity effect and the recovery effect influence the overall performance of the battery, in this work, we focus mainly on the rate capacity effect.

#### B. On-Chip Bus Architectures

On-chip bus architectures are commonly used as a fabric to support inter-component communication among on-chip components. The *topology* of on-chip bus architectures range from a single shared bus, to which all system components are connected, to a network of busses interconnected by *bridges*. Components connected to an on-chip bus include those that can initiate a bus transaction, called *bus masters* (e.g., CPUs, DSPs, etc.), and those that merely respond to transactions initiated by a master, or *slaves* (e.g., on-chip memories). Since busses are often shared by several masters, *bus protocols* are used to specify the exact manner in which bus transactions occur. These include arbitration mechanisms (e.g., *round-robin* access, *priority based* selection, *time-division multiplexed access*), which are implemented in centralized or distributed *bus arbiters*. The protocol also defines handshaking conventions, burst transfer modes, etc.

### III. MAC PROCESSOR ARCHITECTURE

In this section we present a high-performance architecture of a system that implements an illustrative subset of the 802.11 MAC layer functionality. We start with a functional description of the various tasks, and then go on to describe the architectural components to which they are mapped. Finally, we describe the on-chip bus architecture that we use to implement the inter-component communication needs.

#### A. Functional Overview

A task graph showing the various communicating tasks in the MAC processor is shown in Figure 1. We next briefly describe each of these tasks (full details are available in [15]). For ease of explanation we consider only outgoing frames. For incoming frames the tasks are identical, except the direction of data flow is reversed.

**Logical Link Control Interface (LLC):** This task is responsible for receiving frame bits from the Logical Link Control Layer, and storing them in the MAC processor memory until they are operated on by various MAC layer tasks.

**Wired Equivalent Privacy Encryption (WEP):** This task is responsible for encrypting the frame body whenever the encryption bit is set to 1 (this includes all data frames, and certain management frames). The encryption algorithm used to calculate the encrypted data is RC4, a stream cipher symmetric key algorithm [16]. For each new frame to be encrypted, WEP\_INIT generates a new seed value based on the secret key (constant) and an Initialization Vector (IV), specified in the first 4 bytes of each frame. Next, WEP\_ENCRYPT encrypts the frame body by modulo 2 addition with a stream generated by the RC4 algorithm. Next, WEP\_ENCRYPT encrypts the Integrity Check Value (ICV) and appends it to the encrypted frame body, and sends the (IV, frame body, ICV) triplet for subsequent processing.

**Integrity Checksum Value Computation (ICV):** This task works in conjunction with the WEP task in order to prevent unauthorized data modification. It reads the frame body and calculates a 32 bit Integrity Checksum Value, based on a CRC-32 algorithm [15]. The computed ICV is returned to the WEP\_ENCRYPT task for encryption.

**Header (HDR):** This task is responsible for generation of the complete MAC header for outgoing frames.

**Frame Check Sequence (FCS):** This task is responsible for computing the Frame Check Sequence over all the frame bits for each outgoing frame (including the body and the header), and writing the result to the FCS field in the MAC header of the frame. The algorithm for calculating the FCS value is the same as the one used to calculate the Integrity Check Value, namely, CRC-32.

**Media Access Control (MAC\_CTRL):** This task implements the access method — the distributed coordination function or CSMA/CA. This includes interfacing with physical carrier sense mechanisms, virtual carrier sense mechanisms, deferrals, back-offs, RTS/CTS exchanges, etc.

**Physical Layer Interface (PLI):** This task is responsible for retrieving frames stored in the MAC processor memory, and passing them on to the physical layer. For outgoing frames, the PLI task awaits communication from the MAC\_CTRL task which indicates that the channel is available, and it is safe to transmit.

#### B. HW/SW Implementation

Figure 2 shows the HW/SW architecture onto which the previously described set of tasks are mapped. In the following discussion, we consider in turn the various components (e.g., dedicated HW blocks, embedded CPUs) that implement the system's functionality, followed by the on-chip bus architecture that implements the inter-component communications.

**Function-Architecture Mapping:** The functional unit `llc_c` is a HW implementation of the Logical Link Control Interface (LLC) task. It write frame data to `mem_l`, and enqueues addresses of written frames in the queue `f_queue_l`. Frame addresses are read off the queue by (i) `wep_c`, a component consisting of dedicated logic to implement the WEP subtasks, and (ii) `sparc_l`, an embedded CPU running an efficient SW implementation of the ICV task. While computation of the ICV proceeds in parallel with the encryption of the frame, the encryption is not complete until the generated ICV is encrypted. To minimize performance loss due to bus conflicts, `wep_c` and `sparc_l` operate on data stored in lo-

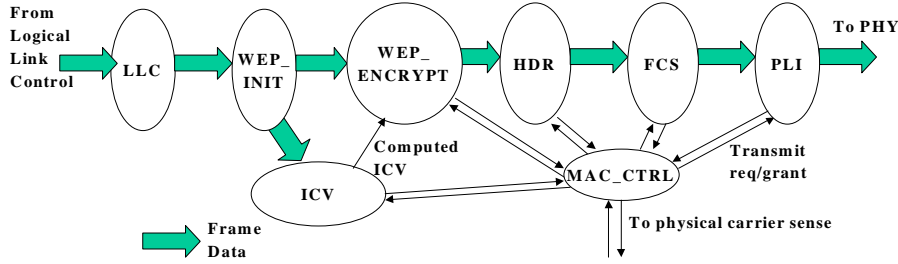


Fig. 1. Functional view of the 802.11 MAC processor

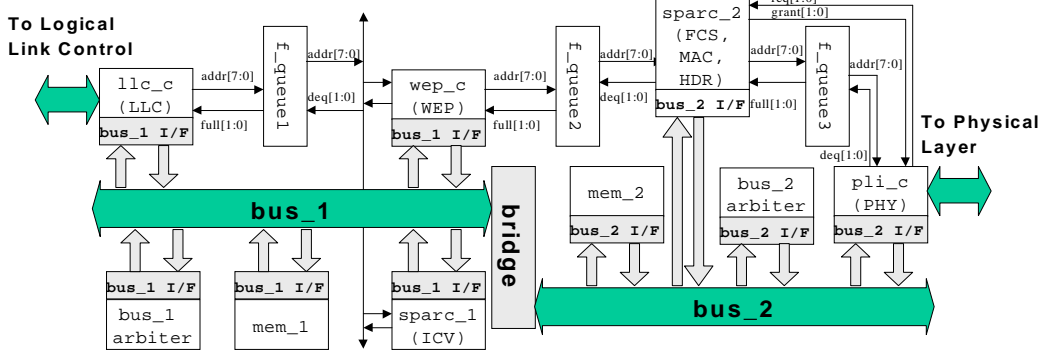


Fig. 2. HW/SW Architecture of 802.11 MAC processor

cal memory. Once `wep_c` has finished encrypting the frame, the resulting triplet (IV (not encrypted), frame body (encrypted), ICV (encrypted)) are written to `mem_2`, and the address to `f_queue2`. A second embedded processor core `sparc_2` implements three tasks: HDR (generation of the MAC header), FCS (calculation of the CRC checksum over all the frame bits, and insertion in the header), as well as `MAC_CTRL`. Once the processing by the FCS and HDR tasks is complete, and `MAC_CTRL` determines that transmission can proceed, `pli_c` hardware, (which implements the PLI task) removes an address from `f_queue3` and reads frame data from `mem_2` and passes it to the PHY implementation.

**On-chip Bus Architecture:** The architecture uses dedicated communication channels to implement low volume communications between functional units (*e.g.*, communications between `sparc_1` and `wep_c`) and 32-bit shared busses to implement large volume communications between system components. Instead of using a flat shared bus, we employed a multiple shared bus architecture with a bridge interconnecting the shared busses, to enable greater parallelism, and hence higher performance. A static priority based protocol is employed on each bus [17]. Components `llc_c`, `wep_c`, `sparc_1`, and the bridge constitute the set of masters that contend for access to `bus_1` in order to access the slaves `mem_1` and bridge. Components bridge, `sparc_2`, `pli_c` use `bus_2` to access slaves `mem_2` and the bridge. Note that the bridge needs to support master and slave interfaces, in order to carry out data transfers initiated by masters on either bus.

#### IV. BATTERY DRIVEN POWER MANAGEMENT OF THE MAC PROCESSOR

In this section, we describe our battery driven power management approach that makes use of the on-chip bus architecture to improve the battery efficiency of the MAC processor.

##### A. Overview

The various steps executed in obtaining a battery-efficient design of the MAC processor are as follows:

1. Generation of the current profile of the initial MAC processor (described in Section III) under a set of typical input stimuli.
2. Analysis of the MAC processor current profile to identify time intervals where the battery is inefficiently discharged, causing degraded battery capacity.
3. Identification of “battery-critical” states of the MAC processor, *i.e.*, combinations of component states that correspond to the time intervals identified in step 2.
4. Modification of the MAC processor bus protocols to regulate the occurrence of battery-critical states.
5. Repetition of steps 1 through 5 till no new battery-critical states are generated.
6. Configuration of the bus protocol to achieve the desired level of battery efficiency.

We next describe each of the above steps in detail.

**Generation of System Current Profiles:** Using a methodology described in Section V, we generate current profiles for the MAC processor for a large number of test cases, consisting of several thousand MAC frames with varying frame sizes and inter-frame spacings. The purpose of this extensive test trace is to force the MAC processor to pseudo-exhaustively enumerate all possible combinations of concurrently executing components.

Figure 3(a) shows the current profile of the initial MAC processor design (described in Section III), as it processes a sequence of 3 MAC frames: the first is of length 54 bytes, and next two (which arrive back to back) are of length 1414 bytes each. Note that, the current profile used is obtained by averaging the instantaneous current over an appropriate time window (in this case, 0.5 ms) to take into account internal time constants of the battery and power supply circuitry.

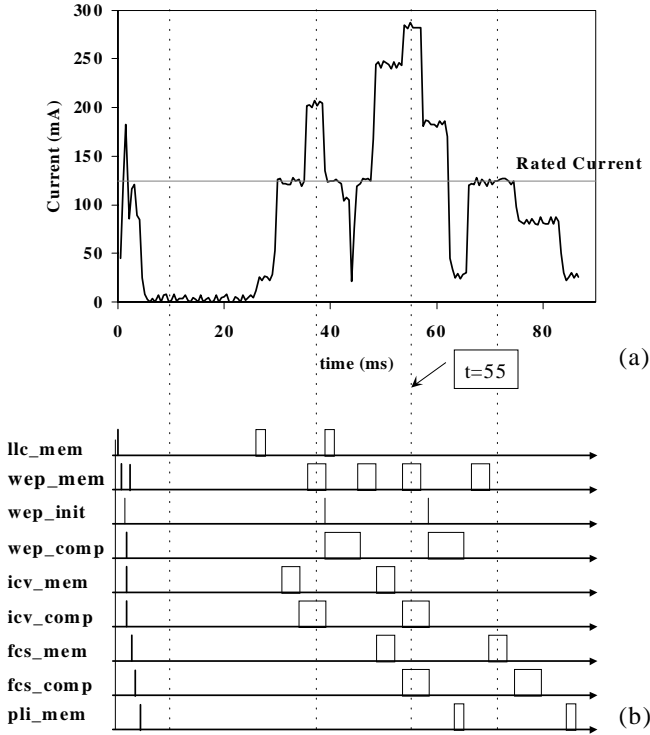


Fig. 3. (a) System current profile of MAC Processor with no power management, and (b) symbolic execution traces to identify battery-critical states

TABLE I  
PERFORMANCE AND BATTERY DATA FOR INITIAL MAC PROCESSOR ARCHITECTURE

Battery life	3209.2 sec
Battery Capacity	154.3 mAh
Discharge Efficiency	62.2%
Performance	475.5 Kbps

**Analysis of the System Current Profile:** In this step, the aim is to identify intervals of time where the MAC processor current profile causes inefficient use of the battery, leading to degraded lifetime and battery capacity. This is achieved by simulating battery discharge using the current profile obtained in the previous step, and a stochastic model of a Lithium-ion battery. The battery modeled has 250 mAh standard capacity at a rated current of 125 mA [5]. Table I shows that the battery capacity is only 154.3 mAh, or 62.2% of the standard capacity of the battery. Upon further analysis, we observe that the profile frequently exhibits regions where the current consumption is significantly higher than the rated discharge current of 125 mA (Figure 3(a)). The entire trace is subsequently analyzed to identify all time intervals where such violations of rated current occur.

**Identifying Battery Critical MAC Processor States:** The aim in this step is to analyze time intervals which are detrimental to battery efficiency, and map them to battery-critical system states. The state of the MAC processor any point of time is determined by examining the states of its individual components. Each component could be in one of several states. For example, component `wep_c`

may be in one of 7 states, depending on whether it is reading a frame body from `mem_1`, initializing the key state array, encrypting a frame body, etc.

To identify processor states that cause violations, a symbolic execution trace is generated. A portion of such a trace is shown in Figure 3(b), corresponding to the profile of Figure 3(a). The trace shows the values of 9 boolean state variables. For example, the value of the variable `llc_mem` indicates the time intervals over which the component `llc_c` is reading to/writing from memory. The overall state of the system at any point of time is determined by a vector of the state variables of all components in the system (e.g., the system state at time  $t = 55$  ms is given by 010001010).

From Figures 3(a) and (b), it is clear that different system states exhibit wide variation in the current drawn. From the list of violating time intervals, and the symbolic execution trace of system states, we identify system states that significantly contribute to inefficient battery discharge. For example, at  $t = 55$  ms, the system state is given by 010001010, which indicates that two functional units in the architecture are simultaneously engaged in lengthy iterative computations (state variables `icv_comp` and `fcs_comp` are 1 in Figure 3(b)), while a third is accessing memory over a shared bus (`wep_mem`), leading to a drawn current of 289 mA (Figure 3(a)).

Analysis of the traces in this manner yields a list of battery-critical states (which violate the rated current). Next we describe how the MAC processor bus architecture is enhanced to regulate the occurrence of such states.

**On-Chip Bus Architectures for Power Management:** The power management strategy is designed based on the following observation. As long as a component in the MAC processor has a pending communication request, its execution is blocked, and all its subsequent states are delayed. This provides a way to control the overall state of the MAC processor. By judiciously withholding communication grants, the occurrence of certain battery-critical MAC processor states can be regulated. An extreme bus protocol could be one where a communication request is granted only if the rest of the MAC processor is idle. At the other extreme, a protocol may grant communications so as to maximize the number of processor components executing in parallel.

Several battery-critical system states are identified for the MAC processor. For example, the state  $\{\text{wep\_mem} = \text{TRUE and icv\_comp} = \text{TRUE}\}$  is found to be one of them. The occurrence of this state is regulated by the following policy:

```
policy_1(int min_delay) /* set by designer */
on every wep_c request:
if (bus1_free = TRUE and icv_comp = TRUE) {
/* check time since last violation */
if
((current_time-last_violation)>min_delay )
grant wep_c;
else
deny wep_c;
}
```

For larger values of the parameter `min_delay`, the MAC processor enters the state  $\{\text{wep\_mem}=\text{TRUE and icv\_comp} = \text{TRUE}\}$  less often, leading to fewer rated current violations, and improved battery efficiency. Smaller values make the policy less stringent, allowing more frequent violations. Using similar policies, the occurrence of selected battery critical states of the MAC processor are regulated.

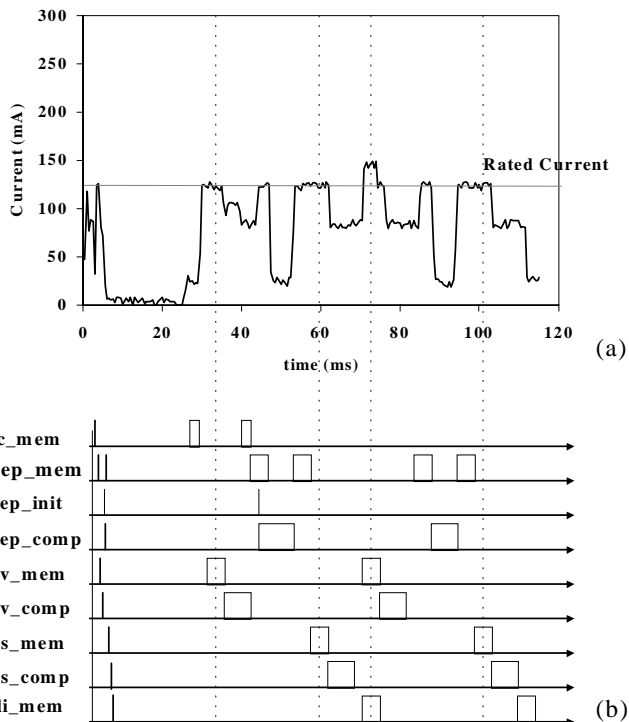


Fig. 4. (a) System current profile of MAC Processor with battery driven power management, and (b) Symbolic execution traces to identify battery-critical states

TABLE II  
PERFORMANCE AND BATTERY DATA FOR BATTERY-EFFICIENT MAC  
PROCESSOR ARCHITECTURE

Battery life	10199.9 sec
Battery Capacity	225.4 mAh
Discharge Efficiency	90%
Performance	205 Kbps

To illustrate the effect of such policies, the on-chip bus architecture of the MAC processor is modified to incorporate an aggressive power management policy in which a large number of battery-critical states are ruled out. A snapshot of the new current profile is shown in Figure 4(a), while the corresponding symbolic execution trace is shown in Figure 4(b). Clearly the occurrence of states that violate the rated current are greatly reduced. The results after simulating the discharge of the battery are presented in Table II, illustrating the impact of the new MAC processor architecture on battery life. The battery capacity improvement is  $1.46X$ , and battery life improvement is  $3.18X$ .

**Iterative exploration of system states:** Modifying the bus protocol has a ripple effect on the timings of component states, potentially causing new processor states to occur. Hence, in order to identify as many battery-critical states as possible, steps 1-5 are performed iteratively till no new states are generated.

**Configuring the power management policy:** This requires setting policy parameters (`min_delay` in the above policy) to regulate the the occurrence of battery critical MAC processor states. As

demonstrated in Section V, such parameters can be used to flexibly trade off performance for battery efficiency.

## V. EXPERIMENTAL RESULTS

In this section, we present results of experiments conducted on the proposed MAC processor architecture. We first describe the methodology used to conduct the experiments. We then investigate the effectiveness of the battery driven power management technique in trading off performance for battery efficiency. Finally, we investigate the impact of variation in network traffic characteristics on the selection of different power management policies.

### A. Experimental Methodology

The MAC processor architecture was developed in the POLIS HW/SW co-design environment [18]. The current consumption profiles for the MAC processor were obtained using a modified version of the system-level power estimation framework presented in [19]. The methodology consists of a discrete-event HW/SW co-simulation environment, power models for various system components, and speedup techniques to enable efficient power estimation for complex systems.

The current profiles of the MAC processor were provided as an input to a stochastic model of a Lithium-ion battery, which accurately and efficiently models rate capacity effects (based on a table lookup technique) as well as recovery effects [5]. The battery for our experiments had standard capacity  $250\text{ mAh}$ , at a rated current of  $125\text{ mA}$ . These are typical values for commercial batteries commonly found in cell phones, PDAs *etc* [20]. To get an idea of the battery efficiency and the performance of the MAC processor under real-life scenarios, we used traces of actual 802.11 traffic to drive the experiments. These traces were obtained by running the *Etherereal* packet capture software [21] on a laptop PC connected to a wireless LAN network, under various applications (streaming audio/video, file transfer, telnet *etc*). To quantify the battery efficiency of the MAC processor, we measure battery capacity. Performance of the MAC processor is expressed in terms of the average data rate.

### B. Battery Efficiency and Performance Trade-Off Analysis

To evaluate the effectiveness of the techniques described in Section IV, we implemented different versions of the MAC processor, each with a different power management policy. These policies differ in terms of the system states each of them detects as being “battery critical”, and the exact component(s) whose communications the bus protocol chooses to regulate.

Figure 5 shows (i) battery capacity (in *mAh*) as well as (ii) data rate (in *Kbps*) of the MAC processor under different power management policies. The trace consists of 4818 frames generated by 1.5 minutes of streaming video over an 802.11 based wireless network on a laptop running the *Windows Media Player* [22]. From the graph we make the following observations:

- The power managed architecture of the MAC processor shows a substantial increase in battery efficiency. Capacity improvements of  $1.45X$  are observed, from  $154\text{ mAh}$  under policy 0 (no power management) to  $225\text{ mAh}$  under policy 6.
- Performance penalties are reasonable, considering that small improvements in battery capacity imply substantially longer run times. For example, while policy 6 represents a  $1.46X$  improvement in capacity, it yields a  $3.18X$  improvement in battery life (from  $3209\text{ sec}$  to  $10199\text{ sec}$ ). However, the

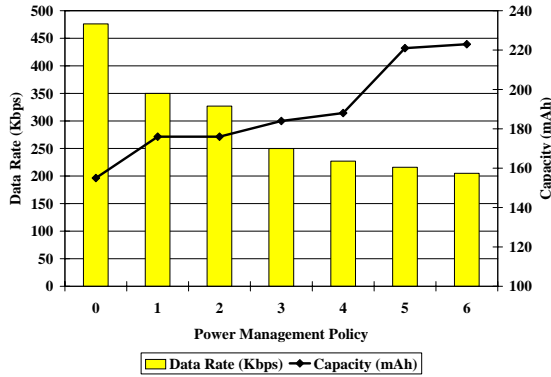


Fig. 5. Battery efficiency vs performance of the MAC processor for different battery driven power management policies.

performance penalty is only 57% (down to 205 Kbps from 475.5 Kbps).

- Certain policies work better than others for specific classes of network traffic. For this video stream, policies 4 and 5 yield comparable data rates, whereas policy 5 is significantly more battery-efficient than policy 4.

### C. Effect of Network Traffic Characteristics

We investigated the battery efficiency of the power managed MAC processor under varying network traffic characteristics. For this experiment, we constructed artificial sequences of frame arrivals, consisting of fixed size frames, arriving in a periodic manner, with constant inter-frame spacing. We generated several such traces, under different values of the inter-frame spacing, and for each, we measured the battery capacity under the set of power management policies described earlier.

The results of this experiment are shown in Figure 6. The figure shows a set of plots, one for each policy. From the figure, we observe that policies differ in their effectiveness across different arrival rates. Policy 1 results in high efficiency for traffic with large inter-frame spacing. Policies 3 and 5 provide intermediate efficiency, while policy 6 (the most aggressive policy) provides high efficiency across traffic with a wide range of inter-frame spacing. While one may be tempted to use policy 6, one must configure the policy carefully (as described in Section IV) so as to balance performance and battery efficiency. This is done by regulating the rate at which bus transactions associated with the policy are inhibited. When this rate is set to a high value, the system incurs larger losses in performance, while significantly gaining in battery efficiency. However, by varying this rate, one can generate arbitrary points in the trade-off space. These trade-off parameters could be under the control of the user, or may be set by higher layers in the protocol stack, or the application, with one extreme yielding a battery-efficient mode of operation and the other yielding a high performance mode of operation.

## VI. CONCLUSIONS

In this paper we presented the design of a battery-efficient architecture for an 802.11 MAC processor, based on a new battery driven power management technique, implemented in the on-chip bus protocols of the MAC processor. Experimental results show that the proposed architecture is highly battery efficient, and that it can easily be configured to trade off performance for battery efficiency, and adapt to network characteristics

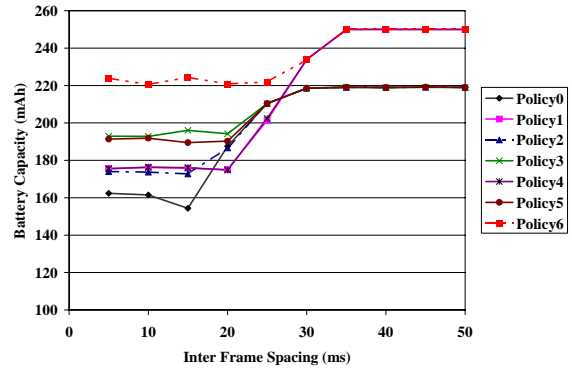


Fig. 6. Effect of inter-frame spacing on battery capacity for different MAC processor power management policies

## REFERENCES

- [1] K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi, "Battery driven system design: a new frontier in low power design," in *Proc. ASP-DAC/Int. Conf. VLSI Design*, pp. 261–267, Jan. 2002.
- [2] M. Doyle, T. F. Fuller, and J. S. Newman, "Modeling of galvanostatic charge and discharge of lithium/polymer/insertion cell," *J. Electrochem. Soc.*, vol. 140, pp. 1526–1533, June 1993.
- [3] S. Gold, "A PSPICE macromodel for lithium-ion batteries," in *Proc. Annual Battery Conference on Applications and Advances*, pp. 9–15, 1997.
- [4] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "A discrete-time battery model for high-level power estimation," in *Proc. Design Automation & Test Europe (DATE) Conf.*, pp. 35–39, Mar. 2000.
- [5] D. Panigrahi, C. F. Chiasserini, S. Dey, R. R. Rao, A. Raghunathan, and K. Lahiri, "Battery life estimation for mobile embedded systems," in *Proc. Int. Conf. VLSI Design*, pp. 55–63, Jan. 2001.
- [6] S. Park, A. Savvides, and M. B. Srivastava, "Battery capacity measurement and analysis using lithium coin cell battery," in *Proc. Int. Symp. Low Power Electronics & Design*, pp. 382–387, Aug. 2001.
- [7] P. Lettieri, C. Schurgers, and M. Srivastava, "Adaptive link layer strategies for energy efficient wireless networking," *Wireless Networks*, vol. 5, no. 5, pp. 339–355, 1999.
- [8] M. Zorzi and R. R. Rao, "Energy-constrained error control for wireless channels," in *Proc. IEEE Personal Communications*, pp. 27–33, Dec. 1997.
- [9] K. M. Sivalingam, M. B. Srivastava, P. Agrawal, and J. C. Chen, "Low-power access protocols based on scheduling for wireless and mobile atm networks," in *Proc. IEEE Universal Personal Communications*, pp. 420–433, Oct. 1997.
- [10] J.-C. Chen, K. M. Sivalingam, P. Agrawal, and S. Kishore, "Comparison of MAC protocols for wireless local networks based on battery power consumption," in *Proc. of INFOCOMM*, pp. 150–157, 1998.
- [11] C. F. Chiasserini and R. R. Rao, "Energy efficient battery management," in *Proc. of INFOCOMM*, Apr. 2000.
- [12] J. Rabaey and M. Pedram (Editors), *Low Power Design Methodologies*. Kluwer Academic Publishers, Norwell, MA, 1996.
- [13] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer, 1998.
- [14] T. F. Fuller, M. Doyle, and J. S. Newman, "Relaxation phenomena in lithium-ion insertion cells," *J. Electrochem. Soc.*, vol. 141, pp. 982–990, Apr. 1994.
- [15] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Computer Society LAN MAN Standards Committee, IEEE Std 802.11-1999 Edition.
- [16] R. L. Rivest, "The RC4 Encryption Algorithm (Proprietary)." Mar. 1992.
- [17] "Peripheral Interconnect Bus Architecture." <http://www.omimo.be>.
- [18] F. Balarin et al, *Hardware-software Co-Design of Embedded Systems: The POLIS Approach*. Kluwer Academic Publishers, Norwell, MA, 1997.
- [19] M. Lajolo, A. Raghunathan, S. Dey, and L. Lavagno, "Efficient Power Co-Estimation Techniques for System-on-Chip Design," in *Proc. Design Automation & Test Europe (DATE) Conf.*, pp. 27–34, Mar. 2000.
- [20] "Panasonic Lithium Ion Batteries: Individual Datasheet CGR 17500." <http://www.panasonic.com/industrial/battery/oem/chem/lithion/index.html>.
- [21] "Ethereal 0.8.18." <http://www.ethereal.com>.
- [22] "Microsoft Windows Media Player." <http://www.microsoft.com/windows/windowsmedia>.