Server Controlled Power Management for Wireless Portable Devices

Andrea Acquaviva[1], Tajana Simunic, Vinay Deolalikar,
Sumit Roy
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-82
April 17th , 2003*

low-power,
wireless,
multimedia

In this report we present a new power management infrastructure developed for the Linux operating system release running on the HP's SmartBadge4 wearable computer. Based on this infrastructure, we implemented a new methodology for optimizing the energy consumption of wireless portable devices. The server knowledge of the workload is exploited to efficiently drive the wireless interface of the client during data transfer over the wireless channel. We tested our methodology on HP's SmartBadgeIV wearable device running a HP's MPEG4 streaming video application. Using our technique we measured energy savings of more than 67% when no power management is used on the WLAN interface. In addition, we save as much as 50% of energy with respect to using power management implemented as a part of the 802.11b standard. All of the energy savings are obtained with no performance loss on the video playback.

* Internal Accession Date Only Approved for External Publication
[1] DEIS – University of Bologna, Viale Risorgimento 2, Bologna, 40136, Italy

Server Controlled Power Management for Wireless Portable Devices

Andrea Acquaviva Tajana Simunic† Vinay Deolalikar† Sumit Roy†

DEIS - University of Bologna
Viale Risorgimento 2
Bologna, 40136, Italy

†HP Laboratories
1501 Page Mill Road
Palo Alto, 94304 CA, USA

Abstract

In this report we present a new power management infrastructure developed for the Linux operating system release running on the HP's SmartBadge4 wearable computer. Based on this infrastructure, we implemented a new methodology for optimizing the energy consumption of wireless portable devices. The server knowledge of the workload is exploited to efficiently drive the wireless interface of the client during data transfer over the wireless channel. We tested our methodology on HP's SmartBadgeIV wearable device running a HP's MPEG4 streaming video application. Using our technique we measured energy savings of more than 67% when no power management is used on the WLAN interface. In addition, we save as much as 50% of energy with respect to using power management implemented as a part of the 802.11b standard. All of the energy savings are obtained with no performance loss on the video playback.

1. Introduction

Portable devices spend a considerable amount of energy in order to support power hungry peripherals such as wireless local area network (WLAN) interfaces and liquid crystal displays (LCD). Since battery size and lifetime are finite, it is of primary importance to optimize the energy consumed by these components. To illustrate the magnitude of this problem, we report in Table 1 the percentage of energy consumed by different system components in a typical portable device, the SmartBadgeIV wearable computer [3]. SmartBadgeIV consists of a StrongARM-1110 processor and SA-1111 coprocessor, memory, WLAN interface, audio codec and 2.2" LCD. Clearly, the major energy contributor is the WLAN. As can be seen from the Table, the WLAN energy consumption contribution grows as the amount of network traffic increases. One way to reduce the energy consumption is to use power management included in the 802.11b standard (802.11b PM).

The 802.11b standard for WLAN requires that a central access point (AP) send out a beacon every 100ms followed by a traffic indication map (TIM). Each wireless client actively listens for the beacon in order to sychronize the clock with the AP, and then checks the TIM to find out if needs to send or receive any data. When the client does not need to communicate, it can go into the doze mode until the next beacon. Unfortunately, the protocol power management is not always able to save a lot of power due to the following reasons:

1. **Inefficient scalability.** The energy efficiency of the 802.11b PM decreases as the number of mobile hosts increases since multiple concurrent attempts at synchronization to the beacon of AP cause contention for access to the medium. As a result, the receiver wait time increases, since clients must wait with their radio turned on until they can access the medium and retrieve packets [31].
2. **Performance penalty.** The response time of the wireless link with 802.11b PM grows because of the delay imposed by the sleep periods [31].

	CPU	LCD	WLAN	OTHER
No Traffic	19%	10%	36%	35%
Heavy Traffic	11%	6%	63%	20%

Table 1. Power contributors for SmartBadgeIV

3. **Broadcast traffic dependency.** In a typical wireless network the broadcast traffic can significantly reduce WLAN's chances to enter the doze mode. In Figures 1 and 2 we show the power measurements with 802.11b PM enabled under light (collected at Stanford U.) and heavy (Bologna U.) broadcast traffic conditions. Clearly, as the amount of broadcast traffic increases, the WLAN spends a large amount of energy listening to it even though no other application on the wireless device is running. As a result, very little or no energy savings are obtained when using 802.11b PM in heavier broadcast traffic conditions.

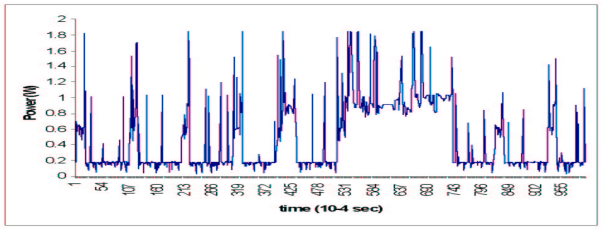

Figure 1. 802.11b PM in light traffic

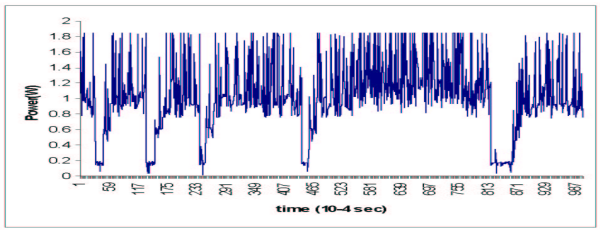

Figure 2. 802.11b PM in heavy traffic

The first two issues with 802.11b PM discussed above can be improved by careful scheduling of communication between the server system and the client WLAN. The problem with WLAN listening to unnecessary broadcast traffic and thus keeping the card from entering doze mode can be solved by just turning off the card. Since the overhead of waking up WLAN from the off state is much larger than the transition from the doze mode, it becomes important for the server to schedule the data transmission so the idle periods are enlarged and thus the transition overhead is minimized.

Many today's wireless local area networks, such as in-home networks, are organized in a client-server fashion. Servers connect multiple WLAN clients to a wired network via APs. Client applications send and receive data in tandem with the server applications (e.g. MPEG4 video decoder, email). While clients run on power constrained device, servers are typically not as power constrained. In addition, servers can have access to the information about both wired and wireless network conditions. For these reasons, servers are the best candidates for efficient scheduling of data transmission to clients.

In this work, we present a power management infrastructure composed by couple of power manager modules, local and remote. The local power manager provides an application program interface to power related parameters of the system. In particular, it can control WLAN power states (off, doze, active), CPU speed and power states (active, idle, sleep). The interface is realized as a set of `ioctl()` and for this reason is enaugh general to be extended to other devices.

We present also a possible utilization of this interface by implementing a server controlled power management strategy. The proposed technique exploits the server knowledge of the workload, traffic conditions and feedback information from the client in order to minimize WLAN power consumption. Our methodology is applicable to a wide variety of applications, ranging from video and audio streaming, to web browsing and email. We define two new entities: a server power manager (server PM) and a client power manager (client PM). Both are implemented as a part of the Linux OS and they provide the power control interface to the applications. Server PM uses the information obtained from the client and the network to control the parameters of 802.11b PM and to perform energy efficient traffic reshaping so that WLAN can be turned off. Client PM communicates through a dedicated low-bandwidth link with the server PM and implements the power controls by interfacing with the device drivers. It also provides a set of APIs that client applications can use to provide extra information back to the server.

To allow a direct control of WLAN power states to the power manager, we modified the WLAN driver to export power related functions to be visible by the other components of the operating system (like the power manager).

In order to illustrate the effectiveness of our approach, we tested our methodology with the MPEG4 streaming video application. MPEG4 video has very tight real-time constraints and high decoding rate variability. Typical implementations of

MPEG4 continually stream video frames, thus keeping the WLAN awake through the whole decoding process. We show that when our methodology is implemented on both the server end (source of MPEG4 data), and on the client end (SmartBadge IV wearable device), we measure savings of more than 67% in power with respect to leaving the card always on, and more than 50% relative to using default 802.11b PM. Even larger savings are possible for applications that inherently have longer idle periods, such as email or web browsing. In addition, our methodology can be easily extended to manage other system components (e.g. I/O peripherals, processor).

The rest of the paper is organized as follows. In Section 2 we present some related work. In Section **??** we give an overview of the methodology we propose and we describe server and client power managers. In Section 4 we present the experiment results and Section **??** concludes the paper.

2. Related Work

A significant amount of work has been published in the context of system-level power management techniques. A major distinction can be made between policies aimed to shut-down components and those that scale down processing voltage and frequency dynamically. In [7, 30, 20, 9] predictive shut-down methods are presented. Run-time policy adaptation is presented in [5]. Shutdown policies are not very effective in periods of extended system activity due to a large power and performance cost incurred during state transitions. For such situations, Dynamic Voltage Scaling, DVS, can provide energy savings [8, 10, 24, 12, 2, 30, 1, 22].

Specific solutions for wireless interfaces have been proposed recently. The wireless network power optimization problem has been addressed at different abstraction layers, starting from physical to system and application level. In [6] authors propose energy efficient channel coding and traffic shaping to efficiently exploit battery lifetime of portable devices. Physical layer aware scheduling algorithm aimed at efficient management of sleep modes in sensor network nodes is illustrated in [27]. Energy efficiency can be improved at the data link layer by performing adaptive packet length and error control as in [13]. At the protocol level there have been attempts to improve the efficiency of the standard 802.11b and proposals of new protocols [17, 29]. A survey of energy efficient network protocols can be found in [11]. Packet scheduling strategies can also be used to reduce the energy consumption of trasmit power. In [23] authors propose the E^2WFQ scheduling policies based on DMS (Dynamic Modulation Scaling) [**?**]. A small price in packet latency is traded-off for the reduced energy consumption.

At the system level researchers have proposed energy-performance trade-offs based on application needs [18]. Several authors exploit the energy-QoS trade-off [**?**, 31, 16]. A different approach is taken in [26]. Here authors perform transcoding and traffic smoothing at the server side by exploting estimation of energy budget at the clients. A completely new communication system is presented in [28]. The system composed by server, clients and proxy that reduces the energy consumption of 802.11b compliant portable devices by exploiting a secondary low-power channel. Since multimedia applications are often most demanding of system resources, such as WLAN, a few researchers studied the cooperation between multimedia applications and the OS with the goal of controlling system resources in order to save energy [14, 4, 25, 15].

We present a new methodology where the server knowledge of workload is exploited to control the power configuration of the radio interface. Compared to physical and protocol layer strategies, the power control is performed at the application level, so it does not requires hardware modifications. Compared to client-centric approaches, we exploit additional information available at the server, and thus we obtain large energy savings without loosing performance. Moreover, with respect to previous application-driven policies, our infrastructure can be used with a wide range of application, since it exploits parameters that are not related to a particular application, but are common to a wide range of network application. In the following section we outline our power management methodology.

3. Server Controlled Power Management

Server controlled power management methodology exploits the server knowledge of the workload, the traffic conditions and the feedback information from the client in order to minimize the WLAN power consumption. Our approach has quite a few advantages:

1. The server can schedule communication sessions with the client based on the knowledge it has of both wired and wireless network conditions. This is espeically important when multiple clients with different application needs are present. For example, the server can make sure that it communicates with each client only under favorable wireless channel conditions, instead of attempting to stream data continually. In addition, it can easily account for the differences

in bandwidth needs of various client applications. In contrast, client centric power management has a much more limited view - it can only observe the patterns of its own traffic arrivals.

2. By monitoring traffic conditions, the server can decide when to enable the 802.11 PM. This is especially important for situations when broacast traffic needs to be monitored, such as when an operating system needs to update the arp cache.
3. Running on a power unconstrained system, the server can take complex power management decisions with minimum overhead to the client.
4. By performing on-time wake-up of the client WLAN, the server avoids the performance penalty typically incureed by the client-centric approaches.
5. Our application driven infrastructure can be used to manage power consumption in stand-alone and ad-hoc applications. Moreover, the power control strategy can easily be exended to include other system components, such as other peripherals or the processor.

In order to exploit the extra information available at the server and the client, the traditional client-centric power manager model had to be extended. Thus, we define two different power managers, one running on the client (Client PM) and the other on the server (Server PM). The two PMs exchange power control information through the wireless channel using a dedicated TCP connection. The overall structure of the system is shown in Figure 3. The client PM interfaces directly with device drivers on the portable device. In addition, it also interacts with the client applications in order to collect the application dependent information. The server PM interfaces with the server application and the client PM.

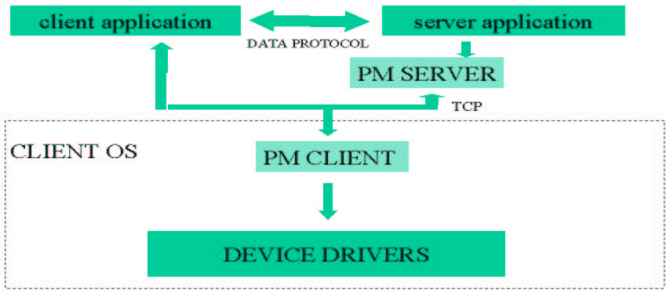

Figure 3. Server Controlled PM Architecture

The communication protocol between server and client is shown in Figure 4. Power control commands are issued by the server PM, interpreted by the client PM and translated in the appropriate device driver's function calls. Upon request from the server PM, device specific information is fetched by the client PM through device driver's calls. In addition, the application specific information can be retrieved by the client PM via application API calls. More details on both server and client PM are discussed in the next subsections.

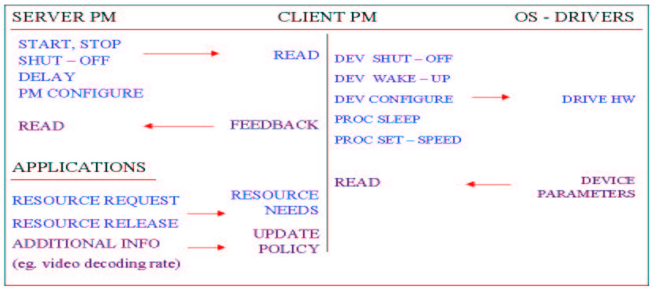

Figure 4. Communication protocol

3.1. Server Power Manager

The server power manager interfaces with the application server in order to exploit application specific information available on the host system, in addition to the knowledge of the overall network conditions and the specific feedback information

provided by each client. Based on all this knowledge, the server PM can control the power configuration of the client's WLAN card by communicating with the client PM. The server uses the following power control commands: i) switch-off the WLAN; ii) set the off time; iii) enable the 802.11b PM policy; iv) set 802.11b PM parameters, such as the period between wake ups and the timeout before going back to doze mode **??**.

The functions provided by the server PM to the application server are outlined below:

- `pmWriteCommand(command)`: send a remote command to the client
- `pmServerInit(command)`: initialize the server and starts main threads
- `pmServerClose(command)`: close the server connection to the remote client
- `pmReadCommand(command)`: receive a remote command from the client

When `pmServerInit()` is called, two threads are created and resumed: `pmAcceptingThread` and `pmReceivingThread`. Their function is to accept connections and informations from the client power manager. The `pmReceivingThread()` uses `pmReadCommand` to interpret an information coming from the client PM. `pmWriteCommand()` instead is used directly by the application server to send commands to the client power manager. Both `pmReadCommand()` and `pmWriteCommand()` exploit the established TCP/IP connection to transfer the command string. For example, it is used to request the shutdown of the WLAN.

The main server PM characteristics are outlined below.

Client Adaptation. Once loaded, the server PM make available to the application server a set of additional APIs. The application server can decide when to initialize the server PM by calling an initialization routine (`pmServerInit()`). This places the server in a state of waiting for incoming client PM requests. Each accessing client PM has a dedicated TCP connection to the server PM. As soon as the two are connected, the clients PM informs the server PM of the client's application and network interface specific information. Examples of device specific information are the WLAN card status and its on/off transition time. Application specific information examples are client's input buffer size and the expected value and variance of the service rate.

Traffic Adaptation. Server PM monitors both the wired and wireless the traffic conditions with minimum overhead. By accounting for the broadcast packet rate, the server decides when to enable the 802.11b PM. In very light traffic conditions, the 802.11b PM can be used instead of a switch-off policy as we will see in the Results section.

Traffic Shaping. The server PM schedules the transmission to the client in bursts in order to compensate for the client's performance and energy overheads during the transitions between on and off states. The client's WLAN card is switched off once the server has sent a burst of data that will keep the client application busy until the next communication burst. Burst size and delay between bursts must be precomputed at the server. The goal is to have a delay large enough to almost empty the client input buffer and small enough burst size to avoid overflow while keeping the buffer sufficiently filled. In this way we can maximize the off time of the card and reduce the number of transitions between on and off states. An illustration of the shut-off policy is shown in Figure 5.

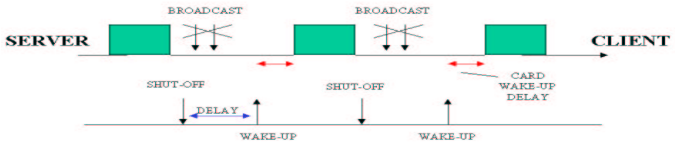

Figure 5. Example Policy

We can compute the time until buffer is emptied:

$$T_{burst} = \frac{1}{\lambda_{s,mean}} \cdot SIZE_{burst} \tag{1}$$

where the average service rate (or buffer depletion rate) at the client is $\lambda_{s,mean}$, and the burst size is $SIZE_{burst}$. If we add to that the time needed for the transition between WLAN on and off states, T_{tran}, and the "cushion" time so that the buffer never gets completely empty, $T_{cushion}$, we can get the total delay we should place between bursts, D_{burst} as shown

below. The cushion time helps accomodate variations of the service rate and arrival rate, and is defined as a percentage of the total delay. We found 10% to be sufficient for most test cases.

$$D_{burst} = T_{burst} + T_{tran} + T_{cushion} \tag{2}$$

We also need to determine if turning off the client's WLAN is beneficial before scheduling the communication burst. Thus, we compute the total energy we could save if the client's WLAN is turned off. We define $\lambda_{a,mean}$ to be the average arrival rate of data into the client's buffer before traffic reshaping. It corresponds to the average network bandwidth. When $\lambda_{a,mean} > \lambda_{s,mean}$ the system is unstable as it would cause an overflow of the client's input buffer. With traffic reshaping we introduce extra idle time between bursts that in turn decrease the average arrival bandwith and allow us to save power by turning off the client's WLAN. The total time the system is off corresponds to the time the client program takes to consume the bufferized data as computed before (D_{burst}).

The total energy spent with traffic reshaping and turning off the client's WLAN can be written as following:

$$E_{tot} = P_{on} \cdot T_{on} + P_{off} \cdot T_{off} - N_{tran} \cdot E_{tran} \tag{3}$$

where E_{tran} is the transition energy (hardware dependent) and N_{tran} is the number of transitions between on and off states:

$$N_{tran} = \frac{SIZE_{tot}}{SIZE_{burst}} \tag{4}$$

Now, if we define $\lambda_{a,mean}$ the average arrival rate at the client input buffer, the total time to send a burst of data to the client will be:

$$T_{on} = \frac{SIZE_{burst}}{\lambda_{a,mean}} \tag{5}$$

We can write the total energy as:

$$E_{tot} = P_{on} \cdot \frac{SIZE_{burst}}{\lambda_{a,mean}} + P_{off} \cdot \frac{SIZE_{transfer}}{\lambda_{s,mean}} + P_{tran} \cdot T_{tran} \tag{6}$$

This must be compared to the energy spent being always on.

In addition to scheduling communication so that the client's WLAN can be turned off with no performance overhead, we can also perform the same scheduling while 802.11b PM is enabled. This allows us to save energy during the burst periods, but the overall burst time is grows because of the decreased responsivness and increased contention probability between data and broadcast packets.

In this section we described the server power manager and we outlined the three main tasks the server PM performs: client adaptaion, traffic adaptation and traffic shaping. In the next section we outline the main characteristics of the client power manager.

3.2. Client Power Manager

Main job of the client power manager is to communicate on one end with the server PM, and on the other end to interface with the device drivers and client's applications.

The interface between the power manager and the applications is realized through a set of dedicated `ioctl` system calls. For each requested functionality, there is a particular command. The main commands are outlined below:

- **CPOWMAN_AIRO_OFF**: request WLAN shutdown
- **CPOWMAN_AIRO_ON**: request WLAN wake-up
- **CPOWMAN_AIRO_OFF_TIME**: set the off time of the WLAN
- **CPOWMAN_AIRO_READ_SNR**: read the SNR level from card's registers

The main tasks of the client's power manager are outlined below.

Server Interface The client application decides when to set-up the communication between the server and the client PM by calling `pmClientInit()`. Once established, the client's application provides the information to be forwarded to the server such as the buffer size and the depletion rate through `pmClientSend()`. Similarily, the device drivers feedback the main characteristics of the devices to be managed, such as the transition time between on and off states for client's WLAN.

Device Interface The client PM calls the appropriate device driver function depending on the command sent by the server PM. Possible actions taken by the client are changing the parameter of the 802.11b PM, switching on and off the WLAN, and reading the interface statistics such as the signal to noise ratio. In addition, the client PM can also interact with the CPU by changing its power mode or setting its clock speed.

Application Interface Applications can feedback information that can be exploited by both the server and the client PM. An example is sending the current backlog level to the server, so the server knows exaclty how much data to provide in a burst in order to refill the buffer. In addition, the the application can directly request a WLAN wake-up when its input buffer reaches a minimum value.

Application Driven Infrastructure The client power manager can also be used in standalone mode (with no server). While in this mode, the applications provide their resource needs to the power manager. The PM then turns on or off devices appropriately. In this way some of the overhead needed to turn on a device can be masked, as many applications have extra latency due to the initial set-up.

In this section we described the main characteristics of server and client power managers and how they interact to provide energy efficient power management. In the next section we outline the experimental results we obtained when we implemented our methodology on the MPEG4 video streaming client-server system.

4. Results

In this section we present the results of testing our methodology with the MPEG4 streaming video application **??**. MPEG4 video represents a difficult test condition for any WLAN power management methodology because of its tight real-time contraints and the decoding rate variability.

The streaming media server used for this work is a research prototype developed by the Streaming Media System Group at Hewlett-Packard Laboratories. The control handshake between the media client and server uses the Real Time Streaming Protocol (RTSP) **??** for session initiation and termination. The media data units are carried using the RTP **??** protocol over UDP. Note that this protocol does not grant delivery of packets (just like the underlying UDP protocol), however the packets are numbered in a sequence, and there is a back channel that sends regular RTCP messages with packet reception statistics. The MPEG4 file format includes the media presentation, as well as structure to facilitate streaming of media. Timestamps on the individual video frames are used to determine the deadline for the data packets that have to go from the server to the client over the network. When a client has buffering capabilities, the server can exploit them to reschedule packet transmission times based on a cost function.

In our experiments the media server transmits MPEG4 compressed video data to a client portable device via wireless link. Since our client is able to decode data faster than real time, it has to implement the decoding rate control. Thus, the decoder takes data from the input buffer with a rate dependent on the output frame rate (e.g. 15 frames/sec). Since the number of packets per frame is variable, the depletion rate of the buffer varies as well. For both streams, the buffer depletion rate is much less than the arrival rate. We measured the average depletion rate of about 25Kbit/sec, while the average arrival rate is 1Mbit/sec. Because of such a large difference in depletion and arrival rates, the server can reduce client's power consumption by scheduling bursts of data transmission followed by longer idle periods during which the client's WLAN can be turned off. We use the equations presented in the previous section to compute the the length of idle time between bursts. The burst size varies in our experiments, but the maximum is set to 80 RTP packets, as it represents the limit of the UDP buffer. When burst size is set to 80 RTP packets, the following idle time (or delay until waking up the client's WLAN card) is about 2 seconds. Based on these parameters, the server transmits a burst of data to the client followed by the switch off WLAN command together with the length of time until the next transmission session. The client responds by turning off the WLAN and turning it back on at the pre-scheduled time. As a result, no performance penalty due to longer turn on time is incurred.

Our client is the HP's SmartBadgeIV wearable device **??**, equipped with a CISCO Aironet 350 PCMCIA WLAN card. Cisco WLAN card's off/on transition time has been measured at 300ms with a average transition power consumption of 0.1W. MPEG4 server is a PentiumIII PC connected directly to the wireless access point. In order to perform repeatable experiments, we isolated our network from other network traffic. We introduced the broadcast traffic in a controlled manner into our isolated network by using real traces collected from other open networks (CSL LAB-Stanford, HP LABS-Palo Alto,

DEIS-Bologna). All power measurements are performed with a DAQ board connected to the client and to the PC running a program accumulating and averaging the current and voltage samples (10 ksample/sec).

Experimental results can be divided in two parts. First we computed the average power consumed by the WLAN card when receiving the MPEG4 stream in different burst sizes and for medium broadcast traffic conditions (HP LABS). We performed this experiment for three different situations: i) WLAN always on; ii) WLAN with only 802.11b PM; iii) WLAN controlled by the server. As shown in Figure 6, the server controlled approach (blue plot) saves 67% of average power compared to leaving WLAN always on (green plot) and 50% compared to the default 802.11b PM (red plot). The average power savings increase as the burst size increases since that enables longer times between bursts, and thus better compensation for the transition delay between WLAN's on and off states. Note that in all three cases the video plays back in the same amount of time, as it continues to be real time. Thus the reported average power savings directly correspond to energy savings.

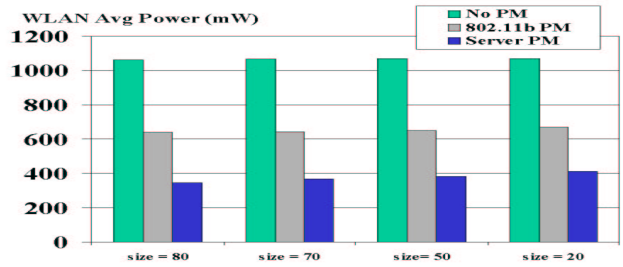

Figure 6. Power consumption comparison for different burst sizes

The next set of results, presented in Figure 7, we compare the average power consumed during streaming video under two different broadcast traffic conditions. In the plot we show the difference between using only the default 802.11b PM and server controlled switch off policy for the WLAN card (with no 802.11b PM). The switch off policy is almost insensitive to the traffic conditions, while the 802.11b PM performs better only in light traffic conditions. Clearly, our server controlled power management approach is more efficient than either policy alone as it always selects the best of the two policies.

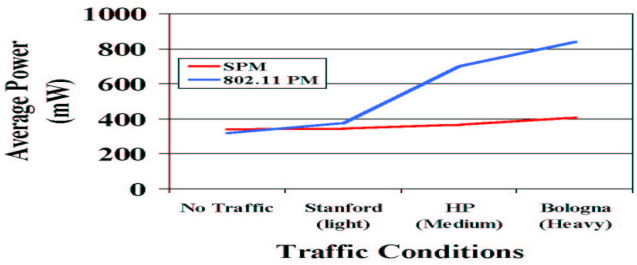

Figure 7. 802.11b PM vs Server Controlled PM with traffic levels

5. Conclusions

In this work we presented a new methodology to improve the energy efficiency of the portable wireless systems that operate in client-server fashion. Our methodology enables the server to exploit the knowledge of the workload, traffic conditions and feedback information from the client in order to minimize client's WLAN power consumption. We tested our methodology on HP's SmartBadgeIV wearable device running MPEG4 streaming video application. Our server controlled approach saves 67% of energy as compared to leaving the client's WLAN card always on, and 50% as compared to the simple 802.11b PM policy.

References

[1] A. Acquaviva, L. Benini, B. Riccó, "Software Controlled Processor Speed Setting for Low-Power Streaming Multimedia," *IEEE Transactions on CAD*, pp. 1283–1292, November 2001.

[2] A. Sinha, A. Chandrakasan, "Energy efficient real-time scheduling," *ICCAD*, pp. 458–463, Nov 2001.

[3] www.it.kth.se/ maguire/badge4.html

[4] F. Bellosa, "Endurix: OS-Direct Throttling of Processor Activity for Dynamic Power Management," *Technical Report TR-I4-99-03, University of Erlangen*, June 1999.

[5] E. Chung, L. Benini and G. De Micheli, "Dynamic Power Management for non-stationary service requests", *Proceedings of DATE*, pp. 77–81, 1999.

[6] C. Chiasserini, P. Nuggehalli, V. Srinivasan, "Energy-Efficient Communication Protocols", *DAC*, pp. , 2002.

[7] C. H. Hwang, A. C. H. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation," *ICCAD*, pp. 28–32, November 1997.

[8] I. Hong, M. Potkonjak, M. B. Srivastava, "On-Line Scheduling of Hard Real-Time Tasks on Variable Voltage Processors," *ICCAD*, pp. 653–656, November 1998.

[9] S. Irani, R. Shukla, R. Gupta, "Competitive Analysis of Dynamic Power Management Strategies with Multiple Power Saving States," *DATE*, pp. 117–123, March 2002.

[10] T. Ishihara, H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processor," *ISLPED*, pp. 197–202, August 1998.

[11] C. Jones, K. Sivalingam, P. Agrawal, J. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks", *Proceedings of DATE*, pp. 77–81, 1999.

[12] S. Lee, T. Sakurai, "Run-time voltage hopping for low-power real-time systems," *ISLPED*, pp.806–809, July 2000.

[13] P. Lettieri, C. Schurgers, M. Srivastava, "Adaptive Link Layer Strategies for Energy Efficient Wireless Networking", *Wireless Networks*, no. 5, pp.339 355, 1999.

[14] J. Lorch, A. J. Smith, "Software Strategies for Portable Computer Energy Management," *IEEE Personal Communications*, pp 60–73, June 1998.

[15] Y. Lu, L. Benini, G. De Micheli, "Operating System Directed Power Reduction," *ISLPED*, pp 37–42, July 2000.

[16] C. Luna, Y. Eisenberg, R. Berry, T. Pappas, A. Katsaggelos, "Transmission Energy Minimization in Wireless Video Streaming Applications,"

[17] R. Krashinsky, H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown", *MOBICOM*, pp. , 2002.

[18] R. Kravets, P. Krishnan, "Application-Driven Power Management for Mobile Communication,"

[19] R. Min, A. Chandrakasan, "A Framework for Energy-Scalable Communication in High-Density Wireless Networks,"

[20] Q. Qiu and M. Pedram, "Stochastic Modeling of a Power Managed System-Construction and optimization", *IEEE Transactions on CAD*, pp. 1200–1217, vol. 20, Oct 2001.

[21] "Voltage Scheduling in the lpARM Microprocessor System," *ISLPED*, T. Pering, T. Burd, R. Brodersen, July 2000.

[22] J. Pouwelse, K. Langendoen,H. Sips, "Energy Priority Scheduling for Variable Voltage Processors," *ISLPED*, pp. 28–33, Huntington Beach, USA 2001.

[23] V. Raghunathan, S. Ganeriwal, C.Schurgers, M. Srivastava, "E^2WFQ: An Energy Effiecient Fair Scheduling Policy for Wireless Systems", *ISLPED*, pp. , 2002.

[24] Y. Shin, K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems," *DAC*, pp. 134–139, June 1999.

[25] J. Flinn, M. Satyanarayanan, "Energy-aware adaptation for mobile applications," *In Proceedings of SOSP*, December 1999.

[26] P. Shenoy, P. Radkov, "Proxy-Assisted Power-Friendly Streaming to Mobile Devices,"

[27] E. Shih, P. Bahl, M. Sinclair, "Dynamic Power Management for non-stationary service requests", *MOBICOM*, pp. , 2002.

[28] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy Efficient Wireless Sensor NEtworks", *SIGMOBILE*, pp. , 2001.

[29] K. Sivalingam, J. Chen, P. Agrawal, M. Srivastava, "Design and Analysis of low-power access protocols for wireless and mobile ATM networks", *Wireless Networks*, no.6, pp.73 87, 2000.

[30] T. Simunic, L. Benini, P. Glynn, G. De Micheli, "Event-driven Power Management," *IEEE Transactions on CAD*, July 2001.

[31] E. Takahashi, "Application Aware Scheduling for Power Management on IEEE 802.11"