Power Management Techniques for Mobile Communication

Robin Kravets
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30309
robink@cc.gatech.edu

P. Krishnan
Bell Labs, Lucent Technologies
101 Crawfords Corner Rd.
Holmdel, NJ 07733-3030
pk@research.bell-labs.com

Abstract

In mobile computing, power is a limited resource. Like other devices, communication devices need to be properly managed to conserve energy. In this paper, we present the design and implementation of an innovative transport level protocol capable of significantly reducing the power usage of the communication device. The protocol achieves power savings by selectively choosing short periods of time to suspend communications and shut down the communication device. It manages the important task of queuing data for future delivery during periods of communication suspension, and decides when to restart communication. We also address the tradeoff between reducing power consumption and reducing delay for incoming data.

We present results from experiments using our implementation of the protocol. These experiments measure the energy consumption for three simulated communication patterns and compare the effects of different suspension strategies. Our results show up to an 83% savings in the energy consumed by the communication. This can translate to a 6-9% savings in the energy consumed by an entire high end laptop or a savings of up to 40% for current hand-held PCs. The resulting delay introduced is small (0.4-3.1 seconds depending on the power management level).

1 Introduction

In today's world of mobile communications, one of the most precious commodities is power. The mobile host can only operate as long as its battery maintains power. New machines are being made to use less power allowing for smaller batteries with smaller capacities. The trend in mobile computing is towards more communication-dependent activities, with mobile users switching from traditional wired Ethernet communication to wireless communication (using wireless Ethernet cards, for example). When inserted, many wireless communication devices consume energy continuously. Although dependent on the specific machine and wireless device, this energy consumption can represent over 50% of total system power for current hand-held computers and up to 10% for high-end laptops. These trends make it imperative that we design power-efficient communication subsystems.

Various techniques, both hardware and software, have been proposed to reduce a mobile host's power consumption during operation. Most software-level techniques have concentrated on non-communication components of the mobile host, such as displays, disks and CPUs. In particular, researchers have looked at methods to turn off the display after some period of inactivity (as often implemented in BIOS or screen savers), to spin down the hard disk of the mobile host [7, 9, 16], and to slow down or stop the CPU depending on work load [8, 17, 24]. The principle underlying the techniques for controlling these components is to estimate (or guess) when the device will not be used and suspend it for those intervals. Stemm et. al [23] have identified the problem of excess energy consumption by network interfaces in hand held devices, and have provided trace-driven simulation results for simple software-level time-out strategies. The new IEEE 802.11 standard that is being adopted by some vendors adopts lower level solutions (at the MAC and PHY layer) to support idle-time power management. Hardware-level solutions for managing the communication device focus on modulating the power used by the mobile transmitter during active communication [21, 22, 19].

Our research presented in this paper focuses on software-level techniques for managing the mobile host's communication device through suspension of the device during idle periods in the communication. We present a novel transport level protocol for managing the suspend/resume cycle of the mobile host's communication device in an effort to reduce power consumption. The management of communication devices creates a new and interesting challenge not present when managing other devices' power consumption. Similar to hard disks and CPUs, the communication devices continuously draw power unless they can be suspended. A suspended hard disk or CPU can be restarted by any user requiring that device. However, when a communication device is suspended, the mobile host is effectively cut off from the rest of the

MOBICOM 98 Dallas Texas USA

network. A mobile host with a suspended communication device can only guess about when other hosts may have data destined for it. If the suspension of the mobile host's communication does not match prevailing communication patterns, the isolation can cause buffers to overflow both in the mobile host and in other hosts trying to communicate with it. Additionally, other hosts may waste precious resources trying to communicate with the mobile host if they have no knowledge about whether or not the mobile host's communication is suspended.

Our goal is to provide mechanisms for managing and reducing the power consumption of the communication device. We present a simple model for mobile communication that provides adaptable functionality at the transport layer for suspending and resuming communication. By exposing this functionality to the application, we enable application-driven solutions to power management. Power savings are attained by suspending communications and the communication device for short periods of time. During these suspensions, data transmissions are queued up in both the mobile host and any other host trying to communicate with the mobile host. The key to balancing power savings and data delay lies in identifying when to suspend and restart communications. By abstracting power management to a higher level, we can exploit application-specific information about how to balance power savings and data delay.

Intuitively, power conservation is achieved by accumulating the power savings from many small idle periods. We, however, need to be careful to monitor any additional energy consumption caused while executing the suspend/resume strategies. Additionally, we need to consider the effect on other hosts who are trying to communicate with the suspended mobile host. A base station using our protocol has enough knowledge about the state of the mobile host to know when it is suspended and can use this information to help employ scheduling techniques. We implemented our protocol and experimentally determined its effect on power consumption and the quality of communication. Using three simulated users designed to capture typical mobile communication patterns, we obtained 48–83% savings in the power consumed by the communication subsystem, while introducing a small additional response delay (0.4–3.1 seconds depending on the power management level) that is acceptable for many applications, like web browsing.

In Section 2, we present our basic mobile communication model, and the important issues in power management for communication. In Section 3 we present our power management protocol and discuss the effect of timing issues on the effectiveness of our protocol. Section 4 describes our experimental setup and the communication patterns used in our experiments. We then present measurements from the implementation of our protocol and discuss the results in the context of several real systems. In Section 5, we discuss adaptive control strategies.

2 Communication Model and Power Management

The introduction of wireless links into communication systems based on wired links has posed a number of problems. These problems include different loss characteristics and different bandwidth capabilities on the wired and the wireless line, synchronization of disconnected operations, and issues involving packet forwarding. These problems pose significant challenges for end-to-end communication protocols. Two types of models have been studied [2]. The first model exploits the natural hop existing in the communication route to a mobile host. Standard communication protocols are used by wired hosts to a base station and specialized protocols are used for the final hop from the base station to the mobile hosts [1]. The second model utilizes and tunes existing end-to-end protocols, providing help and hints along the way [3].

We target our approach at the transport layer, where we provide a set of mechanisms that allow communication to be suspended and resumed. We assume a model where the mobile host is communicating with the rest of the network through a base station. This base station may be a proxy, or it may be the connection point for end-to-end communication with other hosts. Often, dealing with mobility does not fit into the standard seven layer model. By exposing power management techniques to the application, we provide a system-level solution aimed at end-to-end communication. For our experiments in this paper, we concentrate on the communication between the mobile host and the base station, and for clarity assume that all communication to and from the mobile host is directed through one specific base station. This work can be extended to include changing base stations through techniques similar to those used in [1, 3].

Current wireless communication devices typically operate in two modes: transmit mode and receive mode. The transmit mode is used during data transmission. The receive mode is the default mode for both receiving data and listening for incoming data. Much of the time, the wireless communication device sits idle in receive mode, and, while the power required for reception is less than the power required for transmission, this power consumption is not negligible. The IEEE 802.11 standard provides for some power management at the lower (MAC or PHY) layers. Compliant cards can exchange information about outstanding data to decide on when to wake up suspended cards. There are ongoing efforts to provide IEEE 802.11 compliant support for power management by introducing new features into the next generation wireless communication cards [11]. An approach that relies solely on techniques provided by the device (e.g., the 802.11 standard) cannot take application specific information into consideration when determining power management strategies. Researchers have also considered hardware-level solutions to provide low power communication capabilities [21, 22, 19]. Such solutions reduce the power cost of operating in either one of the modes, and are orthogonal to our approach which addresses the amount of time the device spends in each mode.

Logical areas to look for software-level power conservation in communication are two-fold. Since data transmission is expensive, we can reduce the time spent in transmission. This can be achieved by data reduction techniques and intelligent data transfer protocols. The obvious technique of data compression reduces the amount

of transmission time, but requires additional CPU cycles for performing compression. The connection between compression and communication rates is studied in [5]. Through simple experiments, we observed that, considering the current power requirements of CPUs versus wireless communication devices, the benefit in terms of power savings from reduced communication time often outweighs the increased energy consumption costs for compression. Intelligent data transfer protocols can be used to reduce the effect of noisy connections that cause power-expensive retransmission of lost messages. Our continuing research addresses the assessment of the effects of different techniques for data reduction, including reduced reliability requirements, and their effect on both power reduction and communication quality.

The second area, and the emphasis of this paper, is the cost of leaving the communication device sitting idle during periods of no communication activity. During such idle periods, the communication device draws power listening for incoming data. Our goal in this work is to reduce the amount of time the device sits idle drawing power by judiciously suspending it. Suspending a wireless communication device is similar to slowing a CPU in that there are some small power costs associated with suspension and resumption. As mentioned in Section 1, the difficult part here is to deal with when to suspend and resume the communication device, how to deal with the mobile host being unreachable at times, and how to address the issue of not losing en-route data. Our protocol and its implementation presented here address these problems. Since the protocol itself generates additional communication during these idle periods, there needs to be a balance between when it is beneficial to use the power management techniques, and when we should leave the device on continuously.

In contrast to the solutions proposed by the IEEE 802.11 standard, we believe that power management should be controlled by the mobile host, potentially even the application. By providing power control at the transport layer (or above), we can provide power management interfaces to the application, allowing the application to better control the communication, enabling adaptive power management driven by the needs of the application. Specifically, communications using the IEEE 802.11 standard will always pay the overhead of delays imposed by using power management, while our techniques allow the application to determine when such delays are too high, and so adapt power management levels. Stemm et. al [23] have also investigated methods for reducing power consumption of network interfaces, specifically targeting their research at hand-held devices. Their research suggests application-specific solutions to such problems. In contrast, our research provides a general solution capable of hosting various strategies, both static and adaptive. Our measurements are with a real implementation of a power management protocol in an experimental setup. We are, therefore, able to observe the effects of the queuing of data and the real effect of extra energy consumption by such a protocol. We measure the power consumption in the context of the entire system, considering such costs as message processing and disk accesses, for various simulated workloads that we expect mobile users to perform.

3 Communication-Based Power Management

Currently, a typical mobile host leaves its wireless Ethernet card in receive mode during the time it is not being used, unless the user explicitly removes the card. The technique described in this section provides mechanisms to extend battery lifetime by suspending the wireless Ethernet card during idle periods in communication. At the heart of the technique lies a protocol where the mobile host acts as a master and tells the base station when data transmission can occur. When the mobile host wakes up, it sends a query to the base station to see if the base station has any data to send. This permits communication device suspension at the mobile host, and enables the implementation of communication scheduling techniques at the base station. The suspend/resume cycle results in bursts of communication that may be followed by periods of inactivity. Although producing such bursty communication may incur additional delay, bursty communication patterns lend themselves well to efficient scheduling techniques.

With the suspension of a communication device, a mobile host will experience an additional delay in data transmission since data on both the sending and receiving sides may be held up during suspension. The mobile host can monitor its own outgoing communication patterns to insure that, despite these suspension times, communication continues smoothly without buffer overflow. The base station, on the other hand, has no means to restart communication if it notices that it is running out of buffer space. It is up to the mobile host to understand the base station's expected communication patterns so that the buffers at the base station do not overflow. In order to efficiently use our power management techniques, our communication layer must monitor the communication patterns of the mobile host and match the suspend/resume cycle to these patterns.

The protocol we describe in this section allows a mobile host to suspend a wireless communication device. Periodically, or by request from the application, the protocol wakes up and reinitiates communication with the base station. In the rest of this section, we will describe our power management protocol in detail and discuss the significance of some of the timing parameters. Appendix A describes in detail the commands used by the protocol and the possible states and state transitions for both the master and slave.

3.1 Power Management Control Protocol

In this protocol, the mobile host is the master and the base station acts like a slave. The slave is only allowed to send data to the master during specific phases of the protocol. During non-transmit phases, the slave queues up data and waits for commands from the master. Idle periods for both the master and the slave can be detected through the use of idle timers or indicated to the protocol from the application. In the protocol state diagrams for the master and the slave (Figure 1 and Figure 2), IN: indicates an input event that can be either an incoming message or a timeout, Q: indicates the state of the queue, and OUT: indicates an outgoing response message.

As shown in Figure 1, the slave is initialized to be in the SLEEPING mode. It can only leave that mode upon a WAKE_UP message from the master. If the slave has data to send, it will enter the SEND_RECV mode. The slave will stay in this mode until it has detected that it has no more data to transmit, whereupon, it will send a DONE message to the master, enter the RECEIVING mode, and continue receiving until it receives a SLEEP message. If during this time the slave detects that there is new data to transmit, it will send a NEW_DATA message to the master and enter the RECEIVING_WAIT mode. The slave can only start to transmit when it receives a WAKE_UP message. If a SLEEP message is received first, the waiting data stays buffered and is not transmitted until the next resume cycle.

Although the state diagram for the master (Figure 2) is much more complex, we can see that the states may be partitioned into three sets. The first set (SLEEPING) concerns the master when it is sleeping. When the master is in the SLEEPING mode, it can be woken up by one of two triggers: a wakeup timer or new data to transmit. If the wakeup timer expires, the master sends a WAKE_UP message along with any new data to the slave. If there is new data to transmit to the slave before the wakeup timer expires, the master has the option to wake up and transmit this new data, or continue sleeping and queue up the data until the timeout expires.

The second set of states (SENDING_WAIT, WAITING, and WAIT_FOR_OK) concerns the master when it is waiting for a response from the slave about whether or not the slave has data to send. In the SENDING_WAIT mode, the master is transmitting data and in the WAITING mode it has no data to transmit. When the master receives a response from the slave in the form of a DATA or a NO_DATA message, the master enters the appropriate state in the third set. Additionally, if while in the SENDING_WAIT mode an idle timer expires indicating that the master has no more data to send, the master enters the WAITING mode and continues waiting for a response from the slave. In the WAIT_FOR_OK mode, the master has told the slave that it should sleep and is waiting for a SLEEP_OK message.

When the master is in one of the final set of states (SENDING, SEND/RECV, and RECEIVING), it is actively sending and/or receiving data. In the SENDING mode, the master may receive a NEW_DATA message from the slave. The master responds with a WAKE_UP message and enters the SENDING_WAIT mode. When neither the master nor the slave have any more data to send, the master sends a SLEEP message and enters the WAIT_FOR_OK mode.

Wireless connections are very susceptible to interference from both external devices and other wireless devices using the same settings or talking to the same base station. By using this protocol, we provide the base station with useful information about the communication patterns of the mobile host. Although not required by the protocol, the master can inform the slave of its sleep time, or the slave can suggest appropriate sleep times to the master. If the protocol is used such that only prespecified timeouts trigger restarting communication, the slave can design a communication scheduling algorithm based around the known sleep time of the master. Additionally, if the sleep times for the master are sufficiently long, the slave can save any data destined for the master to disk. This will free the buffer space being used by the data destined for the master so it can be used for other active communications.

3.2 Timing Considerations

Timing is a key issue for both the performance of the mobile host as well as the amount of power that can be saved. If the wireless Ethernet card is suspended too often, the user will see lags in data transfer performance. On the other hand, if it is not suspended long enough, the gain in battery life time may be undetectable.

In order to determine when the card should be suspended, the protocol needs to determine the communication patterns for both sender and receiver. There are two ways by which idle periods in the communication can be detected. The first, and simplest, is when the application can actually inform the protocol that it doesn't have any data to send. This requires a more complex application that has information about its communication patterns. The second method is to use a timer set with a *timeout period*. If the timer expires and no communication has occurred since the last expiration, the protocol concludes that there is an idle period in the communication. The appropriate timeout period depends on the requirements of the application. Timeout periods that are too short may cause the protocol to go to sleep prematurely, resulting in poor response time for applications dependent on communication. On the other hand, timeout periods that are too long may cause the protocol and the communication device to remain active for unnecessarily long periods of time, wasting precious energy.

The other timing parameter is the *sleep duration* which defines how long the master should keep the communication suspended. The appropriate sleep duration also depends on the requirements of the application. Longer sleep periods will cause longer lags in any interactive applications. Shorter sleep periods will not extend battery lifetime appreciatively. The application needs to determine the appropriate tradeoff for battery lifetime versus delay. In many instances, the expected time and data size for the response to a request initiated by the mobile host can be estimated. This includes, for example, applications like mail, web browsing, and file transfer. In this context, hints provided by the application could be very helpful. In our experiments reported in Section 4, we examine the effects of fixed timeouts that require no application support and can be implemented within the transport layer. Adaptively varying the timeouts or using learning techniques are discussed in Section 5.

A mobile host that is running multiple applications cannot base its power strategy on the expected communication patterns of a single application. In this situation, the power management protocol must take hints about sleep/wake up durations for all executing applications. By exposing power management to the application, and hence to the user, our power management protocol can be guided in the appropriate allocation of resources.

A final consideration is the time required to wake up and shut down the specific wireless network card. Our protocol is designed to be independent of the specific card

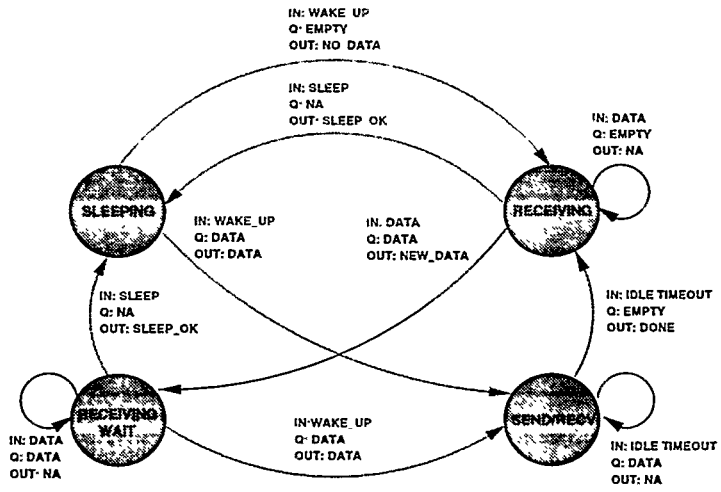

Figure 1: Slave (Base Station) Protocol State Diagram

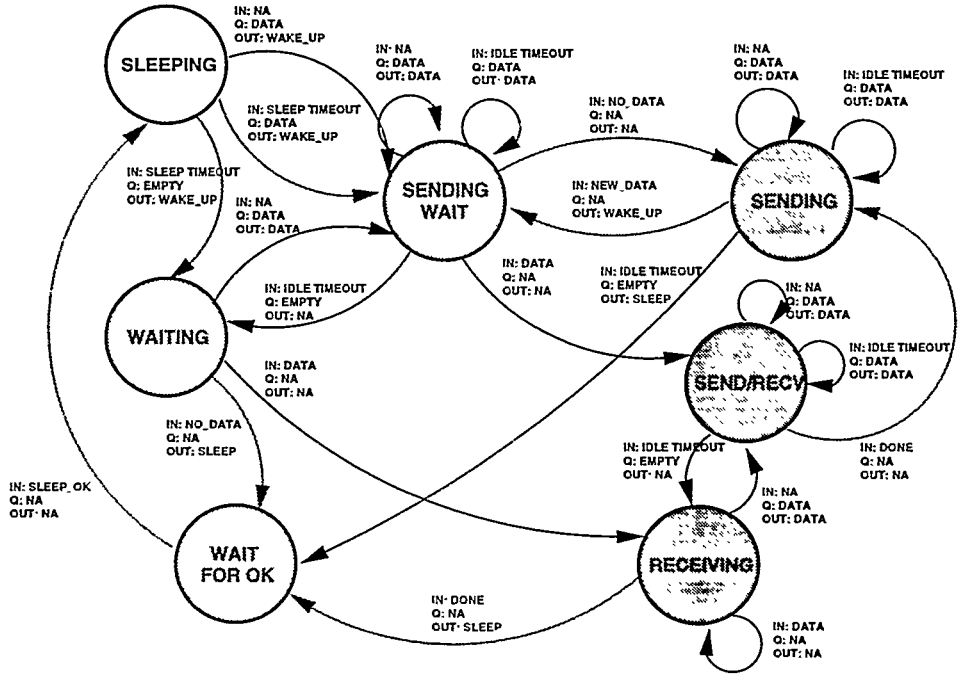

Figure 2: Master (Mobile Host) Protocol State Diagram

being used. Since our techniques address issues regarding the end-to-end transmission of data, we assume that this wakeup time is minimal in comparison to the total transfer time. Although this may not be true for all devices currently, the interface standards proposed in [10] suggest that future devices will provide relatively inexpensive transitions between waking and sleeping states.

4 Experiments

The goal of our experiments is to show that, by using our power management techniques, we can save a significant amount of the power consumed by the wireless Ethernet card. The tradeoff is an increased transmission delay observed by the receiver. First we will present our experimental setup and the user communication patterns used in our experiments. We will then show the impact of the power management techniques in the context of these user communication patterns. Finally, we will discuss our results in the context of several real systems.

4.1 Experimental Setup

In order to determine the impact of our power management techniques, we measure the power consumption of a wireless Ethernet card under varying conditions. In our experiments, we use a 915MHz Lucent WaveLAN PCMCIA wireless Ethernet card that can transmit data up to 150KBps. It provides three power modes: transmit, receive and suspend, and does not perform power management at the MAC layer. The system is configured as shown in Figure 3, with a wireless Ethernet in a NEC Versa 6320 laptop (the mobile host) communicating with a Gateway Solo 2200 (the base station) using a second WaveLAN PCMCIA card, both machines running Linux. We plugged the laptop into a universal power supply (UPS) to filter out voltage fluctuations. Our multimeter samples the current 11–12 times a second. From these samples and the output voltage of the UPS, we can monitor the power being used by the computer.

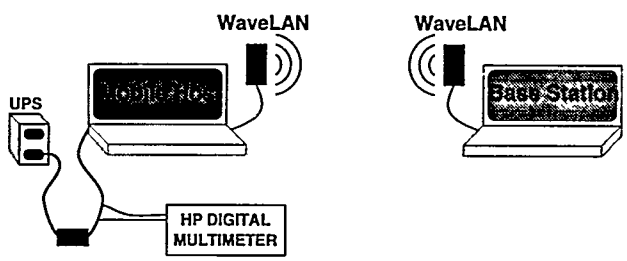

Figure 3: Experimental Setup

To determine the power consumption of the entire computer, we monitor the current being drawn from the transformer by the computer. This trace of current readings (11–12 readings a second), when integrated over time, provides us with the total energy and average power consumed during that time period. From baseline information we collected about the necessary energy to run an idle computer, we can compute the cost of communication. This cost of communication includes the energy consumed by the communication device and any energy consumed by the CPU and hard disk due to the communication. Each experiment was performed over a period of 30 minutes to provide sufficiently long samples. To ensure stability in our reported numbers, we repeated our experiments several times for each scenario. The results presented in this section were taken from specific sample runs. Each individual run was chosen from a set of qualitatively similar runs of a particular experiment

According to specifications from the manufacturer [18], the power requirements of the WaveLAN card are those shown in Table 1, Column 2. Column 3 in Table 1 shows the power requirements measured during our experiments without any power management. The measurements for receive mode were taken while the computer was idle, which implied no extra disk or CPU activity. As mentioned earlier, the power consumption for transmission includes any incidental CPU and hard disk power consumed to effect communication. It is interesting to note that the transmitter is rarely at full power for long periods of time. We observe that our measurements of the power required while the device is in either mode are very close to the documented specification.

State	Documented	Measured
WaveLAN - suspended	0W	0W
WaveLAN - receive	1.48W	1.52W
WaveLAN - transmit	3.00W	3.10W

Table 1: Power Requirements of the Lucent WaveLAN PCMCIA Wireless Ethernet card.

We chose Linux as a research platform because of the available source code for both the PCMCIA driver and the WaveLAN driver. In order to suspend the WaveLAN card in Linux, a system call to the kernel is used to send a suspend command to the WaveLAN driver. Suspension stops the receive unit, turns off the card, and updates the status of the PCMCIA device, removing its entry from any routing tables. This update generates an unwanted disk access. We modified the WaveLAN driver to update the status, but to leave the routing tables untouched, and called this the "sleep mode". Switching from active mode to sleep mode is now a matter of only the system call to the kernel and does not access the disk. Similarly, to wake up the WaveLAN card, a system call to the kernel is used to restart the receive unit on the WaveLAN card. The next generation of WaveLAN cards will have a DOZE mode [11] that will provide a quicker transition from active to DOZE than the transition from active to suspended in the current model. Our power management protocol was implemented in the context of an adaptive communication framework that provides dynamic protocol configuration support to the application [14, 13]. Through the use of the framework interface, the application can set and change specific protocol parameters.

4.2 Communication Patterns

The communication patterns used in our experiments are designed to simulate different types of users. The

amounts of data transmitted and received and the idle patterns of the users are varied randomly over time. The communication patterns are chosen to model three "typical" users. Table 2 presents the minimum, maximum and average amount of data in a transmission for each of these users. For the simulated web users, a transmission from the mobile host triggers multiple responses from the base station to simulate the multiple files needed for a web page. The average number of responses is shown in the count column of Table 2. Table 3 shows the timing patterns for the same users. Each user is described by a transmission cycle. During this cycle, the mobile host transmits and receives the amounts of data as described in Table 2. When considered with the idle times shown in Table 3, we can see the total amount of time per cycle that the mobile host spends transmitting, receiving and sitting idle. Our goal is to suspend the communication device during as much of these idle times as possible.

The first pattern (WEB) simulates a user browsing the web. The amount of data transmitted is relatively insignificant in comparison to the amount of data received. The sleep time represents the amount of time the user would spend reading a page before going on to the next one. Each request transmitted by the mobile host triggers a number of responses. The delay between these responses is varied from 0 to 15 seconds to simulate responses from a busy web server. The second pattern (JW) simulates a user working on a joint project over the wireless LAN. This user occasionally transmits and receives large pieces of their work. There is no connection between transmissions from the mobile and transmissions from the base station. The third pattern (EMAIL) simulates a user that mostly transmits and receives e-mail messages. This user is idle most of the time between transmissions, and the size of the transmissions are relatively small.

Typical mobile users tend to perform each of the above activities to some degree. From that point of view, the patterns presented above categorize users according to their main activity. The goal of the adaptive techniques discussed in Section 5 are to dynamically find appropriate timeouts for user performing such activities.

4.3 Results

In this section, we consider the results from our experiments in the context of the effects of our power management protocol on the energy consumed by the communication. This energy consumption is affected by the use of the CPU and the hard disk during communication. The savings we see come predominantly from the reduced consumption by the wireless Ethernet card. In Section 4.4, we discuss our results in the context of three real systems. The effects of the power management on a real system will depend on the power requirements of the system itself.

4.3.1 Protocol Power Consumption

When running the experiments with our management techniques turned on, we incur some overhead in terms of energy consumption. During idle periods in the communication, the overhead is due to the cost of waking up the WaveLAN card, transmitting a query to the base station and putting the card to sleep immediately since there is no data to receive. Our results show that, even with relatively short sleep times, this overhead is still significantly less than the energy consumed by the WaveLAN card had it been left in receive mode.

Our experiments produce a trace of the power measurements from the multimeter. Plotting these traces gives us a good, intuitive understanding of the effect of allowing the wireless Ethernet card to sleep for short periods of time. In each graph in Figure 4, we compare two traces of the power consumed by the idle communication subsystem (i.e., when there is no actual transmission or reception). In Figure 4a, the sleep duration is 1 second, and in Figure 4b, the sleep duration is 2 seconds. The first trace, the flat line at 1.5W marked by the diamonds, shows the power consumed by the communication when no power management is performed. The second trace, the line at 0W with occasional spikes marked by the plus signs, shows the power consumption with our power management protocol turned on.

We can see from the two traces that the power consumption is approximately 1.5W less when the WaveLAN card is suspended. For the first trace, the power consumptions stays at 1.5W since the communication device is always powered on. For the second trace, the power consumption is near zero most of the time, but spikes up at regular intervals. These spikes are caused by the protocol waking up and sending a query to the base station to see if there is any data waiting to be sent. By comparing the two graphs, we can see that the overhead for transmitting the queries is going to increase as the sleep duration gets shorter. Figure 5(a) shows the percent savings of using our protocol during idle periods when compared to no power management.

Figure 5(b) shows a trace during the transmission of a message to the base station. As we can see, the power consumption for both traces is the same during the transmission. The difference lies in the fact that after the transmission is complete, the trace using power management shows the effect of suspending the wireless Ethernet on the power consumption.

4.3.2 Power Savings for Communication Patterns

In order to determine the longer term effects of power management, we measure power consumption during communication generated in the patterns discussed in Section 4.2. In our experiments, we do not queue data at the master; i.e., we always wake up the communication device at the master if there is data to send. Figure 6 shows the results from our experiments in terms of the percent of the total energy consumed by the communication that was saved by using our power management protocol. For WEB, we see a 48-57% savings in the energy consumed by the communication. For JW, we see a 54-78% savings in the energy consumed by the communication. In the case of EMAIL, we compare the results of no power management with those of power management with sleep durations of 1 and 5 minutes. (Due to

	Data Transmitted					Data Received				
Pattern	Min (KB)	Max (KB)	Avg (KB)	Count	Avg Total (KB)	Min (KB)	Max (KB)	Avg (KB)	Count	Avg Total (KB)
WEB	5	30	17.5	1	17.5	300	1200	750	10	7500
JW	5	500	227.5	1	227.5	5	500	227.5	1	227.5
EMAIL	5	300	152.5	1	152.5	5	300	152.5	1	152.5

Table 2: Communication Patterns for Three Simulated Users

	User Sleep Time			Average Time			Average
Pattern	Min (sec)	Max (sec)	Avg (sec)	Transmitting (sec)	Receiving (sec)	Sleeping (sec)	Percent Sleeping
WEB	10	300	155	0.116	50	104.9	67.7%
JW	10	300	155	1.52	1.52	151.96	98%
EMAIL	10	600	305	1.02	1.02	302.96	99%

Table 3: Timing Patterns for Three Simulated Users

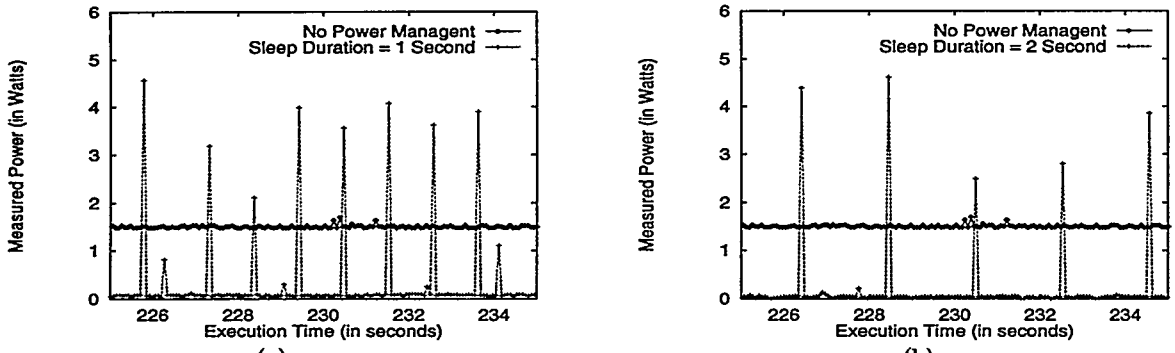

Figure 4: Power Consumption During a Sample Idle Period

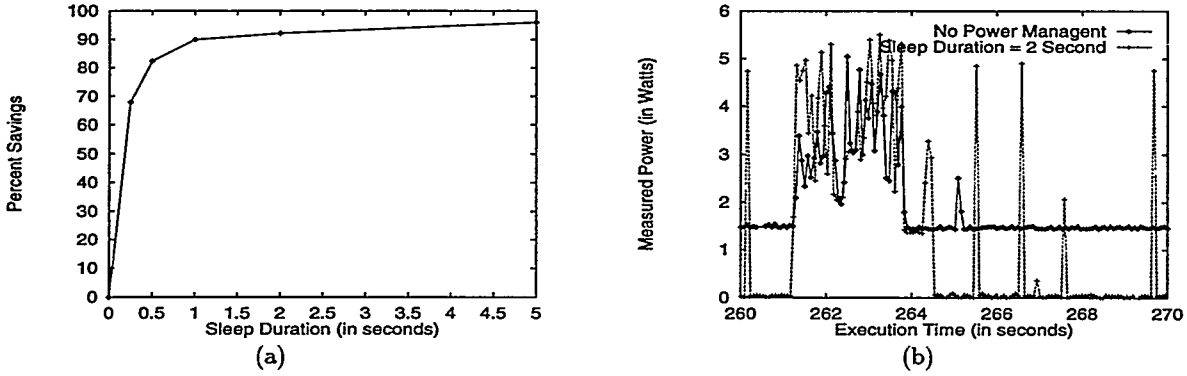

Figure 5: Power Savings During Idle Periods and a Sample Transmission

the different time scale, these measurements are not included in Figure 6.) These sleep durations result in a savings of 81% and 83% of the power consumed by the communication.

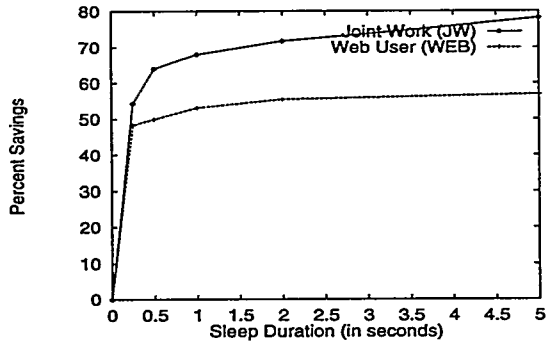

Figure 6: Savings for Communication Power Consumption

4.3.3 Delay

Power savings never come for free. In the context of communication, this cost can be measured in delay. A sleep duration of any length will impose, on average, a delay of half the duration. This cost must be taken into consideration when deciding how much power management to use. In the context of a user that solely uses communication for email, a sleep duration of 1 minute is acceptable. In reality, many email programs check mail on the order of every 5 minutes, and hence, such a sleep duration is more representative of the common email user. In contrast, a user who is working jointly across the network or who is accessing web pages may not be willing to accept such delays. In these cases, the sleep durations should reflect the tolerance of the users to delays in receiving data.

In the context of these experiments, we measure the communication delay in terms of additional transmission time per user data block at the base station. (We never queued data at the mobile machine.) We calculate this delay by determining the amount of time that the first message in the data block was delayed. Once the first message is sent, the rest will follow, and the delay to those messages will depend purely on how quickly the data can be transmitted. It is easy to show that the maximum additional delay imposed by our protocol on any data can never exceed the delay for the first packet, and hence, the numbers we present are a conservative estimate. We measure this delay by determining the time the transmission request was sent to the communication subsystem and the time when the data is finally sent. Since multiple transmission requests may queue up at the base station during large bursts of traffic, delays may be incurred even when no power management is used. Figure 7 shows the average added delay for a given sleep duration for the WEB pattern. When no power management is used, the average added delay is approximately 1 second. This increases, as we would expect, with increased sleep duration.

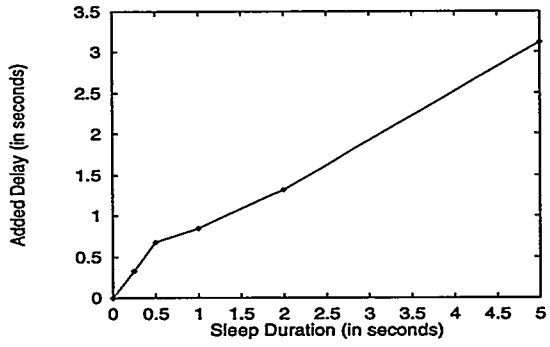

Figure 7: Added Delay based on Sleep Duration

4.4 Impact on System Costs

Now we consider the power savings in the context of three real machines (see Table 4). The experiments that we have described were performed on the NEC Versa 6320. We also measured the idle power consumption for the Toshiba Libretto 60 with and without the WaveLAN card and the idle power consumption for the HP Palmtop PC 320LX. We then used the results from Section 4.3, running the experiments on the NEC, to estimate the effect of the communication patterns on the other two machines. The Toshiba Libretto 60 is a small laptop that can run either Windows 95 or Linux. We would expect similar power costs for the hard disk and cpu for this machine as we did for the NEC. The HP Palmtop PC 320LX, on the other hand, runs Windows CE and has no internal hard disk. This will probably change the effect of power management on this type of machine. It may be argued that a wireless card with the power profile of WaveLAN is unlikely to be used with an HP Palmtop-like machine. We simply use this as an example of the trend towards lighter, more power efficient machines that have small battery capacity.

	Power Requirements	
Machine	Idle w/o WaveLAN Card	Idle w/ WaveLAN Card
NEC Versa 6320	14W	15.5W
Toshiba Libretto 60	7W	8.5W
HP Palmtop PC 320LX	1.2W	2.7W

Table 4: Measured Power Requirements for Three Machines

The first machine, the NEC Versa 6320, is a high end machine that consumes 14W while sitting idle. In this context, the 1.5W consumed by the WaveLAN card only represents approximately 10% of the power consumed by the computer when it is idle. In comparison, a machine like the Toshiba Libretto consumes only 7W when idle. The 1.5W of the WaveLAN card now represents approximately 18% of the power consumed by the computer when it is idle. If we take this one step further, we can see that for a machine like the HP Palmtop PC, this percentage increases to over 50%. To compensate

for this problem, some manufacturers have introduced wireless Ethernet cards that have an internal battery, although they still draw some amount of power from the main battery. In this case, the power consumed from the main battery by the idle device represents approximately 10% of the entire system power of the HP. Although this reduces the effect on the lifetime of the main battery, we still need to consider the effects on the battery for the card itself. Our techniques will work to extend the lifetimes of both batteries.

Considered in the context of the NEC, the results discussed in Section 4.3 show a 6.2–8.9% savings for the JW pattern and a 8.0–9.5% savings for the WEB pattern. If we now project these results onto the other two machines, we see even better savings of the power consumed by the entire system. Figure 8 compares the results for the percent saved of the total system power for these three machines over varied sleep durations. The crossing of the plots for the WEB and JW patterns for the HP is due to the fact that the power consumption of the communication device is more than the power consumption of the HP. Therefore, the better communication power savings for the JW pattern dominates the results. This is simply an artifact of the fact that the two patterns will eventually cross for all cases, as can be seen by the plots for the Toshiba. Most importantly, we can see that the trend toward machines like the Toshiba and the HP make it imperative that we properly manage communication devices, since the power consumed by these devices represents more and more of the total system power.

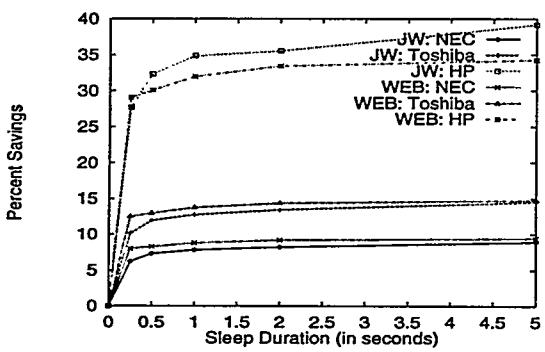

Figure 8: Savings for Three Types of Machines

5 Adaptive Mobile Power Management

We have shown that power management for communication devices can extend battery life. We also see that there is not one scheme that will fit all users or all applications. This leads us to investigate mechanisms for adapting power management levels during communication. The goal of this paper is not to address the issue of prediction, but provide the mechanism by which predictive algorithms can be used to adjust power management parameters; in particular, the timeout and sleep duration parameters in our implementation.

The ideal power management technique would sleep whenever there is no data to receive from the base station and wake up for any incoming receptions as well as tell the base station exactly when to expect transmissions from the mobile host. The goal of adaptive power management techniques is to be able to estimate when there is data to transmit from either side. Poor prediction can cause unsuccessful power management and waste resources such as buffer space and bandwidth.

With our protocol, the sleep duration can be adapted to fit the communication patterns of the application. As the sleep durations increase, we can see that the curve for the amount of energy saved will level off. This happens as the savings reach the theoretical maximum savings for the particular communication pattern. The theoretical limit is reached when the communication device is only active when there is actual data transfers occurring. For smaller sleep durations that are much smaller than the expected time between transmissions, the communication device is still active during some of the idle period. As the sleep duration increases, the probability that there is data waiting at the base station increases. As an example, consider Figure 9 which shows the percent of the total time spent sleeping for both the JW and the WEB patterns. As the sleep duration increases, the percent of the total time spent sleeping approaches the theoretical limit (98% for JW and 67% for WEB; see Table 3). From the fact that the sleep time for the WEB pattern comes closer than the JW pattern to the optimal sleep time, we can see that the power management techniques that we used were more successful for the WEB pattern.

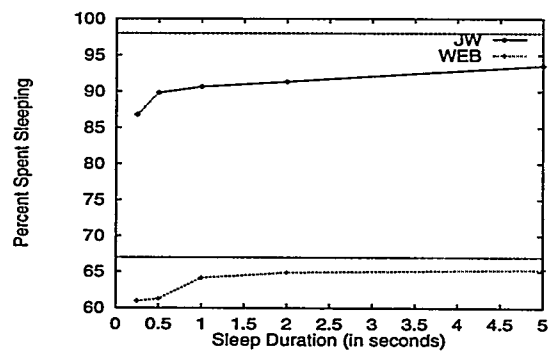

Figure 9: Percent of Total Time spent Sleeping

By providing an application-level interface to our power management protocol, applications can control the policies used for determining sleep durations. In this way, application-specific information can be used to determine optimal adaptation strategies.

In the context of our experiments, we implemented a simple adaptive algorithm similar to [6] for the web user. The algorithm responds to communication activity by reducing the sleep duration to 250 milliseconds and reacts to idle periods by doubling the sleep duration up to 5 minutes. We can use this simple algorithm because the communication patterns of the web user are somewhat predictable. A request from the mobile host expects multiple responses from the base station. As the mobile host notices that no more responses are available, it can deduce that either there are no more responses or that the server is busy and the responses may be delayed. For this communication pattern, such an adaptive algorithm pro-

vides a 58% savings in the power consumed by the communication device. This savings is an improvement over the 5 sec static sleep duration. While for the 5 second sleep duration, we saw a 3.12 second additional delay, for the adaptive algorithm, we only saw a 2.77 second delay. This suggests that adaptive and predictive techniques have merit in mobile communication power management for communication applications. It also demonstrates that applying power management at the transport or application layer has benefits. These techniques can be used in conjunction with adaptive techniques used at the MAC layer [4, 19].

An important aspect to keep in mind is that efficient prediction or estimation is not always simple or useful. Taking the communication patterns of multiple applications would also make adaptation more challenging. Learning-theory based estimation techniques [15, 12] can provide better adaptive algorithms for deciding when to power off and when to turn back on the communication device. Many such techniques inherently try to estimate the distribution generating the communication packets, and hence application provided hints help such estimation techniques.

6 Conclusions

In this paper, we have studied the important issue of power management in mobile wireless communication. We have presented a novel transport-level protocol by which a mobile host can judiciously suspend and restart its communication device, and by informing the base station appropriately, not lose en-route data. We have presented experimental results from an implementation of this protocol, and shown power savings of up to 83% for communication. This translates to a savings of 6–9% in terms of total system power for high end laptops, and can represent up to 40% savings for current hand-held PCs. When we consider the subset of our results for email and web browsing applications in the context of hand-held PDAs, our implementation results agree in large measure with the simulation results in [23]. It is important to note that if other components of the mobile machine are managed better, the relative improvement numbers due to efficient power management of the communication device will be more prominent. For most applications the resulting small additional delay (e.g., 0.4–3.1 seconds for some web browsing responses) should be acceptable.

Open problems include the development of intelligent techniques (e.g., learning-based methods) to estimate when there is queued data at the base station, so that reception delays can be reduced. Such techniques might also adapt the timeout choice for users with varied communication patterns. It will be interesting to explore the correct APIs to provide to applications so that they can give hints to the protocol about their communication patterns (in the spirit of transparent informed prefetching [20]).

Acknowledgements

Research by Robin Kravets was sponsored in part by an AT&T/Lucent Technologies Ph.D. Fellowship. The authors would like to thank Rob Kooper and Don Allison for their assistance in the setup, configuration and building of our power measurement equipment.

References

[1] A. Bakre and B.R. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *IEEE International Conference on Distributed Computing Systems (ICDCS) '95*, 1995.

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proceedings of the SIGCOMM '96 Symposium*, 1996.

[3] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP performance over wireless networks. In *First ACM International Conference on Mobile Computing and Networking (MOBICOM)*, November 1995.

[4] I. Chlamtac, C. Petrioli, and J. Redi. Energy conservation in access protocols for mobile computing and communication. *Microprocessors and Microsystems Journal*, 1998.

[5] F. Douglis. On the role of compression in distributed systems. *ACM Operating Systems Review*, 27(2):88–93, April 1993.

[6] F. Douglis, P. Krishnan, and B. Bershad. Adaptive disk spindown policies for mobile computers. In *Proceedings of the Second USENIX Symposium on Mobile and Location Independent Computing*, April 1995.

[7] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the power hungry disk. In *Proceedings of the 1994 Winter USENIX Conference*, January 1994.

[8] K. Govil, E. Chan, and H. Wasserman. Comparing algorithms for dynamic speed-setting of a low-power cpu. In *First ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 1995.

[9] D. Helmbold, D. D. E. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In *Second ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 1996.

[10] Intel Corporation, Microsoft Corporation and Toshiba Corporation. *Advanced Configuration and Power Interface Specification*, revision 1.0a edition, July 1998.

[11] A. Kamerman and L. Monteban. WaveLAN-II: A high performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, Summer 1997.

[12] S. Keshav, C. Lund, S. J. Phillips, N. Reingold, and H. Saran. An empirical evaluation of virtual circuit holding time policies in ip-over-atm networks. In *Proceedings of IEEE INFOCOM 95*, 1995.

[13] R. Kravets, K. Calvert, P. Krishnan, and K. Schwan. Adaptive variation of reliability. In *the Seventh IFIP Conference on High Performance Networking (HPN'97)*, April 1997.

[14] R. Kravets, K. Calvert, and K. Schwan. Payoff adaptation of communication for distributed interactive applications. *Journal on High Speed Networking: Special Issue on Multimedia Communications*, 1998.

[15] P. Krishnan, P. Long, and J. S. Vitter. Adaptive disk spindown via optimal rent-to-buy in probabilistic environments. In *Proceedings of the Twelfth International Machine Learning Conference*, July 1995.

[16] K. Li, R. Kumpf, P. Horton, and T. Anderson. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the 1994 Winter USENIX*, 1994.

[17] J. R. Lorch and A. J. Smith. Reducing processor power consumption by improving processor time management in a single-user operating system. In *Second ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 1996.

[18] Lucent Technologies. *WaveLAN/PCMCIA Card User's Guide*, October 1996.

[19] B. Narendran, J. Sienicki, S. Yajnik, and P. Agrawal. Evaluation of and adaptive power and error control algorithm for wireless systems. In *IEEE International Conference on Communications (ICC'97)*, 1997.

[20] R. H. Patterson, G. A. Gibson, and M. Satyanarayanan. A status report on research in transparent informed prefetching. *ACM Operating Systems Review*, (27):21–34, April 1993.

[21] J.M. Rulnick and N. Bambos. Mobile power management for maximum battery life in wireless communication networks. In *Proceedings of IEEE INFOCOM 96*, 1996.

[22] A. Sampath, P.S. Kumar, and J. Holtzman. Power control and resource management for a multimedia cdma wireless system. In *The Sixth International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'95)*, 1995.

[23] M. Stemm and R. Katz. Reducing power consumption of network interfaces in hand-held devices. In *Third International Workshop on Mobile Multimedia Communications (MoMuc-3)*, December 1996.

[24] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Proceedings of the First Symposium on Operating System Design and Implementation (OSDI) '94*, November 1994.

A Power Control Protocol

Protocol Commands: During the course of communication both the master and the slave use commands to inform the receiving side of state changes. Protocol commands are as follows:

`PMC_CMD_WAKE_UP`: Used by the master to inform the slave that it can wake up and transmit data if it has any messages queued up.
`PMC_CMD_NO_DATA`: Used by the slave to inform the master that upon wake up, the slave had no data to send.
`PMC_CMD_DATA`: Used for any messages that simply contains data.
`PMC_CMD_NEW_DATA`: Used by the slave to indicate to the master that it now has data to transmit.
`PMC_CMD_DONE`: Used by the slave to indicate the end of data transmission.
`PMC_CMD_SLEEP`: Used by the master to inform the slave that it should go to sleep.
`PMC_CMD_SLEEP_OK`: Used by the slave to indicate that it completed the sleep command.

Master Protocol: The mobile host has the responsibility of determining when communication takes place. At any point in time, the master can be in one of the following states:

`PMC_STATE_SLEEPING`: The protocol is sleeping and no data can be transmitted or new data requests will be allowed and a `PMC_CMD_WAKE_UP` is sent. Additionally, no data should be received when the protocol is sleeping.
`PMC_STATE_SENDING`: Only the master is sending data. Subsequent data requests are queued for transmission.
`PMC_STATE_SENDING_WAIT`: Only the master is sending data. The slave has been queried for new data to send. Subsequent data requests are queued for transmission.
`PMC_STATE_RECEIVING`: Only the slave is sending data. New data requests will be queued for transmission and the master will enter the `PMC_STATE_SEND_RECV` state.
`PMC_STATE_SEND_RECV`: Both the master and the slave are sending data. Subsequent data requests are queued for transmission.
`PMC_STATE_WAITING`: The master woke up and has nothing to send. It has sent a query to the slave to see if it has any new data to send.
`PMC_STATE_WAIT_FOR_OK`: The master has determined that communication should be suspended and is waiting for a response from the slave.

Slave Protocol: The base station follows the commands of the master. At any point in time, the slave can be in one of the following states:

`PMC_STATE_SLEEPING`: The protocol is sleeping and no data can be transmitted. Upon receiving a message, the slave wakes up and enters either the `PMC_STATE_RECEIVING` state or the `PMC_STATE_SEND_RECV` state, determined by whether or not the slave has data to send.
`PMC_STATE_RECEIVING`: Only the master is sending data. New transmissions will not be allowed. If new transmissions are requested, the slave sends a `PMC_CMD_NEW_DATA` message and enters the `PMC_STATE_RECEIVING_WAIT` state.
`PMC_STATE_RECEIVING_WAIT`: Only the master is sending data. New transmissions will not be allowed. Upon receipt of a `PMC_CMD_WAKE_UP` message, the slave starts transmitting and enters the `PMC_STATE_SEND_RECV` state.
`PMC_STATE_SEND_RECV`: Both the master and the slave are sending data. Subsequent data requests are queued for transmission.