PAMAS – Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks*

C.S. Raghavendra
Aerospace Corporation
El Segundo, CA 90245
email: raghu@aero.org

Suresh Singh
Department of Computer Science
University of South Carolina
Columbia, SC 29205
singh@cs.sc.edu

October 19, 1997

Abstract

In this paper we develop a new multiaccess protocol for ad hoc radio networks. The protocol is based on the original MACA protocol with the adition of a separate signalling channel. The unique feature of our protocol is that it conserves battery power at nodes by intelligently powering off nodes that are not actively transmitting or receiving packets. The manner in which nodes power themselves off does not influence the delay or throughput characteristics of our protocol. We illustrate the power conserving behavior of PAMAS via extensive simulations performed over ad hoc networks containing 10–20 nodes. Our results indicate that power savings of between 10% and 60% are attainable in most systems. Finally, we discuss how the idea of power awareness can be built into other multiaccess protocols as well.

1 Introduction

Ad Hoc networks are multi-hop wireless networks where all nodes cooperatively maintain network connectivity. These type of networks are useful in any situation where temporary network connectivity is needed. For instance, consider the problem of establishing a temporary wireless network in a region hit by some natural disaster. An ad hoc network here would enable medics in the field to retrieve patient history from hospital databases (assuming that one or more of the nodes of the ad hoc network are connected to the Internet) or allow insurance companies to file claims from the field. Other examples of such *ad hoc* networks include internetworking participants in a meeting to enable them to exchange data, battlesite networks, etc.

Nodes in an ad hoc network communicate via radio and since the radio channel is shared by all nodes, it becomes necessary to control access to this shared media. Several authors have developed channel access protocols for multi-hop radio networks where the goal has been maximizing throughput and minimizing transmission delay. Unlike this previous work, however, in this paper we present a channel access protocol that reduces the **power consumption** at each of the nodes. Reducing power consumption is clearly an important goal because battery life is not expected to increase significantly in the coming years. In an ad hoc network, it is even more important to reduce power consumption because these networks are typically established in mission critical environments (such as disaster relief).

*The second author's work was supported by the NSF under grant number NCR-9706080.

Significant power is consumed at a node when it either transmits a packet or when it receives a packet. Thus, the DEC Roamabout radio [1] consumes approximately 5.76 watts during transmission, 2.88 watts during reception and 0.35 watts when idle. The radio used in [6] consumes 15 watts while transmitting, 11 watts while receiving and 50mW in idle mode. Now, notice that in ad hoc networks, a transmission from one node to another is potentially overheard by all the neighbors of the transmitting node – *thus all of these nodes consume power even though the packet transmission was not directed to them*! For example, in the ad hoc network illustrated in Figure 1, node A's transmission to node B is overheard by node C because C is a neighbor of A. Node C thus expends power receiving a packet not sent to it! In our protocol, node C *turns itself off* during the transmission from A to B to conserve power. It is easy to see that the potential savings of this simple approach can be enormous – particularly in dense networks.

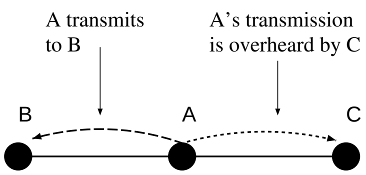

Figure 1: Unnecessary power consumption.

The remainder of this paper is organized as follows. In the next section we survey several multiaccess protocols that have been devised for ad hoc networks. Section 2.1 presents other work related to conserving power. Our protocol is presented in section 3 and its performance is discussed in section 4. Section 5 discusses how our approach for conserving power can be extended to other multiaccess protocols. We also discuss possible extensions of our basic protocol. Finally, our conclusions are presented in section 6.

2 Channel Access Protocols for Ad Hoc Networks

Channel access protocols for ad hoc networks have to contend with the problem of *hidden terminals* in addition to the problem of contention as in the Ethernet. Figure 2 illustrates the hidden terminal problem. Here, node A begins transmitting a packet to node B. However, since node C is out of range of node A, it begins transmitting a packet some time later. This results in collisions at the *receiver* B. Observe that neither of the two senders is aware of the collision and therefore cannot take preventive measures. It is noteworthy that this type of a problem does not arise in the Ethernet (for instance) because all nodes can hear one another. Several authors have developed different solutions to the hidden terminal problem. In this section we examine some of these proposals.

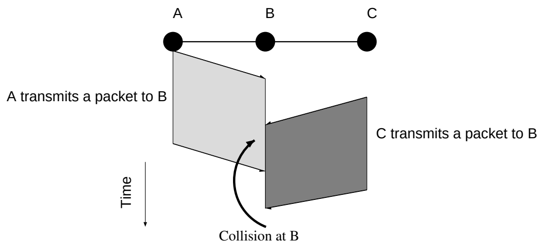

Figure 2: The hidden terminal problem.

MACA[11] was one of the early channel access protocols developed for ad hoc networks. Here, whenever a node wishes to transmit a packet to a neighbor, it first transmits a RTS (Request To Send) message. The receiver responds with a CTS (Clear To Send) message. Upon receiving the CTS message, the sender begins transmitting the packet. How does this RTS-CTS message exchange alleviate the hidden terminal problem? In Figure 2, C would have received the CTS transmission from B before A begins transmitting the packet to B. Thus C can hold off transmitting until B receives A's packet completely (the RTS and CTS messages contain the length of the packet). It is possible that the RTS message or its CTS may suffer a collision. In this case the sender executes a binary exponential backoff algorithm and tries to send a RTS again, later.

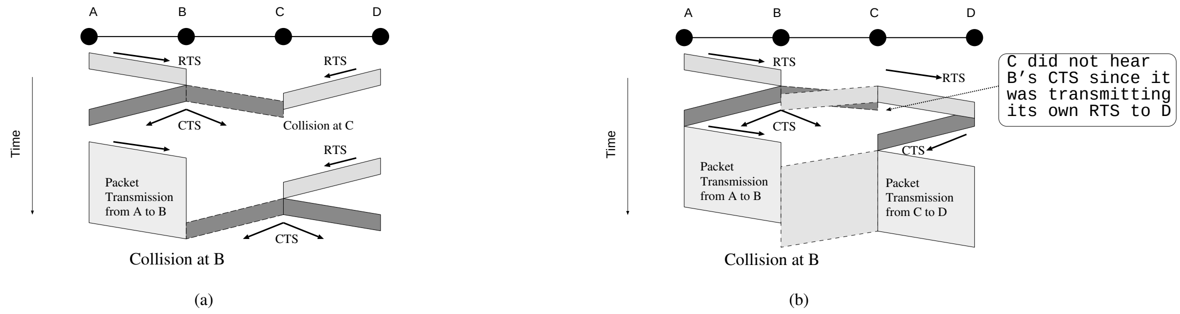

Figure 3: Hidden terminal problem in MACA.

Even though MACA solves the hidden terminal problem illustrated earlier, it creates another! In Figure 3(a), A sends a RTS to B who responds with a CTS. However, D sends a RTS to C at about the time that B is sending a CTS to A. Thus C hears a collision and does nothing. A receives the CTS and begins transmitting the data packet. D, on the other hand, retransmits a RTS after some time. Since C does not know that B is receiving a packet, it responds with a CTS. This CTS transmission, unfortunately, collides with the packet transmission at node B.

When collisions occur in MACA, recovery is left up to the transport layer thus greatly reducing throughput. MACAW[2] is a modified version of MACA where link layer ACKs have been added. Thus, a sender can retransmit a packet that was not successfully received at the receiver. In the above example, B will not send an ACK for the packet and A will retransmit it again. FAMA[5] is a refinement of the MACAW protocol in that it includes a non-persistent CSMA at the beginning of each free slot. In addition, the length of the CTS is made longer than the RTS to deal with the situation illustrated in Figure 3(b). If the length of the CTS is longer, node C will receive a part of the CTS transmission (either the initial part or the end). It will interpret this as noise wait for the length of one (maximum length) packet transmission before doing anything (it will not transmit a packet even if it receives a CTS from D). This solution also fixes the problem illustrated in Figure 3(a). Here, node C will hear noise (when control packets collide) and will ignore all transmissions for the length of time taken to transmit one maximum length packet. MACA/PR[13] is a protocol based on MACAW with the provision of non-persistent CSMA (as in FAMA). In addition, MACA/PR supports real-time data traffic by including a reservation mechanism in the RTS-CTS-Packet-ACK sequence. Finally, the IEEE 802.11 standard for wireless LANs includes the collision aviodance of MACA and MACAW. In addition, all directed traffic uses positive ACKs (again as in MACAW).

A different approach to the problem is described in [15]. In this system, there is a separate channel used for transmitting "busy tones". Thus, when a node wants to transmit a packet, it listens for a busy tone. If that channel is free, it transmits the preamble of its packet (this contains the receivers address). The receiver responds with a busy tone. On hearing the busy tone, the sender continues sending the packet. It is easy to see that the hidden terminal problems described above are handled with this solution. Other approaches to channel access include splitting the network into clusters (fully connected subnetworks) and using a different spreading code in each cluster (see [7]).

2.1 Power Conserving Research

In this section we review work related to conserving power in devices used in mobile environments. Battery life imposes a severe constraint on the deployment and large-scale use of mobile computing technology in the future and has prompted several researchers to develop approaches for conserving power on mobile computers. Significant amount of power is consumed by the display, by spinning disks, by the CPU and by the radio. In the event a camera or other recording device is attached to the mobile that I/O device is also likely to be a power drain. [10] discusses the problem of power consumption in displays and proposes several solutions. [4, 12, 19] study the issue of power consumption by spinning disks and propose algorithms for spinning up or spinning down disks based on the expected disk access patterns. [3] is one example of research on the development of power-efficient I/O devices – in this case the design of a power-efficient wireless digital camera suitable for use in the field.

Recently, some researchers have begun studying the problem of reducing power consumption by the wireless interface. Thus, [16] observes that the average life of batteries in an *idle* cellular phone is one day. [17] studies power consumption of several commercial radios (WaveLAN, Metricom and IR) and observes that even in Sleep mode the power consumption ranged between 150-170 mW while in Idle state the power consumption went up by one order of magnitude. [16] observes that the only way to reduce power consumption in radios is to *shut them off* and use this observation to propose a power efficient MAC layer protocol suitable for use within one cell for communication between a base station and the mobiles. This protocol is based on the paging protocols POCSAG and FLEX.

3 The PAMAS Protocol

The PAMAS protocol is a combination of the original MACA protocol (see [11]) and the idea of using a separate *signalling channel* (as in [15]). Thus, we assume that the RTS-CTS message exchange takes place over a signalling channel that is separate from the channel used for packet transmissions. This separate signalling channel enables nodes to determine when and for how long to turn off (section 5 shows why the lack of a separate signalling channel, in most other protocols, results in sub-optimal power conservation).

The state diagram outlining the behavior of our protocol is illustrated in Figure 4. As indicated in the figure, a node may be in any one of six states – *Idle, AwaitCTS, BEB* (Binary Exponential Backoff), *Await Packet, Receive Packet,* and *Transmit Packet.* When a node is not transmitting or receiving a packet, or does not have any packets to transmit, or does have packets to transmit but cannot transmit (because a neighbor is receiving a transmission) it is in the Idle state. When it gets a packet to transmit, it transmits a RTS and enters the AwaitCTS state. If the awaited CTS does not arrive, the node goes into binary exponential backoff (the BEB state in the figure). If a CTS does arrive, it begins transmitting the packet and enters the Transmit Packet state. The intended receiver, upon transmitting the CTS, enters the Await Packet state. If the packet does

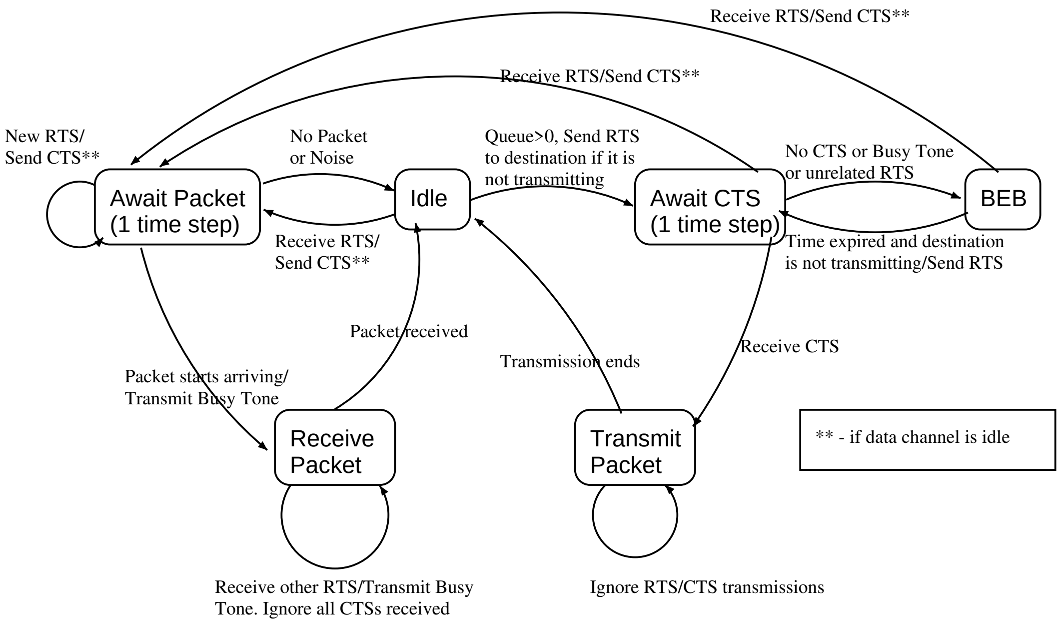

Figure 4: The PAMAS Protocol.

not begin arriving within one roundtrip time (plus processing time), it returns to the Idle state. If the packet does begin arriving, it transmits a busy tone over the signalling channel and enters the Receive Packet state. Let us now look at the functioning of the protocol in some more detail.

When a node in the Idle state receives a RTS, it responds with a CTS if no neighbor is in the Transmit Packet state or in the AwaitCTS state. It is easy for a node to determine if any neighbor is in the Transmit Packet state. However, it is not always possible for a node to know if a neighbor is in the AwaitCTS state (the transmission of the RTS by that neighbor may have collided with another transmission over the control channel). In our protocol, if the node heard noise over the control channel within τ [1] of the arrival of the RTS, it does not respond with a CTS. If, however, it does not hear a packet transmission begin within the next τ, it assumes that none of its neighbors is in the AwaitCTS state anymore.

Now consider a node that is in the Idle state and has a packet to transmit. It transmits an RTS and enters the AwaitCTS state. If, however, a neighbor is receiving a packet that neighbor responds with a busy tone (twice as long as a RTS/CTS) that will collide with the reception of the CTS. This will force the node to enter the BEB state and not transmit a packet. If no neighbor transmits a busy tone and the CTS arrives correctly, transmission begins and the node enters the Transmit Packet state.

Say a node that transmitted a RTS does not receive a CTS message. It enters the BEB state and waits to retransmit a RTS. If, however, some other neighbor transmits a RTS to this node, it leaves the BEB state, transmits a CTS (if no neighbor is transmitting a packet or is in the AwaitCTS state) and enters the Await Packet state (i.e., it waits for a packet to arrive). When the packet begins arriving, it enters the Receive Packet state. If it does not hear the packet in the expected time (i.e., round trip time to the transmitter plus some small processing delay at the receiver), it goes back to the Idle state.

[1] τ = one roundtrip time plus transmission time for a RTS/CTS.

When a node begins receiving a packet, it enters the Receive Packet state and immediately transmits a busy tone (whose length is greater than twice the length of a CTS). If the node hears a RTS transmission (directed to some other node) or noise over the control channel at any time during the period that it is receiving a packet, it transmits a busy tone. This ensures that the node transmitting the RTS will not receive the expected CTS. Thus its transmission (which would have interfered with the node receiving a packet) is blocked.

It is easy to see that our protocol handles the hidden terminal problems illustrated in Figure 2. For instance, in the first example, node B's reception of a packet from A will not be affected by the transmission of a CTS by node C (since these transmissions occur over different channels). In the second example, when node B begins receiving the packet from A, it transmits a busy tone that is heard by node C. If the busy tone overlaps with the CTS transmission from node D to node C, node C hears only noise and will enter the BEB state and transmit a RTS again, later. This retransmission of the RTS will be met by another busy tone from B if B is still receiving the packet. This continues until either B finishes receiving or D sends a RTS to C (in this case C may begin receiving a packet from D).

3.1 Powering off radios

We noted in section 1 that nodes consume power while transmitting or even while receiving a packet. Unfortunately, in an ad hoc network, it is frequently the case that a packet transmission from one node to another will be overheard by all the neighbors of the transmitter. All of these nodes will thus consume power needlessly. Consider a simple example where the network is fully connected (i.e., all nodes are within transmission range of each other) with n nodes. A transmission here will be heard by all $n-1$ nodes. If the power consumed in transmitting a packet is t and r is the power consumed while receiving, we see that the total power consumed (system-wide) for one packet transmission is $t+(n-1)r$. This is a huge waste because the total power consumed for a single transmission should be no more than $t+r$ (ignoring the power consumed in the CTS-RTS-Busy Tone transmissions).

In order to conserve power and extend the lifetime of mobile nodes, the PAMAS protocol requires nodes to shut themselves off if they are in a situation where they overhear transmissions. Thus, our protocol ensures that in the fully connected example above, $n-2$ nodes will shut themselves off for the duration of the transmission. We have identified two conditions under which it is beneficial for a node to turn itself off.

$\oplus$ Observe that if a node has no packets to transmit, then that node ought to power itself off if a neighbor begins transmitting.

$\oplus$ Similarly, if at least one neighbor of a node is transmitting and another is receiving, the node ought to power off because it cannot transmit or receive a packet (even if its transmit queue is non-empty).

Every node in our system makes the decision to power off independently. A node knows if a neighbor is transmitting because it can hear the transmission (over the data channel). Likewise, a node (with a non-empty transmit queue) knows if one or more of its neighbors is receiving because the receivers transmit a busy tone when they begin receiving a packet (and in response to RTS transmissions). Thus, a node can easily decide when to power off. There are, however, two additional questions to be answered:

1. For how long is a node powered off?

2. What happens if a neighbor wishes to transmit a packet to a node that has powered itself off?

Let us answer the second question first using an example. Say we have a line network with four nodes (A–B–C–D) and node B is transmitting to node A. The transmission is overheard by node C (who powers itself off). In this situation, it is easy to see that any transmission *to* node C (say node D wants to transmit to node C) will be garbled at C whether C is powered off or not. So the answer to the second question above is – the neighbor will have to wait for node C to wake up (which happens when the transmission from B to A completes).

The first question is also easy to answer, however, we need to consider several cases. Ideally, a node ought to stay powered off whenever any of the two conditions ($\oplus$) hold. However, collisions in the signalling channel and the data channel may make it difficult for a node to determine the length of a transmission. Nodes follow the following protocol to determine the length of time for which they can power off.

⊙ When a packet transmission begins in the neighborhood of a node, it knows the duration of that transmission (say l). If the node has an empty transmit queue, it powers itself off for l seconds.

⊙ It is possible that one or more neighbors may begin data transmission while the node is powered off. In this case, when the node powers back on, it will continue hearing transmissions over the data channel. If the node still has an empty transmit queue, it ought to power itself off again. But for how long?

Figure 5 illustrates the case when three neighbors begin transmission after a node powers off. These transmissions are ongoing when the node powers back on and it needs to find out the remaining transmission time (i.e., it needs to find out the value of l_2). To do this we add additional functionality in our protocol. The node, upon waking up, transmits a *t_probe(l)* packet over the control channel where l is the maximum packet length. All transmitters with transmissions completing in time period $[l/2, l]$ respond with a *t_probe_response(t)* packet (t is the time when this transmitter's transmissions will end). If the packet is received without collision, the node powers itself off until time t. Otherwise, if there was a collision, it probes interval $[3l/4, l]$. If there is silence, it probes $[l/2, 3l/4]$, and so on. If there was silence in response to the initial probe, it probes interval $[0, l/2]$. In effect, the node does a *binary search* to determine the time when the last (current) transmission will end.

A simplification that can be built into the probe protocol just described is the following. When the node hears a collision in response to a probe of the interval $[t_1, t_2]$, it powers itself off for the period t_1. This simplification attempts to reduce the probing time by sacrificing some power (since the node will power back on while a transmission is ongoing). Observe that if the node powers itself off until time t_2 (instead of t_1), there is a likelihood that packet delays will increase. This is because the packet transmissions may cease soon after time t_1 but the node is powered off until t_2. Thus if another node has a packet for this node, that packet cannot be delivered.

⊙ Next consider the case when a node has a non-empty transmit queue. When the node powers back on (after it first powered off), it transmits a RTS (rather than probing because it needs to transmit a packet). If any node in its neighborhood is receiving a transmission, that node responds with a busy tone (containing the length of the remaining transmission). If the busy tone collides with another busy tone or a CTS or some other RTS, the node attempts to probe the *receivers* using the same binary search algorithm described above but using a *r_probe(l)* packet (the prefix r denotes a receiver probe packet). It probes the transmitters next using the *t_probe(l)* packet. Then it powers itself off for $min\{r,t\}$ where r is the time the last receiver finishes receiving and t is the time the last transmitter finishes transmitting.

To understand the reason for taking a min above, consider the following two cases. If $t < r$ (i.e., all transmitters finish before the receivers finish) then we need to power on this node so that, if some other node has a packet for this node, that node can go ahead with its transmission (to reduce delays). If, on the other hand, $t > r$, we need to power back this node after r so that it can begin transmitting packets from its queue (again to keep delays small).

⊙ Finally, it is important to observe that the probe messages could get corrupted (say more than one node powers on at the same time and transmits a probe message). In this case there will be no response to the probes and the nodes will stay on. A workaround we suggest (but have not implemented) is to use p-persistent CSMA when transmitting a probe packet (with p choosen appropriately). This will reduce the possibility of collisions of probe packets. We have not implemented this scheme because of two reasons:

- Under light load conditions, the probability of hearing a new transmission after a node powers back on is low. Hence there is no need to build a sophisticated protocol.
- Under heavy loads, it is almost always the case that when a node powers on there will be ongoing transmissions. In this case, it is unlikely that the node will have an empty transmit queue. So it will try to transmit RTS messages which will evoke busy tone responses from receivers. This will quickly inform the node of the additional time it needs to power off for (we assume that the busy tone transmissions include the length of the remaining transmission).

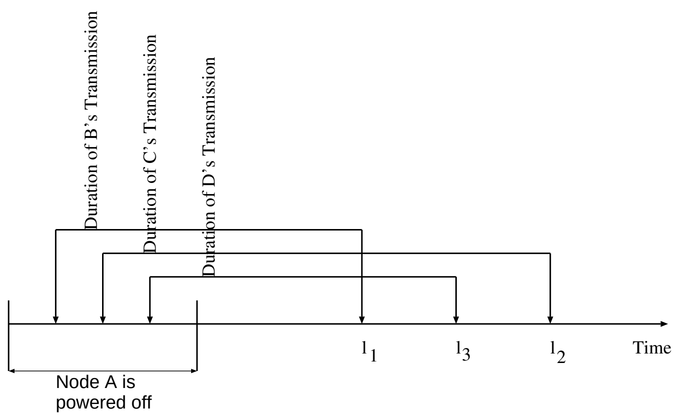

Figure 5: Situation when a node powers back on.

It is noteworthy that the above *probe protocol* can be simplified considerably if we assume that the node only powers off its data interface but always leaves the signalling interface powered on. This will enable the node to always know the length of new transmissions and keep the data

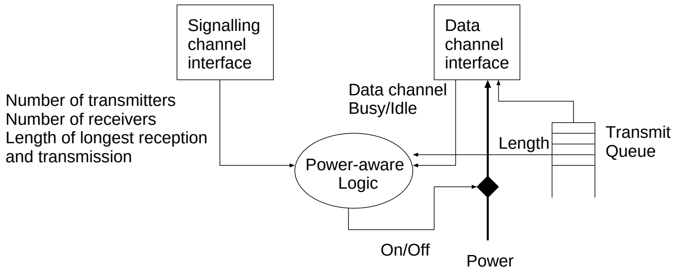

Figure 6: Separate interfaces for signalling and data.

interface powered off appropriately. Figure 6 illustrates the block diagram needed for this type of a communications device. Here, the signalling interface listens to all RTS/CTS/Busy Tone transmissions and records the length of each transmission and reception. This information (along with the length of the transmit queue) is fed to the power aware logic which determines whether to turn the data interface off or on. The power aware logic that can be used is as described above with the exception of the probe algorithm.

3.1.1 Effect of powering off on Delay and Throughput

An important concern related to powering off radios is whether the delay or throughput behavior of PAMAS changes. We claim that powering off radios, as we have described, does not have any effect because a radio powers itself off if it either cannot receive data transmissions directed to it (because a neighbor is transmitting a packet) or if it cannot transmit a packet (because a neighbor is receiving another transmission). In these cases, even if the radio was not turned off, it could not receive/transmit a packet. Thus, powering off the radio has no effect on the behavior of PAMAS. This statement does have an important caveat, however. The *length* of time that a radio is powered off should be no longer than necessary (i.e., the two conditions mentioned above hold for this time period) otherwise the delay and throughput behavior of PAMAS will be changed. It is for this reason that in section 3.1 we use the probe algorithm to enable nodes to estimate the length of time that a radio can turn itself off. In fact, as we noted in the discussion earlier, we err on the side of caution and may underestimate the length of this period. This results in sub-optimal power savings but ensures that the delay/throughput behavior of PAMAS remains unchanged.

4 Power Conserving Behavior of PAMAS

In order to characterize the power conserving behavior of our protocol we conducted extensive simulations where we compared the power expended by PAMAS without power conservation and PAMAS with power conservation. The simulations were conducted in three different network topologies that, we believe, represent most ad hoc networks. Thus, we ran PAMAS in a *random network* topology, a *line topology* and a *fully connected network* topology. We conjectured that networks where nodes are densely connected will show the most power savings while networks that are sparse will show the least amount of power savings. The reason is that, in a dense network, if one node transmits most of its neighbors can power off. In a sparse network, on the other hand, fewer nodes can power off. An additional implication of this is that the throughput will typically be higher in sparse networks because more transmissions can go on simultaneously (note that throughput and delay are *not* affected when nodes are powered off – see section 3.1.1).

In the simulations, we used fixed size packets (512 bytes). The RTS and CTS packets were 32 bytes each and the busy tone was twice as long. The bandwidth was assumed to be 12.8Kbps (observe that our results also hold for higher data rates – we used this rate to keep the length of the simulation time small). In terms of power conservation, we assumed that no power is consumed when a node is idle (i.e., powered on but not hearing any transmissions). 1 unit of power is consumed for 32 bytes of transmission (over either the data channel or the signalling channel). 0.5 units of power are consumed at a receiver for every 32 bytes processed (again, data or control).

We ran simulations for networks with 10 and 20 nodes. For the random networks, we generated edges randomly uniformly with probabilities between 0.1 to 0.9 for different experiments. A probability of 0.1 generates sparse networks (allowing more parallel transmissions and hence less power savings) while an edge probability of 0.9 yields dense networks with much better power conserving behavior. Traffic arrived at each node according to a poisson process. The destination was choosen randomly uniformly from the remaining $n-1$ nodes and the packet was routed using the shortest path.

Finally, to measure the power savings, we calculated the total number of bytes transmitted B_t during a run of the experiment, the total number of bytes received B_r(note that a packet may be received by more than one node and is therefore counted more than once) and the total number of packets P transmitted during the experiment. The power expended is then,

$$\mathcal{P} = (B_t + B_r/2)/P$$

we divide B_r by two because we assume that one unit of power is consumed per byte transmitted and half a unit per byte received per node. If we use power conservation, the number of bytes received tends to be smaller, say it is B_r^c. Then, the power expended is,

$$\mathcal{P}^c = (B_t + B_r^c/2)/P$$

and the power *savings* can be written as,

$$\text{Percentage of Power Saved } = (\mathcal{P} - \mathcal{P}^c)/\mathcal{P}$$

We ran each experiment 150 times and computed 95% confidence intervals for the percentage of power saved. The interval half-widths kept to less than 5% of the point values in all cases.

Figure 7 plots the power saved in a complete network topology as a function of λ. We ran the experiments with 10 nodes and with 20 nodes. It is easy to see that PAMAS reduces power consumption by almost 50% for high loads and by even more for low loads. The reason for the higher savings at low loads is that, at low loads, there is less contention for the channel resulting in fewer control message transmissions (i.e., RTS transmissions). At high loads, almost all the nodes have packets to send and thus contention for the channel is high. This results in fewer actual packet transmissions (which is the only time when $n-2$ of the n nodes not involved in the transmission can power off) and lower power savings. Finally, observe that the power savings at low loads are higher for networks with more nodes. This is because the number of nodes that can power off is greater when there are more nodes in the network.

The complete network case illustrates the best case performance of PAMAS. At the opposite end of the spectrum we have a line network where, as expected, the savings in power were not as dramatic, see Figure 8. The savings here range from 20% at light loads to less than 10% at heavy loads. The reason for these lower savings is that in a line network, a large number of packet transmissions can go on simultaneously. Thus, fewer nodes are in a position to overhear unintended transmissions.

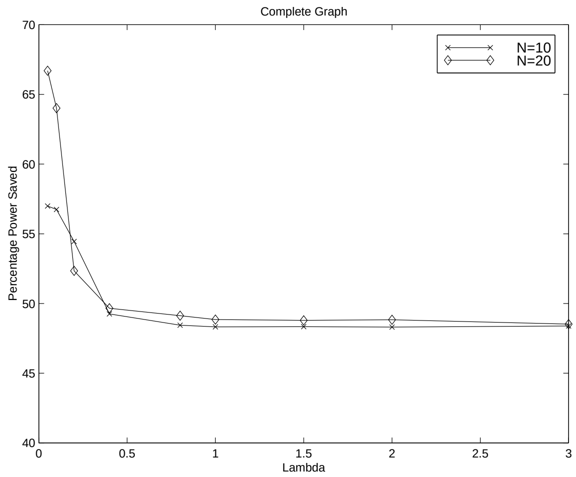

Figure 7: Power saved in complete networks with 10 or 20 nodes.

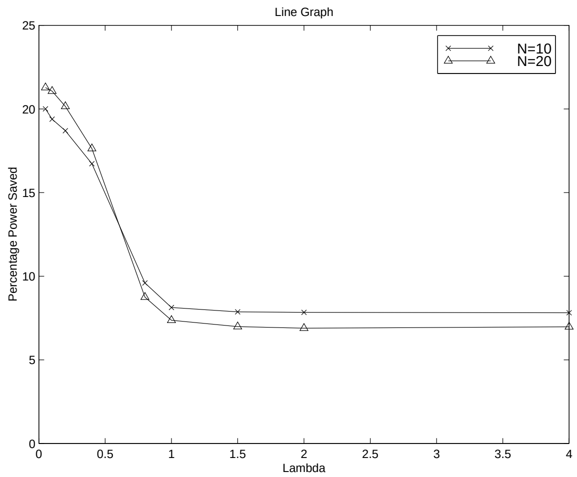

Figure 8: Power saved in line networks with 10 or 20 nodes.

Finally, to get an idea of the dependence of power savings on node connectivity, we ran simulations with random networks. Figure 9 illustrates the power saved as a function of edge probability (low edge probability means sparse networks) for $\lambda = 0.05, 0.1, 0.5, 1.0$ and 4.0. Figure 10 plots the same values for a network with 20 nodes. As these networks show, power savings increase as the network connectivity increases (note that with an edge probability of p, the expected connectivity of a node in a network with 10 nodes is $9p$ whereas it is $19p$ in a network with 20 nodes). This is not surprising because in dense networks, more nodes can power off during transmissions (the extreme case is represented by the fully connected network where all but the single sender and receiver power off).

In the following subsection we develop approximate bounds on the optimal power savings that can be achieved in the different types of networks we have studied. The mathematical formulation of these bounds illustrates the reasons for the high degree of power savings PAMAS achieves.

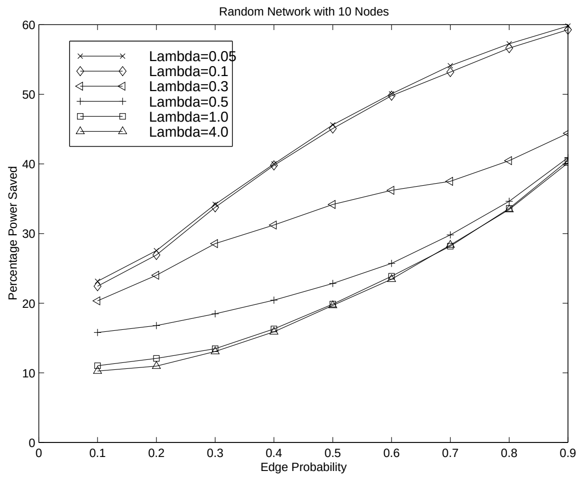

Figure 9: Power saved in random networks with 10 nodes.

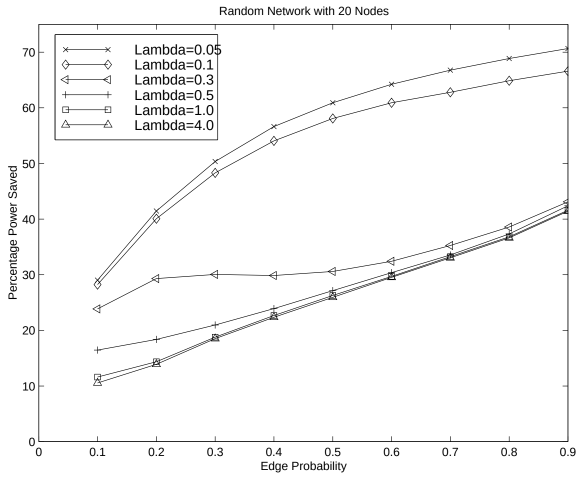

Figure 10: Power saved in random networks with 20 nodes.

4.1 Bounds on Power Savings

An ad hoc network can be modeled by a network, where the nodes represent the mobile radio units and edges represent neighboring nodes. Battery power is consumed by a radio when transmitting a packet and receiving a packet. Most radios available in the market operate by consuming about twice as much power (P_t) when in transmit mode compared to being in receive mode (P_r). A radio also consumes some small amount of power when in idle mode (P_i), i.e., powered on, but neither transmitting nor receiving. Whenever a node k is powered on and a neighbor transmits a packet, it will consume power P_r even if the transmission is not intended for k. Battery power can be saved if we can turn off the radios whenever a neighbor transmits packets not intended for that node.

Assuming point-to-point communication, we can establish some bounds on power savings in an ideal situation. When a radio transmits a packet, it will be intended for one of its neighbors. Therefore, ideally, for each packet transmission exactly one intended neighbor should be powered up to receive the packet and the rest of the nodes should be powered off to maximize power savings.

In a fully connected topology of n nodes, for each packet transmission, $n-2$ nodes can be powered off yielding a power savings of,

$$\frac{(n-2)P_r}{\delta + P_t + (n-1)P_r}$$

where δ is the power consumed by nodes in contending for the channel. The PAMAS protocol achieves the maximum power savings for this case because the unintended nodes power themselves off during each packet transmission. Therefore, our protocol is optimal for the fully connected topology.

Next consider a line topology where n nodes are on a line. Each node has at most two neighbors. In this topology, when a middle node is transmitting, one of its neighbors should be turned off if it is neither transmitting nor able to receive without interference. In the Figure below, we show various scenarios with transmitters and receivers being paired off. A node with symbol T indicates it is a transmitting node, a node with symbol R indicates a receiving node, and a node with symbol O can be powered off. Note that two adjacent nodes can both be transmitting to their neighbors on the opposite sides without interference. Figure (a) shows the tightest packing where almost all nodes are transmitting or receiving. Figure (c) shows the situation where maximum power savings is possible. Here every transmitting node is transmitting to the same side neighbor and in this case one neighbor can be powered off. This situation leads to saving of 25the situation which is somewhere in between – one or more non-participating nodes. Thus, the amount of power that can be saved varies from 0 to 25

```
(a)  T----R----R----T----T----R----R----T----T----

(b)  T----R----R----T----O----T----R----R----T----

(c)  O----T----R----O----T----R----O----T----R----

(d)  R----T----O----O----T----R----R----T----O----
```

Let us analyze this further to calculate power savings on the average. Under ligh load conditions, we expect few nodes to be involved in transmitting a packet. In this case, we expect the pattern O—T—R to be the *typical* pattern most of the time resulting in 25% power savings. Thus the power savings under light load are bounded above by 25%.

When the system load is high, we can analyze the system as follows. Consider a node that is transmitting a packet. One of its neighbor will be receiving this packet. The other neighbor can either be a transmitting node or it can be non-participating node and can be powered off. The different cases with respect to a specific transmitting node are shown below.

```
----R----T----T----R----

---------R----T----T----R----

---------O----T----R-------

---------R----T----O-------
```

We assume that all these 4 cases are equally likely when the load is high. In the first two scenarios no power is saved and in the last two cases there is power saving. The ratio of power saved to total power consumed due to powering off of nodes is given by (.5+.5)/(3+3+2+2) which is 10%. From this analysis, we can conclude that the power savings can range from about 25% under very light load conditions to about 10%under heavy load.

We will now perform similar analysis to determine the power savings in the case of a random network. Let us consider random networks with N nodes. Again, we will consider two extreme situations one where the system is lightly loaded and another where the load is high. In the light load case we assume that transmitters are widely separated while in the heavy load case we assume that as many nodes will transmit as possible without interference.

Let us first consider the light load situation where a transmitting node will not find another transmitter nearby. Therefore, all its neighbors except one receiver can be powered off. We can calculate the power savings as follows. Given n nodes there can be possibly $n(n-1)/2$ edges, and thus the number of distinct networks is $2^{n(n-1)/2}$. Let a specific node be the transmitter. If this node has exactly k neighbors this accounts for k edges. We have accounted for k edges and this node cannot have edges to the remaining $(n-k-1)$ nodes. Therefore, the number of distinct networks with this node having k neighbors is $2^{n(n-1)/2-(n-1)}$. The fraction of all possible networks with this scenario is then simply $2^{-(n-1)}$. There are many ways to choose these k neighbors. Under this light load situation, $k-1$ neighbors can be powered off with the other one being the receiver. Amount of power saved is given by the fraction (k-1)/(k+2). We can calculate the average power saved under light load condition to be

$$\sum_{k=1}^{n-1} (nk)\, 2^{-(n-1)}(k-1)/k+2)$$

Let us now consider the heavy traffic load situation where as many nodes as possible will be transmitting simultaneously without interference. For each transmitting node another node will be a receiver. If we know this maximum number m_t nodes that can transmit simultaneously, then potentially we can power off $n-2*m_t$ nodes. This maximum number of simultaneous transmitters is closely related to the maximum independent set of a network. In a network G, an independent is a set of nodes such that no two nodes in this set have an edge between them. However, this set of nodes cannot be transmitters in a radio network because of two possibilities. One is that two of these nodes may have a common neighbor which can be the intended receiver for one of these two transmitting nodes. Another case is that two neighboring nodes can simultaneously transmit provided their receivers are not neighbors. In fact, in the ad hoc radio network modeled as a network the direction of edges become significant.

We model an n node ad hoc network as a directed network with two directed edges between a pair of nodes that are within the range. If there are e edges in the original network, we will now have 2e directed edges. We label all the edges and the direction of the edge represents the transmitter and the receiver. We construct a new network with 2e nodes where each node in this network corresponds to a directed edge in the original network. We will draw an undirected edge between a pair of nodes in this network if there cannot be simultaneous data transmssions on those two corresponding directed edges in the original network. Finally, we will have 2e node network and nodes in any independent set in this new network can lead to simultaneous transmissions.

4.2 Throughput behavior of PAMAS

Even though the focus of this paper is on power aware protocols, we believe that it is necessary to include a brief discussion of the throughput of PAMAS (a complete analysis will appear in a future paper). We conducted experiments, as described earlier in this section, for random network topologies and counted the number of successful packet transmissions per unit time. This data was collected for random networks with edge probabilities ranging from 0.1 (sparse) to 0.9 (dense) as well as for five values of load $\lambda = 0.05, 0.1, 0.5, 1.0, 4.0 (pkts/sec/node)$.

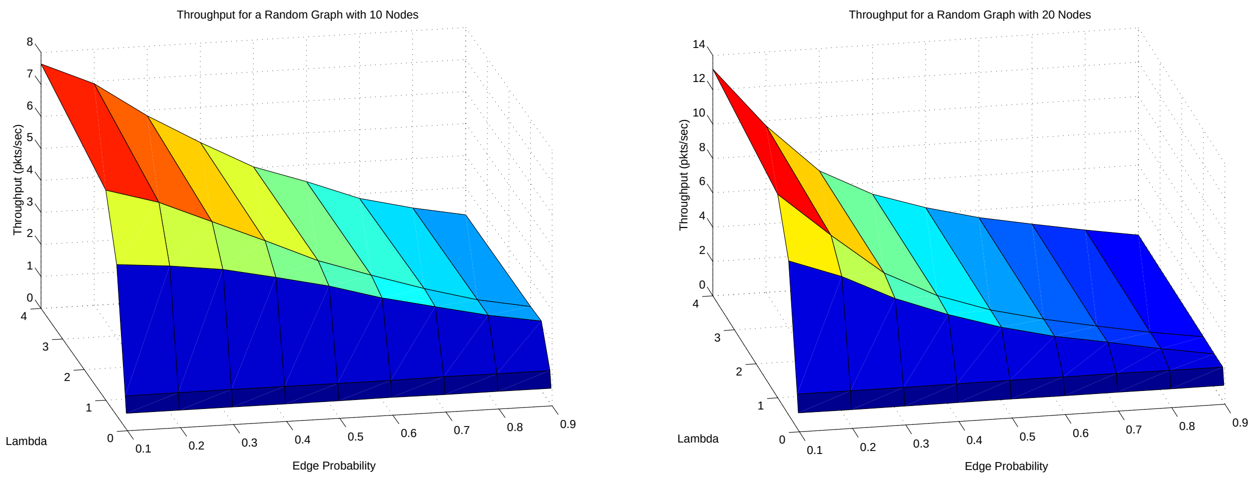

Figure 11: Throughput of PAMAS.

Figure 11 plots the throughput (pkts/sec) transmitted in the network as a function of λ and edge probability. As we can see, sparse networks achieve high throughput at high loads because more simultaneous transmissions can occur simultaneously. Consider the extreme case of a line network with n nodes. Here a maximum of $n/2$ simultaneous transmissions can occur yielding a maximum throughput of $2.6 \times n/2$ packets/sec (in our system one node can transmit a maximum of 2.6 packets per second including the minimum RTS/CTS/Busy Tone exchange). For 10 nodes this value is 13 packets/sec and for 20 nodes it is 26 packets/sec. As Figure 11 illustrates, PAMAS achieves 50% of this theoretical maximum for sparse networks. In dense networks, a node has a large number of neighbors. As a result few simultaneous transmissions can be supported. In a fully connected network (representing the extreme case of dense networks), the maximum obtainable throughput is 2.63 packets/sec. Again, as the figure illustrates, PAMAS achieves over 70% of this theoretical maximum[2].

5 Extensions to PAMAS

From the discussion thus far, it is clear that the PAMAS protocol has very good power conserving behavior. However, it is also clear that the ideas of power awareness that we have developed can be used to make other multiaccess protocols power conserving as well. This is because nodes are powered off only when they are blocked from transmitting or receiving. Thus, the delay characteristics of these protocols will not change. These ideas are discussed further in the next subsection where we outline how power awareness can be added onto protocols such as FAMA, MACAW, and others

[2]In our simulations reported here, all nodes maintain a FIFO queue of packets. New arrivals are put in behind existing arrivals. Thus, even when the system gets overloaded, the *offered load* remains constant. In other words, with a fixed number of nodes n, no more than n packets can compete for the channel regardless of packet arrival rates. This is why the plots for throughput remain constant (at overload) even as λ increases.

discussed earlier in section2. Then, in the following subsection, we discuss possible extensions to PAMAS to handle *broadcasts* and other ad hoc network models.

5.1 Using Power Awareness in other Multiaccess Protocols

In order to conserve power, PAMAS powers off the radio interface in the event that a node is unable to either transmit or receive a packet. In section 3.1 we discussed how a node knows when and for how long to power off. If we are to incorporate power awareness in other multiaccess schemes [2, 5, 13] we need to develop similar protocols that will enable a node to determine when and for how long it needs to power off. Interestingly, however, it turns out that using the same channel for signalling and data limits the extent of power awareness that can be built into these protocols.

Based on our discussion in section 3.1, it is clear that, ideally, a node ought to power off either if it has no packets to transmit and a neighbor is transmitting or if at least one neighbor is transmitting and another is receiving. However, in order to implement this idea, we need to develop a *protocol* for powering off that answers the following questions based on feedback received from the radio channel:

1. When does a node power off?
2. For how long does it power off?
3. What happens when a node powers on and sees an ongoing data transmission?

Let us first look at the FAMA[5] protocol. Here, a node is in the *Passive* state if it does not hear anything on the channel and does not have a packet to transmit. If it receives a packet to send in the *Passive* state, it transmits a RTS and transitions to the *RTS* state awaiting a CTS. If no CTS arrives, it enters a *BACKOFF* state. It stays here for the appropriate interval, and if it does not hear anything on the channel for the entire period, upon coming out of backoff, it transmits a RTS. If it hears a transmission, it goes into the *Remote* state. A station from transitions from the *Passive* state to the *Remote* state upon hearing a transmission or noise as well. In the *Remote* state, the node node waits for a timeperiod before returning to the *Passive* state. The time period is determined as follows:

- If the station hears a RTS, it waits for the time needed to transmit a CTS plus the start of a packet. If it does not hear anything after this time, it goes back to the *Passive* state (or transmits a RTS if it has a packet to send).
- If it hears a noise it waits for the time to send a maximum sized data packet. If it hears a CTS, it waits for the time to send the data packet.

In the *Passive* state, there is no need to power off the node because it is not expending power receiving a transmission. However, power savings can be obtained by turning off the node when it is in the *Remote* state. In the first case, when the node hears an RTS and hears the start of packet transmission, the node could power off for the duration of the transmission (since it knows the packet length). Likewise, if it hears a CTS/noise, it can power off for the appropriate time period. It is clear that the delay and throughput behavior of FAMA does not change with these modifications while its power conserving properties do improve. However, the power savings are not the best possible. This is because, when a node powers back on, it may continue hearing noise/transmissions. This can happen if a node has two or more neighbors (who are not neighbors of each other) who begin transmissions at different times. When the node powers back on, it will not know the remaining length of the current transmissions (this is true even if it had not powered off because it will not hear the CTSs due to collisions with ongoing packet transmissions). Therefore, it will have to remain powered on until the transmissons complete. Observe that under heavy load conditions, this situation will occur frequently resulting in relatively poor power savings. In contrast, in our protocol, the node transmits *probe* packets to determine the additional length of time for which it can power off.

In the MACA[11] and MACAW[2] protocols, a node enters the *Quiet* state when it hears a RTS or a CTS. In the former case, it waits for the packet transmission to begin and once started, it waits for the packet transmission to end. In the latter case, it waits for the packet transmission to end before transitioning out of this state. In either case, it makes sense to power off the node for the duration of the packet transmission. As in the FAMA, however, if new transmissions begin while a node is waiting for an ongoing one to complete, it does not know when the new transmission(s) will end. Thus, once it powers on after staying powered off for the duration of the first transmission, it will have to remain powered on until all the current transmissions finish. This results in needless power consumption. Finally, the MACA/PR [13] protocol can also be modified in a similar fashion (since it is based on MACAW and FAMA) but it suffers from the same drawbacks when it comes to conserving power.

The clustering mechanisms used in [7, 8], in contrast to the schemes discussed above, are far more amenable to power savings. Clustering divides the network into fully-connected components and a different spreading sequence is used for transmission within each cluster. Transmission within a cluster is accomplished using TDMA. Thus, every node is assigned a slot for transmitting its packets (modifications to the basic TDMA allow for the implementation of QoS guarantees). As such, the TDMA scheme is not amenable to power saving because a node cannot really power off (some node may be transmitting a packet to it). However, if we add minislots to the start of each TDMA cycle, we can implement power awareness as follows. For each node in the cluster, allocate a one-bit minislot. All the minislots are set to zero initially. If a node A wants to transmit to node B, A sets the bit in B's minislot. Thus B will remain powered oon for the duration of the TDMA cycle. If, on the other hand, a node's minislot has not been set and it does not have a packet to transmit, it powers itself off for the length of the TDMA cycle.

5.2 Enchancements to PAMAS

Several enhancements are possible to the basic PAMAS protocol we have described. In this section we outline some of the more obvious ones. The first modification would be to add ACKs as has been done in MACAW[2]. Thus, the receiver transmits an ACK when a packet is received correctly. In addition, if the sender does not receive the ACK and transmits a RTS with the same packet number again, the receiver responds with an ACK instead of a CTS. Another modification that will improve throughput is to allow a node to transmit *multiple* packets when it has acquired the channel. This will serve to reduce time spent on channel access (but may increase delays). In order to implement power savings, we will need to ensure that the RTS/CTS/Busy Tone messages include the *total* length of the transmission (length of all packets being transmitted). Thus, a node will know the length of time for which it can power off.

An interesting network model, where power conservation is very important, is where some nodes are either receivers only or only transmitters (e.g., sensors in the field). Our protocol works well here because transmitters can be powered off when they have no packets to transmit (since they do not expect to receive any). The receivers, on the other hand, will never have packets to transmit.

Thus, they always power off when it begins overhearing a neighbors transmission. However, it will need to use the *t_probe* packets to determine the length of the remaining transmission whenever it powers back on again and hears an ongoing transmission.

5.2.1 Support for Broadcasting

Broadcasting is often necessary in networks and, even though broadcasting is typically a network layer function, it is often necessary to provide MAC layer support. In PAMAS and other ad hoc network MAC layer protocols, a sequence of message exchanges (RTS-CTS) preceedes transmission of a packet. This message exchange ensures that the receiver is ready to receive the transmission. However, if a node needs to broadcast a message to all its neighbors, using the RTS-CTS sequence is meaningless because all the neighbors are potential receivers of the broadcast. If they all respond with a CTS (or if some respond with a CTS and others with a Busy Tone), the transmitter will hear noise and will be unable to decide what to do.

In PAMAS, the transmitter transmits a RTS_B message when it needs to transmit a broadcast. As in the basic PAMAS protocol, however, it transmits this message if no neighbor is transmitting or is scheduled to transmit. Upon hearing this, a node receiving another transmission responds with a Busy Tone. Nodes that are free to receive the broadcast *do not respond*. If the transmitter does not hear any response, it transmits the broadcast message. If it hears a Busy Tone or noise, it refrains from transmitting. It waits for the ongoing transmission to end (i.e., it waits for a maximum packet transmission time in case it heard noise or for the length of time specified in the Busy Tone) and retries. When a node begins receiving the broadcast packet, it transmits a Busy Tone to warn other neighbors to refrain from transmitting.

A potential problem in this protocol is that a broadcast may collide with another transmission at some receiver. This is because nodes do not transmit CTSs in response to a RTS_B. Thus, their neighbors, two hops away from the transmitter of the broadcast, are unaware that it is about to receive a broadcast. If a node that heard a RTS_B heard noise when it expected the broadcast, it waits for the transmission to cease and transmits a NACK_B packet to the sender over the data channel (after the usual RTS-CTS exchange). We leave recovery from this situation to the network layer because the network layer is aware of the network topology and is in the best position to decide whether the broadcast need be repeated.

Power awareness can be easily incorporated here as follows. If we assume that every broadcast packet has a unique identifier (that is included in the RTS_B message) then a neighbor of the transmitter who has already received the packet can power itself off for the duration of the transmission.

6 Conclusions

In this paper we developed a novel multiaccess protocol for ad hoc networks that conserves power by turning off radios under certain conditions. We implemented and measured the performance of this protocol and showed that power savings range from 10% (in cases where the network is sparsely connected) to almost 50% in fully-connected networks. The noteworthy aspect of our protocol is that it achieves these power savings without affecting the delay or throughput behavior of the basic protocol. Finally, we discussed the applicability of our power saving ideas to other multiaccess protocols and showed how our ideas may be easily incorporated into these protocols, again without affecting their delay/throughput performance.

References

[1] `http://www.networks.digital.com/npb/html/products_guide/roamwir2.html`

[2] V. Bharghavan, A. Demers, S. Shenkar and L. Zhang,"MACAW: A Media Access Protocol for Wireless LANs", *Proceedings ACM SIGCOMM'94*, pp. 212-225, 1994.

[3] A. Chandrakasan, T. Simon, J. Goodman and W. Rabiner,"Signal Processing for an ultra low power Wireless Video Camera", *3rd International Workshop on Mobile Multimedia Communications*, Princeton, NJ, September 25-27, 1996.

[4] F. Douglis, F. Kaashoek, B. Marsh, R. Caceres, K. Lai and J. Tauber, "Storage Alternatives for Mobile Computers", *Proc. 1994 Symposium on Operating Systems Design and Implementation*, OSDI, November 1994.

[5] Chane L. Fullmer and J.J. Garcia-Luna-Aceves,"Solutions to Hidden Terminal Problems in Wireless Networks", *Proceedings ACM SIGCOMM'97*, Cannes, France, Sept. 14-18, 1997.

[6] J.J. Garcia-Luna-Aceves, Chane L. Fullmer and Ewerton Madruga, "Wireless Mobile Internetworking", Manuscript.

[7] Mario Gerla and J.T.-C. Tsai,"Multicluster, Mobile, Multimedia Radio Network", *ACM-Baltzer Journal of Wireless Networks*, Vol. 1, No. 3, pp. 255-265, 1995.

[8] C. R. Lin, M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," IEEE Jour. Selected Areas in Communications, pp. 1265-1275, Sept. 1997.

[9] S. Murthy, J. J. Garcia-Luna-Aceves, "A routing protocol for packet radio networks," Proc. MOBICOM'95, pp. 86-94. Nov. 1995.

[10] E.P. Harris and K.W. Warren,"Low Power Technologies: A System Perspective", *3rd International Workshop on Mobile Multimedia Communications*, Princeton, NJ, September 25-27, 1996.

[11] P. Karn,"MACA – a New Channel Access Method for Packet Radio", in *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pp. 134-140, 1990.

[12] K. Li, R. Kumpf, P. Horton and T. Anderson,"A Quantitative Analysis of Disk Drive Power Management in Portable Computers", *Proceedings 1994 USENIX*, San Francisco, CA, pp. 279-291, 1994.

[13] Chunhung Richard Lin and Mario Gerla,"Asynchronous Multimedia Multihop Wireless Networks", *Proceedings IEEE INFOCOM'97*, (1997).

[14] A. Sen, M. L. Huson, "A New Model for Scheduling Packet Radio Networks," Wireless Networks, Vol. 3, No. 1, March 1997.

[15] Cheng-shong Wu and Victor O.K. Li,"Receiver-Initiated Busy-Tone Multiple Access in Packet Radio Networks", *Proceedings ACM SIGCOMM'87 Workshop*, Stowe, Vermont, Aug. 11-15, Vol. 17(5), pp. 336-342, 1987.

[16] W. Mangione-Smith and P.S. Ghang,"A low power medium access control protocol for portable multi-media systems", *3rd International Workshop on Mobile Multimedia Communications*,September 25-27, 1996.

[17] M. Stemm and P. Gauthier and D. Harada,"Reducing power consumption of network interfaces in hand-held devices",*3rd International Workshop on Mobile Multimedia Communications*,September 25-27, 1996.

[18] H. S. Wilf, Algorithms and Complexity, Prentice Hall, 1986.

[19] S. Zdonik, M. Franklin, R. Alonso and S. Acharya,"Are "disks in the air" just pie in the sky?", *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, pp. 12-19, December 1994.