

P- 2158



**Power Aware Multiple Access Protocol with
Signaling for Ad hoc Networks**



A PROJECT REPORT

Submitted by

Jeshu Jayanthe C Reg.No: 71204205013

Paul Antony T Reg.No: 71204205026

Vevin Kumar S Reg.No: 71204205057

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2008

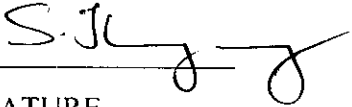


P- 2158

ANNA UNIVERSITY: CHENNAI 600 025

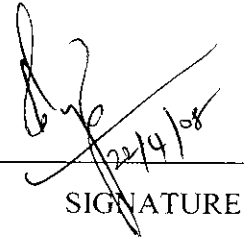
BONAFIDE CERTIFICATE

Certified that this project report “Power Aware Multi Access protocol with Signaling for ad-hoc networks” is the bonafide work of “**Jeshu Jayanthe C, Paul Antony T and Vevin Kumar S**” who carried out the project work under my supervision.



SIGNATURE

Dr.S.Thangasamy



SIGNATURE

Prof.L.S.JayashreeM.E.,(Ph.D.)

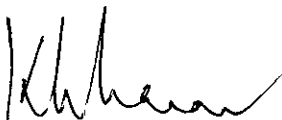
DEAN

Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.

PROJECT GUIDE

Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.

The candidates with University Register No **71204205013**,
71204205026 and **71204205057** were examined by us in the project viva-
voice examination held on 23.04.2008..



INTENAL EXAMINER



EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere thanks to our Vice - Chairman **Dr. K. Arumugam B.E., M.S., M.I.E.**, Correspondent **Shri. M. Balasubramaniam** and Joint Correspondent **Dr. A. Selvakumar** for all their support and ray of strengthening hope extended. We are immensely grateful to our Principal **Dr. Joseph V. Thanikal M.E., (Ph.D.), PDF., CEPIT.** for providing us with all the facilities to complete our project.

We are deeply obliged to **Dr.S.Thangasamy**, Dean, The Department of Computer Science and Engineering for his valuable guidance and useful suggestions during course of this project.

We also extend our heartfelt thanks to our Project Coordinator **Prof. K. R. Baskaran B.E., M.S.**, Department of Information Technology for providing us his support which really helped us.

We are indebted to our Project Guide **Prof. L. S. Jayashree M.E., (Ph.D.)**, Department of Information Technology for her helpful guidance and invaluable support given to us throughout this project.

We thank all the teaching and non-teaching staff of our department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

DECLARATION

We,

Jeshu Jayanthe C	Reg.No:	71204205013
Paul Antony T	Reg.No:	71204205026
Vevin Kumar S	Reg.No:	71204205057

hereby declare that the project entitled “Power Aware Multi Access protocol with Signaling for ad-hoc networks”, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) degree, is a record of original work done by us under the supervision and guidance of the Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

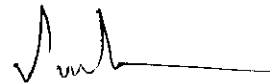
Date: 22/4/08



[Jeshu Jayanthe C]

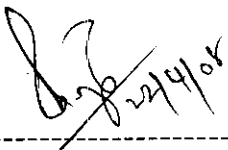


[Paul Antony T]



[Vevin Kumar S]

Project Guided by



[Prof. L. S. Jayashree M.E., (Ph.D.)]

Table of Contents

Chap No.	Title	Page No.
	Abstract	1
	List of Figures	2
	List of Abbreviations	3
1	Introduction	
	1.1 General	4
	1.2 Problem Definition	5
	1.3 Objective of Project	6
2	Literature Review	
	2.1 Feasibility Study	8
	2.1.1 Current Status of Problem	10
	2.1.2 Proposed System and its Advantages	15
	2.2 Hardware Requirements	26
	2.3 Software Requirements	27
	2.4 Software Review	28
3	Details of the Methodology Employed	36
4	Performance Evaluation	
	4.1 Simulation Environment	42
	4.2 Simulation Outcome	44
5	Conclusion	47
6	Future Enhancements	48
7	Appendix	55
8	References	58

Abstract:

In our project, we develop a new multi-access protocol for ad hoc radio networks. The protocol is based on the original MACA protocol with the addition of a separate signaling 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 70% are attainable in most systems. Finally, we discuss how the idea of power awareness can be built into other multi-access protocols as well.

List of Figures

S.No.	Title	Page No.
1.1	Unnecessary Power Consumption	7
2.1	The Hidden Terminal Problem	11
2.2	The PAMAS Protocol	15
2.3	Situation when a node powers back on	21
2.4	Separate interfaces for signaling and data	24
2.5	GloMoSim Architecture	34
3.1	Power saved in complete networks with 10 or 20 nodes	38
3.2	Power saved in line networks with 10 or 20 nodes	39
7.1	Glomosim Visualization Tool	55
7.2	MACA Statistics	56
7.3	PAMAS Statistics	57

List of Abbreviations

PAMAS	-	P ower A ware M ulti A ccess protocol with S ignaling for ad-hoc networks
GLOMOSIM	-	G lobal M obile I nformation S ystem S IMulator
PARSEC	-	P ARallel S imulation E nvironment for C omplex systems
MAC	-	M edium A ccess C ontrol
MACA	-	M ultiple A ccess C ollision A voidance
MACAW	-	M edium A ccess C ollision A voidance W ireless
FAMA	-	F loor A cquisition M ultiple A ccess
MANET	-	M obile A d-hoc N ETwork
RTS	-	R equest T o S end
CTS	-	C lear T o S end
SIR	-	S ignal- I nterface R atio
CDMA	-	C ode D ivision M ultiple A ccess
TDMA	-	T ime D ivision M ultiple A ccess
FIFO	-	F irst I n F irst O ut
mW/hr	-	m illi W atts per h our

1. Introduction:

1.1 General:

Ad Hoc networks are multi-hop wireless networks where all nodes cooperatively maintain network connectivity. These types 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.

1.2 Problem Definition:

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 persons have developed channel access protocols for multihop radio networks where the goal has been maximizing throughput and minimizing transmission delay. Unlike this previous work, however, 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.

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. 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)

1.3 Objective of the Project:

The protocol is based on the original MACA protocol with the addition of a separate signaling 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 70% are attainable in most systems. Finally, we discuss how the idea of power awareness can be built into other multi-access protocols as well.

Significant power is consumed at a node when it either transmits a packet or when it receives a packet. Thus, the DEC Roam about radio consumes approximately 5.76 watts during transmission, 2.88 watts during reception and 0.35 watts when idle. The radio used 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 the following figure, 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.

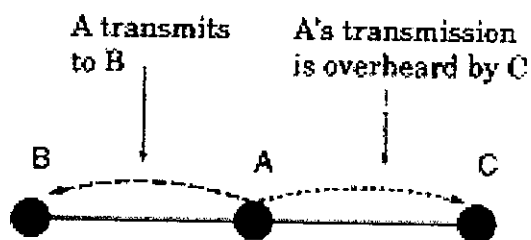


Figure 1.1: Unnecessary power consumption

2. Literature Review:

2.1 Feasibility Study:

An Ad hoc routing protocol is a convention or standard that controls how nodes come to agree which way to route packets between computing devices in a mobile ad-hoc network (MANET). In ad hoc networks, nodes do not have a prior knowledge of topology of network around them and they have to discover it. The basic idea is that a new node (optionally) announces its presence and listens to broadcast announcements from its neighbors. The node learns about new near nodes and ways to reach them, and may announce that it can also reach those nodes. As time goes on, each node knows about all other nodes and one or more ways how to reach them.

Mobile ad-hoc network

A mobile ad-hoc network (MANET) is a kind of wireless ad-hoc network, and is a self-configuring network of mobile routers (and associated hosts) connected by wireless links – the union of which form an arbitrary topology. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet.

Mobile ad hoc networks became a popular subject for research as laptops and 802.11/Wi-Fi wireless networking became widespread in the mid to late 1990s. Many of the academic papers

evaluate protocols and abilities assuming varying degrees of mobility within a bounded space, usually with all nodes within a few hops of each other, and usually with nodes sending data at a constant rate. Different protocols are then evaluated based on the packet drop rate, the overhead introduced by the routing protocol, and other measures. The Children's Machine One Laptop per Child program has developed a cheap laptop for mass distribution to developing countries for education. The laptops will use IEEE 802.11s based ad hoc wireless mesh networking to develop their own communications network out of the box.

Wireless mesh network

A wireless mesh network is a communications network made up of radio nodes in which there are at least two pathways of communication to each node. The coverage area of the radio nodes working as a single network becomes a mesh cloud. Access to this mesh cloud is dependent on the radio nodes working in harmony with each other to create a radio network. A mesh network is reliable and offers redundancy. When a node can no longer operate, all the rest can still communicate with each other directly or through one or more intermediate nodes. The diagrams below illustrate how wireless mesh networks can self form and self heal. Wireless mesh builds routes between nodes only as desired by originating nodes. It maintains these routes as long as they are needed by the originating node. Wireless mesh nodes forms paths in term of hops which connect together to form the wireless mesh network.

2.1.1 Current Status of Problem:

Channel Access Protocols for Ad Hoc Networks

Distributed power control schemes are extensively employed in the cellular networks and are capable of improving the capacity of the network. However, the power control schemes from the cellular networks suffer from performance degradation due to self and direct-interference and hidden-terminal problems when directly employed in ad hoc networks. Most of the existing channel reservation-based power control protocols for ad hoc networks employ incremental power allocation rather than global allocation of the power to the incoming links; thus, they may not effectively utilize the spatial frequency reuse in the network. This paper presents a distributed channel access protocol that couples the channel reservation and the iterative/global transmission power control schemes in ad hoc networks. The designed protocol considers the convergence problem of the global power control in ad hoc networks. The designed access criteria employ the local admission control based on the sufficient criteria for admissibility and global power control for balancing the SIR (signal to interference ratio) of the links. In the performance evaluation study of the designed protocol, an almost twofold increase in the throughput and capacity is observed compared to the existing power-controlled protocol 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. 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. Before doing so, however, it is important to observe that research in building ad hoc packet radio networks was initiated by DARPA. Many access protocols developed as part of this program used some form of CSMA. Suggested approaches for dealing with hidden terminals include using appropriate "randomization delaying" to reduce the probability of receiver-side collisions, the use of CDMA and the use of adaptive transmission scheduling (based on node connectivity) which ensures that the probability of two nodes transmitting to a common receiver is small. We will focus on how the protocols deal with (or not) the hidden terminal problem.

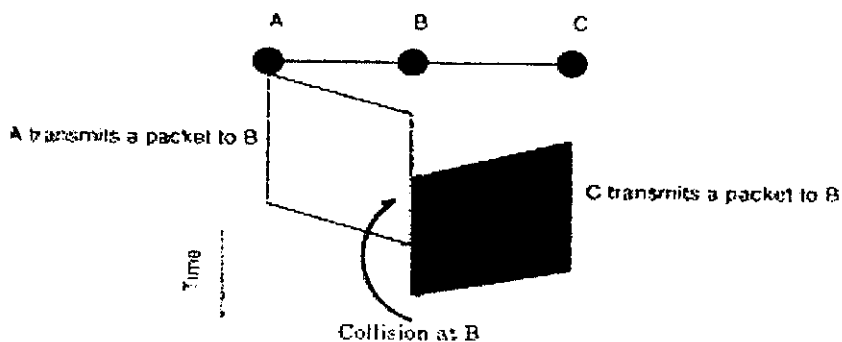


Figure 2.1: The hidden terminal problem.

MACA is a protocol that many other, more recent, protocols are based on. 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 the above figure, 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 back off algorithm and tries to send a RTS again, later.

Even though MACA solves the hidden terminal problem illustrated earlier, it creates another! 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 is a modified version of MACA where link layer ACKs has 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 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. 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 fixes the previous problem also. 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/PR20 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 avoidance of MACA and MACAW.

In addition, all directed traffic uses positive ACKs (again as in MACAW). A different approach to the problem is described later. In this system, there is a separate channel used for transmitting "busy tones". Thus, when a node wants to transmit a packet, 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 and using a different spreading code in each cluster.

2.1.2 Proposed System and its Advantages:

The PAMAS Protocol

The PAMAS protocol is a combination of the original MACA protocol and the idea of using a separate signalling channel. 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 they can power themselves off. In this section we first present the PAMAS protocol. Later, we add power conserving behavior to the protocol in a way that does not change the delay or throughput behavior of PAMAS.

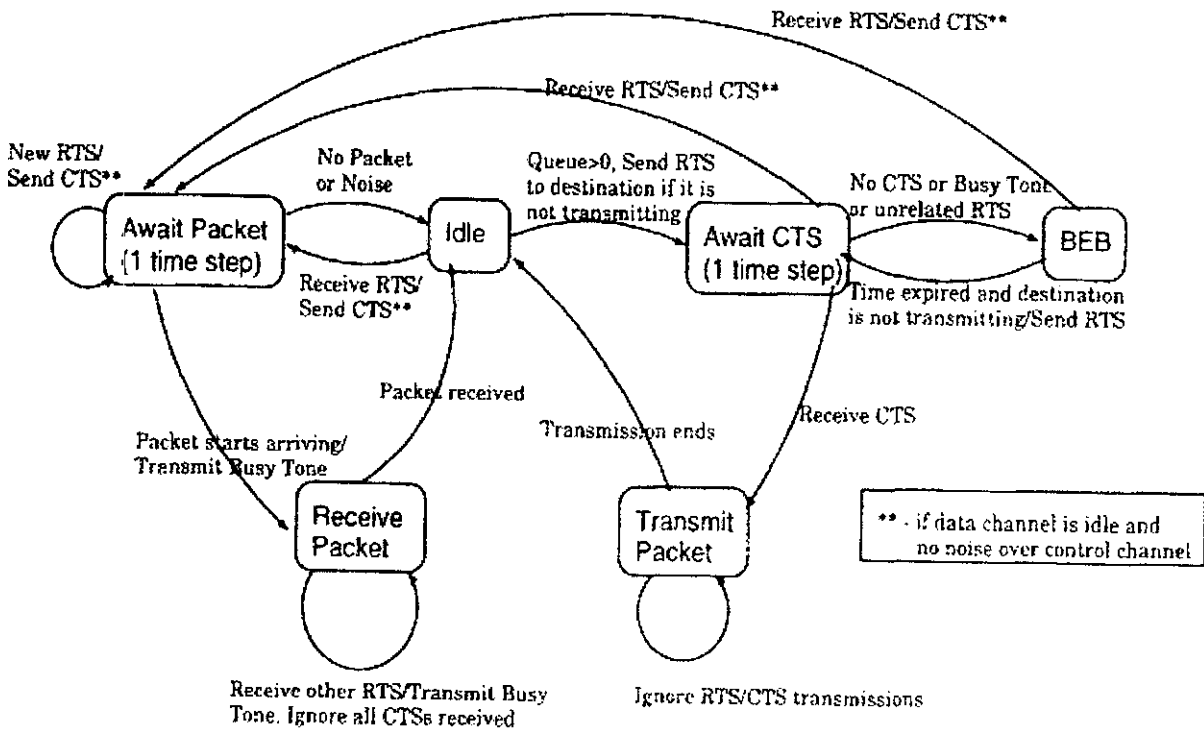


Figure 2.2: The PAMAS Protocol

The state diagram outlining the behavior of our protocol is illustrated in the above Figure. As indicated in the figure, a node may be in any one of six states - Idle, AwaitCTS, BEB (Binary Exponential Back off), 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 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 back off (the BEB state in the figure). If the 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 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. 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. In our protocol, if the node heard noise over the control channel within $T/2$ 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 T , 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

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. 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 neighbor transmitting the RTS will not receive the expected CTS. Thus, the neighbor's 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 the last diagram.

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. 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).

Powering off radios

We found 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).

We assume that the data channel and the control channel have identical conditions (e.g., noise). 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.

- If a node has no packets to transmit, then that node ought to power itself off if a neighbor begins transmitting.
- 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:

- For how long is a node powered off?
- 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). Say node D has a packet to transmit to node C. Since C is powered off, D's RTSs go unanswered causing D to go into BEB.

What happens if C was not powered off? In this case, since C's neighbor B is transmitting a packet, C will not respond to D's RTSs. Thus, C's behavior, from the viewpoint of D, is the same irrespective of whether C is powered off or not! As a corollary, we can see that packet delays do not increase as a result of powering off nodes. This is because the period of time when a node is powered off is one where it can neither receive packets nor can it transmit packets.

To answer the first question we need to consider several cases. Ideally, a node ought to stay powered off whenever any of the two conditions (0) hold. However, collisions in the signaling 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?

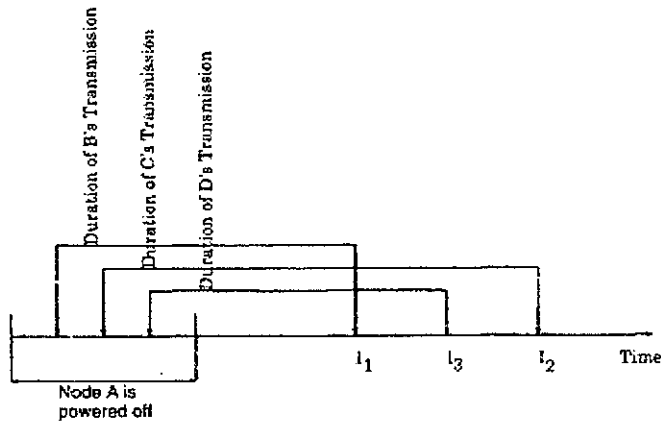


Figure 2.3: Situation when a node powers back on

Figure 2.3 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} 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 $t_{\text{probe_response}}(t)$ 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 battery 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 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 (powered off) node, that packet cannot be delivered. The node powers off its control channel as well and therefore does not know the length of the remaining transmissions.

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 chosen 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).

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 signaling interface powered on. This will enable the node to always know the length of new transmissions and keep the data interface powered off appropriately.

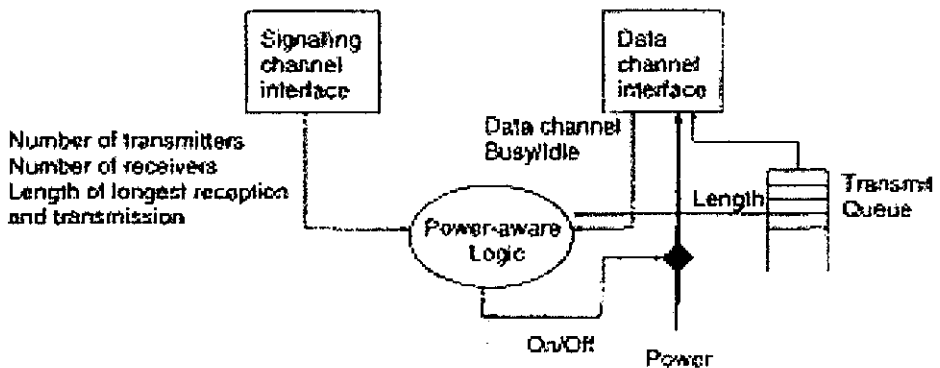


Figure 2.4: Separate interfaces for signaling and data

Figure 2.4 illustrates the block diagram needed for this type of a communications device. Here, the signaling 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.

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 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.

2.2 Hardware Requirements:

Processor	:	Pentium IV
Speed	:	Above 500 MHz
RAM capacity	:	128 MB
Floppy disk drive	:	1.44 MB
Hard disk drive	:	20 GB
Key Board	:	108 keys
Mouse	:	Optical Mouse
CD Writer	:	Optional
Printer	:	Optional
Motherboard	:	Intel
Monitor	:	17"

2.3 Software Requirements:

Operating System	:	Windows 2000 and above
Front end used	:	Java
Simulator used	:	Glomosim
Software needed	:	MS Visual C++ 6.0
Coding language	:	Parsec C

2.4 Software Overview:

JAVA Overview:

This language was initially called “Oak” but was renamed as “Java”. The primary motivation of was the need for a platform-independent language that could be used to create software to be embedded in various customer electronic devices. An easier and most cost-efficient was needed. In an attempt to find such a solution, the programmers began work on a portable platform-independent language that could be used to produce code that run on a verity of CPU’s under differing environments. This effect led to the creation of Java.

However, with the emergence of World Wide Web, Java was propelled to be forefront of computer language design, because the web too demanded portable programs. Java is designed, tested and refined by real, working programmers. It is a language grounded in the needs and experience of people who devised it. Thus Java is a programmer’s language. Java is cohesive and logically consistent. Except for those constraints imposed by the internet environment, Java gives you, the programmer full control. If you program well, your programs reflect it. If you program poorly, your program reflects that too.

Java Applet and Application

Java can used to create two types of programs called “applet and application”. An application is a program that runs on your

computer, under the operating system of that computer. An applet is an application designed to be transmitted over the internet and executed by Java compatible web-server.

Security

When a Java compatible web-server is used, the user can download applets without fear of virus infection. Java archives this protection by confining a Java program to the Java execution environment and not allowing it access to other parts of the computer.

Portability

Many types of computers and operating systems are in use throughout the world and many are connected to the internet. For programs to be dynamically downloaded to all various type of platform connected to the internet, some means of generating portable executable code is needed. The same mechanism that helps ensure security also helps create portability. Indeed, Java's solution to these two problems is both elegant and efficient. Java was designed to be easy for the professional programmer to learn and use effectively.

Object-Oriented

The object model in Java is simple and easy to extend, while simple types, such as integers are kept as high-performance nonobjective. The ability to create robust programs was given a high priority in the

design of Java. To gain reliability, Java restrict user in a few key areas, to force to find mistakes in early in program development. At the same time, Java frees the user from having to worry about many of the most common causes of programming errors. Because Java is strictly typed language, it checks the user code at compile time and it also checks the code at runtime.

Multithreaded

Java was designed to meet the real-world requirements of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows the user to write programs that do work simultaneously.

Architecture-neutral

A central issue for Java designers was that of code longevity and portability. One of the main problems facing programmers that no guarantee exists that if you write a program today, it will run tomorrow, even in the same machine, operating system upgrades, processor upgrades, changes in core system resources can all combine to make program malfunction. But in Java the goal was “write once, run anywhere, any time, forever”.

Interpreted and High Performance

Java enables cross-platform program by compiling into a intermediate representation. The code can be interpreted on any system that provides Java virtual machine. Cross-platform solution has done at the expense of performance. Java, however, was designed to perform well on very low-power CPUs. Java runtime machine that provides this feature lose none of the benefits of platform-independent code. “High performance cross-platform” is no longer an oxymoron.

Distributed

Java is designed for distributed environment of the internet, because it handles TCP/IP protocols. The original version of Java (Oak) includes features for intra-address-space messaging. This allows objects on two computers to execute procedures remotely. Java has revived these interfaces in a package called remote method invocation (RMI).

Dynamic

Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at runtime. This makes it possible to dynamically link code in a safe and expedient manner.

GLAMOSIM Overview:

Our project has been done using a scalable simulation environment called GloMoSim (for Global Mobile Information System Simulator) that effectively utilizes parallel execution to reduce the simulation time of detailed high-fidelity models of large communication networks. GloMoSim has been designed to be extensible and composable: the communication protocol stack for wireless networks is divided into a set of layers, each with its own API. Models of protocols at one layer interact with those at a lower (or higher) layer only via these APIs. The modular implementation enables consistent comparison of multiple protocols at a given layer. The parallel implementation of GloMoSim can be executed using a variety of conservative synchronization protocols, which include the null message and conditional event algorithms.

Parsec

PARSEC (for PARAllel Simulation Environment for Complex systems) is a C-based simulation language developed by the Parallel Computing Laboratory at UCLA, for sequential and parallel execution of discrete-event simulation models. It can also be used as a parallel programming language. PARSEC runs on several platforms, including most recent UNIX variants as well as Windows. PARSEC adopts the process interaction approach to discrete-event simulation. An object (also referred to as a physical process) or set of objects in the physical system is represented by a logical process. Interactions among physical processes

(events) are modeled by time-stamped message exchanges among the corresponding logical processes. One of the important distinguishing features of PARSEC is its ability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures. PARSEC is designed to cleanly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute it. Thus, with few modifications, a PARSEC program may be executed using the traditional sequential (Global Event List) simulation protocol or one of many parallel optimistic or conservative protocols. In addition, PARSEC provides powerful message receiving constructs that result in shorter and more natural simulation programs.

About Glomosim

GloMoSim is a mobile simulator built using C language. All message transfers and other network elements are handled by the individual layer coding built in C. To make the concepts clear, GloMoSim provides users with a Visualization Tool (VT) built using Java. The VT helps us understand the network environment, the node positions, message transfers, clustering details, etc.

Glomosim Architecture

The networking stack is decomposed into a number of layers as shown in Figure 2.5. A number of protocols have been developed at each

layer and models of these protocols or layers can be developed at different levels of granularity.

In our project we deal with all these layers, but most of the coding has been implemented in the Clustering and Routing layers. The dynamic clustering algorithm is implemented in these two layers. The clustering algorithm has been fully implemented in the Clustering layer. The coding present in the Routing layer has been largely modified according to the cluster formation. The algorithm used for routing is the Bellman-Ford algorithm. This algorithm maintains a routing table for every node being simulated.

The communication between the various layers is accomplished by means of the various APIs available. A common API between two layers helps in the communication between those two layers.

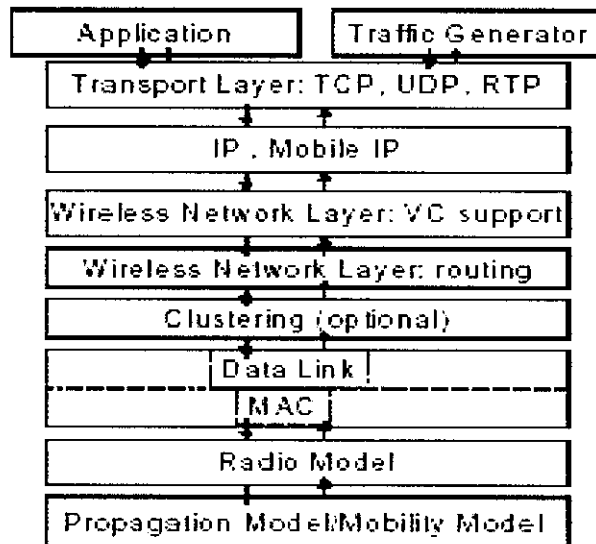


Figure 2.5: GloMoSim Architecture

Glomosim Library

GloMoSim is a scalable simulation library for wireless network systems built using the PARSEC simulation environment. GloMoSim also supports two different node mobility models. Nodes can move according to a model that is generally referred to as the “random waypoint” model. A node chooses a random destination within the simulated terrain and moves to that location based on the speed specified in the configuration file. After reaching its destination, the node pauses for a duration that is also specified in the configuration file. The other mobility model in GloMoSim is referred to as the “random drunken” model. A node periodically moves to a position chosen randomly from its immediate neighboring positions. The frequency of the change in node position is based on a parameter specified in the configuration file.

3. Details of the Methodology Employed:

In order to characterize the power (or energy) conserving behavior of our protocol we conducted extensive simulations where we compared the energy 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 call power off because more simultaneous transmissions are possible. An implication of this is that the throughput will typically be higher in sparse networks because more transmissions can go on simultaneously.

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 energy is consumed for 32 bytes of transmission and., 0.5 units of energy are consumed at a receiver for every 32 bytes processed (again, data or control). We ran simulations

for networks with 10 and 20 nodes to see how energy conservation scaled. For the random networks, we generated edges randomly uniformly with probabilities between 0.1 and 0.9 for different experiments. A probability of 0.1 generates sparse networks while an edge probability of 0.9 yields dense networks with much better energy conserving behavior. Traffic arrived at *each* node according to a poisson process. The destination was chosen randomly uniformly from the remaining $n - 1$ nodes and the packet was routed using the shortest path. All nodes maintain a FIFO buffer of packets awaiting transmission. The length of this buffer is fixed at $2n$ per nodes. Packets arriving at a node with a full buffer are dropped. 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 . A packet may be received by more than one node and is therefore counted more than once) and the total number of the packets P transmitted during the experiment. The energy expended per packet is then,

$$P = (B_t + 0.5 \times B_r)/P$$

If we use power conservation, the number of bytes received tends to be smaller, say it is B_{rc} . Then, the energy expended is,

$$P_c = (B_t + 0.5 \times B_{rc})/P$$

and the power savings can be written as,

$$\text{Percentage of Power Saved} = P - P_c/P$$

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

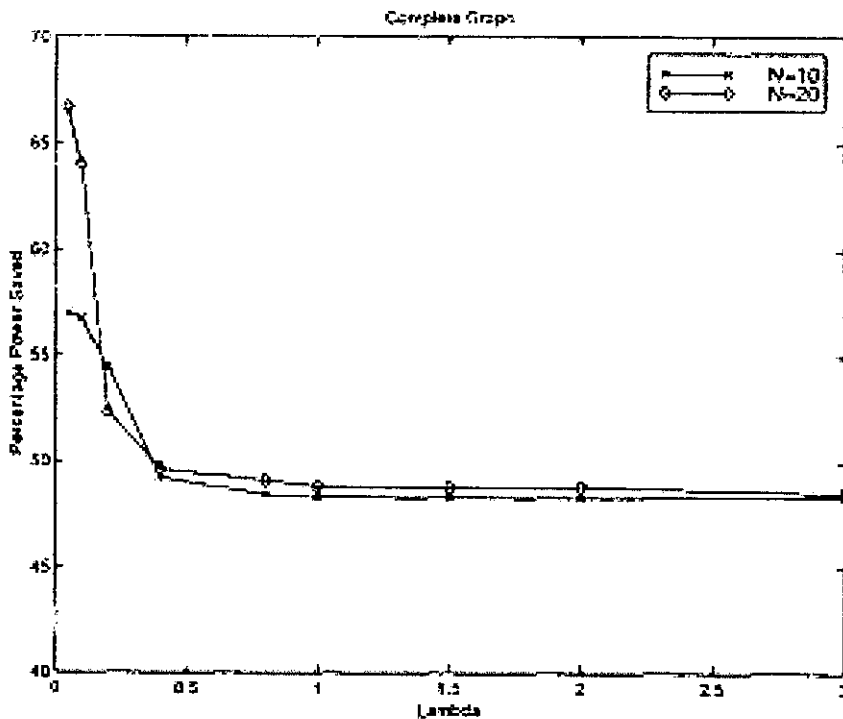


Figure 3.1: Power saved in complete networks with 10 or 20 nodes

Figure 3.1 plots the power saved in a line network (i.e., fully connected) topology as a function of load lambda (packets/sec/node). 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/CTS/Busy Tone 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.

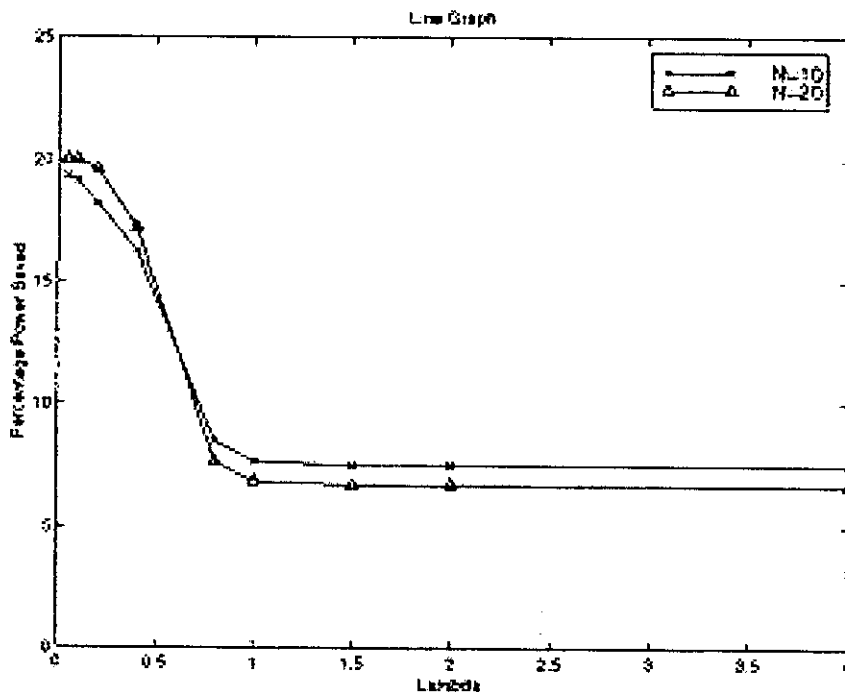


Figure 3.2: Power saved in line networks with 10 or 20 nodes

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. 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.

Bounds and Approximations on Energy Savings

An ad hoc network can be modeled by a graph, where the nodes represent the mobile radio units and edges represent neighboring nodes. Battery power is consumed by a radio when transmitting or 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 a very 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 node 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. For this analysis, we consider normalized power consumption by a node to be 1 unit in transmit mode, 0.5 unit in receive mode, and 0 units in idle mode.

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. Our PAMAS protocol achieves this maximum power savings as the unintended nodes can power themselves off during each packet transmission. In addition, all the nodes know exactly how long to

turn themselves off, and therefore, our protocol is optimal for the fully connected topology.

Bounds for the Line Topology

Consider a line topology with n nodes. 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 that neighbor is neither transmitting nor able to receive without interference. In the following figure, 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 off. Note that two adjacent nodes can both be transmitting to their neighbors on the opposite sides without interference. Figure (a) shows the most tight packing where almost all nodes are transmitting or receiving. Figures (c,d) show the situation where the most energy savings is possible. Here, for every transmitting node one neighbor can be turned off as it is neither transmitting nor receiving. Figure (b) shows the situation which is somewhere in between. In the following, we will derive the bounds on energy savings by considering light load and heavy load conditions.

(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~

4. Performance Evaluation:

4.1 Simulation Environment:

In the simulation environment, 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 energy is consumed for 32 bytes of transmission and., 0.5 units of energy are consumed at a receiver for every 32 bytes processed (again, data or control).

We ran simulations for networks with 5 to 10 nodes to see how energy conservation scaled. For the random networks, we generated edges randomly uniformly with probabilities between 0.1 and 0.9 for different experiments. A probability of 0.1 generates sparse networks while an edge probability of 0.9 yields dense networks with much better energy conserving behavior. Traffic arrived at *each* node according to a poisson process. The destination was chosen randomly uniformly from the remaining $n - 1$ nodes and the packet was routed using the shortest path. All nodes maintain a FIFO buffer of packets awaiting transmission. The length of this buffer is fixed at $2n$ per nodes. Packets arriving at a node with a full buffer are dropped.

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 . A packet may be received by more than one node and is therefore counted more than once) and the total number of the packets P transmitted during the experiment.

The primary purpose of the Java VT is to help network designers debug their protocols. It is written in Java to provide portability across multiple platforms. The GloMoSim simulation can be run with or without the VT. If run without the VT, it can just be executed from the command line. However, if run with the VT, it must be executed through the GUI provided by the VT rather than from the command line.

There are basically two ways to run GloMoSim with the VT real time and play back. If we choose "Real Time" from the Simulate menu, then the VT will display the results of GloMoSim while GloMoSim is running. If we wish to run GloMoSim first and then play the results of the simulation back later, then we need to do the following.

- Choose "Write Trace" from the Simulate menu. A dialog will pop up asking for the name of the executable and the name of the trace file to write the output from GloMoSim to.
- Once the trace file has been written to, you can play it back by choosing "Play Back" from the Simulate menu. A dialog will pop up asking for the name of the trace file to play back.

4.2 Simulation Outcome:

Statistics for Simulation of the MACA Protocol:

Node: 0, RadioAccnoise, Collisions: 1

Node: 0, RadioAccnoise, Energy consumption (in mWhr): 150.012

Node: 0, NetworkIp, Number of Packets Delivered To this Node: 32

Node: 1, RadioAccnoise, Collisions: 0

Node: 1, RadioAccnoise, Energy consumption (in mWhr): 150.007

Node: 1, NetworkIp, Number of Packets Delivered To this Node: 38

Node: 2, RadioAccnoise, Collisions: 0

Node: 2, RadioAccnoise, Energy consumption (in mWhr): 150.005

Node: 2, NetworkIp, Number of Packets Delivered To this Node: 23

Node: 3, RadioAccnoise, Collisions: 5

Node: 3, RadioAccnoise, Energy consumption (in mWhr): 150.005

Node: 3, NetworkIp, Number of Packets Delivered To this Node: 24

Node: 4, RadioAccnoise, Collisions: 7

Node: 4, RadioAccnoise, Energy consumption (in mWhr): 150.000

Node: 4, NetworkIp, Number of Packets Delivered To this Node: 0

Node: 5, RadioAccnoise, Collisions: 1

Node: 5, RadioAccnoise, Energy consumption (in mWhr): 150.000

Node: 5, NetworkIp, Number of Packets Delivered To this Node: 0

Statistics for Simulation of the PAMAS Protocol:

Node: 0, RadioAccnoise, Collisions: 0

Node: 0, RadioAccnoise, Energy consumption (in mWhr): 134.992

Node: 0, NetworkIp, Number of Packets Delivered To this Node: 82

Node: 1, RadioAccnoise, Collisions: 0

Node: 1, RadioAccnoise, Energy consumption (in mWhr): 134.989

Node: 1, NetworkIp, Number of Packets Delivered To this Node: 90

Node: 2, RadioAccnoise, Collisions: 0

Node: 2, RadioAccnoise, Energy consumption (in mWhr): 134.985

Node: 2, NetworkIp, Number of Packets Delivered To this Node: 34

Node: 3, RadioAccnoise, Collisions: 0

Node: 3, RadioAccnoise, Energy consumption (in mWhr): 134.984

Node: 3, NetworkIp, Number of Packets Delivered To this Node: 36

Node: 4, RadioAccnoise, Collisions: 0

Node: 4, RadioAccnoise, Energy consumption (in mWhr): 134.984

Node: 4, NetworkIp, Number of Packets Delivered To this Node: 28

Node: 5, RadioAccnoise, Collisions: 0

Node: 5, RadioAccnoise, Energy consumption (in mWhr): 134.984

Node: 5, NetworkIp, Number of Packets Delivered To this Node: 32

Energy Calculations:

Energy Consumed in MACA Protocol:

Simulation time: 10 minutes

Total energy consumed (in mWhr): 900.029

Total number of packets delivered: 117

Energy consumed per packet (in mWhr): 7.69255

Total number of collisions occurred: 14

Energy Consumed in PAMAS Protocol:

Simulation time: 10 minutes

Total energy consumed (in mWhr): 809.918

Total number of packets delivered: 302

Energy consumed per packet (in mWhr): 2.68185

Total number of collisions occurred: 0

Energy Conserved:

Energy conserved per packet (in mWhr): **5.0107**

Percentage of energy conserved: **65.14 %**

5. Conclusion:

In this project, 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 the PAMAS protocol and showed that power savings range from 10% (in cases where the network is sparsely connected) to almost 70% 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 will discuss the applicability of our power saving ideas to other multi-access protocols and show how our ideas may be easily incorporated into these protocols without affecting their delay/throughput performance.

6. Future Enhancements:

From the discussion so 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 multi-access 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. Now, we discuss possible extensions to PAMAS to improve its power conserving behavior and extensions to handle broadcasts.

Using Power Awareness in other Multi-access 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. If we are to incorporate power awareness in other multi-access schemes, 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, 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:

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

Let us first look at the FAMA 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 back off, it transmits a RTS. If it hears a transmission, it goes into the Remote state. A station transitions from the Passive state to the Remote state upon hearing a transmission or noise. In the Remote state, the node waits for a time period 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 noise it waits for the time to send a maximum sized data packet. If it hears a CTS, it waits for the time required 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 CTS due to collisions with ongoing packet transmissions). Therefore, it will have to remain powered on until the transmissions 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 (on the control channel) to determine the additional length of time for which it can power off.

In the MACA and MACAW 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 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 contrast to the schemes discussed above, are far more amenable power savings. Clustering divides the network into distinct 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 on 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.

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. 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.

Another possible enhancement to PAMAS is to power off the data interface of a node when its signalling interface is trying to acquire the channel. Thus, in a line sub network A-B-C-D, if C is transmitting to D while B is sending an RTS to A, powering off B's data interface ensures that C's transmission does not result in power consumption at B.

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) precedes 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 the RTS_B, a node receiving another transmission responds with a Busy Tone. Nodes those are free to receive the broadcast do not respond. If the transmitter does not hear any response in time equal to one roundtrip time plus processing delay, it transmits the broadcast message. If it hears a Busy Tone or noise in response to the RTS_B, 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.

7. Appendix:

Appendix A: Simulation Snapshots:

Glomosim Simulation Environment:

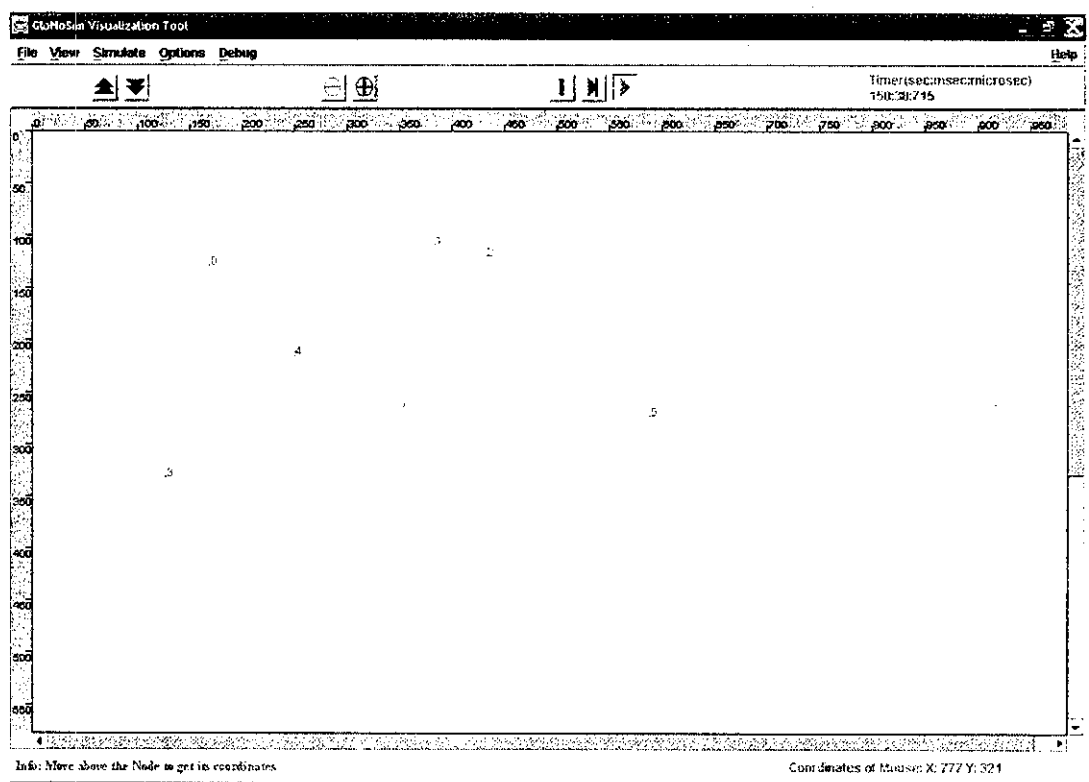


Figure 7.1: Glomosim Visualization Tool

Simulation Time	:	10 Minutes
No. of Nodes	:	6
Terrain Dimensions	:	(600,600)
Node Placement	:	Uniform
Mobility	:	None

Glomosim Statistics File for MACA Protocol:

The screenshot shows a dialog box titled "GlomoSim Statistics File: glomo.stat". It contains a list of statistics for nodes 0 through 5, categorized by layer (RadioAccnoise and NetworkIp). The statistics include collisions, energy consumption (in mWhr), and the number of packets delivered to each node. The dialog has a "Show Stats" button and a "Cancel" button.

Node	Layer	Statistic	Value
0	RadioAccnoise	Collisions	1
0	RadioAccnoise	Energy consumption (in mWhr)	150.012
0	NetworkIp	Number of Packets Delivered To this Node	32
1	RadioAccnoise	Collisions	0
1	RadioAccnoise	Energy consumption (in mWhr)	150.007
1	NetworkIp	Number of Packets Delivered To this Node	38
2	RadioAccnoise	Collisions	0
2	RadioAccnoise	Energy consumption (in mWhr)	150.005
2	NetworkIp	Number of Packets Delivered To this Node	23
3	RadioAccnoise	Collisions	5
3	RadioAccnoise	Energy consumption (in mWhr)	150.005
3	NetworkIp	Number of Packets Delivered To this Node	24
4	RadioAccnoise	Collisions	7
4	RadioAccnoise	Energy consumption (in mWhr)	150.000
4	NetworkIp	Number of Packets Delivered To this Node	0
5	RadioAccnoise	Collisions	1
5	RadioAccnoise	Energy consumption (in mWhr)	150.000
5	NetworkIp	Number of Packets Delivered To this Node	0

Figure 7.2: MACA Statistics

Glomosim Statistics File for PAMAS Protocol:

GloMoSim Statistics File: glomo.stat	
	Node: 0, Layer: RadioAccnoise, Collisions: 0
	Node: 0, Layer: RadioAccnoise, Energy consumption (in mWhr): 134.992
	Node: 0, Layer: NetworkIp, Number of Packets Delivered To this Node: 82
	Node: 1, Layer: RadioAccnoise, Collisions: 0
<input checked="" type="checkbox"/> RadioAccnoise	Node: 1, Layer: RadioAccnoise, Energy consumption (in mWhr): 134.989
	Node: 1, Layer: NetworkIp, Number of Packets Delivered To this Node: 90
	Node: 2, Layer: RadioAccnoise, Collisions: 0
	Node: 2, Layer: RadioAccnoise, Energy consumption (in mWhr): 134.985
	Node: 2, Layer: NetworkIp, Number of Packets Delivered To this Node: 34
	Node: 3, Layer: RadioAccnoise, Collisions: 0
	Node: 3, Layer: RadioAccnoise, Energy consumption (in mWhr): 134.984
	Node: 3, Layer: NetworkIp, Number of Packets Delivered To this Node: 36
	Node: 4, Layer: RadioAccnoise, Collisions: 0
	Node: 4, Layer: RadioAccnoise, Energy consumption (in mWhr): 134.984
	Node: 4, Layer: NetworkIp, Number of Packets Delivered To this Node: 28
<input checked="" type="checkbox"/> NetworkIp	Node: 5, Layer: RadioAccnoise, Collisions: 0
	Node: 5, Layer: RadioAccnoise, Energy consumption (in mWhr): 134.984
	Node: 5, Layer: NetworkIp, Number of Packets Delivered To this Node: 32

Figure 7.3: PAMAS Statistics

8. References:

- [1] Suresh Singh, C.S. Raghavendra, "PAMAS - Power Aware Multi-Access protocol with Signaling for Ad-Hoc Networks", ACM SIGCOMM, Computer Communication Review.
- [2] S.Krishna Murthy, "Power Awareness in Ad Hoc Networks", UCR.
- [3] Rong Zheng, Robin Kravets, "On-demand Power Management for Ad Hoc Networks", Department of Computer Science, University of Illinois.
- [4] Wikipedia – The Free Encyclopedia, <http://www.wikipedia.com> .
- [5] A. Chockalingam and M. Zorzi, 1998, "Energy Consumption Performance of a class of Access Protocols for Mobile Data Networks", Proc. IEEE VTC'98, Ottawa, Canada.
- [6] Chane L. Fullmer and J.J. Garcia-Luna-Aceves, 1997, "Solutions to Hidden Terminal Problems in Wireless Networks", Cannes, France.
- [7] J.J.Garcia-Luna-Aceves, Chane L. Fullmer and Ewerton Madruga, 2000, "Wireless Mobile Internetworking".
- [8] Krishna M. Sivalingam, M. B. Srivastava, P. Agarwal and J-C. Chen, "Low-Power Access Protocols Based on Scheduling for Wireless and Mobile Networks", <http://www.eecs.wsu.edu/~krishna> .

[9] Michele Zorzi and R. R. Rao, 1997, "Energy Management in wireless Communications", Paper presented during workshop on Third Generation Wireless Information Networks.

[10] H.M. Deitel, P.J. Deitel, "Java: How to Program", Fifth Edition, Prentice Hall of India private Ltd.

[11] David J.Kruglinski, George Shepherd, Scot Wingo, 1999, "Programming Visual C++", Microsoft Press.

[12] Jochen Schiller, 2003, "Mobile Communications", Pearson Education, Second edition.

[13] William Stallings, 2002, "Wireless Communications and Networks", Pearson Education.

[14] Web-o-pedia, <http://www.webopedia.com>

[15] Sun Website, <http://www.java.sun.com>



P-2158