The Design and Implementation of a Low Power Ad Hoc Protocol Stack

Gray Girling, Jennifer Li Kam Wa, Paul Osborn
AT&T Laboratories Cambridge
24a Trumpington Street
Cambridge, UK
http://www.uk.research.att.com/
{GGirling, JLi-Kam-Wa, POsborn}@uk.research.att.com

Radina Stefanova
Laboratory for Communications Engineering
Department of Engineering
University of Cambridge
Cambridge, UK
http://www-lce.eng.cam.ac.uk
Radina.Stefanova@cl.cam.ac.uk

Abstract - **Low power consumption is a key design metric for wireless network devices that have limited battery energy. The problem of reducing power consumption needs to be addressed at every level of system design. This paper investigates the issues of designing low power protocols in the context of the PEN system, a mobile ad hoc network developed at AT&T Laboratories Cambridge. It describes the ad hoc protocols that have been implemented, outlining both the design of individual protocols and the structure of the overall stack. The power-relevant mechanisms from the various protocols are collated in a summary.**

I. Introduction

In a mobile embedded networking environment, the communicating nodes are small and rely on limited battery energy for their operation. Since energy is a limited resource, and battery technology has been slow to improve, the design of low power architectures and protocols has become a pressing issue. This paper investigates the issues of designing a low power protocol stack for the PEN[1] system, a mobile ad hoc radio network developed at AT&T Laboratories Cambridge [1]. An embedded network is one made up of a set of autonomous, embeddable, and possibly mobile, nodes that communicate intermittently. Nodes are radio-peers: each node is obliged neither to listen nor to transmit continuously. The maximum achievable bandwidth on an embedded network is quite low, making it more suited for control rather than data applications. To reduce power consumption, PEN nodes are powered down for much of their duty cycle and awaken periodically to rendezvous with other nodes. This has interesting implications on the design of Medium Access Control (MAC) and higher-level protocols, requiring them to deal with the problem of nodes being powered down and thus unable to respond immediately.

A prime design requirement for the protocols is that they are as simple as possible in order to minimise power consumption and maintain the overall energy efficient properties of the stack. For this reason, existing de-facto protocols are not really suited for the envisaged energy-constrained environment. IP, for instance, is just too complex, particularly with respect to its routing functions; a typical implementation yields far greater code size and results in system demands that will be too excessive for a PEN node.

Although they are by no means standard, the function of many of the layers in the PEN protocol stack are similar to those in the Open Systems Interconnection (OSI) seven layer protocol stack [5]. In particular the PEN MAC layer has a similar function to the OSI Link layer – to share the physical communication medium; the PEN Routing layer has a similar function to the OSI Network layer – to route over subnetworks; and, the PEN Transport layer has a similar function to the OSI Transport layer – to provide a required quality of service. One major difference, however, is the presence of an additional layer between the MAC layer and the Routing layer called the Rendezvous layer which is responsible for scheduling and forecasting times of inactivity and thus has a major role in power saving.

Each protocol in the stack has been implemented as an independent module and they are accessed through Application Program Interfaces (APIs) that are supersets of the one used to access the Transport layer in an end system. This means that many of the layers can be omitted from a system build should the function they provide not be required. For example the Routing layer can be omitted when the system requires interaction only with local nodes, or the Transport layer can be omitted if large Service Data Units (SDUs) and data integrity are not required. Reduced software size can have a beneficial effect on power consumption if it results in a lower processor overhead for the periods during which a node is active.

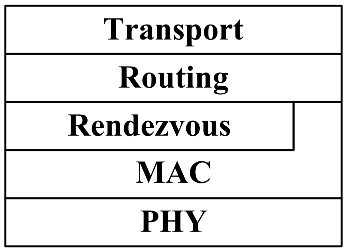

Fig. 1. The PEN protocol stack

[1] Prototype Embedded Network – this is the working title for the Embedded Network previously known as "Piconet"

II. Physical Layer

In order to simplify development, the PEN hardware was designed to make use of off-the-shelf components, whilst at the same time striving to minimise power consumption and maximise functionality. A Radiometrix 'BIM-418' module operating at 418MHz using narrow-band frequency modulation provides the radio interface. This is capable of a raw data-rate of 40kbit/s, half-duplex, but requires a balanced data stream so a 4 bit to 6 bit encoding is used, reducing the usable data-rate to around 25kbit/s.

Although the BIM provides a carrier detect signal, it was found to be extremely unreliable, so a data-sense technique had to be used to detect channel utilisation. This relies upon an out-of-band 62 bit preamble at the start of each packet, which, upon reception, allows synchronization of the receive clock and then detection of a start of packet marker.

A Xilinx 3090 Field Programmable Gate Array (FPGA) implements the encoding which also provides an effective error detection method since only 1 in 4 6-bit codes are valid. The FPGA also handles preamble generation and detection, as well as implementing some of the MAC features discussed in the next section.

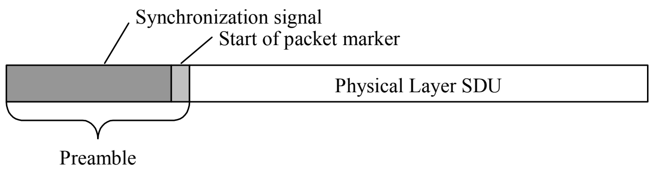

Fig. 2. Physical layer PDU format

III. MAC LAYER

The Media Access Control protocol serves two main functions: the first is to provide a communication service appropriate to the hardware available, and the second is to do so in a manner that will minimise power consumption. Given that data is to be communicated, power can be saved only by mechanisms that attempt to minimise the amount of time that the radio has to be on. One such type of mechanism reduces the amount of radio traffic simply by choosing a time to transmit that minimises the likelihood of failure and therefore retransmission. Note that design choices in favour of such considered timing are likely to result in less aggressive protocols that would not meet a design choice in favour of high throughput or low latency. In this sense low power protocol operation must be traded against these other qualities of service.

The MAC protocol uses a novel contention mechanism which bounds the probability of collision given likely communication scenarios and traffic levels. It is part of the philosophy behind PEN that communication should be possible when all nodes in range turn on their transmitters and receivers only occasionally. This effectively removes the possibility of exploiting continuous synchronization signals, such as might be required in a Time Division Multiple Access (TDMA) scheme. The contention mechanism is novel because it makes use of implicit synchronization signals inevitably created by the normal operation of the radio to schedule transmissions in a way that will result in fewer collisions than a pure Data Sense Multiple Access (DSMA) scheme, but more than a TDMA scheme. In a sense, the protocol is an intermediate hybrid between these two extremes.

When the radio channel is busy, there will be a tendency for transmissions from different nodes to collide. Since the hardware prevents transmissions only once a valid preamble is read, collisions that occur during a preamble will not be detected. In that case, some other mechanism is needed to detect the collision. Acknowledgements are used for this purpose (but only for unicast data units, ACKs from multicast transmissions would be too difficult to implement - MAC multicast transmissions therefore are inherently susceptible to this loss). Unacknowledged unicast data cannot be guaranteed not to have been delivered since the ACK may have collided undetectably; a retried transmission can therefore result in duplicated delivery.

When there are a number of nodes contending for the radio channel, it is probable (although not necessary) that they will all observe the same traffic and many of them will be able to determine the time at which the current transmission has completed. In effect, this event synchronizes the contending nodes that observed it. It is important that these nodes do not simply choose to (re)try their transmissions once the channel becomes free (since this will guarantee an invisible collision that will require an ACK timeout to detect).

To address this problem the MAC protocol introduces a fixed contention period of C ms following these events. Each node chooses a random time within the contention period to start its preamble of P (1.55) ms and does not collide with another preamble starting in that period with a probability of $(1+P/C)^{-1}$. Thus, when there are n contending nodes the probability that there is no collision with the first preamble is $(1+P/C)^{-(n-1)}$, and if there is no collision the transmitting node will have been chosen fairly. The protocol uses a 32 ms contention period yielding a probability of a successful transmission of $0.954^{(n-1)}$ although the effects of unsynchronized nodes will actually reduce this.

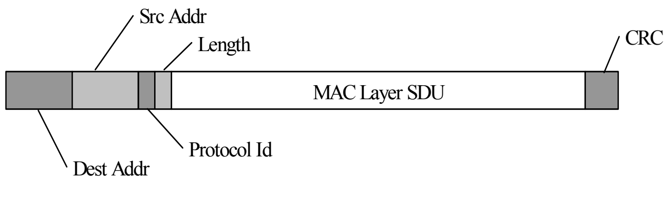

Fig. 3. MAC layer PDU format

In order to prevent ACKs having to contend with other synchronous transmissions, a fixed allowance is made by all nodes that detect the end of a unicast transmission for the time that an ACK is expected to take to be returned. If no ACK is observed, synchronized nodes will begin their contention period only after this allowance has expired. In order for this delay to be minimised, it is important for ACKs to be generated promptly. In the case of unicast data the MAC protocol also handles the retransmission of data units - a maximum number of retransmissions being specified by the protocol's user. There are two main reasons why a data unit might not be acknowledged:

- the addressed node does not exist or is not currently receiving so acknowledgements are not being sent; and,
- the radio channel is congested and data units or their acknowledgements are being lost

To allow for the latter case, random ‘backoff’ times chosen from a retry duration that increases linearly with every retried transmission are introduced.

Power management involves the periodic shutdown of the radio, both its transmitter and its receiver. When power is re-established, the reason is often in order to perform a radio transmission. Because the radio can detect only data unit preambles, and not the data units themselves, no transmission is allowed for the maximum duration that a data unit can take to pass. However, if a preamble is sensed and reported before the expiry of that time, it is possible to contend for the channel earlier, along with the other users of the channel. In certain circumstances this might actually reduce the latency for access to the radio channel when there is already traffic on the channel (particularly if it consists of small multicast data units - many of which can pass in the maximum packet time).

In summary, the MAC protocol is designed to:

- Provide fair contention for the radio channel under likely, lightly loaded, circumstances
- Provide an acknowledged unicast service (with no guarantee of single delivery)
- Provide a best-effort unacknowledged multicast service

IV. RENDEZVOUS LAYER

The MAC layer described above provides a means for nearby nodes to exchange PDUs, attempting to control power usage by minimising the likelihood of collision and retransmission. In order to hear incoming packets, the MAC layer must leave the radio receiver active; with the current hardware, this requires the CPU, FPGA and transceiver to be powered on. Hence, a node actively listening for packets consumes approximately 55mA whilst a node that is asleep consumes only 16µA. If the transceiver were able to recognise and process incoming packets by itself, the power consumption might not have been so high and leaving it active all the time may have been an acceptable burden. To obtain a significant reduction in power usage however, it becomes necessary to power down the CPU, FPGA and transceiver as often as possible.

This is where the Rendezvous layer comes into play. It is a communication protocol designed to sit just above the MAC layer and encapsulate all of these power saving decisions and timings independently of the MAC layer. Given the PEN vision of ad hoc networking, with radio-peer communication between mobile nodes, designing a protocol that allows nodes to spend most of their time with their receivers switched off is difficult. Existing solutions include using a master node to store data whilst the destination is powered down and forwarding the data when it powers up, or providing a framing structure for other nodes to synchronize their active periods to ([2], [3]). The problem with these approaches is that they presuppose the existence of a master node with a large battery source that can consume significantly more power than the slave nodes; they are therefore not radio-peer solutions. Also, the necessary designation of master nodes, and the formation of clusters are expensive procedures given the degree of mobility that is anticipated for PEN nodes. Since node interactions may occur in many varied environments, it is important that communication between low power nodes be possible without relying on the presence of nodes with greater power reserves.

To meet these requirements an asynchronous protocol that supports radio-peer communication between low power nodes is proposed.

A. Beacons

The basis for this design is introduced in [12] which explores the behaviour of two rendezvous systems: 'server beaconing' and 'client beaconing', as the rate of interaction increases. 'Server beaconing' has the server node sleeping most of the time, but periodically waking to send out a broadcast packet and advertise its presence. Following this broadcast, the node will listen briefly for replies from interested clients. Any client that wishes to use a particular server must switch on its receiver and listen for beacons from the server. 'Client beaconing' reverses the above strategy. The server listens continuously for broadcasts from clients wishing to use its service. Whilst this results in high power consumption for a server node, it minimises the power consumption of client nodes. As a result, this scheme offers lower overall power consumption when there are a large number of clients trying to access a relatively small number of servers.

In the PEN environment, it is anticipated that the rate of interaction will be fairly low. A scheduled rendezvous can easily be negotiated in cases where an ongoing association is required, so the basic rendezvous scheme is intended for the remaining first contact scenarios. As such, a variation on 'server beaconing' is the best solution. In PEN, 'server beaconing' is actually applied to any node that could accept Rendezvous layer datagram SDUs, whether or not it is a server. Such nodes will send out periodic beacons using the MAC multicast service, on a well-known multicast address, to all Rendezvous layer equipped nodes. Any node wishing to send Rendezvous layer datagram SDUs would listen for beacons from such nodes, and then forward its SDUs. Since reception is comparatively expensive relative to transmission, this ensures that the default state for a server node is one in which the minimum power is being expended. Nodes wishing to send SDUs must expend more power listening for beacons, and so the penalty, in terms of power, is placed upon the node originating the exchange. Note that this system takes advantage of the fact that PEN operates on a single frequency: frequency hopping systems would need to isolate a control frequency on which to broadcast availability.

Following the transmission of its beacon a node must listen for replies before powering down. This listen window must be long enough to allow all nearby nodes to send their replies, but at the same time be minimised to avoid wasting power. To resolve these requirements a sliding window system is used. A minimum listen duration T_{min} is specified, along with a window duration T_{window}. The node will remain listening for at least T_{min} and until an interval T_{window} has passed with no further PDUs received.

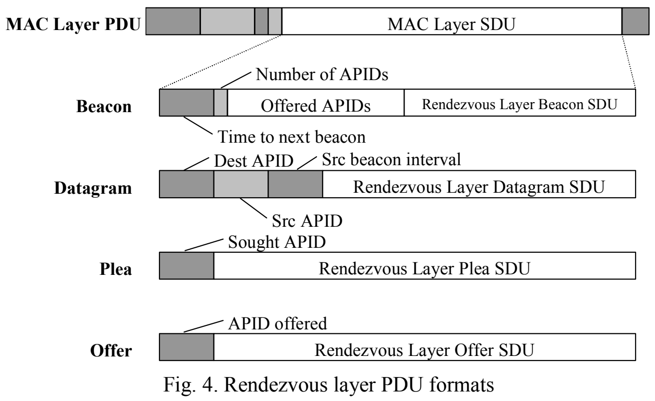

Fig. 4. Rendezvous layer PDU formats

B. Discovery and Description

An important part of PEN is the process of discovering nearby nodes that offer facilities with which one wishes to interact. It is of course important that this is a power efficient process. By having each node send out beacons that contain a list of the services offered by that node, the processes of discovery and description are combined. This avoids the need to expend power querying the capabilities of a node after its presence is discovered. To minimise the size of the beacon PDUs, short identifiers called Application Protocol IDentifiers (APIDs) are used to identify different service types. When an application wishes to receive incoming datagram SDUs, it specifies the APID that it wishes to listen on. This APID is then appended to the beacon PDU, advertising the service to interested clients. APIDs are similar to TCP port numbers in that they determine both the application and the protocol being communicated with.

By varying the interval between beacons it is possible to customise the power usage profile of a node. It is assumed that nodes spend most of their time idle. Whilst idle the only protocol activity is waking up periodically to send out a beacon, so if the interval between beacons is increased, so is battery life. However, if the beacon period is too long, nodes wishing to communicate must spend longer waiting to hear a beacon and so will consume more power. It is important to note that this also increases the latency of the communication. As a result there is a trade off between battery life and communication latency.

C. Application specified QOS

Different applications will have differing requirements for latency and battery life, so the selection of beacon interval is best left to the application. This also ensures that application developers are aware of the power decisions that they are making, leading to power efficient applications [8]. However, there is a need to reconcile differing requirements when a node contains many applications. This is fairly straightforward as the overhead in advertising multiple services in a single beacon is minimal, so only a single beacon is required[2]. The interval for this beacon should be the minimum of those requested by the applications. Since at least one application has requested this interval, there is a justification for expending the power needed to satisfy it, and all other applications will benefit from the reduced latency it provides.

When a node wishes to send a Rendezvous layer datagram SDU, it must listen for a beacon from the destination node. Since nodes may be mobile, it is possible that the destination node may not be within range. In this case it is important to have an upper bound on how long the source node will listen before timing out. It is this issue that brings together APIDs and beacon intervals. All datagram SDUs have a destination APID that denotes which type of application on the destination node the SDU should be sent to. When that destination application opened its APID it had to specify its required beacon interval. In order to send to that application, the application's APID must be known, so it is only a small extra requirement that the associated beacon interval be known. In this way the APID and interval can be considered a tuple that is required for addressing. The actual beacon interval of the node may be smaller than this value, but not greater. Once the beacon interval of the destination application is known, it is possible to timeout after a sensible multiple of this time. The exact multiple will depend upon the reliability of broadcast transmissions in the MAC layer.

[2] Unless the number of APIDs being advertised is too great to fit within a single beacon, in which case a tiered beacon structure could be used, advertising high priority APIDs in one frequently broadcast beacon, and others in a less frequently broadcast beacon.

D. Transmission Modes

The beacon system discussed above offers a number of useful transmission modes when sending a datagram SDU. Firstly there is a unicast form of transmission called TX_SPECIFIC. In this transmission mode the sender specifies a Rendezvous address consisting of the destination MAC address and destination APID (with associated interval I), both of which must match at the receiver. The sender will then listen for a beacon from the requested destination and forward the Rendezvous SDU if the requested APID is advertised. Under ideal conditions, this should result in the sending node listening for an average time of I/2.

A second mode of transmission offers a less directed form of unicast termed TX_ANY. The sender specifies a Rendezvous address consisting of just a destination APID (and associated interval); it does not specify a destination MAC address. As before, the node will listen for beacons but the datagram SDU is forwarded to the first recipient found to offer that APID. This mode will reduce the time spent listening when there are many suitable recipients and is useful for service discovery when any instance of a given service is acceptable.

The final mode offered is the least power efficient, but offers a form of broadcast that may be useful in some cases. It is called TX_ALL and attempts to forward a copy of the Rendezvous SDU to all recipients within range offering the requested APID. To accomplish this, the node must listen for at least a full beacon interval and forward a copy of the datagram SDU to each suitable recipient detected. Upon completion a list of nodes that the SDU was sent to is returned to the application, along with the MAC return code for each attempt.

When a transmission is pending, a node will actively listen for a suitable destination node to be heard. This power consumption accounts for much of that consumed in Rendezvous communications. Whilst it is necessary to listen in order to hear the destination node, an SDU may not be sufficiently important to justify the power consumption. For that reason a 'best effort only' flag is added to SDUs sent to the Rendezvous layer for transmission. This flag indicates that the node should not expend power listening on behalf of this SDU, but if there is no significant penalty for doing so, the SDU should otherwise be treated as normal. Frequently there will be other pending SDUs which result in the node listening, so 'best effort' SDUs would still stand a chance of success. Continuously powered nodes incur no penalty for listening, so can treat 'best effort' SDUs like any other.

E. Pleas and Offers

Nodes that have large and cheap power reserves gain little by going to sleep. However, with the algorithm described above a node wishing to send to a continuously powered node would still wait for it to broadcast a beacon. One solution might be to send the datagram SDU speculatively, in case the destination node happens to be listening. If that fails, the normal approach of waiting for a beacon can be adopted. Whilst this would work, it has the disadvantage of sending a potentially large PDU when the destination is not listening. Because of the design of the MAC layer, this would result in a number of retransmissions, which waste further bandwidth.

The solution adopted is to use small 'plea' PDUs. When a datagram SDU is queued for transmission, a plea PDU containing only its destination addressing information is broadcast. Any node receiving this plea that can satisfy its requirements will respond with an 'offer' PDU. Queued SDUs are therefore sent on receipt of either a beacon, or an offer. This approach allows a node to contact other active nodes almost immediately and ensures that the datagram SDUs are only ever sent when the destination node is known to be listening, and using the MAC acknowledged unicast mode. In the case of the TX_ANY transmission mode, the originating node will not have to wait if a suitable recipient is available, and such datagrams are more likely to be sent to continuously powered nodes that have greater power resources with which to process the datagram.

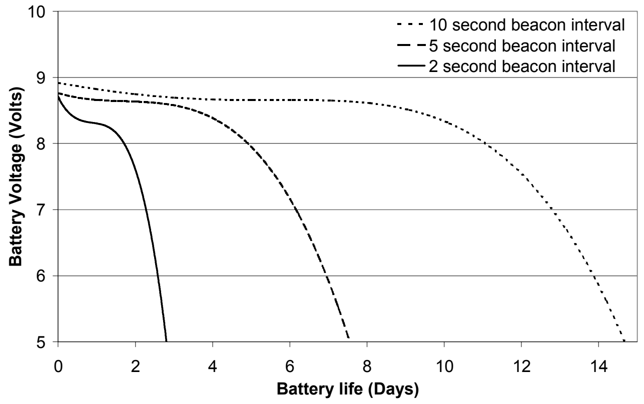

Fig. 5. Battery life for various beacon intervals

Although their occurrence is not necessarily synchronized with a Rendezvous layer user, the beacon, offer and plea PDUs do provide a relatively low power opportunity for (limited) communication. Therefore, each of these PDUs incorporates an SDU that can be used by the user – for example, by the Routing layer.

F. Results

A standard PEN node runs from a 9V lithium battery (Ultralife U9VL), chosen so as to maximise energy density. Whilst asleep the node consumes only 16μA, but this can rise as high as 90mA when active. By using a power efficient idle task, an average consumption of 55mA is achieved for a node with its transceiver active.

A node that does not use the Rendezvous layer must keep its transceiver active at all times, so as to receive PDUs. In the prototype hardware this prevents the MAC layer and CPU from powering down, so such a node will drain its battery in approximately 20 hours. Using the Rendezvous layer, the node can power down in periods that a node has not advertised its availability – resulting in power savings proportional to the time during which it is asleep. These savings are demonstrated in Fig. 5.

Depending upon the application, considerably longer beacon intervals may be acceptable. For example, in a distributed sensor application, a sensor may only need to beacon once an hour to upload its readings. Such behaviour could extend the battery life to several months. Unlike many other protocols (such as those of Bluetooth which has a minimum duty cycle) it is possible for different nodes to exhibit radically different behaviour depending upon their applications' requirements; there is no enforced set of timings to which all nodes must abide.

In summary, the Rendezvous layer:

- permits communication between nodes that spend much of their time asleep
- provides a mechanism for node discovery
- allows nodes to vary their power usage depending upon their application

V. Routing Layer

The PEN Routing and Relay layer has been specifically designed to route data between nodes in a power-managed network. Logically this layer is above the Rendezvous layer, but in practice its implementation is closely integrated with that of the Rendezvous layer.

The lack of fixed infrastructure in PEN complicates the task of routing by requiring each host in the network to act as a relay. The low power requirement adds a further constraint to the routing problem, as the algorithm must be energy efficient. Existing routing algorithms do not perform well in this respect as most of them have not considered the energy of hosts in the network as a resource constraint. They assume that nodes can be powered on continuously and all nodes within range of a source will always receive its transmissions. These assumptions do not hold in a power-managed network like PEN as nodes are powered down for the majority of their duty cycle in order to conserve energy. In fact, it can further be said that broadcasts are not viable in such a power-managed network because of their high energy cost.

The Routing protocol allows nodes to communicate with neighbours more than one radio hop away without placing a significant overhead on the energy resources of the network. Most importantly, the Routing algorithm allows nodes to continue operating in a low power mode. Furthermore, the per-hop routing delay is comparable to the Rendezvous layer synchronization delay between nodes.

In addition to data delivery, the PEN Routing layer consists of two further functions: route discovery and route maintenance. Routes are discovered on demand as this makes a more efficient use of the energy in nodes since they do not have to expend energy discovering and maintaining routes they may never use. The penalty paid for this approach is a greater delivery delay when a node does not have a route to a destination as it must first perform the route discovery. Route maintenance also takes an on demand approach. A route is detected as broken because a node tried to use it and failed, and only then are other nodes notified or the route fixed.

Nodes may be mobile, but one major assumption is that they do not move very frequently or with a great speed. In a large network this is liable to be the case. Node mobility degrades the performance of any routing algorithm as routes are broken more frequently and have to be reconfigured, but it does not add any extra complexity to the routing problem: route maintenance is still required to handle cases where the network topology does change, but more importantly - because of the unpredictable nature of radio as a communication medium - it must be able to reconfigure a route in the case where intermediate relays experience so much radio channel interference that they are unable to function.

When any transmission using the Rendezvous layer is made, a far larger amount of energy is expended in synchronizing with the receiving node than in transmitting the data PDU. This implies that the PDUs periodically used to synchronize nodes should be used to exchange as much routing information as possible. The functions of the different Rendezvous layer PDUs, pleas, offers and beacons, have been described in the previous section and the routing algorithm uses these to carry different types of routing information.

A. Route discovery

Route discovery begins with a node requesting a route to a particular destination that is not available in the routing table. Each node maintains a list of destinations that it requires a route to and includes this list (or as much of it as possible) into every plea PDU it transmits. On receiving such a plea, a neighbouring peer will check every route requested and reply with an offer PDU containing any of the routes it has available. For any route it does not have, it will add the destination to its own list of routes to discover. The routing table stored at each node contains an entry for every destination that it has discovered a route to, consisting of a destination address, the address of the next hop node along the route to the destination, a routing metric and a destination-originated sequence number. Currently, the hop count to the destination is used as the routing metric but other measures could be used. The sequence numbers are used to ensure the loop freedom of the routing algorithm. Although this routing algorithm is designed to discover routes on demand, it is advantageous to cache routes once they have been discovered in order to avoid spending extra energy and time re-discovering them. There is always a danger that a node may use a stale route, but considering the amount of time and energy expended in discovering a new route it is more desirable to rely on route maintenance to detect and fix broken routes.

As described above, non-continuously powered nodes mainly use the beaconing mechanism to synchronize. Beacon PDUs are therefore used to carry two types of routing information: routes that the node needs and routes that it has available. On receiving a beacon PDU containing route requests, a node sends a gratuitous offer containing any of those routes that it has in its routing table. This extra offer is the only protocol exchange that is not due to the normal operation of the Rendezvous layer. As it does not know what route its neighbours may need, the node may include its whole routing table in the beacon PDU, space permitting, or choose a certain number of entries to advertise cyclically. On receiving an offer or a beacon PDU, a node updates its routing table with new entries, or with better ones as long as the sequence number of the new route is higher than that of the one being replaced.

B. Route maintenance

The route maintenance procedure must be able to detect a broken route and either attempt to route around the breakage or inform other nodes trying to route through it that the route is no longer valid. Detecting a broken link is a difficult task because nodes are powered down for most of the time and the failure of a node to respond could mean that it has powered down or that it has moved or that the radio channel is busy. A node must be able to detect reliably that a neighbour has moved out of its transmission range with as few false positives as possible because each time a broken link is detected the route maintenance procedure is invoked at one or more nodes. This is a waste of energy if the route is not broken and may result in a worse route being used (assuming the network has previously converged on optimal routes). While a node has its receiver powered on, it will receive a number of MAC layer PDUs not addressed to it that would normally be discarded. However they contain one important piece of information: namely that the node that sent them was within transmission range at that time. Using this information each node's Routing layer maintains a list of all its neighbours along with the time that it last received a transmission from them and this table is used in determining whether a given neighbour has moved out of range.

A failure to transmit data using the Rendezvous layer indicates that either a route could not be found or the next hop node did not synchronize with this one (either due to radio interference or node migration). The table of neighbours can be used to give some indication of whether that neighbour has moved. If the elapsed time since the last transmission is greater than some pre-defined interval, then it is assumed that the neighbour has moved. If the node assumes that its neighbour has not moved, then the transmission must have failed because the two nodes failed to synchronize, most likely due to radio interference. The PDU is then given another chance at transmission but this time using the 'best effort' facility. This allows a PDU to be retried without exerting any overhead on the node's energy resources and in fact the only extra energy that the node consumes is in transmitting the PDU. Therefore if the assumption that the neighbour has not moved away is correct, the PDU is given another chance at transmission. The penalty paid is in an increase in the route reconfiguration delay due to the node having a second transmission attempt; however this is justified by the aim to minimise energy consumption rather than routing delay.

If a particular route is found to be broken, the source of the failed PDU must be notified. The routing PDU is marked as being an error and returned along a route to its source if such a route is available. If at some point an error PDU fails to be transmitted, it is discarded. This is a rather drastic approach to route maintenance as the source of a route may not be notified for some time that the route is broken and may continue sending PDUs along it. However, the route maintenance procedure should not be a big overhead on the energy resources of the relay nodes as the emphasis is on low power consumption and only a best effort service to the data PDUs.

Furthermore, as the PDU has just used the route, up to the broken link, and as the topology is assumed to vary slowly, it is very likely that that portion of the route is still intact, so this failure scenario is not likely to occur often.

C. Results

The results presented in this section have been obtained using a simulation of a PEN network that has been calibrated using empirical measurements. The reason for presenting simulation results is that each of the curves on the graph represents approximately 500 hours of simulation time. In order to produce more accurate results and because random network topologies are used each point in the curves is averaged over twenty simulation runs.

Fig. 6 shows the average energy consumption, i.e. the average amount of time spent powered on, per node in a network of ten nodes. The network is assumed to have a static topology and therefore the probability of a link is used to define the binomial probability of any two nodes in the network being able to directly communicate with each other. A probability of one corresponds to a fully connected network where all nodes can communicate directly with each other and as the probability is decreased the network becomes increasingly more partitioned. The graph shows that the routing protocol does not present a considerable power consumption overhead, but to the contrary using routing actually results in lower power consumption for the nodes in the majority of cases. The simulation is set up so that a network topology of 10 nodes is initialised at the start of each simulation run. The beacon interval for all nodes in the network is defined to be 10 seconds and the mean time between datagram arrivals at each node is also set at 10 seconds. All datagrams are transmitted using the TX_SPECIFIC primitive and for this purpose a node address from the network is chosen at random, all node addresses being equally likely, and the source node attempts to transmit its datagram to that destination.

As can be seen below, despite the extra information transmitted by the Routing layer, on average, it results in lower power consumption at each node. This is most pronounced for the more sparsely connected network topologies. These results can be explained by considering the operation of the Rendezvous layer: when a Rendezvous transmission is made, the node is kept powered on until it either synchronizes with the intended destination or the datagram times out. For a sparsely connected network the intended destination will not be within one radio hop of the source node and so it will remain powered on for the full timeout period before it decided that this datagram could not be transmitted. In the presence of the Routing layer however, the node will attempt to discover a route to the destination and if one is found it will have been stored

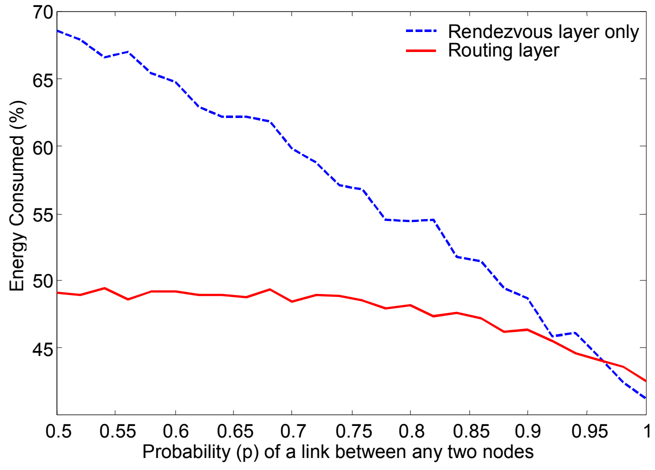

Fig. 6. Energy impact of the Routing layer

by the node and will be used for all datagrams to that destination. On average, the synchronization with a neighbour takes less than a beacon interval and so the node can transmit its datagram and power down. As the network becomes more reliably connected, the power saved by using the Routing layer is reduced, and for a fully connected network the Rendezvous layer consumes less power than the Routing layer. This is because, in this case, all nodes are able to communicate with any node in the network through just one radio hop and a routing function is unnecessary. The routing function uses energy to discover routes and this is unnecessary since all nodes can communicate directly. However, this overhead is relatively small because the Routing layer uses Rendezvous layer beacon, plea and offer SDUs to transport its routing information, which require no explicit and expensive synchronization.

The Routing layer service achieves low power operation by accessing details of lower layers that might otherwise have been unavailable. The first is the use of Rendezvous layer protocol primitives (Beacon, Plea and Offer) to exchange routing information. The second involves the use of information gathered at the MAC layer to make the routing protocol more robust. In summary the Routing layer provides power efficient relay service.

VI. Transport layer

Higher up the stack, the Transport protocol provides a data integrity service that enables reliable end-to-end transmission of application-specific Transport Service Data Units (SDUs) over the inherently unreliable radio channel.

The Transport protocol is designed to operate on a homogeneous network where the endpoints are PEN or PEN-like devices, and are addressed using a combination of MAC address and Transport Port number. To reduce energy consumption the protocol adopts a two, rather than three, phase approach. It uses a two-way handshake to set up a temporary connection between the transmitter or sending end of the connection, and the recipient or receiving end. Setting up the connection involves the transmitter sending a single packet connection request to the recipient specifying the desired Transport Port. The recipient will respond with a confirmation indicating that it is ready to receive data on that port. The data transfer phase can then proceed, with the protocol performing the required segmentation and reassembly functions. In particular, the transport service data unit is segmented into a number of transport datagram PDUs, which are then transmitted block by block. The Maximum Transfer Unit (MTU) size is fixed; since heterogeneous networks are not involved, there is no need to consider refragmentation of PDUs as they go from a subnetwork with a greater MTU size to one with a smaller one, i.e. the MTU size does not need to be negotiated. At the receiving end, transport PDUs are reassembled into a transport SDU that is then passed to the receiving application.

Since the radio channel is prone to errors, and PDUs can be lost or duplicated, it is important that the Transport protocol ensures that the SDU is received intact at the recipient end. To achieve this, each transport PDU is assigned a unique sequence number; the sequence number is incremented after the transmission of the PDU. At the destination, the sequence numbers are used to identify and suppress duplicates. The recipient also maintains a bitmap of the sequence numbers of PDUs it has received so far; it periodically sends the transmitter an acknowledgement containing the sequence number of the first PDU being acknowledged along with a delivery bitmap indicating the receipt status of the next *N* PDUs where *N* is the size of the bitmap. When the transmitter receives an acknowledgement, it examines the bitmap, discards any undiscarded PDU that was successfully received, and retransmits any unsuccessful ones. The Transport protocol ensures that an SDU is delivered only once to its destination by associating a unique connection identifier with the temporary connection created when an SDU is first transmitted; when a recipient receives a subsequent SDU with the same connection identifier, it assumes that it is a duplicate and discards it.

The acknowledgement need not be sent in response to every PDU, and can be relatively infrequent (helping to reduce power use),

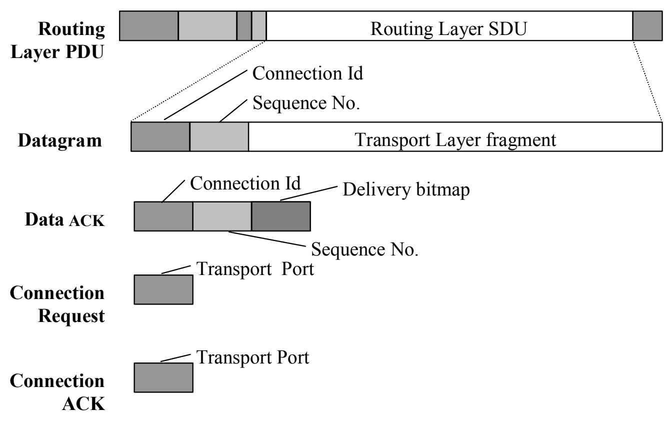

Fig. 7. Transport layer PDU format

but a transmitter is not allowed to have more than a certain number (window size) of outstanding PDUs at a time. At the start of the data transfer, the allowed window size is initialised to a fixed value. Each time the transmitter transmits a PDU, the window size is decremented by one. Once the window size becomes zero, the transmitter is not allowed to transmit any more Transport PDUs until it receives the necessary information from the recipient that it can increase its window again. Such a mechanism provides a basic form of flow control and liveness control indication on a per connection basis.

A transmitter with a pending data transaction has a single outstanding timer associated with it. If the transmitter has just started a transmission and a request for connection has been sent to the intended recipient, the timer is set to the expected response time of the recipient. In case of a timeout, the request is retransmitted and a new timer is set; there are a fixed number of retransmission attempts before the connection request is aborted. Once the connection is set up and data transfer is in progress, a timer is set for every block of PDUs that is sent out. Expiry of the timer indicates that the recipient has not sent back an acknowledgement PDU with a delivery bitmap. When this happens, the transmitter sends out an explicit request for an acknowledgement, setting a new timer for the acknowledgement to come back. If the transmitter fails to receive an acknowledgement even after sending out a number of requests for ACKs, it aborts the connection. If an acknowledgement is received that indicates no new receptions, the transmitter retransmits all the PDUs that have been sent, but have not yet been acknowledged; there are a maximum number of such retransmission attempts before the connection is aborted.

Similarly a recipient with a pending data transfer has one outstanding timer. The timer is set as soon as the recipient has sent back a response to a connection request from the transmitter. Subsequently, the timer is reset each time a new Transport PDU is received. If the timer expires, meaning that a PDU has not been received for some time, the recipient sends back an ACK bitmap to the transmitter indicating that it is still expecting more PDUs. The ACK bitmap specifies the last block of PDUs the recipient has received, so the transmitter knows which PDUs need to be sent again.

As soon as a recipient receives the last PDU for a connection, it sends back an ACK to the transmitter and closes its end of the connection. On receiving that last ACK, the transmitter terminates its side of the connection. There is no explicit 'closing' of a Transport connection, thus avoiding the use of extra protocol primitives.

The Transport protocol integrates with the low power underlying protocols and uses their power saving features. Using the layer below, the Transport service is advertised by a particular application protocol identifier (APID), and an associated beacon interval. Typically the Transport beacon interval is chosen to be large so that beacons are not transmitted too frequently; this ensures that the channel is not hogged by Transport beacons and reduces power consumption. When a transmitter opens a Transport connection on a given Transport Port, it sends out a connection request on the Transport APID. A recipient that is listening on that Transport Port accepts the incoming connection request and uses the Transport APID to send back a reply. For the data transfer phase, the Transport protocol opens a transient APID with a smaller beacon interval. Using that APID, Transport PDUs are then transmitted as quickly as possible, minimising the latency of the connection. Once all the PDUs have been sent, the transient APID is closed, and the Transport protocol reverts back to its normal beaconing mode.

In summary the Transport layer:

- provides a connectionless service that has, in effect, no maximum SDU size
- provides guaranteed exactly once delivery
- uses lazy acknowledgements and windowing to reduce the power used by ACKs
- avoids explicit three-phase connection operation in order to eliminate extra protocol primitives

VII. Related Work

The design of low power architectures and protocols has been the subject of a lot of recent research work. Stemm et al. [10] have observed that a significant fraction of a wireless node's power is consumed by its wireless interface cards, and have investigated methods of reducing the power consumption of network interfaces, with emphasis on hand-held devices.

Various energy efficient protocols have been proposed at the MAC level. In [4], an energy efficient method for broadcasting data through indexing is proposed where a base station computes a transmission schedule and broadcasts it to mobiles. The mobiles are then able to stay in standby mode and conserve energy until it is their time to transmit. Various reservation and scheduling algorithms have also been proposed for wireless ATM networks. Scheduled access is generally good for low power consumption since the number of collisions are reduced and key components of the wireless interface hardware can be powered down based on the knowledge of the location of surrounding nodes and their transmission schedules. In their comparison of various MAC protocols [9], Sivalingam et al. conclude that contention usually results in high energy consumption while reservation and scheduling methods perform better in a low-power environment.

Mangione [7] proposes a power-efficient MAC protocol for use within one cell for communication between a base station and its mobiles. In this protocol, the base station periodically sends a beacon followed by a mini-slot containing the ID of nodes that have a page waiting for them. Such nodes stay awake to receive their messages while the other nodes power themselves off.

Zorzi [13] proposes transmission channel probing to overcome the problem of unnecessary MAC retransmissions when the radio channel quality is impaired. The idea is to send a short low-power probe packet to check the channel quality; if the probe is successfully sent, indicating that the channel quality is good, MAC retransmissions are scheduled. As a result, energy is not wasted for retransmission when the channel is impaired although energy is used to transmit the probes. At the software level, data compression techniques are considered useful for conserving power as they reduce the required transmission time. This is however at the expense of the additional CPU cycles for performing the compression which on smaller, embedded systems can itself make for significant power drain.

An application-level control approach is proposed in [6]. In this solution, the mobile host decides when to suspend and resume the communication device. Hence the mobile host acts as the master while the base station is the slave. Whenever the mobile host is asleep, the base station has to buffer any data sent to the mobile host. When the mobile host wakes up, it queries the base station to receive its data. Since the base station has no way of restarting communication if it has run out of buffer space, it is up to the mobile host to understand the base station's communication patterns so that there is no buffer overflow at the base station. Essentially the suspend/resume cycles have to be matched to the communication patterns between the mobile host and the base station.

IEEE 802.11 [3] defines wireless MAC and PHY layers intended for a range of applications, including mobile stations. The PHY layer specifies a frequency-hopping spread spectrum system at 2.4GHz. This provides a considerably higher data rate and superior resilience to interference than that offered by the existing PEN Physical layer, but is also inherently more power hungry. The MAC specification defines behaviour for an IBSS[3] formed in an ad hoc fashion, with stations collaborating to define a framing structure to which all stations in the IBSS synchronize. This scheme treats all stations as equals, so there is no designated master station. This ensures that no station suffers the burden of managing the entire group, but also requires that all stations exhibit the same behaviour and level of activity regardless of their intended application. Also, only stations within an IBSS can communicate and the 802.11 procedures for association and authentication are very unsuited to supporting the PEN model of high mobility and short transactions.

Bluetooth [2] was initially conceived as a cable replacement architecture, with a range of 10m and sufficient bandwidth to carry a real-time voice channel. It has since evolved from that into an architecture for creating personal ad hoc networks, and has had a higher power mode added giving a range of around 100m. Originally envisaged as having a unit cost of below $5 and ‘low’ power usage, products have been delayed in reaching the marketplace due to difficulties in achieving these requirements.

Other emerging wireless communication standards have also tackled the power saving issue. The HiperLAN is the European Telecommunications (ETSI) RES 10, with an operational frequency band of 5.2 GHz. Since HiperLAN offers a higher bit rate (over 20 Mb/s) compared to 2 Mb/s for IEEE 802.11, receiving data is more expensive as it requires equalisation (an equaliser being the most expensive part of the receiver in terms of power consumption). The solution is to divide the packet into a low-bit-rate part for control information and a high-bit rate part for data. The MAC evaluates the control information and, if the packet is addressed to the node, the equaliser is turned on and the packet is received at the higher data rate. This ensures that no energy is wasted on receiving packets that are not destined for a node. In HiperLAN, there is also the notion of high-power supporter nodes (p-supporter) and nodes that aim to save power (p-saver). Basically there is a contract between a p-supporter and a p-saver. The p-saver is only active during prearranged intervals, and the p-supporter has to queue all packets destined for a p-saver and schedule their transmission during the active intervals of the p-saver.

From this survey of existing power efficient protocols, it can be seen that most existing protocols assume some kind of infrastructure network where there is a base-station or an elected node that co-ordinates the action of the other nodes and helps them to control their power consumption. In the embedded environment considered here, it cannot be assumed that such a node would be available; the PEN algorithms are therefore asynchronous in nature, each node decides by itself when to power on or off. This approach is more flexible since each node can control its own power usage independently and can operate in a truly ad hoc fashion.

VIII. Conclusions

In this paper an energy efficient MAC protocol was first proposed that saves power by minimising the likelihood of collision and reducing the number of packet retransmissions. Above that, a Rendezvous protocol was designed that enables PEN nodes to remain asleep for much of their duty cycle and switch on their transceiver only when it is required. An optional Routing protocol uses routing information opportunistically gleaned during lower layer protocol exchanges and provides a power saving even when used in a relatively well-connected network. A Transport protocol has been built on top of these protocols and provides energy efficient transfer of arbitrarily sized SDUs on an end-to-end basis.

No protocol described is without some mechanism incorporated to reduce power consumption, and there is a strong requirement for applications to be power-aware. This underlines one of the main conclusions of the project – that low power must be considered throughout an embedded system’s design. To summarize the experience gained in the construction of the PEN communications infrastructure the following, although sometimes obvious, were each found to have a role in reducing power consumption.

Turn everything off: The majority of power saving simply comes from the availability of hardware that allows the selective powering down of different system elements and the use of software that monitors the use of each hardware element so as to power down the ones that are not required.

Use simple hardware: There are a number of requirements that protocols can make of hardware, such as timer precision, complex modulation techniques, frequency hopping etc. that, when omitted, permit lower power solutions.

Eliminate radio reception requirements: It is sometimes assumed that transmission is more expensive in terms of power than reception. In the area of low power, low range radio, reception is likely to be at least as expensive as transmission (and is likely to have greater expense in hardware more advanced than that used in the PEN project). Identifying reception as an expensive activity can have a significant impact on protocol and system design.

Schedule Rendezvous times: Protocols that establish periods of availability enable periods of radio dumbness and radio silence to be implemented relatively low down in a protocol stack. In lightly loaded nodes such periods can allow power to be removed from the rest of the system.

Consider re-using events from underlying protocol layers: Although it may break the ideals of ‘information hiding’ it seems that making information, that might otherwise be discarded, available to higher protocol layers can often save power. In terms of the design of a protocol stack this may involve the explicit promotion of details, initially thought to be specific to a protocol, to aspects of their layer service.

Examples from the PEN protocol stack include: the use of the end-of-reception event (no matter for whom the reception was intended) as a ‘free’ synchronization hint; the accumulation of neighbourhood information available from redundant receptions in the MAC protocol (which is used in the Routing protocol to provide a liveness hint); and the use of the protocol primitives used to exchange beacons in the Rendezvous protocol to exchange routing information.

Provide a relaying service: In a radio environment with no global synchronization nodes have to expend a significant amount of energy in synchronizing with communication partners. Relaying through ubiquitous neighbours decreases the amount of time, and thus energy, necessary before communication can occur. If the overhead in operating the relaying protocol is small our measurements show that the overall power consumption in the network (as opposed to per-node power consumption) is also lower.

Provide and use cheap low-quality options: If it is simple to provide ‘low priority’ communication primitives more cheaply (in terms of their power use) than ‘normal’ primitives, consider providing them. Some protocol elements may require only a ‘least effort’ delivery service: so as to carry a hint to peers that can be safely ignored (by the recipients) with a small performance penalty, for example.

[3] Independent Basic Service Set

Acknowledge streamed data infrequently: When protocol data units are expected back-to-back, generating an acknowledgement for each of them uses power unnecessarily. A windowing protocol involving 'lazy' acknowledgements will save power.

Avoid contention: When competing for unique access to a shared channel, determining the level of contention for the channel and delaying transmission for as long as that level is high can save transmission power. The alternative mechanism of relying only on acknowledgements and retransmission wastes power on discarded transmissions although it may provide lower latency.

Use channel availability hints: The 'hidden terminal' problem [11] means that no radio event can be guaranteed to be shared by all those sharing the communications medium so protocols must be designed without an absolute requirement for synchronization via radio signals. However, the probability that most of its users do observe the same events can be used in protocols to provide hints of channel availability.

Keep protocols simple: Protocols are commonly connectionless (single phase: data transfer) or 'soft state' (two phase: including explicit connection establishment) or connection orientated (three phase: including explicit connection release). Additional protocol elements supporting more phases, particularly those involved only in maintaining the shared knowledge that a connection is in place, represent additional power use: not only in their transmission and reception; but also, because their regular occurrence may prevent a node from powering down.

Modular implementation: System size, when it becomes large, may dictate the use of additional power-consuming hardware resources (such as external memory). Software extent and complexity may also require a greater level of activity in the hardware base (for example, coming out of low-power modes more frequently, or accessing relatively 'power-hungry' memory). It is important, therefore, that the software incorporated into an embedded application can be tailored to match the application's requirements closely (such as removing unnecessary protocols or operating system services). This can most conveniently be achieved by constructing the system from largely independent software modules that can be assembled into any particular build as required.

Despite the current limitations of our hardware, our empirical measurements show that these principles have enabled us to extend the battery life of PEN nodes significantly.

Further work is required to evaluate the effect that purpose built, more integrated, hardware would have on power consumption both through a higher quality radio transceiver and through more efficient components. For example, higher bandwidth or faster processors – although they might require additional power – may enable a node's workload to be dispatched more quickly and thus enable it to power down sooner. Also the current Rendezvous layer does not take advantage of the predictable nature of other nodes' availability when scheduling transmissions – modified protocol implementations that do are currently under investigation.

IX. References

[1] Bennett F, Clarke D, Evans J B, Hopper A, Jones A, Leask D, "Piconet – Embedded Mobile Networking", *IEEE Personal Communications*, vol. 4, no. 5, October 1997, pp. 8-15.

[2] *Bluetooth Specification Volume 1*, Bluetooth Consortium, July 1999

[3] Institute of Electrical and Electronics Engineers Std. 802 and International Organization for Standardization, Std. ISO/IEC 8802 "Information Technology – Telecommunications and Information Exchange – Local and Metropolitan Area Networks – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

[4] Imilienski T, Vishwanathan S, Badrinath, "Energy efficient indexing on air", *Proceedings SIGMOD*, May 1994.

[5] International Organization for Standardization, ISO 7498-1, "Information processing systems – Open Systems Interconnection – Basic Reference Model", October 1984

[6] Kravets R, Krishnan P, "Application-Driven Power Management for Mobile Communication", *Proceedings of 4th International Conference on Mobile Computing and Networking MOBICOM98*.

[7] Mangione-Smith W, Chang P S, "A Low Power Medium Access Control Protocol for Multimedia Systems", *Proceedings of 3rd International Workshop on Mobile Multimedia Communications*, September 25-27, 1996.

[8] Schlatter Ellis C, "The Case for Higher-Level Power Management", *Hot Topics in Operating Systems VII*, 1998, Institute of Electrical and Electronics Engineers.

[9] Sivalingam K M, Srivastava M B, Agrawal P, "Low Power Link and Access Protocols for Wireless Multimedia Networks", *Proceedings of IEEE Vehicular Technology Conference VTC97*, May 4-7 1997.

[10] Stemm M, Gauthier P, Harada D, "Reducing Power Consumption of Network Interfaces in Hand-held Devices", *Proceedings of 3rd International Workshop on Mobile Multimedia Communications*, September 25-27, 1996.

[11] Tobagi F A, Kleinrock L, "Packet switching in radio channels: Part II – the hidden terminal problem in carrier sense multiple-access modes and busy-tone solutions", *IEEE Transaction on Communications*, vol. 23, no. 12, pp. 1417-1433, 1975

[12] Todd T D, Bennett F and Jones A, "Low Power Rendezvous in Embedded Wireless Networks", Proceedings of 1st International Workshop on Mobile Ad Hoc Networking & Computing, MobiHOC, August 11, 2000

[13] Zorzi M, Rao R R, "Energy Management in Wireless Communications", *Proceedings of 6th WINLAB workshop*, New Brunswick, NJ, March 1997.