Power-Aware RF Communication with MICA Motes

Micah Colon **Sunil Mathew**

FINAL PROJECT REPORT

Background and Initial Work:

Sensor networks have rapidly become an efficient and convenient means of collecting data from a variety of sources for a variety of purposes. Due to the isolated nature of most sensors and their reliability on a limited power source, maximizing the lifetime of the power source is a major design element in sensor networks. Research and experience have shown that RF communication is one of the greatest draws of power in these systems and much work has gone into minimizing such communication, resulting in a number of different schemes and methods to conserve power.

Initial work on the project involved familiarization with TinyOS, nesC (the language of TinyOS), and the MICA motes themselves. Some efforts were devoted to monitoring and displaying the remaining battery power, with interest in finding ways of conserving power while continuing to execute. However, it soon became apparent that there were limitations in the hardware of the MICA motes. Power saving became primarily a fully on (no power savings) state or a snooze (max power savings) state. Our attention quickly moved to efficient communication schemes that could make use of these limitations.

TDMA-based power schemes for sensor networks are not new, but implementations of them tend to be an integral part of applications, adding to the applications complexity and development time. Our objective was to remove the need to constantly re-implement such schemes by providing a platform which handles all the details of communication and a significant portion of the power savings.

Overview of scheme:

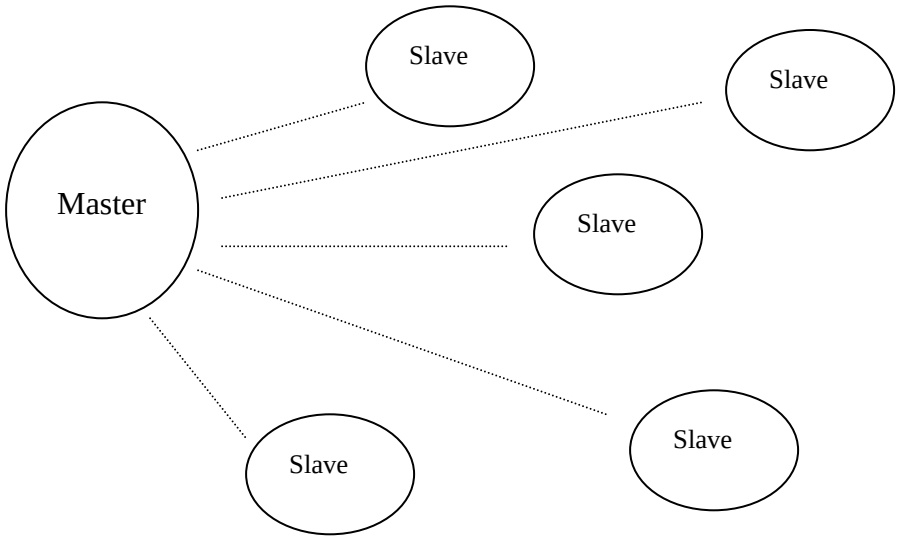
Our scheme consists of a master node which periodically receives sensed information from a number of slave nodes.

The general concept involves dividing time into cycles. Each cycle is further divided into slots. In each cycle, each slave node is allocated one fixed slot during which it transmits

data to the master (via RF). The slave also listens for periodic messages from the master to make sure that it is still in range of the master as well as for time synchronization information. During periods of time when a slave node is not transmitting, or listening for the heartbeat from the master, it is put into a low-power sleep mode. The slave node wakes up again when it needs to transmit data or listen for the heartbeat.

Apart from this, the slave node also saves power by finding the minimum transmission power needed to communicate with the master. The slave node begins with the lowest transmission power and increases it in steps until it is able to hear messages from the master and communicate with it in return.

The master and slave use a series of messages to communicate initially and negotiate the allotment of a particular slot to the slave for further communication. The ultimate objective therefore of the scheme is to ensure that every slave can communicate without being interrupted by anyone else and conserver power whenever possible.



High level view of the network

Details of the scheme

The middleware consists of two modules:

1. The Master Module
2. The Power-aware Slave Module

Master Module:

The details of the Master Module are explained below by describing the states of the master node followed by an algorithm of its operation.

States

INIT	The master node starts its clock and initializes itself to start interaction.
SEND_BCAST	The master node sends a heartbeat broadcast to all nodes with a timestamp and advertising the next available time slot for the slaves.
RECV_WAIT	The master node is waiting for data or requests to be sent by any slave nodes.
RECV_REQUEST	The master node has received a request for a time slot and is deciding whether to grant it.
SEND_ACK	The master node is sending the acknowledgement to a slave telling it that it can use that slot for its data transmission.
RECV_DATA	The master node is receiving data from the slaves.

Algorithm

Step 1: The master node comes on and initializes itself, including its clock. It is in INIT state.

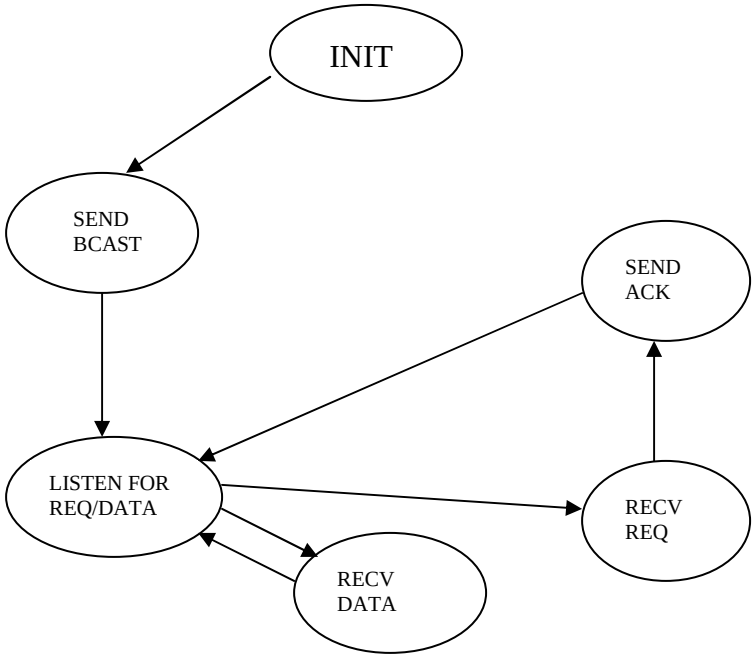
Step 2: The master waits for its slot 0 (first slot of the cycle) and transmits a broadcast message containing the next available slot for an incoming slave to transmit data.

Step 3: The master may receive a request for a slot from a slave during the particular slot.

Step 4: The master sends back an acknowledgement for the slot telling the slave it may utilize the slot to transmit its data.

Step 5: The master listens for data from the slave.

Step 6: If the master receives data, it checks whether it is the right slave talking to it in that slot. It accepts and processes the data if it is the right slot and rejects it if it is not.



State transition diagram of the master

Power-aware Slave Module:

The details of the Slave Module are explained below by describing the states of the slave node followed by an algorithm of its operation.

States

INIT	The slave node has just been turned on and is listening for messages from the master.
RECVD_BCAST	The slave node has received a broadcast message from the master.
SENT_REQUEST	The slave node has sent a request to the master for the slot which it heard advertised.
RECVD_ACK	The slave node has received a message from the master indicating which slot it is to transmit.
SEND_DATA	The slave node is transmitting during the slot which it was allotted to by the master.
LISTEN_FOR_BCAST	The slave node has been previously allocated a slot and is presently listening for a heartbeat broadcast from the master.
SLEEP	The slave node is in the low power (lowest transmission power) state. During SLEEP, the RF power is set to its lowest value.

Algorithm

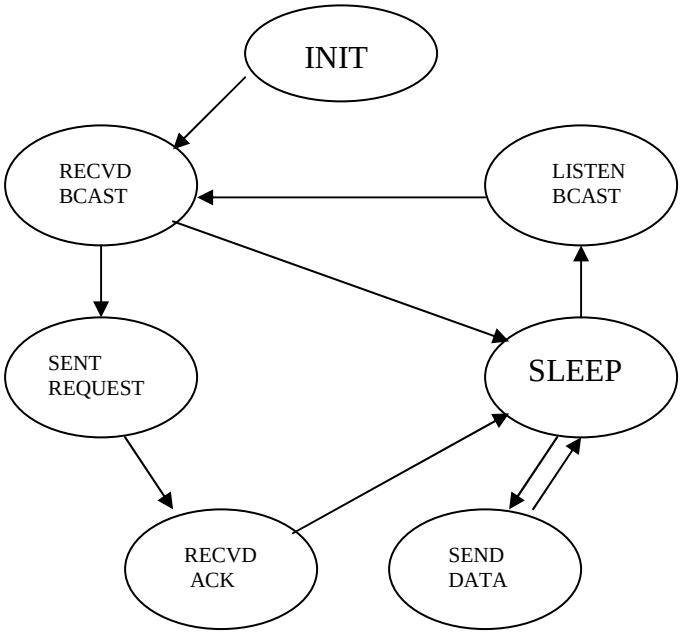
Step 1: Slave node is turned on and is in INIT state. It periodically increases its transmission power till it hears the heartbeat message from the master. It then transitions to WAITING_FOR_SLOT state.

Step 2: The slave node hears a heartbeat message from the master which always advertises an available slot. It then sends a request directed to the master asking for the slot which it heard was advertised. If, following this, the slave hears another broadcast message with an advertised slot, it sends another request to the master with this slot

number. If it is the same advertised slot, it means that the master did not hear the request. If the slot number is different, it means that the slave assumed that the master gave the slot to another slave.

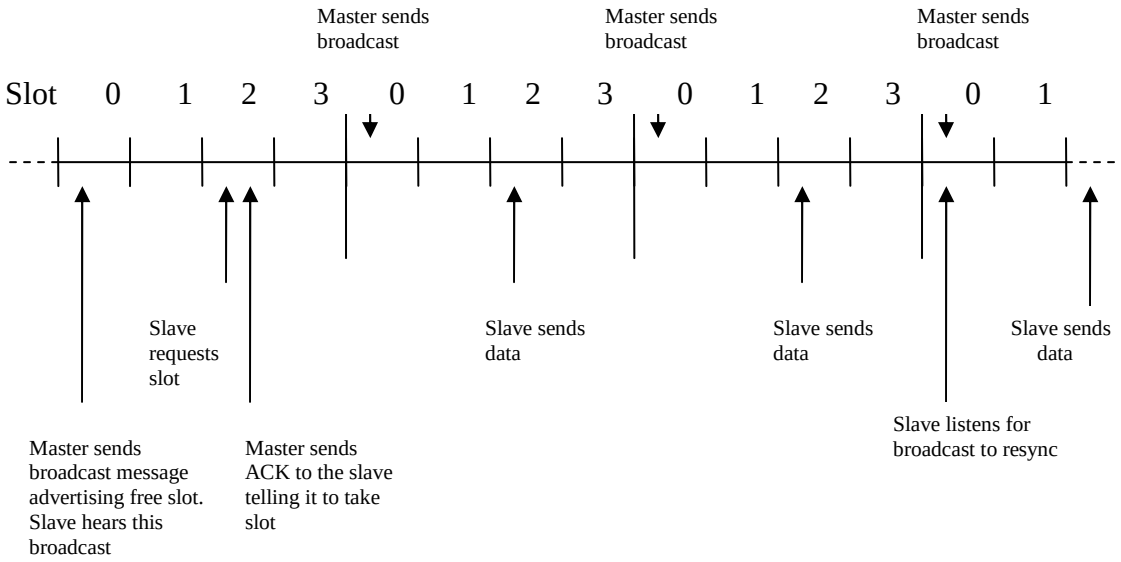
Step 3: The slave receives an acknowledgement from the master indicating the slot which it is to use. It then transitions to a SLEEP state where it sleeps till the next slot when it has to transmit.

Step 4: The slave wakes up and sends its data and goes to sleep till the next time when it has to either transmit data or listen to the heartbeat from the master. If the slave is to transmit data, it does so and goes to sleep again. If it is to receive the heartbeat from the master and the slave receives the heartbeat, it goes to sleep again till it has to either transmit data or listen for the next heartbeat. If the slave does not receive the heartbeat after a particular time, it assumes it is out of range and goes back to INIT and starts the algorithm again.



State transition diagram of the slave

Communications between Master and Slave

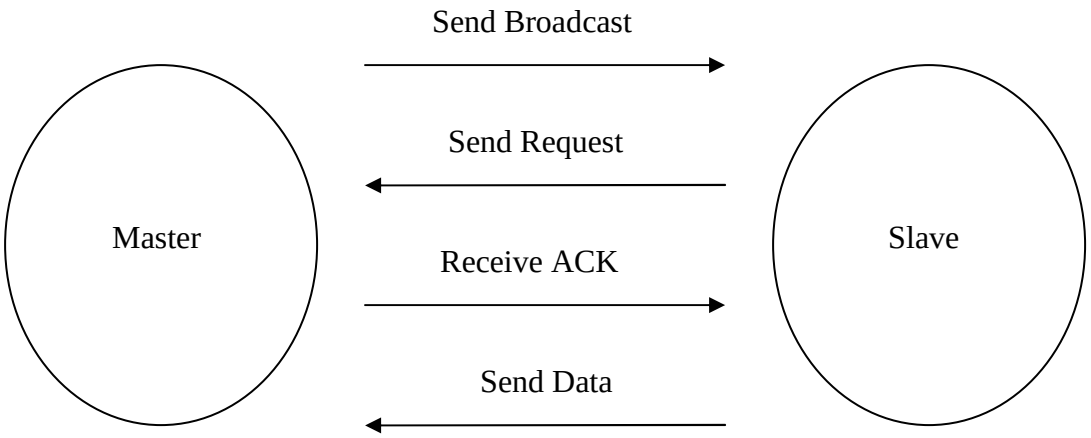


Operation Timeline

The communication between the master and the slave takes place as a 3-way handshake followed by data transmission in the proper time slots by the slave.

1. The master broadcasts heartbeat message containing the next available slot.
2. The slave replies to the broadcast requesting the advertised slot.
3. The master sends an ACK telling the slave to take the slot.
4. The slave starts transmitting data in that slot in the following cycle.

Apart from this, the slave also listens for the master's broadcast message every few cycles to make sure it is still within range of the master and uses this to resynchronize its clock.



Master – Slave Interaction

Power Savings

The power savings occur by adjusting the potentiometer (transmission power). When the slave is turned on, it sets its potentiometer to a low value. It then increases the power until it successfully negotiates and obtains a slot from the master for communication, storing this potentiometer value. The slave then proceeds to decrease its potentiometer value to the lowest possible value during time slots when it is not transmitting or listening for the heartbeat broadcast from the master. When it is time for the slave to transmit its data to the master or listen for the heartbeat, the slave raises the transmission power to the necessary level.

Software Developed

Middleware has been written for both the master and slave modules. Applications can be written on top of this middleware which serves the purpose of systematic, TDMA based communication with a base station (master).

The slave can be passed data by an application written on top of it and send it to the master. The master which receives data will, in turn, pass it on to an application which can choose to do as it wishes with the information. The dummy application provided for the master simply sends the data on through its serial port.

The middleware provides interfaces for applications to use. These are used on the slave to signal to the application that it is ready to accept data to send to the master. On the master, the interface is used to pass the data the master receives from the slaves to the application written on top of it.

Testing Results

Due to the lack of resources, specifically the number of MICA motes available to us, extensive testing of our work was not possible. However, the limited amount of testing we were able to perform revealed issues that would likely need to be resolved before our scheme could be utilized.

Tests revealed an un-expectedly large drop rate of data packets being received at the master node. Utilizing passive nodes that simply dumped received RF packets directly to the serial port it was clear that correct data packets were indeed being sent by the slave nodes – they were simply not being accepted or processed on the master node. Further investigation and analysis of the data pointed to issues relating to the timing, as some data packets were only received shortly after a time sync. Upon removing our block, requiring that only packets received during their correct time slot be processed, we began to receive almost all expected packets.

This apparently points to a lack of proper synchronization between the master and the slave nodes. Our very simple scheme did not account for transmission latency of timestamps, provide any inter-period time buffers, or provide any other good or reliable method and set of checks to ensure closer synchronization. Modifying our scheme to do so should be easily accomplished, once a proper design has been established.

Other Issues and Future Work

- MICA motes have a SNOOZE state where the mote is almost totally shut down. The snooze functionality uses the same timer as the general Timer component and so we had some time-sync problems when using this functionality. This could possibly be overcome with some modifications.
- The middleware developed could be ported to work with MICA2 motes with additional power savings as these motes have more power-saving functionality than the MICA motes.
- The protocol could be extended to become ad hoc capable with a peer-to-peer negotiation and time slot allocation rather than master-slave allocation.
- The present protocol works with a single master and multiple slaves. This could possibly be extended to a network which had multiple masters or a series of sub-masters communicating together. This would increase the maximum possible number of nodes in the network as data from multiple slaves could be accepted during a single time slot.

Project Work Allocation

Setup and installation of tools required – Micah and Sunil

Power analysis of MICA motes – Micah

Design of protocol – Micah and Sunil

Implementation of Master module – Micah

Implementation of Slave module – Sunil

Integration and testing of middleware – Micah and Sunil

Documentation – Micah and Sunil

Important Links

http://webs.cs.berkeley.edu/tos/

http://www4.ncsu.edu/~macolon/CSC714/