Radio Stack Iteration

*How to improve the CC1000*

Joe Polastre

January 15, 2004

NEST Retreat
Outline

- Mica2 TinyOS 1.1 MAC
- New Methods for Clear Channel Assessment
- Improvements to the MAC
- How to develop TinyOS core services across many groups
  *(without breaking everyone else’s applications)*
Mica2

MAC Delay

Switch to TX Mode

Preamble: 18
Sync: 2
Packet Transmission: 36

Switch to RX Mode

56 bytes

1-128 bytes

250μs

250μs
Mica2 Release CSMA

- Set a fixed noise floor threshold
- On transmission:
  - Take RSSI reading
  - If reading < floor, transmit on next byte boundary from radio

- With a few months in the lab, what happens?
RSSI on the Chipcon

signal strength inversely proportional to RSSI

- Noise Floor
- Strength of Packet
- Incoming Packet

Time (ms)

RSSI (Volts)

More Signal

Less Signal
Collisions Oh My!

Channel Clear

Raw signal strength (V)

Time (ms)
Clear Channel Estimation
Adjusted Mica2 CSMA Layer

- Adjust noise floor (AGC):
  - After each packet transmission, sample RSSI and add to 10 sample FIFO queue
  - Select median(Q) and add to EWMA

- On transmit:
  - Take <=5 RSSI samples
  - If one reading < floor, transmit on next byte boundary from radio

- On initialization: timer takes radio samples to fill the FIFO queue
Results

Each node sends as fast as possible

Delivered bandwidth with Different Payloads
Each node sends as fast as possible
AGC Performance

Graph showing signal strength (dBm) over time (ms). The graph compares RSSI Signal and AGC Noise Floor.
New Mica2 CSMA Layer

Let’s call it \textit{B-MAC} ... 

- How does it differ from AGC:
  - No initial backoff
  - Minimize preamble overhead:
    - Woo et al show that communication with nodes in the noise floor is futile
    - set preamble length
    - set radio settling
    - Now: radio hardware only detects nodes with strength greater than the noise floor
## Comparison

<table>
<thead>
<tr>
<th>Mote Granularity</th>
<th>Mica bit</th>
<th>Mica2 Rel byte</th>
<th>AGC byte</th>
<th>B-MAC byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Init MAC Backoff</td>
<td>16 (5.3) 3.2ms</td>
<td>36+128 68.3ms</td>
<td>64 26.6ms</td>
<td>0 0ms</td>
</tr>
<tr>
<td>Max Congest Backoff</td>
<td>16 (5.3) 3.2ms</td>
<td>29*16 193ms</td>
<td>16 6.6ms</td>
<td>16 6.6ms</td>
</tr>
<tr>
<td>Noise Floor</td>
<td>AGC (Hardware)</td>
<td>Fixed</td>
<td>AGC (Software)</td>
<td>AGC (Software)</td>
</tr>
</tbody>
</table>
Results

Each node sends 10 packets/sec
Results

Each node sends 10 packets/sec
MAC Interfaces

- Application controlled parameters!
  - For example: initial CSMA backoff:
    - No backoff for bulk data transfer
    - Random backoff for broadcasts
    - 2 packet backoff for multihop routing (tree)
- Expose MAC parameters enabling cross layer control
Other Improvements

- Link Layer ACKs
  - Small ACK packet immediately following data packet
  - Requested by many services including multihop routing

- Low Power Listening
  - Periodically wake up, sample the channel for activity, and return to sleep
  - Reduced the time to sample the channel
What is the process for getting this into TinyOS?

- “beta” structure
- Defined project with a lifetime
- Expected that the beta project will become part of the main branch
- Enables multiple groups to work on the same evolving code without breaking the main branch
  - eg: radio stack was work between UCLA, USC, UCB, IRB, and JHill
- Moved to the main branch one month before its release
Conclusion

- Systematic design of MAC layer
  - Provides application cross layer control
  - Different services require different MAC parameters
  - Evolving functionality and improvements
    - AGC, Low Power Listening, ACKs, Interfaces
- Beta system now in place for collaborative work on new TinyOS services
webs.cs.berkeley.edu
TinyOS 1.1.3 Release