Approaches to Communication Scheduling

Barbara Hohlt and Eric Brewer
Terence Tong, David Molnar, and Alec Woo
Umesh Shankar and Naveen Sastry
Introduction

- Problem: motes exhaust battery quickly
- Solution: turn radio off to extend mote lifetime
- Technique: distributed explicit scheduling
- Three different approaches from UCB NEST
- Rest of talk
  - Brief review of other techniques
    - Energy-aware MAC, topology management, low-power listening, centralized scheduling, more…
  - Introduce each UCB NEST approach
    - Flexible Power Scheduling (Hohlt and Brewer)
    - STEAM (Shankar and Sastry)
    - DuraNet (Tong, Molnar, and Woo)
  - Outline design space & investigate design choices
- See our posters!
Low-Power Listening

- Implemented in TinyOS 1.1 radio stack
- Deployed at Great Duck Island
- Node sleeps except to probe for message
  - Amount of sleep given by duty cycle
- Sending requires long “preamble”
- Pro: simple, in mica2, big win over CSMA
- Con: Requires long preamble, still has useless overhearing
Centralized Scheduling

• Central point coordinates all nodes
  – Gathers network conditions
  – Pushes out sleep/wake schedules to nodes
• Examples: Dust, Inc., PEDAMACS
• Pro: throw lots of computation at the problem, switch algorithms easily
• Con: Hard to probe network, conditions may change, overhead to push schedules
Flexible Power Scheduling (FPS)

- Barbara Hohlt and Eric Brewer
- Coarse-grained slotted system
- Flows are scheduled: parent nodes advertise (supply bandwidth)
- Child makes reservation with parent for some slot, sends during that slot each period (demands bandwidth)
- Each node makes demands for all flows it forwards => end-to-end reservation
FPS (cont’d)

• Interference handled by underlying MAC
• Supply and demand may be increased/decreased as needed by making or dropping reservation
STEAM

• Umesh Shankar and Naveen Sastry
• Channel = (Rendezvous Time in each period, Frequency)
• Every node picks random channel on which it listens
  – If it hears message, keep listening
• No reservation between parent & children
• Exchange state, i.e., channel selections, using special announcement channel
One neighborhood in STEAM:

- Announcement Channel

Message for B, no conflict

Message for B causes conflict, causes useless overhearing for C and D
STEAM (cont’d)

• Switch channels when useless overhearing exceeds expected amount
• Switch parents using EWMA link estimator
  – Interference detected using Carrier Sense
• Exchange state only when network in flux
• Nodes switch parents without any extra messages, since no parent-child agreement
DuraNet

• Terence Tong, David Molnar, Alec Woo
• Optimize for static networks, constant known traffic & interference patterns
• Separate schedule formation phase
  – Each link schedules separately as needed
  – Use RTS/CTS to avoid one-hop hidden node
  – Resulting schedule is (almost) conflict-free
  – Avoid queue overflows
• Application traffic delayed to fit schedule
Design Issues

- Schedule Formation
- Workload
- Rendezvous Times
- Time Sync Requirements
- Interference
- Adaptation Mechanism
Schedule Formation

- **FPS**
  - Parents advertise bandwidth (as receive slots)
  - Children listen for advertisements, request as needed

- **STEAM**
  - Probabilistically listen/send on broadcast channel
  - Nodes still find parents because of birthday paradox

- **DuraNet**
  - Separate schedule formation phase
    - Each pair of nodes uses RTS/CTS
    - Schedules static and re-used
Workload

• Primary focus: tree-based data collection or monitoring

• Variable vs. Fixed traffic
  – FPS: Traffic increase handled by queueing at cost of increased latency
  – STEAM: Traffic increase limited only by bandwidth; full utilization may require time for adaptation
  – DuraNet: Can decrease traffic only
Latency

• One-hop latency = period length
  – Comm periods must be <= application period
• FPS: depends on queue length (burstiness); periods ~ 2-3 sec.
• STEAM: depends less on queue length; periods ~ 100ms
• DuraNet: depends on schedule formation
  – Assumes constant traffic flow
Rendezvous Time

• Discrete/continuous
  – DuraNet “continuous”
    • Each parent-child link has own time
  – FPS divides time into discrete slots
  – STEAM has discrete rendezvous times, but spacing is less than one message time

• Dedicated vs. Shared
  – FPS: (Parent, child) reservations are unique w.r.t. each member, but other nearby pairs may choose the same slot
  – STEAM: Parents advertise a listening time without any reservation with children; children use feedback to avoid conflict
  – DuraNet: (Parent, child) reservations are unique within interference range
One node in FPS:

<table>
<thead>
<tr>
<th></th>
<th>Idle</th>
<th>Listen to A</th>
<th>Idle</th>
<th>Send to C</th>
<th>Idle</th>
<th>Listen to D</th>
</tr>
</thead>
</table>

One neighborhood in STEAM:

- A
- B
- C
- D

One neighborhood in DuraNet:

- A → B
- C → D
Time Sync Requirements

• All three require only local sync
• Rough granularity required for correctness
  – FPS: 25 ms
  – STEAM: 2-3 ms
  – DuraNet: 50 ms
• In all cases, tighter time synchronization can yield more message packing, better energy savings
Interference

- Interference: Node A is too far to send real messages to Node B, but can still interfere with B’s communication
- Interference *changes over time*
- Responses
  - FPS: rely on underlying CSMA MAC
  - STEAM: incorporated in parent link and channel-switching estimators
  - DuraNet: rebuild schedule
GDI connectivity
Adaptation Mechanisms

• FPS
  – Parents advertise available bandwidth; children request it
  – Workload changes handled by buffering at the source

• STEAM
  – Child switches parents based on parent link estimator
  – Parent switches channels based on observed vs. predicted useless overhearing
  – Parent advertises more/fewer listen times depending on usage

• DuraNet
  – Detect loss via end-to-end ACK
  – Detect node join by base station
  – Recover/join by schedule reformation
See Our Posters!