

# CMSC 724: Query Processing in Wireless Sensor Networks

Amol Deshpande

University of Maryland, College Park

April 12, 2007

# Wireless Sensor Networks

- A collections of devices that can:
  - sense,
  - actuate, and
  - communicate over a wireless network
- Increasingly being deployed everywhere
  - Great Duck Island (one of the first deployments)
  - Redwood forests, precision agriculture, fabrication monitoring
- Lot of research in last 5 years
  - In networking, systems, languages, databases, modeling etc. . .

# Wireless Sensor Networks: A Brief History

- Sensors used for a long time (especially in industrial monitoring)
- Recent CS History:
  - (1998) Pottie + Kaiser: Radio based networks of sensors
  - (1998) Pister et al: Smar Dust
    - Initial focus on optical communication
    - By 1999, radio based networks, COTS Dust, "Motes"
  - (1999) Estrin + Govindan
    - Ad-hoc networks of sensors
  - (2000) Culler/Hill et al.: TinyOS + Motes
  - (2002) Hill/Dust: SPEC, mm<sup>3</sup> scale computing ??
- Many companies providing hardware, support now
  - Crossbow, Moteiv, Arch Rock etc. . .

# Motes vs. Traditional Computing

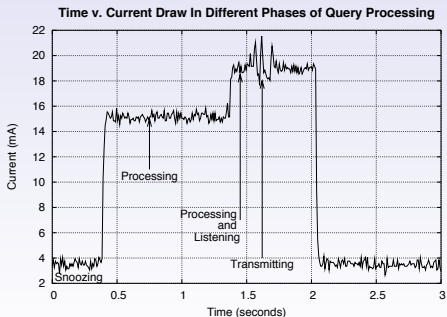
- Lossy, Adhoc Radio Communication
  - Low Bandwidth Shared Radio Channel
    - 40 kBits on motes
    - Much less in practice
  - Very lossy: 30% base loss rate
  - Somewhat different communication patterns (multi-hop, broadcast etc)
- Sensing Hardware
  - “Acquisitional”: can control when/what to sample
- Severe Power Constraints
  - Lasts only days to weeks depending on usage

# Data acquisition in Sensor Networks

- TinyOS-based
  - Write custom programs to acquire data
  - HARD !! Need to program around/for:
    - Limited power budget; Lossy communication
    - Need for distributed algorithms, limited development tools
- TinyDB
  - Provides a declarative querying interface to acquire data
  - Abstracts away much of the complexity

# Wireless Sensors: Energy Consumption

- Primary focus of much research in this area
- Reasoning about power consumption hard - many simplifications typically made
- Hardware dependent: e.g. for motes, “receiving” can be more expensive than “transmitting”



**Figure 2: Phases of Power Consumption In TinyDB**

# TinyDB: Acquisitional Language

- Allows specifying when and what to acquire
- Support for “sliding windows” (“storage points”)
- Supports aggregation (pushed down inside the network)
- Event-based queries: polling not very efficient
- Lifetime-based queries ?: Users specify the lifetime, the query processors decides the frequency

```
SELECT COUNT(*)
  FROM sensors AS s, recentLight AS rl
  WHERE rl.nodeid = s.nodeid
  AND s.light < rl.light
  SAMPLE INTERVAL 10s

ON EVENT bird-detect(loc):
  SELECT AVG(light), AVG(temp), event.loc
  FROM sensors AS s
  WHERE dist(s.loc, event.loc) < 10m
  SAMPLE INTERVAL 2 s FOR 30 s
```

# TinyDB: Optimization

- Maintain cost information about the sensors
- Use *rank ordering* type algorithms to optimize the order of acquisition
- Event query batching
- Many open optimization problems
  - For aggregates like *min* etc
- Conditional Plans (Deshpande, Guestrin, Hellerstein, Hong, Madden, ICDE 2005)
  - Acquire cheaper attributes to avoid sampling expensive attributes

# TinyDB: Aggregates

- Processed by propagating “partial state” up the tree
- Nature of aggregate determines the optimizations
- Differentiating Dimensions:
  - duplicate sensitivity (min - not sensitive, avg - sensitive)
  - exemplary (min) vs summary (avg)
  - monotonic (e.g. max)
  - “partial state required” (similar issues to one-pass vs not-one-pass)
    - distributive (sum) vs algebraic (avg) vs holistic (median)
    - vs unique (no. of distinct elements) vs content-sensitive

	MAX, MIN	COUNT, SUM	AVERAGE	MEDIAN	COUNT DISTINCT <sup>4</sup>	HISTOGRAM <sup>5</sup>
Duplicate Sensitive	No	Yes	Yes	Yes	No	Yes
Exemplary (E), Summary (S)	E	S	S	E	S	S
Monotonic	Yes	Yes	No	No	Yes	No
Partial State	Distributive	Distributive	Algebraic	Holistic	Unique	Content-Sensitive

Table 1: Classes of aggregates

# TinyDB: Query/Result Dissemination

- Standard approach:
  - Build a routing tree starting at the root
  - Use the tree to send messages back
- Semantic routing trees
  - A form of index on constant attributes

# Model-Driven Data Acquisition

- VLDB 2004, Deshpande, Guestrin, Madden, Hellerstein, Hong.
- Incorporate a statistical model of the underlying physical phenomenon in query processing
- Needed to handle:
  - Noisy Data
  - Missing/incomplete data
  - Misrepresentation of data
- More energy-efficient
  - Use spatio-temporal correlations to avoid acquiring data
- Pre-cursor to the MauveDB work