

## TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks

Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong  
UC Berkeley  
OSDI '02

Presented by Svilen Mihaylov  
Database Group, University of Pennsylvania  
Slides courtesy of Samuel Madden

1

## TAG Introduction

- What is a sensor network?
- Programming sensor nets is hard!
- Declarative queries are easy
  - **Tiny Aggregation (TAG):** In-network processing via declarative queries
- Connection to previous papers in class
  - Declarative query routing. Few lines of SQL, many lines of C.
  - Do as much processing in the network as possible
- Example:
  - » Vehicle tracking application: 2 weeks for 2 students
  - » Vehicle tracking query: took 2 minutes to write, worked just as well!



```
SELECT MAX(mag)
FROM sensors
WHERE mag > thresh
EPOCH DURATION 64ms
```

## Sensor Networks (cont)

- Sensor networks vs. wired networks
- Unreliable communication:
  - Messages get lost quite often (20 - 30%).
  - Payload is small (e.g. 30 bytes) - maximizes total throughput
  - Packet overhead large proportional to payload
  - Why?
    - » Weak signal (note - two AA batteries); small antenna.
    - » No forward error correction (e.g. Turbo Codes in cell phones); limited power and memory.
    - » Larger packet means larger probability of collision (and more lost messages).
- Power restrictions:
  - Can do processing but not too-sophisticated (e.g. Chase and BackChase)
  - Real-time requirements - need to process and send while others are still listening.
  - Clock is slow - tens of Mhz
  - Must operate for months on a couple of batteries.

3

## Overview

- **Sensor Networks**
- **Queries in Sensor Nets**
- **Tiny Aggregation**
  - Overview
  - Simulation & Results

4

## Device Capabilities

- "Mica Motes" (2002, still widely used)
  - 8bit, 4Mhz processor
    - » Roughly a PC AT
  - 40kbit CSMA radio
  - 4KB RAM, 128KB flash, 512KB EEPROM.
  - TinyOS based
- Variety of other, similar platforms exist
  - UCLA WINS, Medusa, Princeton ZebraNet, SmartIts
  - Newer ones: Telos - 10KB, research: Intel Mote 2 (520 Mhz, 32Mb RAM).



5

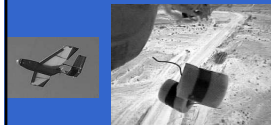
## Sensor Net Sample Apps

Habitat Monitoring: Storm petrels on great duck island, microclimates on James Reserve.



Earthquake monitoring in shake-test sites.

Vehicle detection: sensors along a road, collect data about passing vehicles.

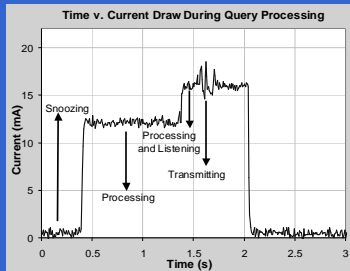


Traditional monitoring apparatus.

6

## Metric: Communication

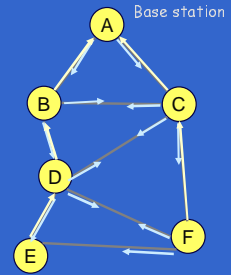
- Lifetime from one pair of AA batteries
  - 2-3 days at full power
  - 6 months at 2% duty cycle
- Communication dominates cost
  - < few mS to compute
  - 30mS to send message
- Our metric: communication!



7

## Communication In Sensor Nets

- Radio communication has high link-level losses
  - typically about 20% @ 5m
- Ad-hoc neighbor discovery
- Tree-based routing
  - Aggregation tree, rooted at the base station
  - Choose parent at minimum distance to base station



8

## Overview

- Sensor Networks
- Queries in Sensor Nets
- Tiny Aggregation
  - Overview
  - Optimizations & Results

9

## Declarative Queries for Sensor Networks

- Examples:
  - SELECT nodeid, light FROM sensors WHERE light > 400 EPOCH DURATION 1s

		Sensors			
Epoch	Nodeid	Light	Temp	Accel	Sound
0	1	455	x	x	x
0	2	389	x	x	x
1	1	422	x	x	x
1	2	405	x	x	x

"Where" predicate is local, can be pushed down the aggregation tree

10

## Aggregation Queries

- SELECT AVG(sound) FROM sensors EPOCH DURATION 10s



Epoch	AVG(sound)
0	440
1	445

- SELECT roomNo, AVG(sound) FROM sensors GROUP BY roomNo HAVING AVG(sound) > 200 EPOCH DURATION 10s



Epoch	roomNo	AVG(sound)
0	1	360
0	2	520
1	1	370
1	2	520

Rooms w/ sound > 200

11

## Overview

- Sensor Networks
- Queries in Sensor Nets
- Tiny Aggregation
  - Overview
  - Optimizations & Results

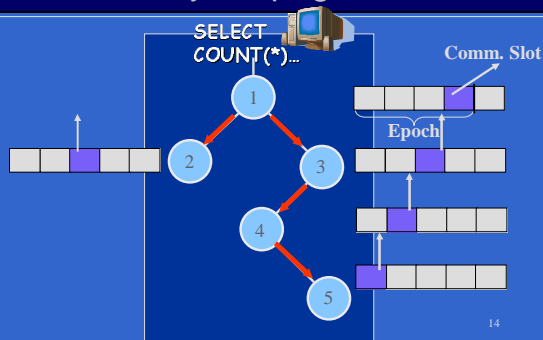
12

## TAG

- In-network processing of aggregates
  - Common data analysis operation
    - » Aka *gather* operation or *reduction* in || programming
  - Communication reducing
    - » Operator dependent benefit
  - Across nodes during same epoch
- Exploit semantics improve efficiency!

13

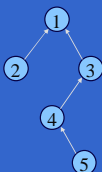
## Query Propagation



14

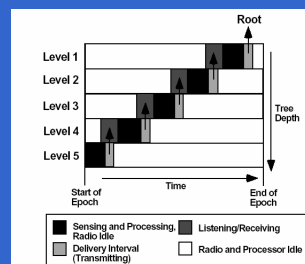
## Basic Aggregation

- In each epoch:
  - Each node samples local sensors once
  - Generates partial state record (PSR)
    - » local readings
    - » readings from children
  - Outputs PSR during its comm. slot.
- At end of epoch, PSR for whole network output at root
- (In paper: pipelining, grouping)



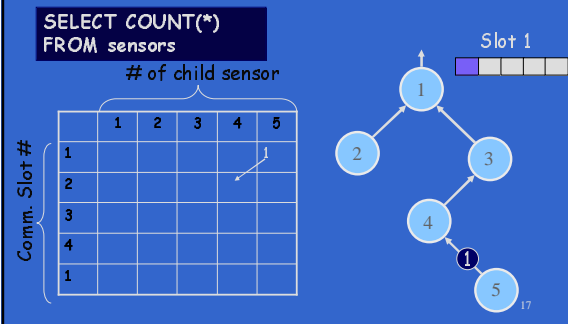
15

## Communication Staggering



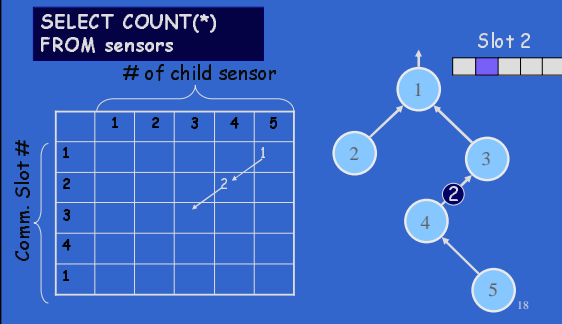
16

## Illustration: Aggregation



17

## Illustration: Aggregation



18

### Illustration: Aggregation

**SELECT COUNT(\*)  
FROM sensors**

# of child sensor

Comm. Slot #

	1	2	3	4	5
1					1
2				2	
3		1	3		
4					
5					

Slot 3

19

### Illustration: Aggregation

**SELECT COUNT(\*)  
FROM sensors**

# of child sensor

Comm. Slot #

	1	2	3	4	5
1					1
2				2	
3		1	3		
4	5				
5					

Slot 4

20

### Illustration: Aggregation

**SELECT COUNT(\*)  
FROM sensors**

# of child sensor

Comm. Slot #

	1	2	3	4	5
1					1
2				2	
3		1	3		
4	5				
5					1

Slot 1

21

### Aggregation Framework

- As in extensible databases, we support any aggregation function conforming to:
  - $Agg_n = \{f_{init}, f_{merge}, f_{evaluate}\}$
  - $f_{init}(a_0) \rightarrow \langle a_0 \rangle$  Partial State Record (PSR)
  - $f_{merge}(\langle a_1 \rangle, \langle a_2 \rangle) \rightarrow \langle a_{12} \rangle$
  - $f_{evaluate}(\langle a_i \rangle) \rightarrow \text{aggregate value}$
- (Merge associative, commutative)
- Example 1: Average
  - $AVG_{init}(v) \rightarrow \langle v, 1 \rangle$
  - $AVG_{merge}(\langle S_1, C_1 \rangle, \langle S_2, C_2 \rangle) \rightarrow \langle S_1 + S_2, C_1 + C_2 \rangle$
  - $AVG_{evaluate}(\langle S, C \rangle) \rightarrow S/C$
- Example 2: Min
  - $AVG_{init}(v) \rightarrow \langle v \rangle$
  - $AVG_{merge}(\langle v_1 \rangle, \langle v_2 \rangle) \rightarrow \langle \min(v_1, v_2) \rangle$
  - $AVG_{evaluate}(\langle v \rangle) \rightarrow v$

22

### Types of Aggregates

- SQL supports MIN, MAX, SUM, COUNT, AVERAGE
- Any of those functions *can* be computed via TAG
- In network benefit for many operations
  - E.g. Standard deviation, top/bottom N, spatial union/intersection, histograms, etc.
  - Compactness of PSR necessary for efficient operation

23

### Taxonomy of Aggregates

- TAG insight: classify aggregates according to various functional properties
  - Yields a general set of optimizations that can automatically be applied

Property	Examples	Affects
Partial State	MEDIAN: unbounded, MAX: 1 record	Effectiveness of TAG
Duplicate Sensitivity	MIN: dup. insensitive, AVG: dup. sensitive	Routing Redundancy
Exemplary vs. Summary	MAX: exemplary, COUNT: summary	Applicability of Sampling, Effect of Loss
Monotonic	MAX: monotonic, AVG: non-monotonic	Hypothesis Testing, Snooping

24

## TAG Advantages

- Communication Reduction
  - Important for power and contention
- Continuous stream of results
  - Smooth transient faults across epochs
    - » Interpolate PSRs between epochs
- Lots of optimizations
  - Via operator semantics

25

## Simulation Environment

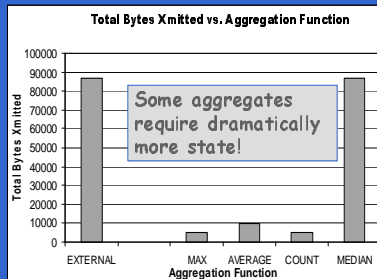
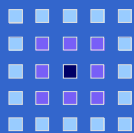
- Evaluated via simulation
- Coarse grained event based simulator
  - Sensors arranged on a grid
  - Two communication models
    - » Lossless: All neighbors hear all messages
    - » Lossy: Messages lost with probability that increases with distance

26

## Benefit of In-Network Processing

### Simulation Results

2500 Nodes  
50x50 Grid  
Depth = ~10  
Neighbors = ~20



28

## Optimization: Channel Sharing ("Snooping")

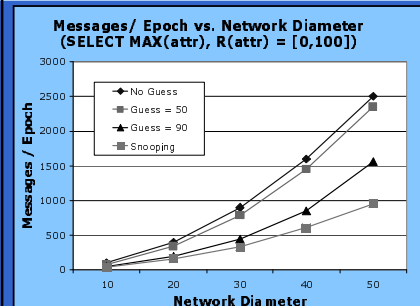
- Insight: Shared channel enables optimizations
- Suppress messages that won't affect parent aggregate - "horizontal aggregation"
  - E.g., MAX
  - Applies to all exemplary, monotonic aggregates

## Optimization: Hypothesis Testing

- Insight: Guess from root can be used for suppression
  - E.g. 'MIN < 50'
  - Works for monotonic & exemplary aggregates
    - » Also summary, if imprecision allowed within some error bound
- How is hypothesis computed?
  - Blind or statistically informed guess
  - Observation over network subset

29

## Experiment: Hypothesis Testing

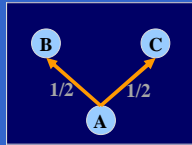


- Uniform Value Distribution
- Dense Packing
- Ideal Communication

30

## Optimization: Use Multiple Parents

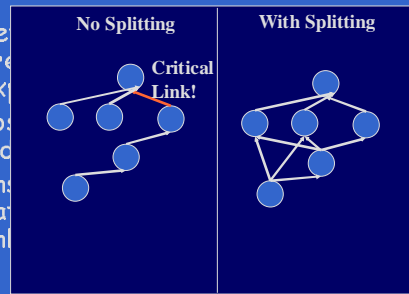
- For duplicate insensitive aggregates
- Or aggregates that can be expressed as a linear combination of parts
  - Send (part of) aggregate to all parents
    - In just one message, via broadcast
  - Decreases variance



31

## Multiple Parents Results

- Better performance
- Lower variance
- Increased data link utilization



Splitting
No Splitting
6 parents per

32

## Summary

- TAG enables in-network declarative query processing
  - State dependent communication benefit
  - Transparent optimization via taxonomy
    - Hypothesis Testing, Parent Sharing
- Declarative queries are the right interface for data collection in sensor nets!
  - Easier to program *and* more efficient for vast majority of users

TinyDB Release Available - <http://telegraph.cs.berkeley.edu/tinydb>

33

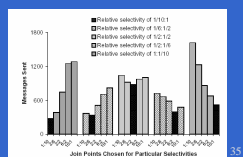
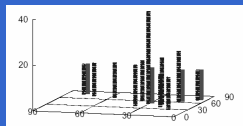
## Shortcomings

- No "joins". Just aggregate functions
  - Difficult to find correlations between readings
  - Difficult to use for signaling events (e.g. "propane tank approaching building on fire")
- Communication problems
  - No guarantees for delivery. Not acceptable in the case of the propane tank
  - Retransmissions will destroy the time slotting scheme and skew the results
  - Difficult to overcome inherent packet length (for larger readings, more sophisticated partial records)
- Questionable simulation (by today's standard)
  - Event-level Java simulator? No bit-level TOSSIM or Avrona
  - Only grid arrangement tested on a large scale. Hardly the case in a real deployment
  - Did not study load distribution, only total load on the system
- Addressed in later work (e.g. the Database group at Penn)

34

## Some Later Research

- Load distribution
- Join optimization



35