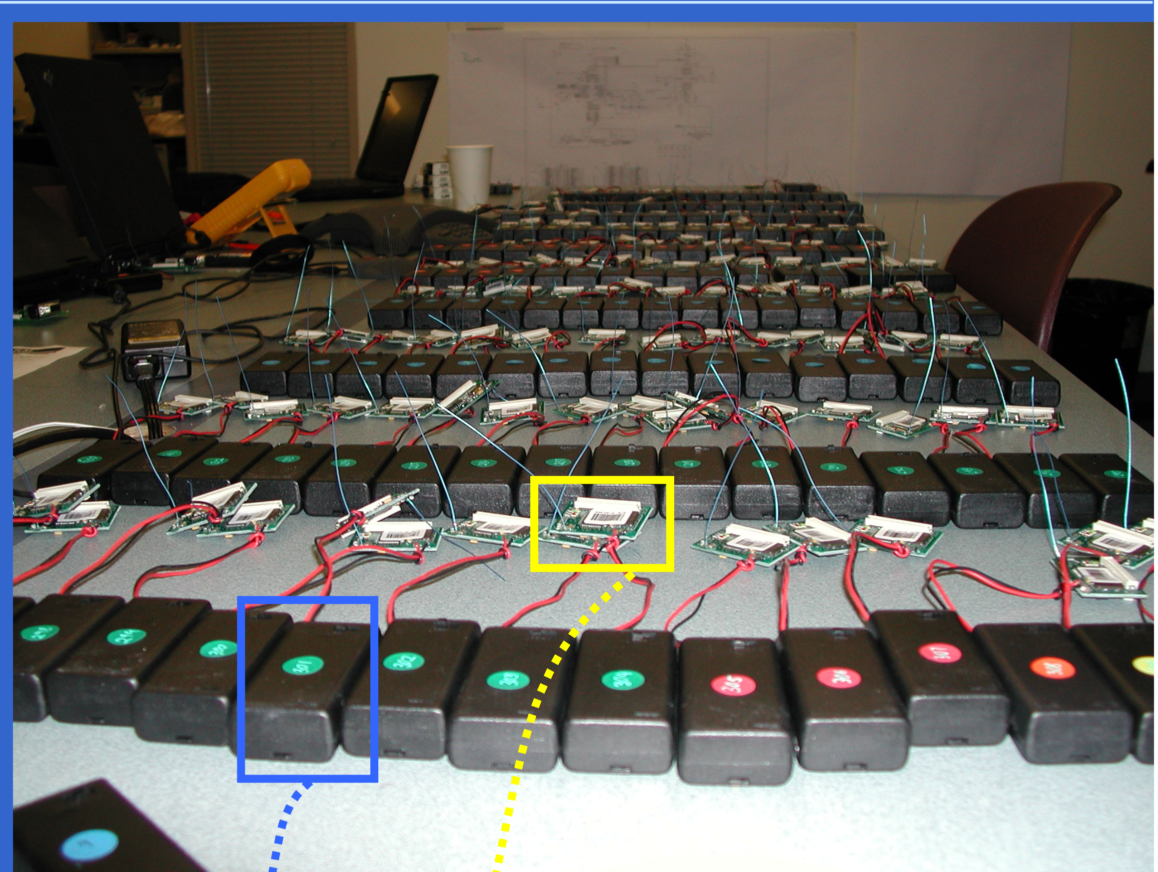
Querying Sensor Networks

Sam Madden

Sensor Networks

- Small computers with:
  - Radios
  - Sensing hardware
  - Batteries

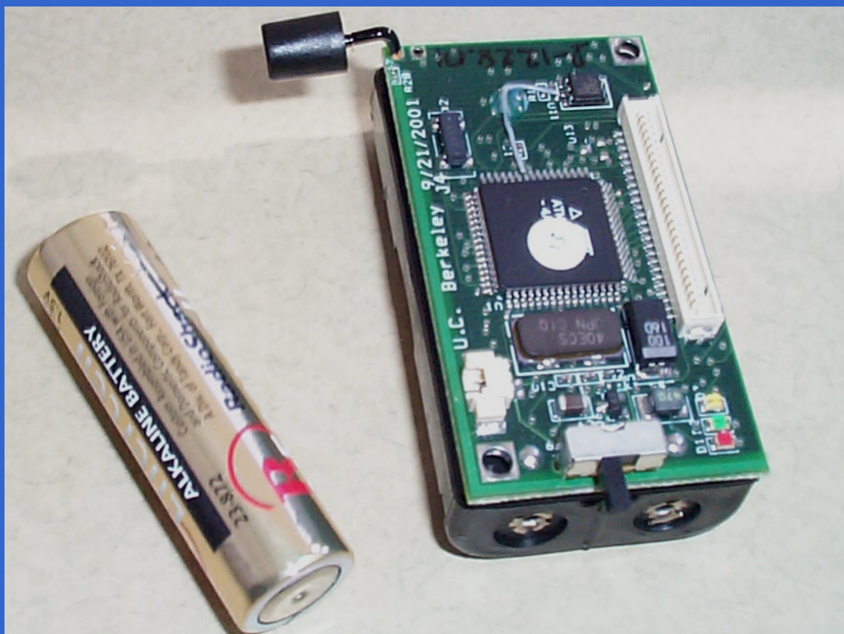
- Remote deployments
  - Long lived
  - 10s, 100s, or 1000s



Battery Pack

Smart Sensor, aka "*Mote*"

Motes



Mica Mote

4Mhz, 8 bit Atmel RISC uProc

40 kbit Radio

4 K RAM, 128 K Program Flash, 512 K Data Flash

AA battery pack

Based on TinyOS*

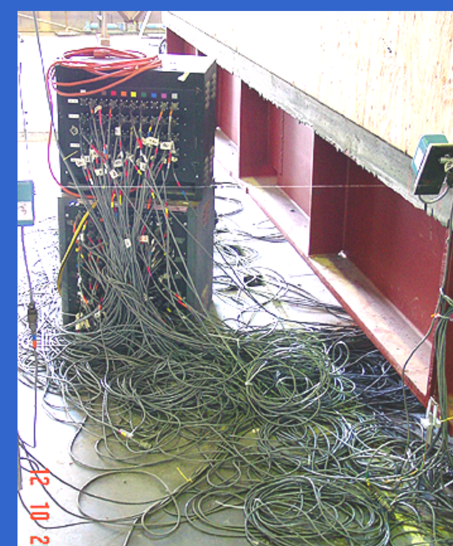
*Hill, Szewczyk, Woo, Culler, & Pister. "Systems Architecture Directions for Networked Sensors." ASPLOS 2000. http://webs.cs.berkeley.edu/tos

Sensor Net Sample Apps

<u>Habitat Monitoring</u>: Storm petrels on Great Duck Island, microclimates on James Reserve.

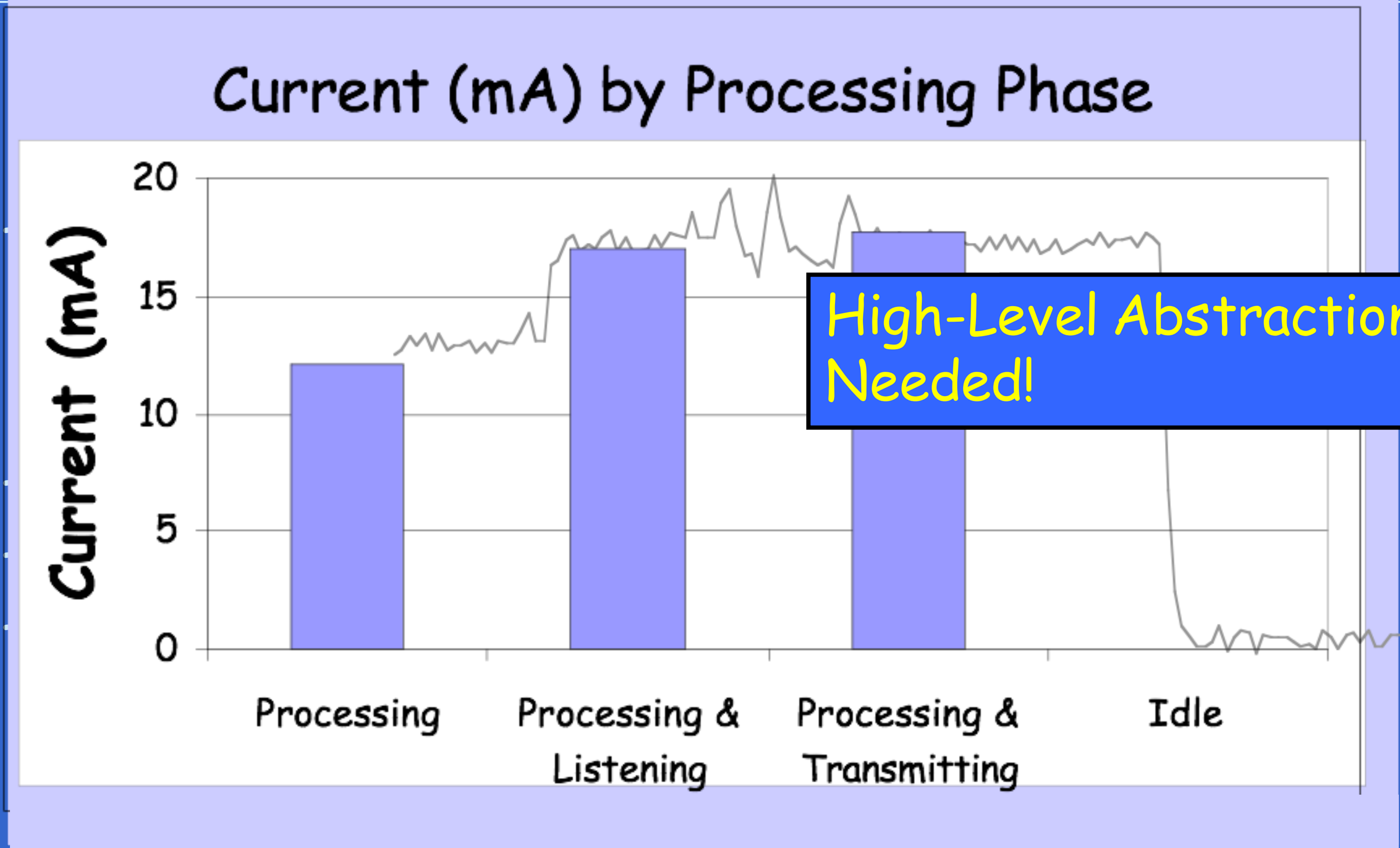
<u>Earthquake monitoring</u> in shake-test sites.

<u>Vehicle detection</u>: sensors along a road, collect data about passing vehicles.



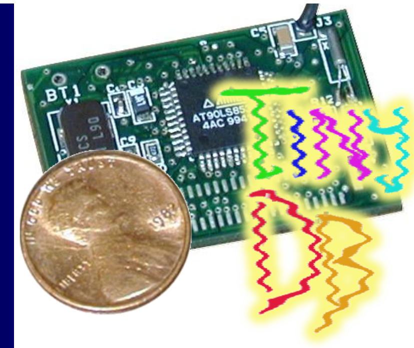
Traditional monitoring apparatus.

Programming Sensor Nets Is Hard



High-Level Abstraction Is Needed!

A Solution: Declarative Queries



- Users specify the data they want
  - Simple, SQL-like queries
  - Using predicates, not specific addresses
  - Same spirit as *Cougar* – Our system: TinyDB
- Challenge is to provide:
  - Expressive & easy-to-use interface
  - High-level operators
    - » Well-defined interactions
    - » "Transparent Optimizations" that many programmers would miss
      - Sensor-net specific techniques
  - Power efficient execution framework
- Question: do sensor networks change query processing? Yes!

Overview

- TinyDB: Queries for Sensor Nets
- Processing Aggregate Queries (TAG)
- Taxonomy & Experiments
- Acquisitional Query Processing
- Other Research
- Future Directions

Overview

- TinyDB: Queries for Sensor Nets
- Processing Aggregate Queries (TAG)
- Taxonomy & Experiments
- Acquisitional Query Processing
- Other Research
- Future Directions

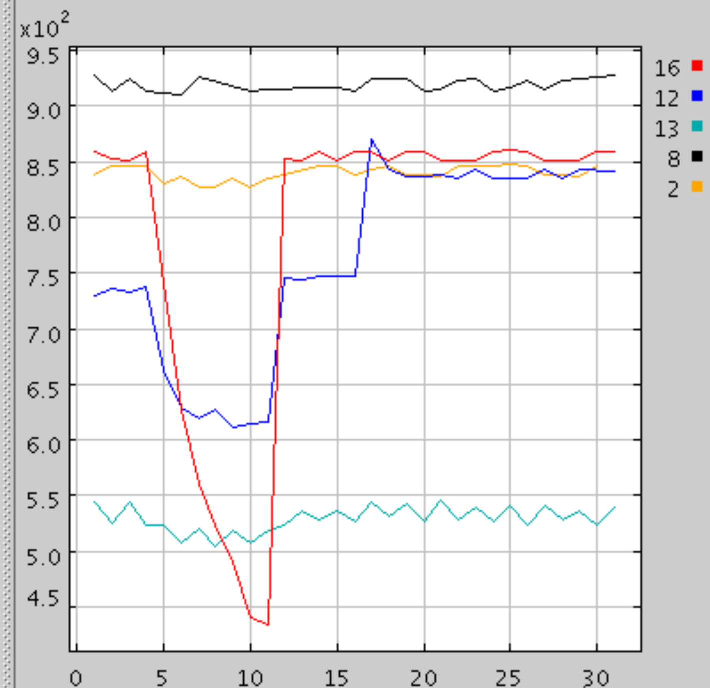
TinyDB Demo

Query 8

SELECT s.nodeid, s.light FROM sensors AS s

EPOCH DURATION 1024

Epoch	nodeid	light
26	12	836
26	16	859
26	2	846
26	8	923
26	13	523
27	16	851
27	8	915
27	13	541
27	12	844
27	2	838
28	2	838
28	12	836
28	13	528
28	16	852
28	8	923
29	13	536
29	12	843
29	2	837
29	16	851
30	13	523
30	16	859
30	2	846
31	8	928
31	13	540
31	12	841



light | Reset Graph | Clear Graph

Stop Query | Resend Query

TinyDB Architecture

SELECT
AVG(temp)
WHERE
light > 400

T:1, AVG: 225
T:2, AVG: 250

Schema:

- "Catalog" of commands & attributes

get (

getTempF

() ...

~10,000 Lines Embedded C Code

~5,000 Lines (PC-Side) Java

~3200 Bytes RAM (w/ 768 byte heap)

~58 kB compiled code

(3x larger than 2nd largest TinyOS Program)

Declarative Queries for Sensor Networks

"Find the sensors in bright nests."

1 Examples:

```
SELECT nodeid, nestNo, light
FROM sensors
WHERE light > 400
EPOCH DURATION 1s
```

Sensors

Epoch	Nodeid	nestNo	Light
0	1	17	455
0	2	25	389
1	1	17	422
1	2	25	405

Aggregation Queries

② SELECT AVG(sound)

FROM sensors

EPOCH DURATION 10s

> "Count the number occupied nests in each loud region of the island."

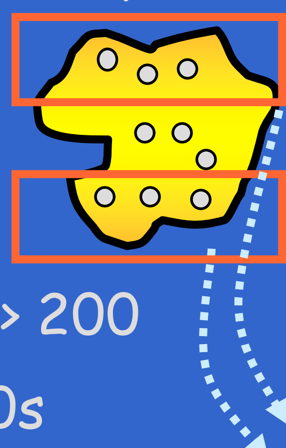
③ SELECT region, CNT(occupied)
AVG(sound)

FROM sensors

GROUP BY region

HAVING AVG(sound) > 200

EPOCH DURATION 10s



Epoch	region	CNT(…)	AVG(…)
0	North	3	360
0	South	3	520
1	North	3	370
1	South	3	520

Regions w/ AVG(sound) > 200

Overview

- TinyDB: Queries for Sensor Nets
- Processing Aggregate Queries (TAG)
- Taxonomy & Experiments
- Acquisitional Query Processing
- Other Research
- Future Directions

Tiny Aggregation (TAG)

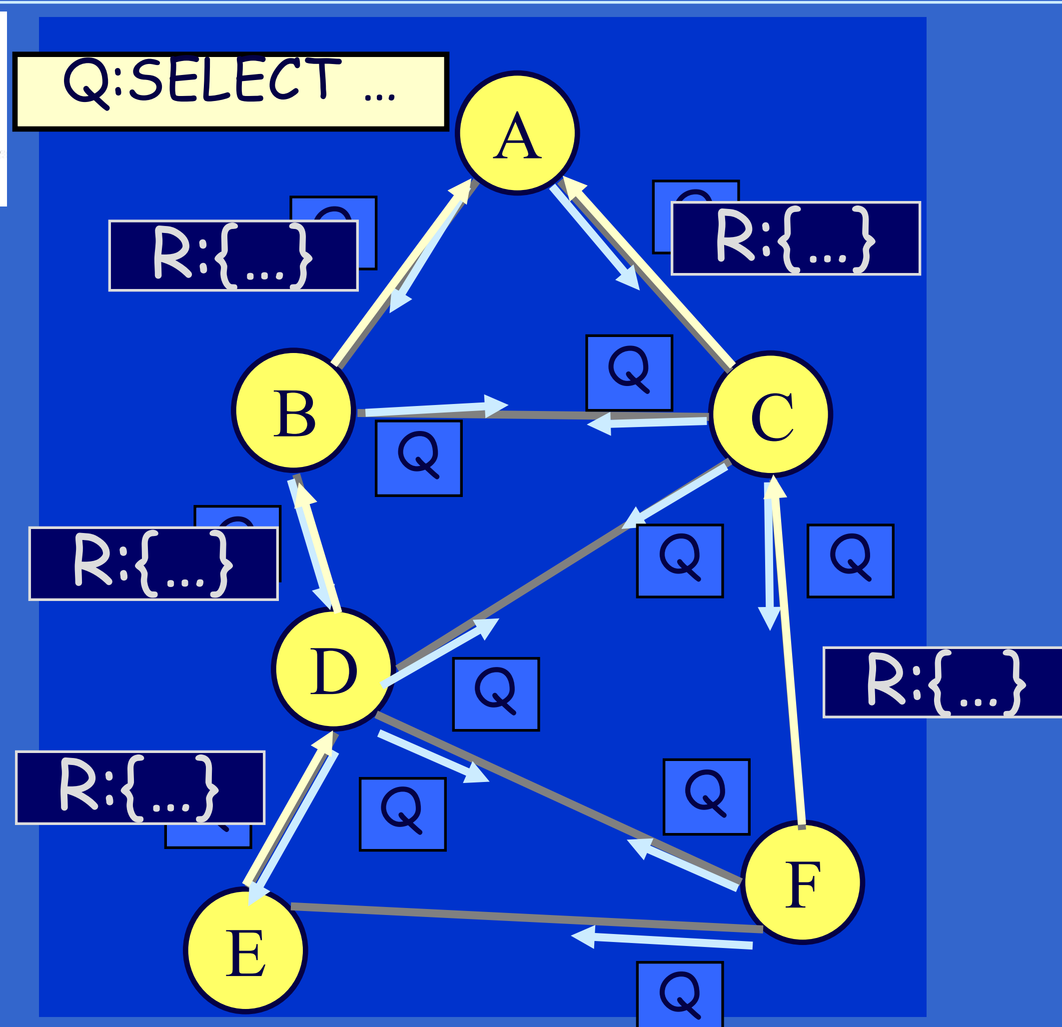
- In-network processing of aggregates
  - Common data analysis operation
    - » Aka *gather* operation or *reduction* in || programming
  - Communication reducing
    - » Operator dependent benefit
  - Across nodes during same epoch

- Exploit query semantics to improve efficiency!

Madden, Franklin, Hellerstein, Hong. Tiny AGgregation (TAG), OSDI 2002.

Query Propagation Via Tree-Based Routing

- Tree-based routing
  - Used in:
    - » Query delivery
    - » Data collection
  - Topology selection is important; e.g.
    - » Krishnamachari, DEBS 2002, Intanagonwiwat, ICDCS 2002, Heidemann, SOSP 2001
    - » LEACH/SPIN, Heinzelman et al. MOBICOM 99
    - » SIGMOD 2003
  - Continuous process
    - » Mitigates failures



Basic Aggregation

- In each epoch:
  - Each node samples local sensors once
  - Generates partial state record (PSR)
    - » local readings
    - » readings from children
  - Outputs PSR during assigned comm. interval

- At end of epoch, PSR for whole network output at root
- New result on each successive epoch

- Extras:
  - Predicate-based partitioning via GROUP BY

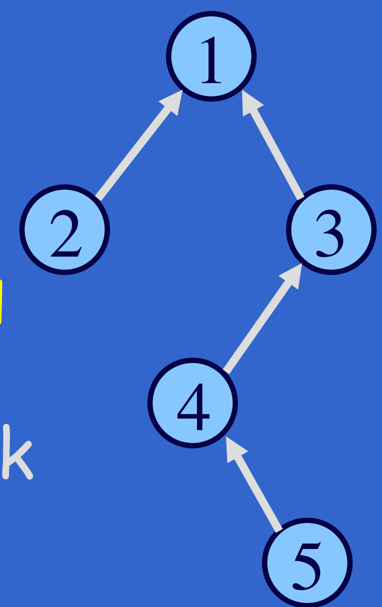


Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```

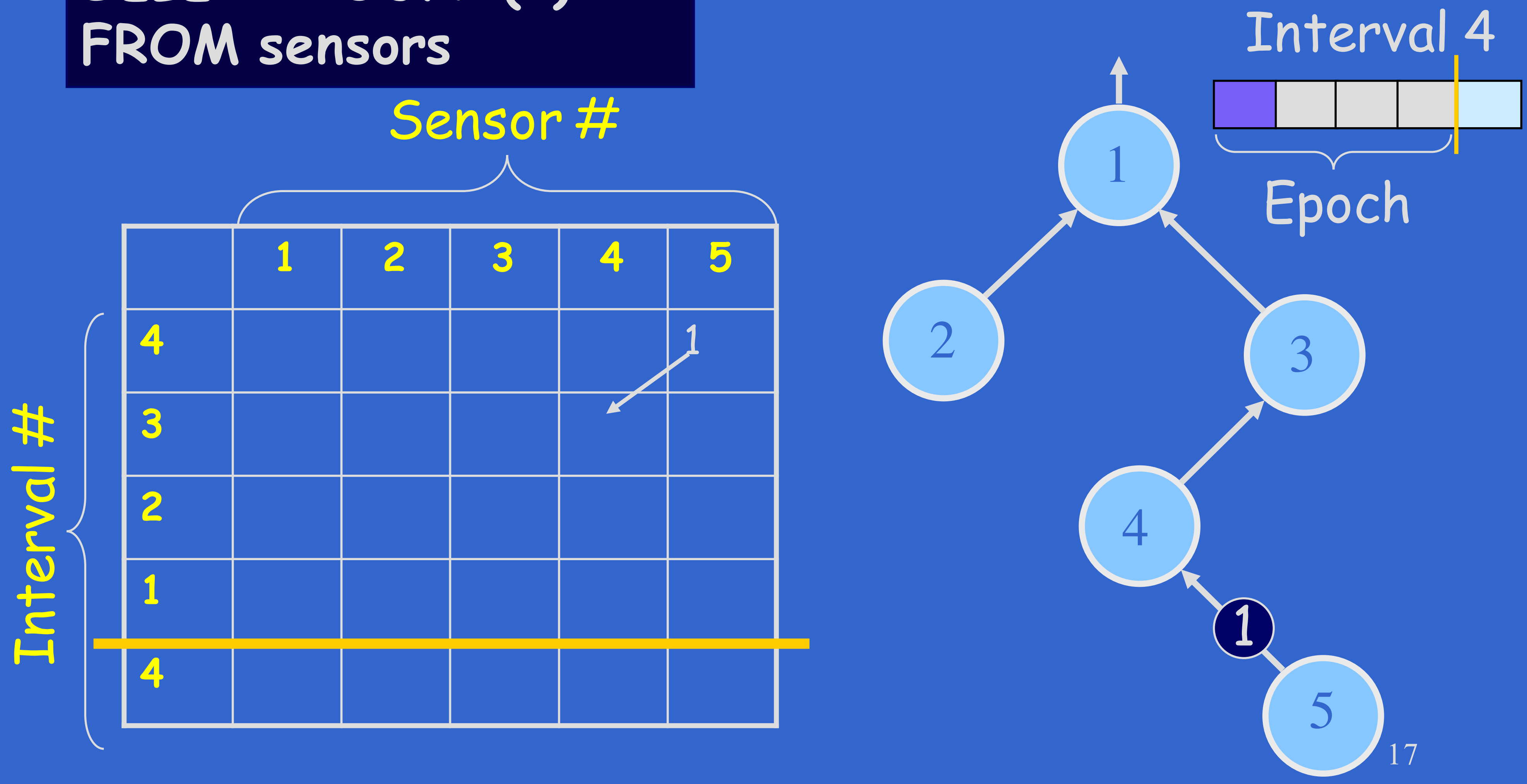
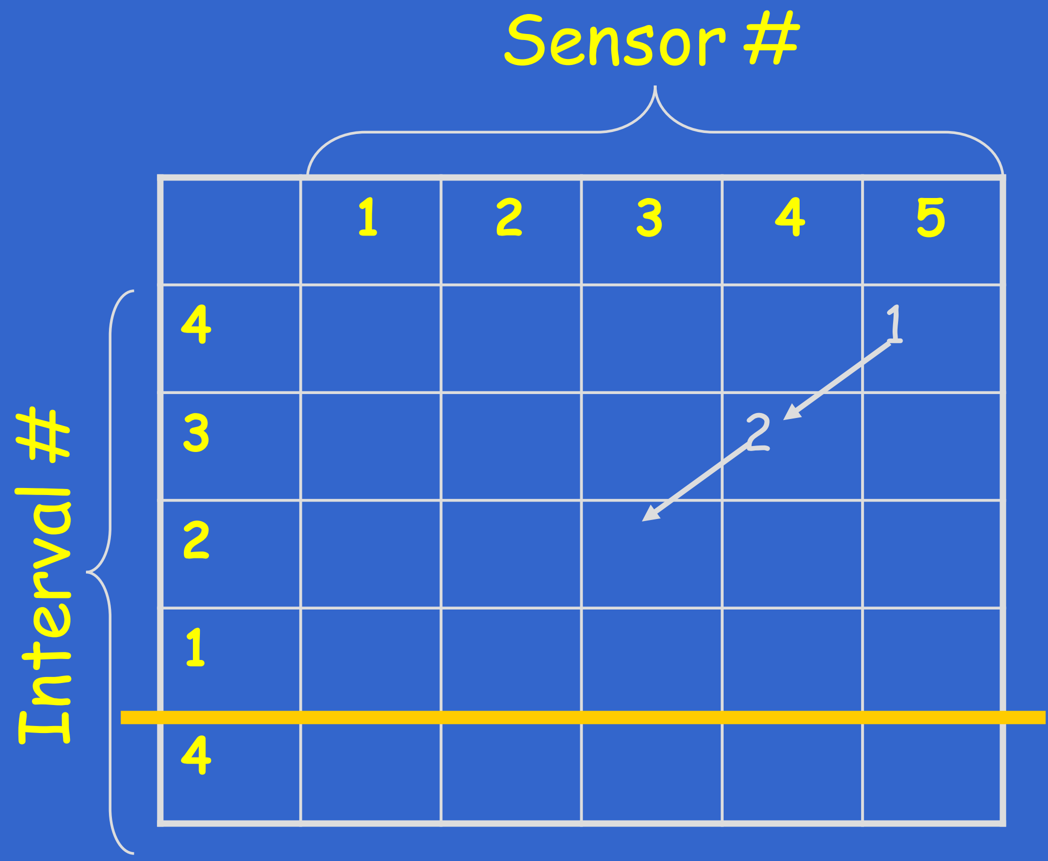


Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```



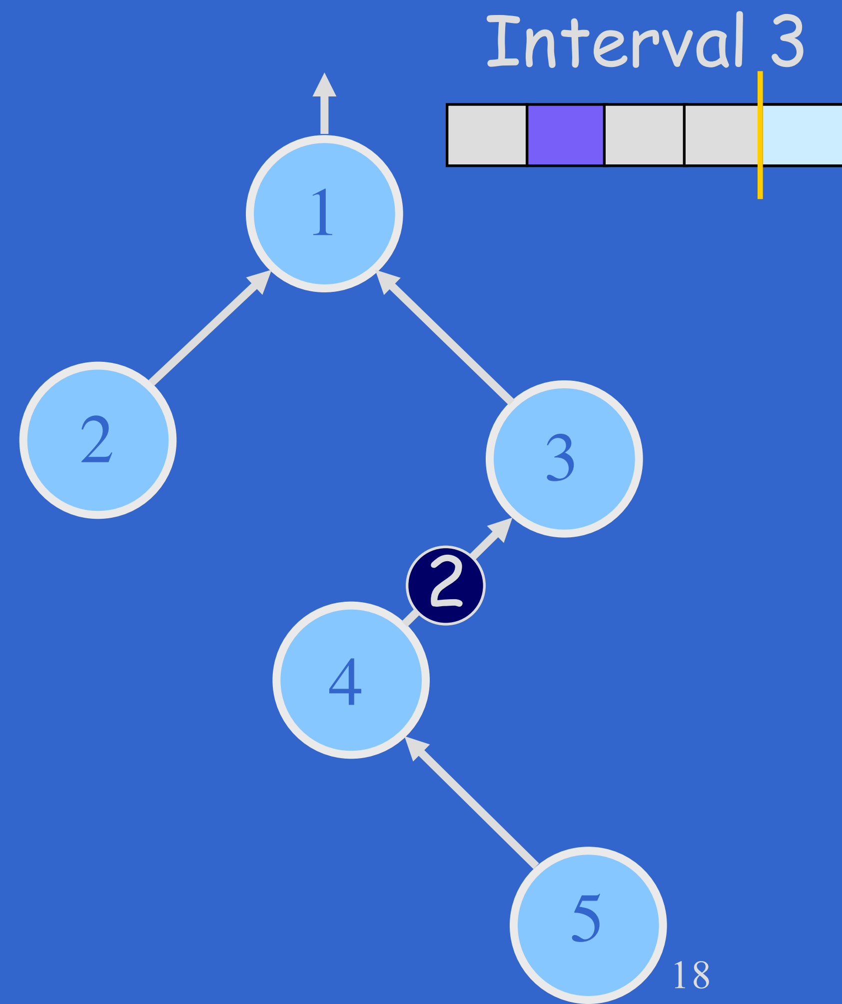
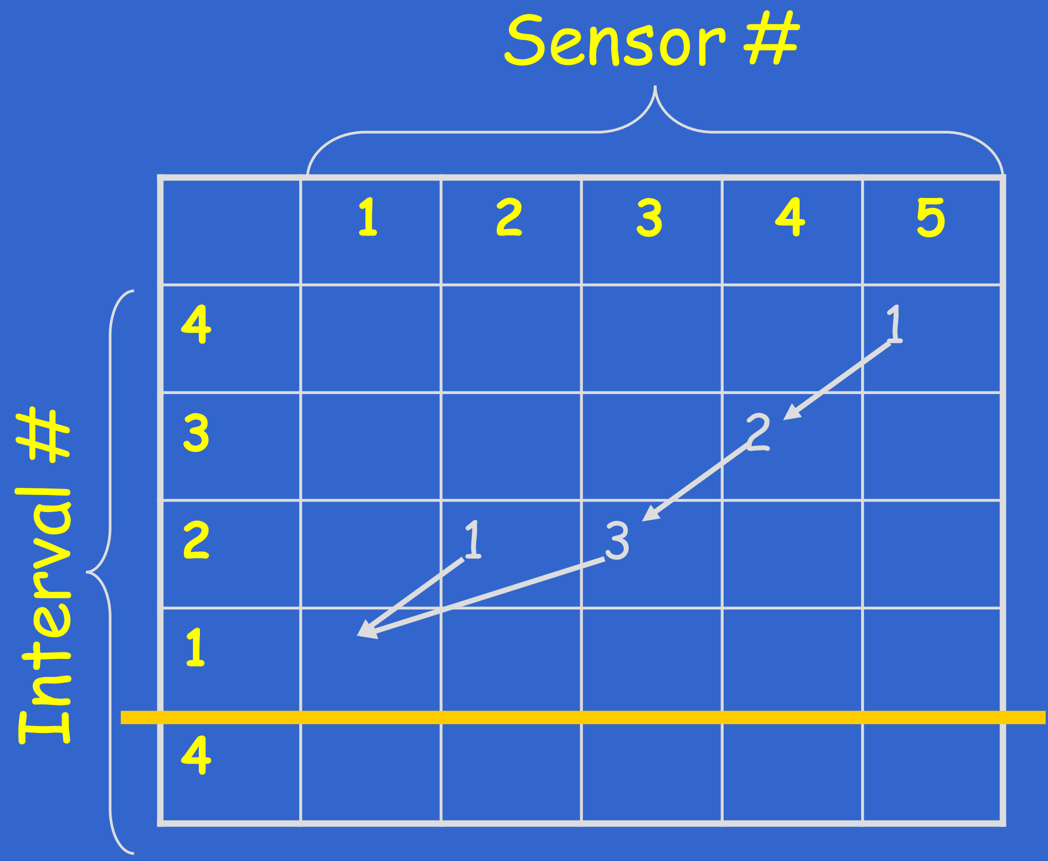


Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```



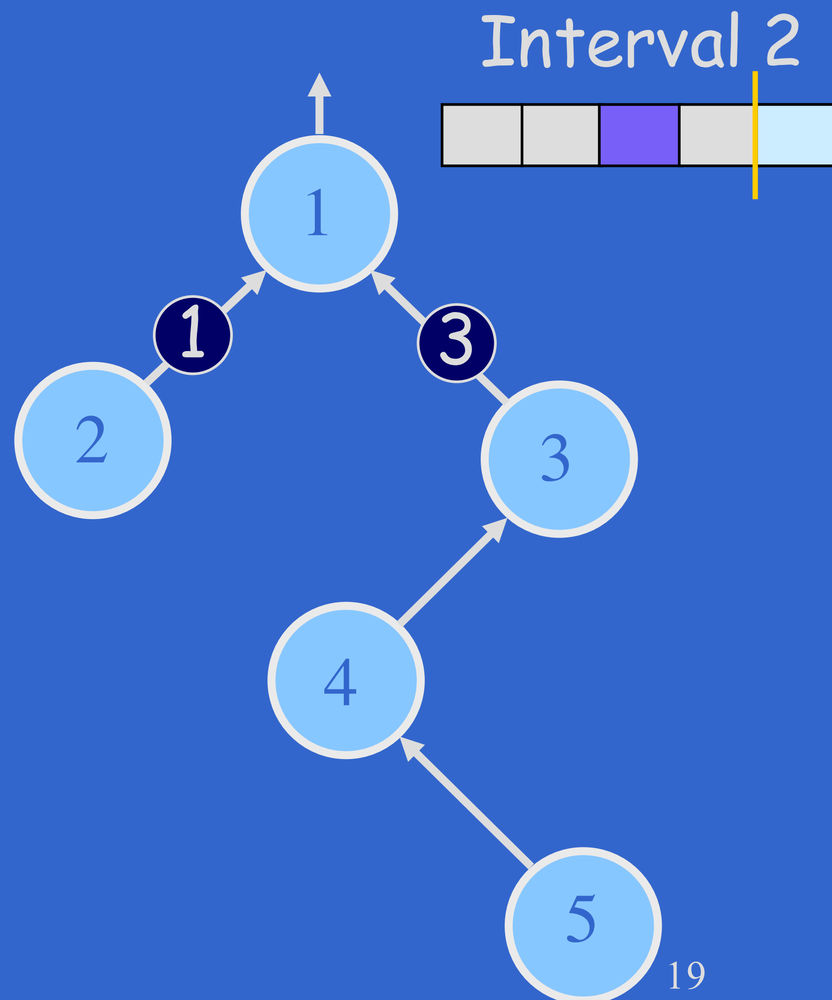
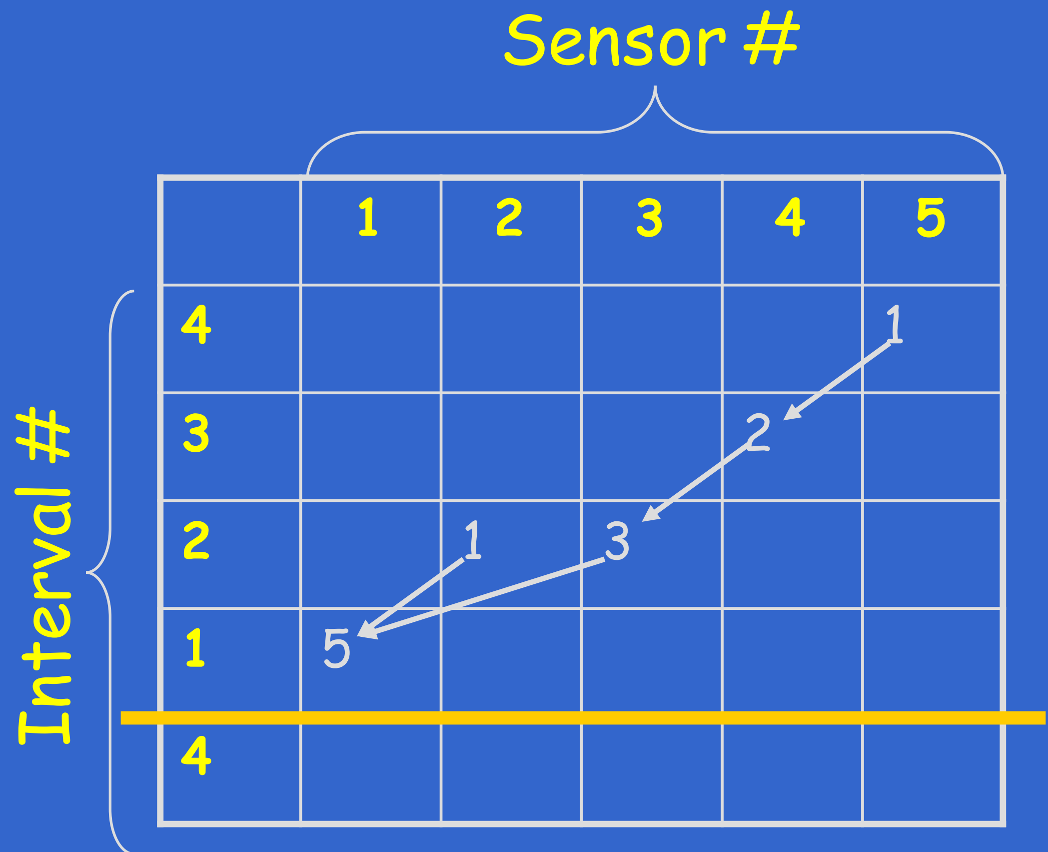


Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```



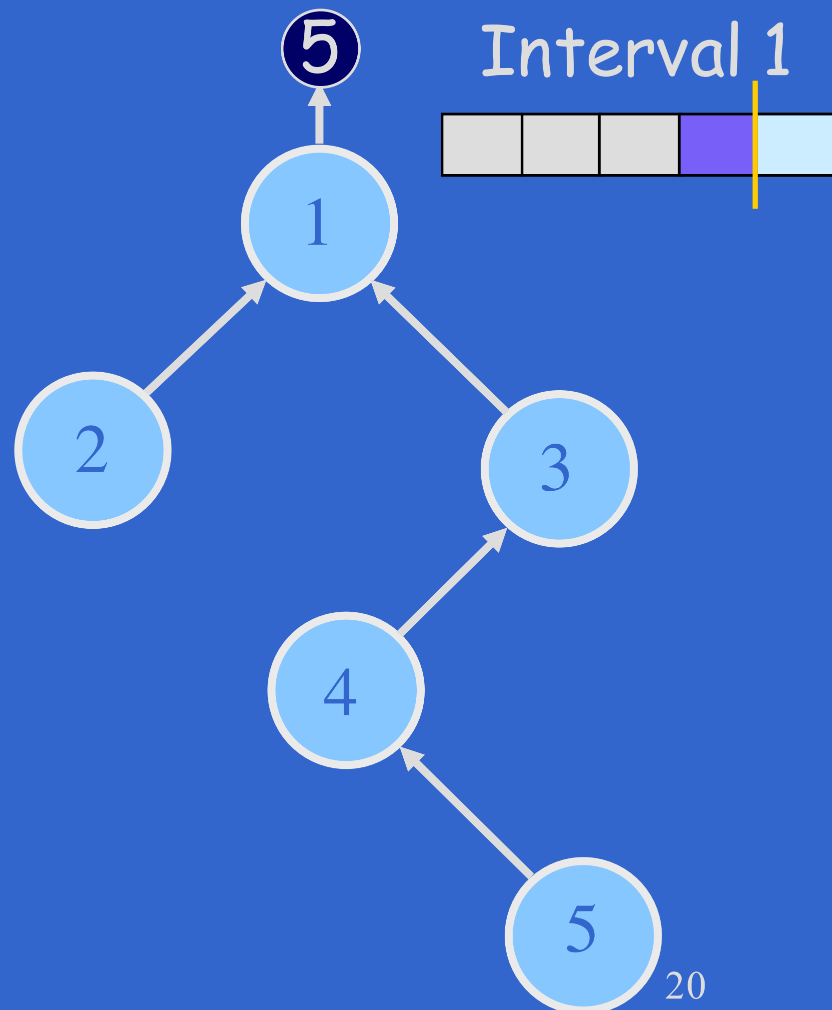
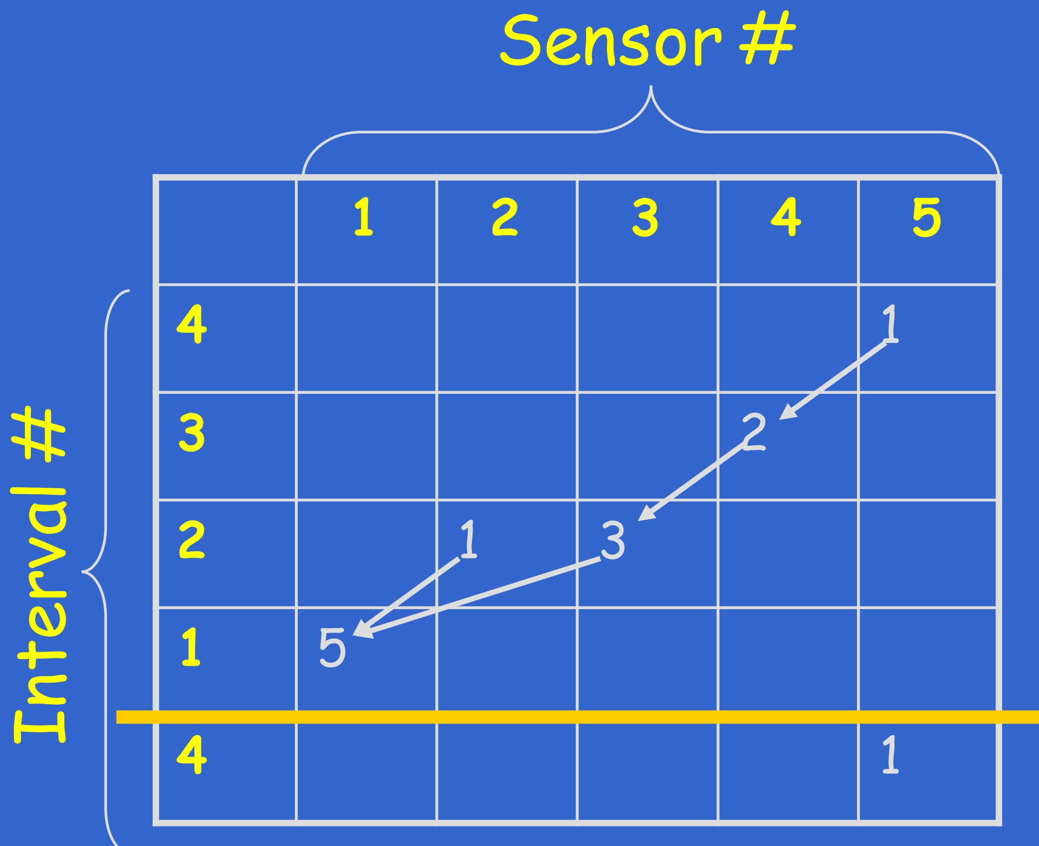
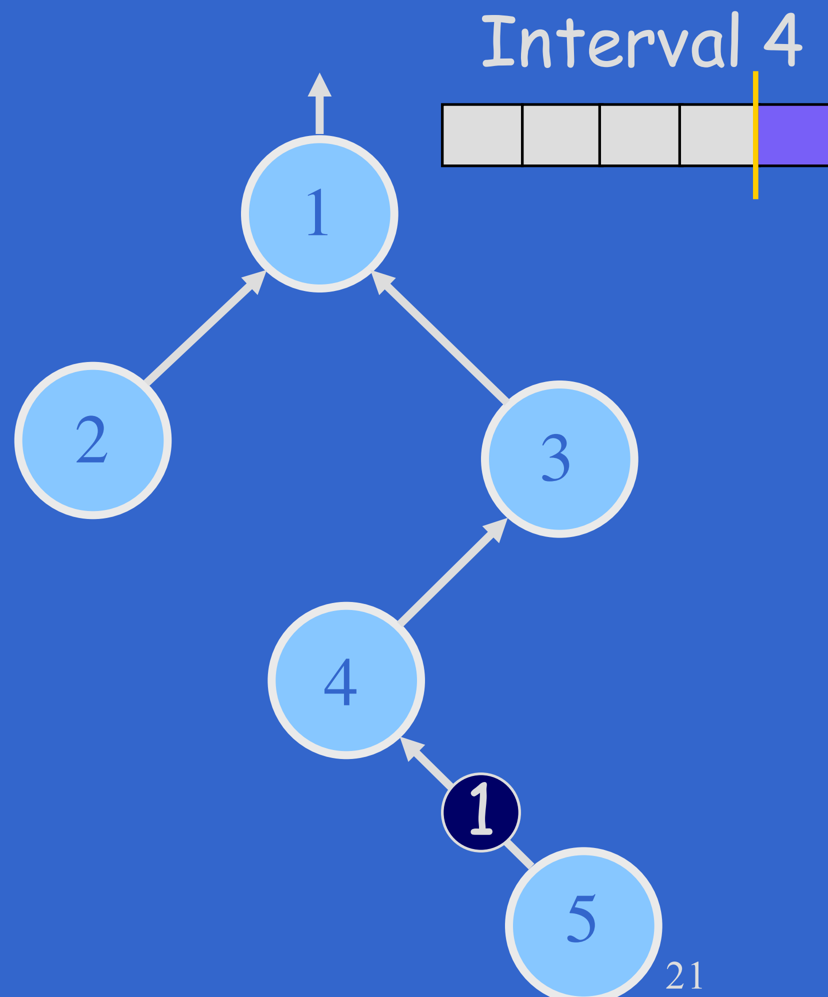


Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```



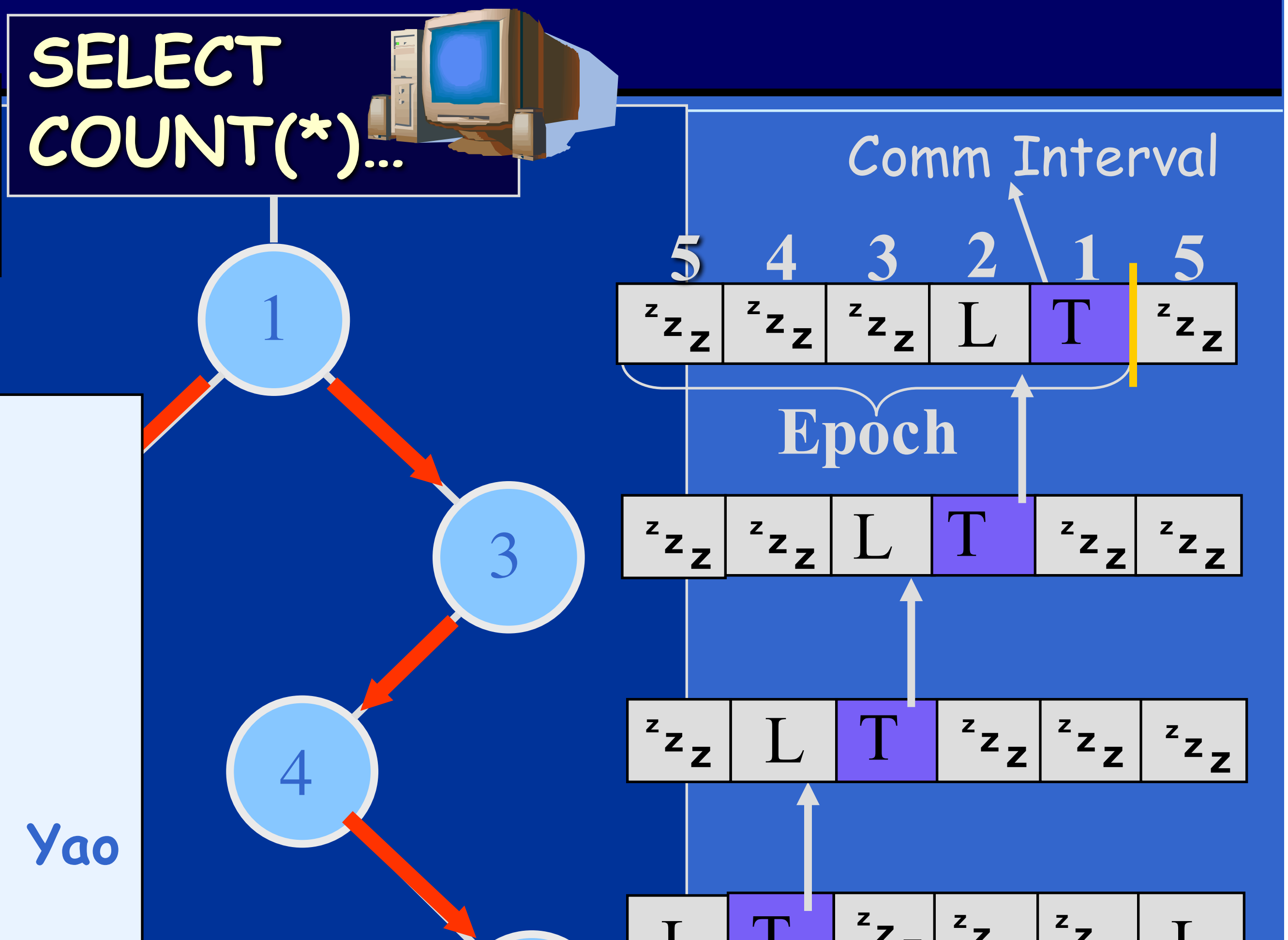


Interval Assignment: An Approach

4 intervals / epoch

Interval # = Level

Level = 1



- **CSMA for collision avoidance**
- **Time intervals for power conservation**
- **Many variations(**e.g. Yao & Gehrke, CIDR 2003**)**
- **Time Sync (**e.g. Elson & Estrin OSDI 2002**)**

Pipelining: Increase throughput by delaying result arrival until a later epoch

Madden, Szewczyk, Franklin, Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. WMCSA 2002.

Aggregation Framework

- As in extensible databases, we support any aggregation function conforming to:

$Agg_n = \{f_{init}, f_{merge}, f_{evaluate}\}$

$F_{init}\{a_0\} \rightarrow <a_0>$ → Partial State Record (PSR)

$F_{merge}\{<a_1>,<a_2>\} \rightarrow <a_{12}>$

$F_{evaluate}\{<a_1>\} \rightarrow$ aggregate value

Example: Average

$AVG_{init}\ \{v\} \rightarrow <v,1>$

$AVG_{merge}\ \{<S_1, C_1>, <S_2, C_2>\} \rightarrow < S_1 + S_2 , C_1 + C_2>$

$AVG_{evaluate}\{<S, C>\} \rightarrow S/C$

Restriction: Merge associative, commutative

Types of Aggregates

- SQL supports MIN, MAX, SUM, COUNT, AVERAGE
- Any function over a set *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

Overview

- TinyDB: Queries for Sensor Nets
- Processing Aggregate Queries (TAG)
- Taxonomy & Experiments
- Acquisitional Query Processing
- Other Research
- Future Directions

Simulation Environment

- Evaluated TAG 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
- Communication (message counts) as performance metric

Taxonomy of Aggregates

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

Properties
Partial State
Monotonicity
Exemplary vs. Summary
Duplicate Sensitivity

Drives an API!

Partial State

- Growth of PSR vs. number of aggregated values (n)
  - Algebraic: |PSR| = 1 (e.g. MIN)
  - Distributive: |PSR| = c (e.g. AVG)
  - Holistic: |PSR| = n (e.g. MEDIAN)

  "Data Cube", Gray et. al

  - Unique: |PSR| = d (e.g. COUNT DISTINCT)
    - » d = # of distinct values
  - Content Sensitive: |PSR| < n (e.g. HISTOGRAM)

Property	Examples	Affects
Partial State	MEDIAN : unbounded, MAX : 1 record	Effectiveness of TAG

Benefit of In-Network Processing

Simulation Results

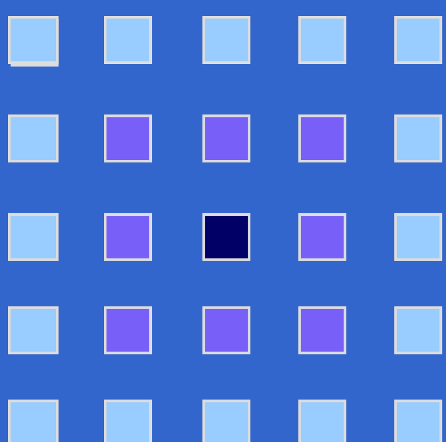
2500 Nodes

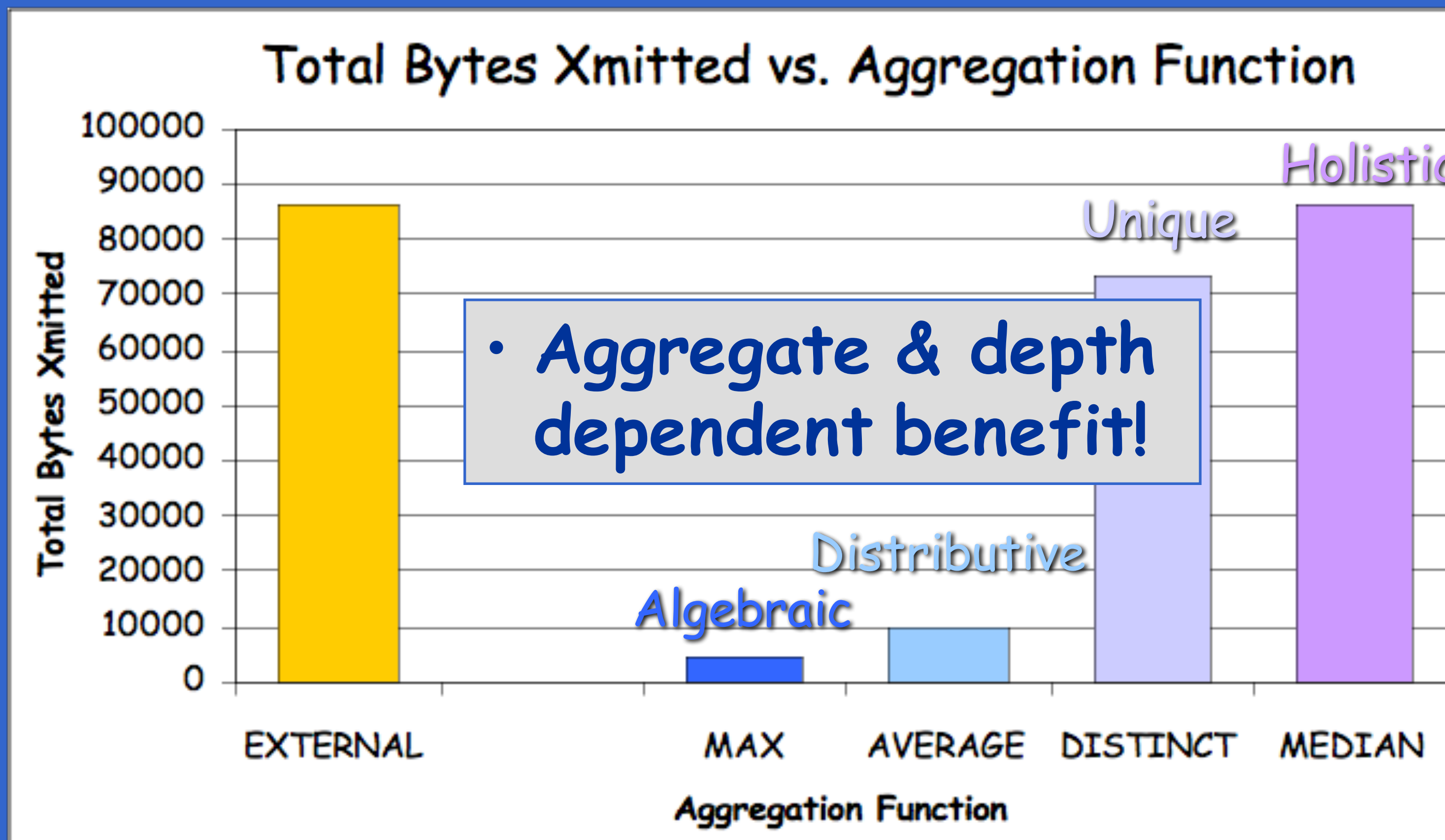
50x50 Grid

Depth = ~10

Neighbors = ~20

Uniform Dist.





- Aggregate & depth dependent benefit!

Monotonicity & Exemplary vs. Summary

Property	Examples	Affects
Partial State	MEDIAN : unbounded, MAX : 1 record	Effectiveness of TAG
Monotonicity	COUNT : monotonic AVG : non-monotonic	Hypothesis Testing, Snooping
Exemplary vs. Summary	MAX : exemplary COUNT: summary	Applicability of Sampling, Effect of Loss

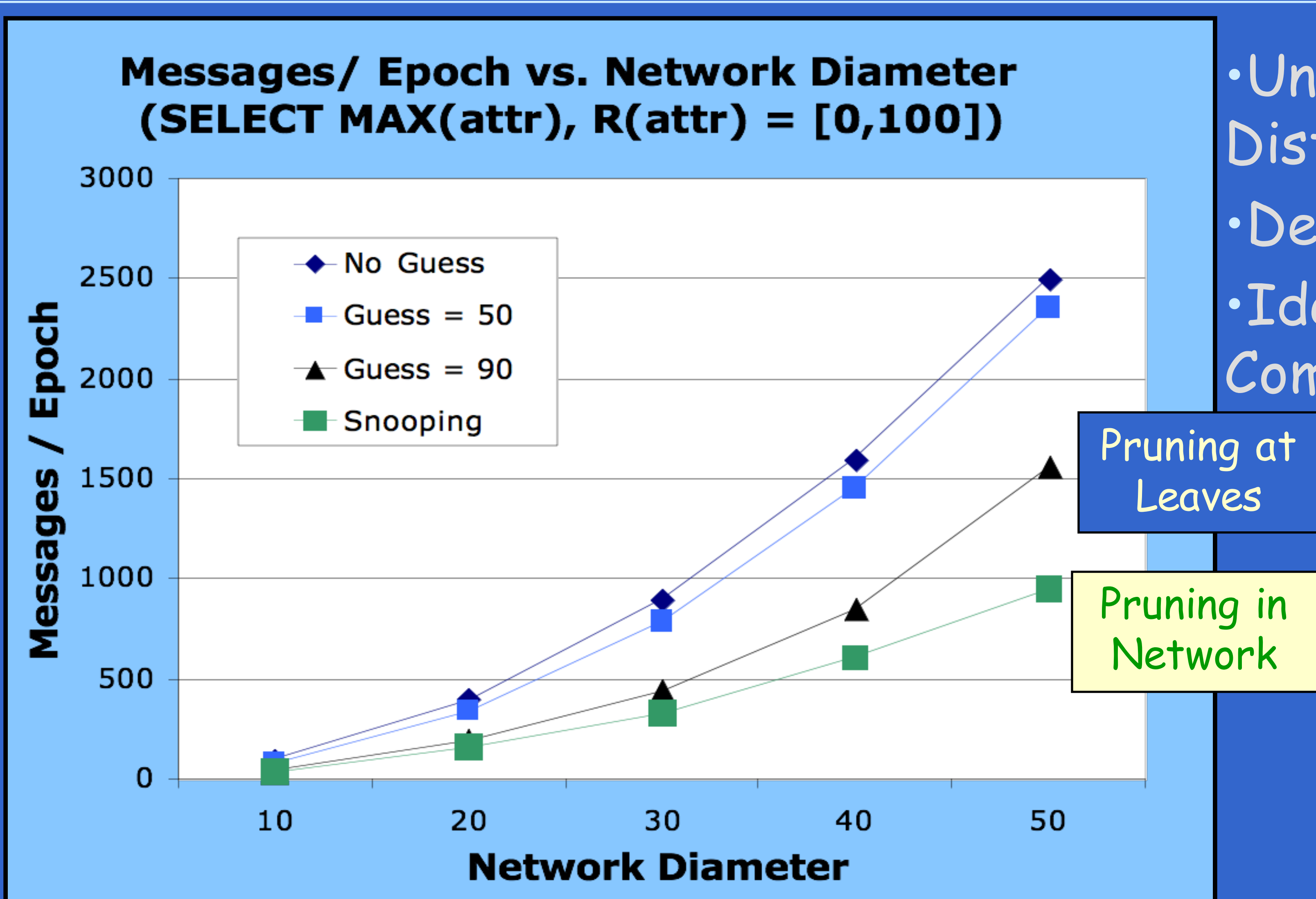
Channel Sharing ("Snooping")

- Insight: Shared channel can reduce communication
- Suppress messages that won't affect aggregate
  - E.g., MAX
  - Applies to all exemplary, monotonic aggregates
- Only snoop in listen/transmit slots
  - Future work: explore snooping/listening tradeoffs

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
- How is hypothesis computed?
  - Blind or statistically informed guess
  - Observation over network subset

Experiment: Snooping vs. Hypothesis Testing



- Uniform Value Distribution
- Dense Packing
- Ideal Communication

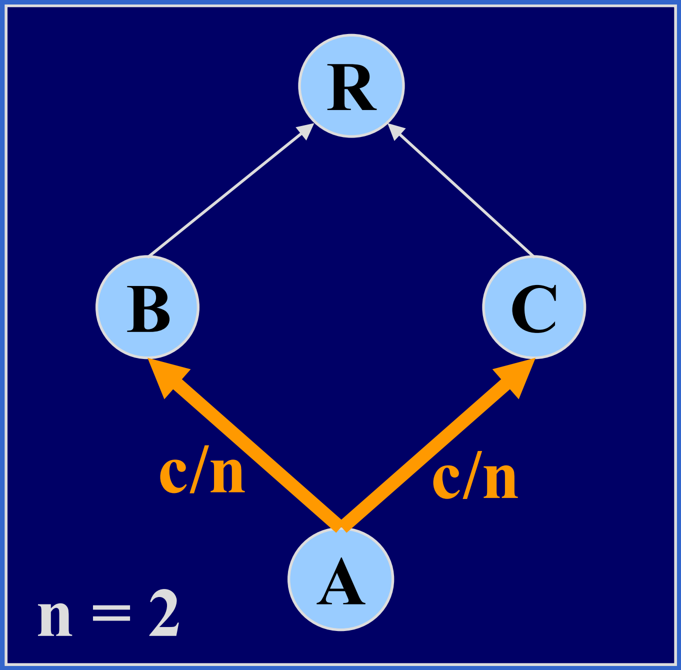
Duplicate Sensitivity

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

Use Multiple Parents

- Use graph structure
  - Increase delivery probability with no communication overhead
- For duplicate insensitive aggregates, or
- Aggs expressible as sum of parts
  - Send (part of) aggregate to all parents
    - In just one message, via multicast
  - Assuming independence, decreases variance

SELECT COUNT(*)



P(link xmit successful) = p

P(success from A->R) = p^2

E(cnt) = c * p^2

Var(cnt) = $c^2 * p^2 * (1 - p^2)$
≡ V

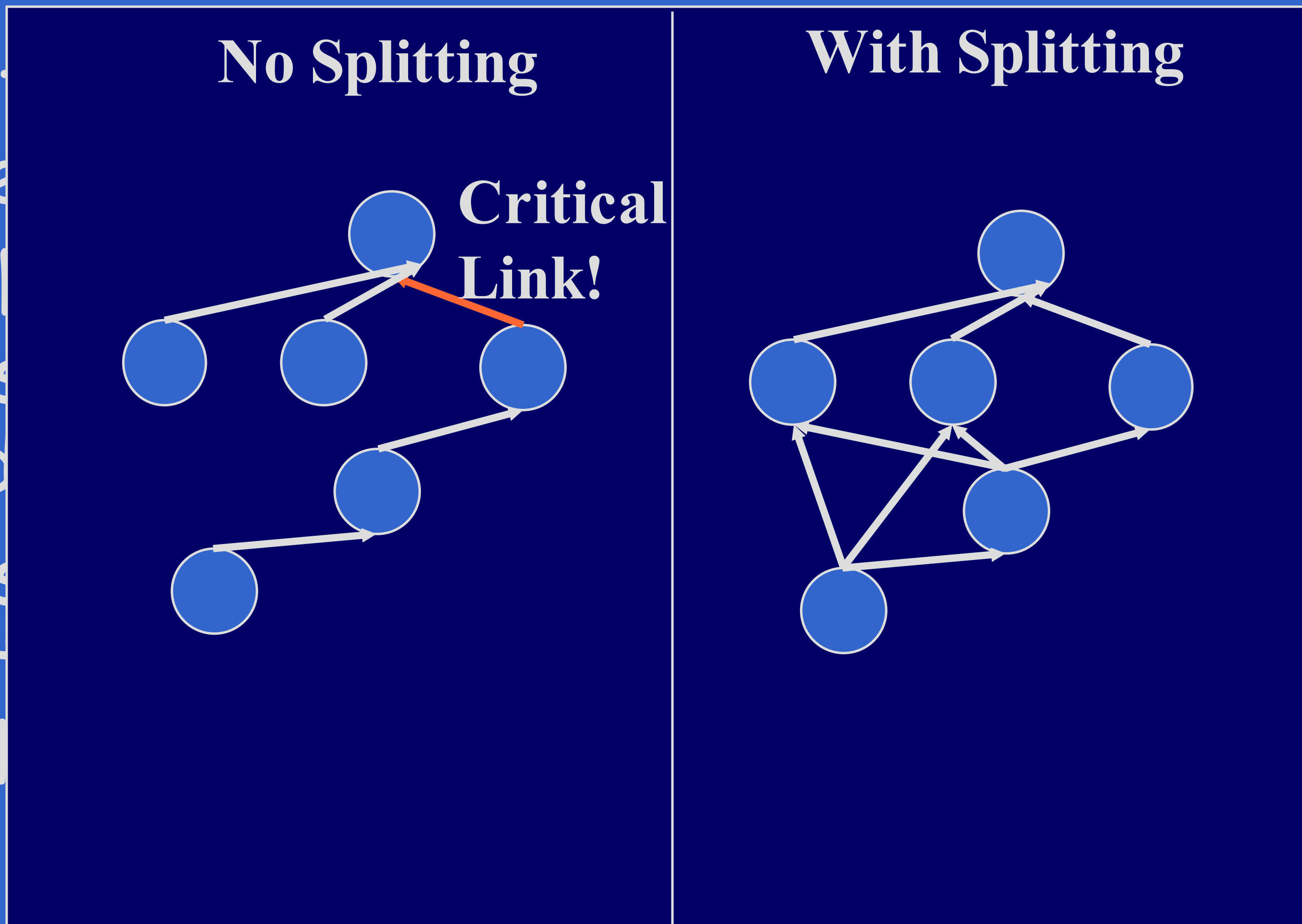
# of parents = n

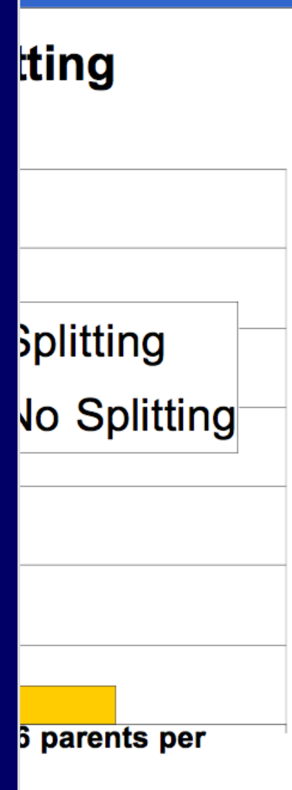
E(cnt) = n * (c/n * p^2)

Var(cnt) = n * $(c/n)^2 * p^2 * (1 - p^2)$ = V/n

Multiple Parents Results

- Be
  pre
  ex
- Los
  ind
- Ins
  da
  link





Taxonomy Related Insights

- Communication Reducing
  - In-network Aggregation (Partial State)
  - Hypothesis Testing (Exemplary & Monotonic)
  - Snooping (Exemplary & Monotonic)
  - Sampling

- Quality Increasing
  - Multiple Parents (Duplicate Insensitive)
  - Child Cache

TAG Contributions

- Simple but powerful data collection language
  - Vehicle tracking:

    SELECT ONEMAX(mag,nodeid)
    EPOCH DURATION 50ms

- Distributed algorithm for in-network aggregation
  - Communication Reducing
  - Power Aware
    - » Integration of sleeping, computation
  - Predicate-based grouping
- Taxonomy driven API
  - Enables transparent application of techniques to
    - » Improve quality (parent splitting)
    - » Reduce communication (snooping, hypo. testing)

Overview

- TinyDB: Queries for Sensor Nets
- Processing Aggregate Queries (TAG)
- Taxonomy & Experiments
- Acquisitional Query Processing
- Other Research
- Future Directions

Acquisitional Query Processing (ACQP)

- Closed world assumption does not hold
  - Could generate an infinite number of samples

- An acqusitional query processor controls
  - when,
  - where,
  - and with what frequency data is collected!
- Versus traditional systems where data is provided *a priori*

Madden, Franklin, Hellerstein, and Hong. *The Design of An Acqusitional Query Processor.* SIGMOD, 2003 (to appear).

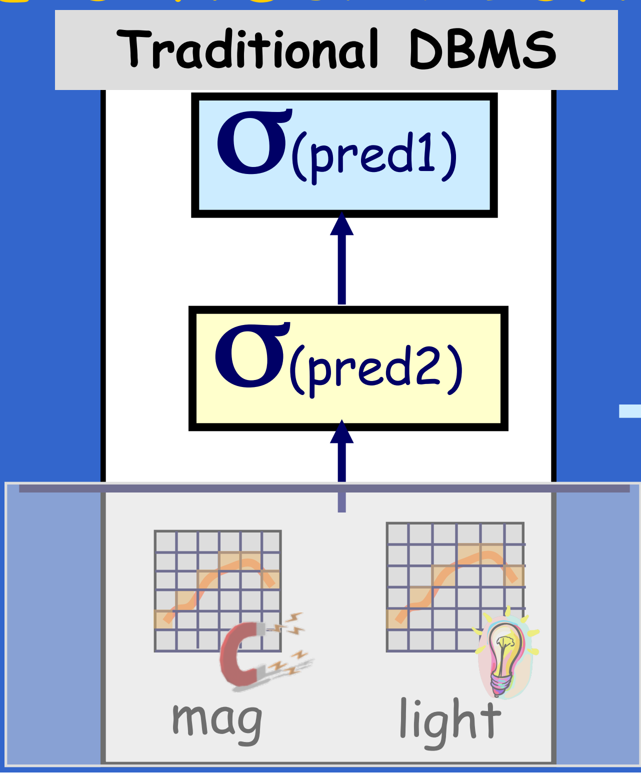
ACQP: What's Different?

- How should the query be processed?
  - Sampling as a first class operation
  - Event – join duality
- How does the user control acquisition?
  - Rates or lifetimes
  - Event-based triggers
- Which nodes have relevant data?
  - Index-like data structures
- Which samples should be transmitted?
  - Prioritization, summary, and rate control

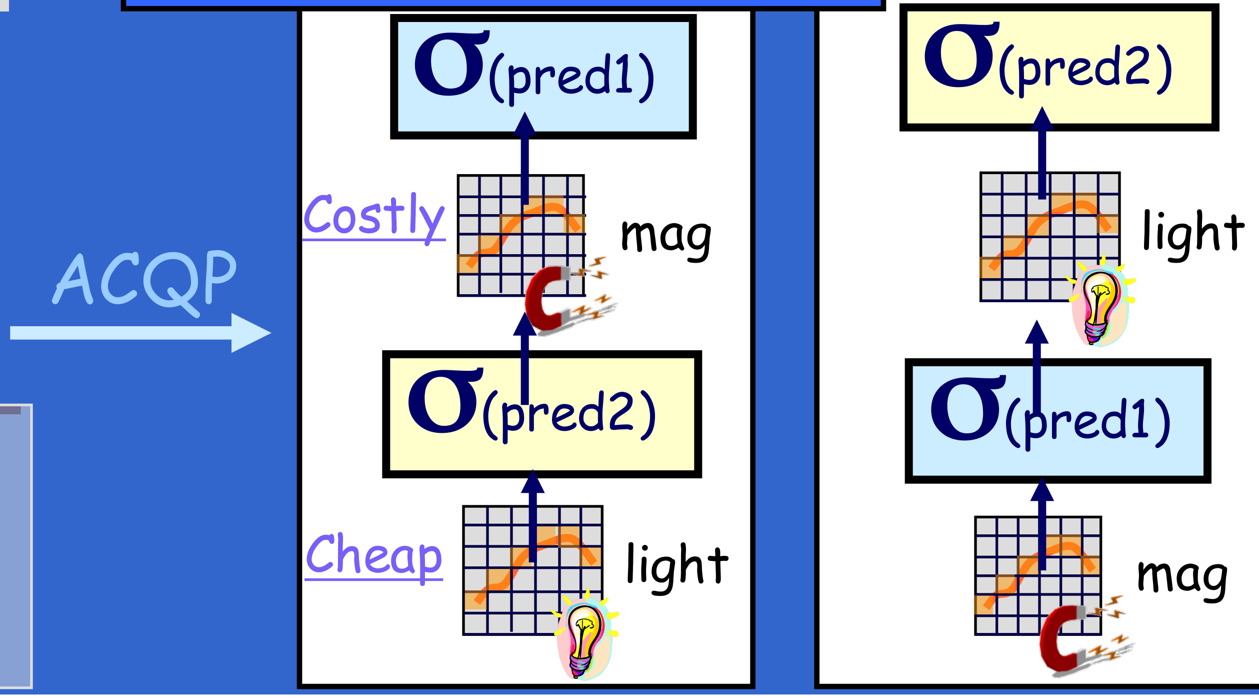
Operator Ordering: Interleave Sampling + Selection

SELECT light, mag
FROM sensors
WHERE pred1(mag)
AND pred2(light)
EPOCH DURATION 1s

At 1 sample / sec, total power savings could be as much as 3.5mW → Comparable to processor!

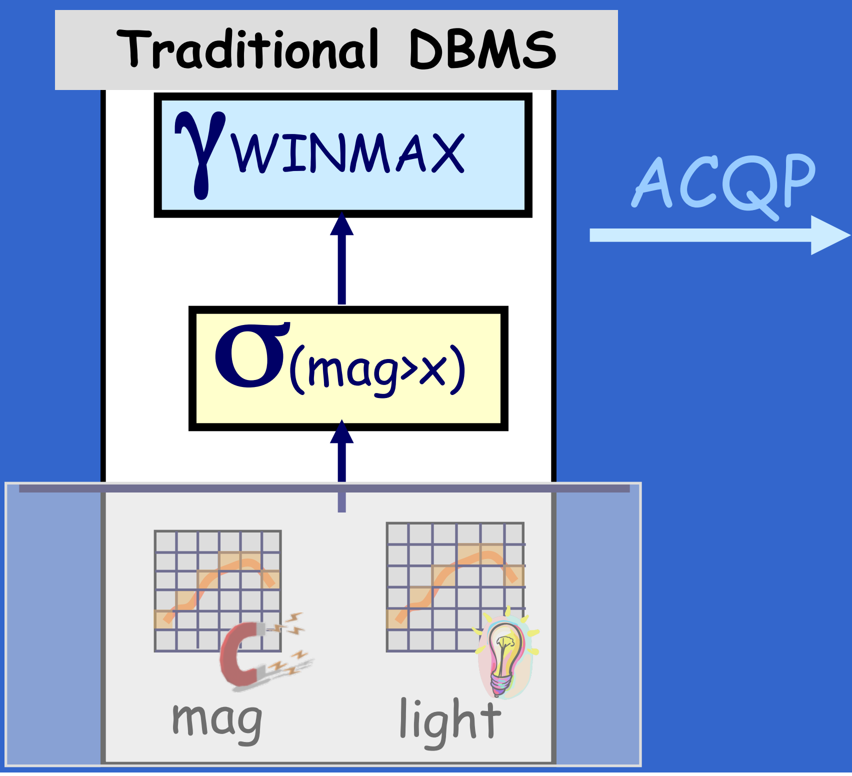


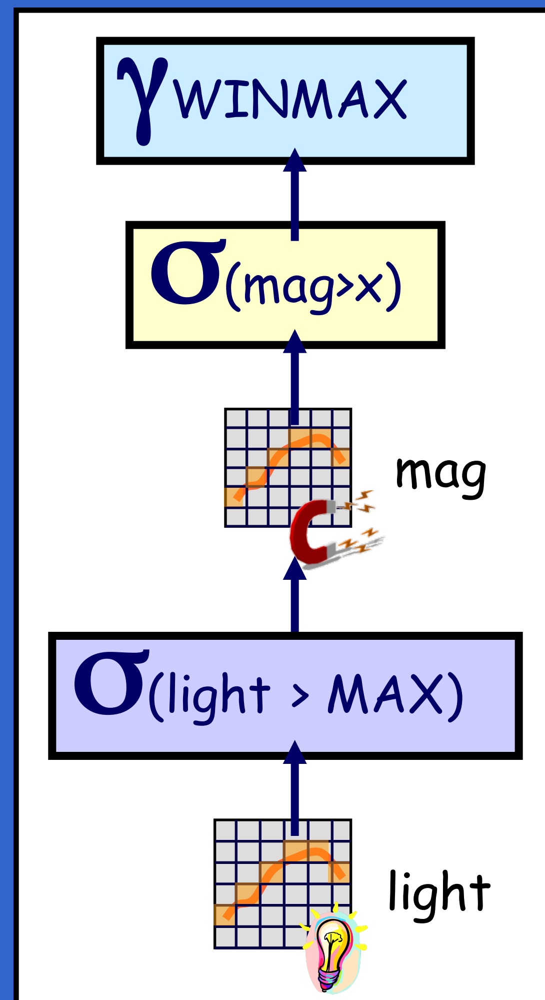
Correct ordering (unless pred1 is *very* selective and pred2 is not):



Exemplary Aggregate Pushdown

SELECT WINMAX(light,8s,8s)
FROM sensors
WHERE mag > x
EPOCH DURATION 1s





- Novel, general pushdown technique
- Mag sampling is the most expensive operation!

Lifetime Queries

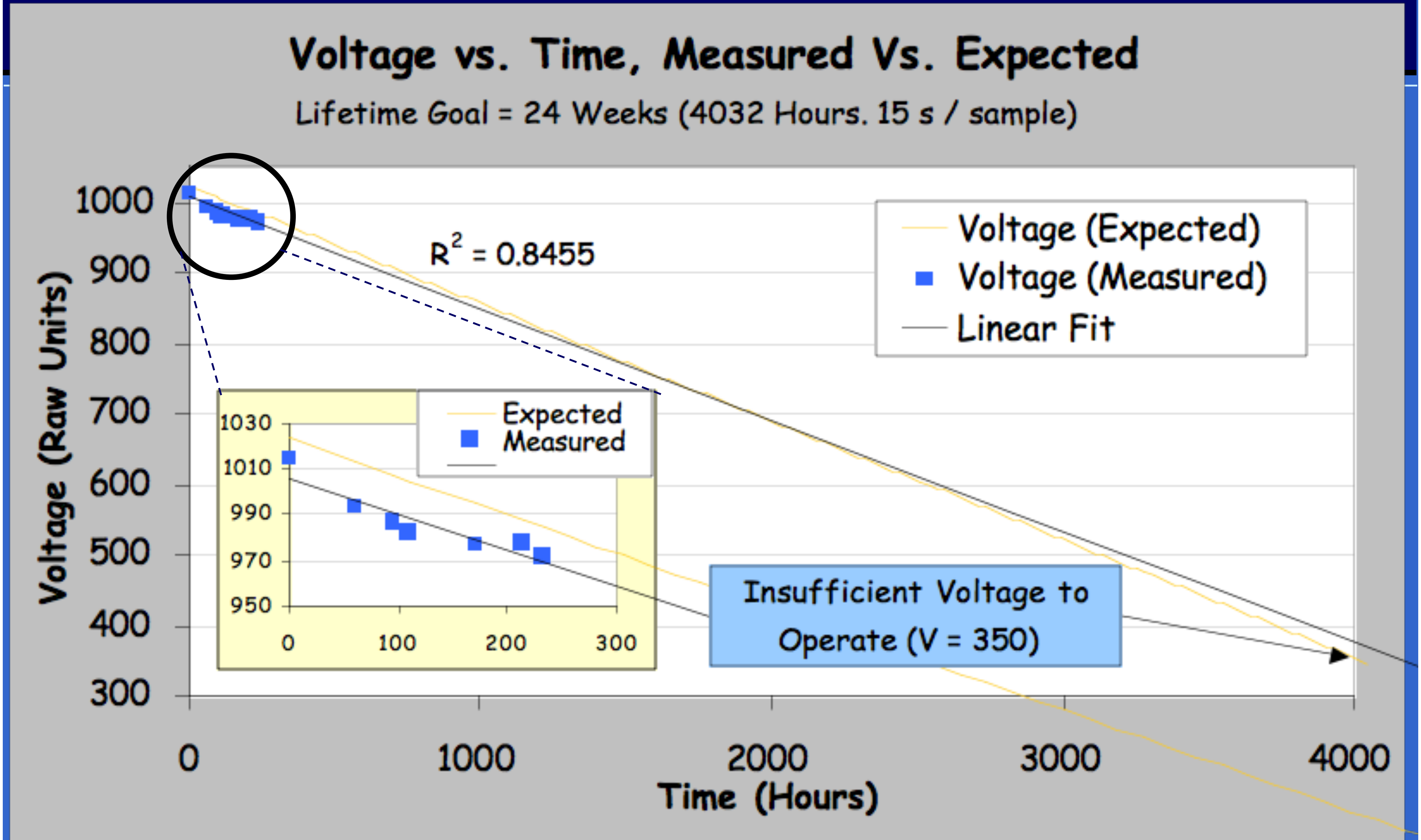
- Lifetime vs. sample rate

```
SELECT …
EPOCH DURATION 10 s

SELECT …
LIFETIME 30 days
```

- Extra: Allow a MAX SAMPLE PERIOD
  - Discard some samples
  - Sampling cheaper than transmitting

(Single Node) Lifetime Prediction

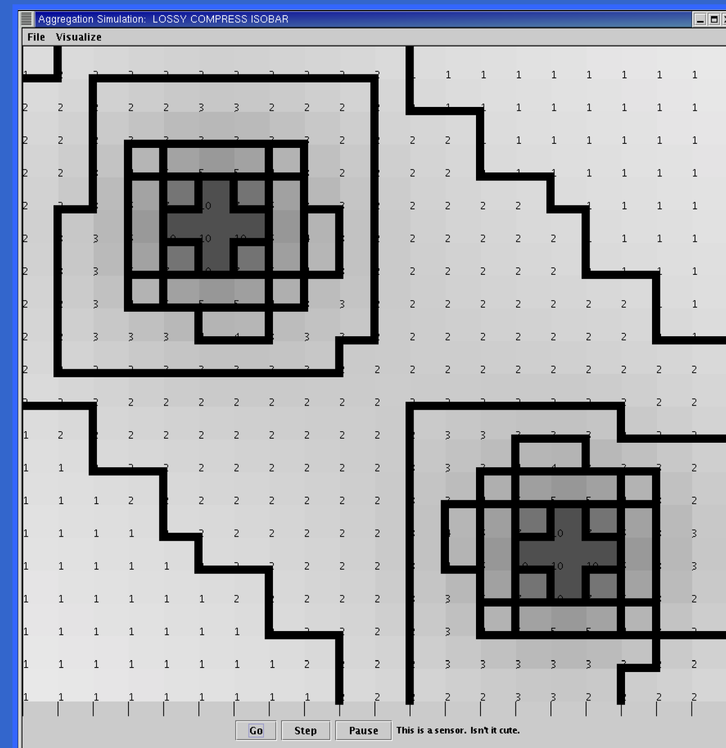


Overview

- TinyDB: Queries for Sensor Nets
- Processing Aggregate Queries (TAG)
- Taxonomy & Experiments
- Acquisitional Query Processing
- Other Research
- Future Directions

Sensor Network Challenge Problems

- Temporal aggregates
- Sophisticated, sensor network specific aggregates
  - Isobar Finding
  - Vehicle Tracking
  - Lossy compression
    - » Wavelets



"Isobar Finding"

Hellerstein, Hong, Madden, and Stanek. *Beyond Average*. IPSN 2003 (to appear)

Additional Research

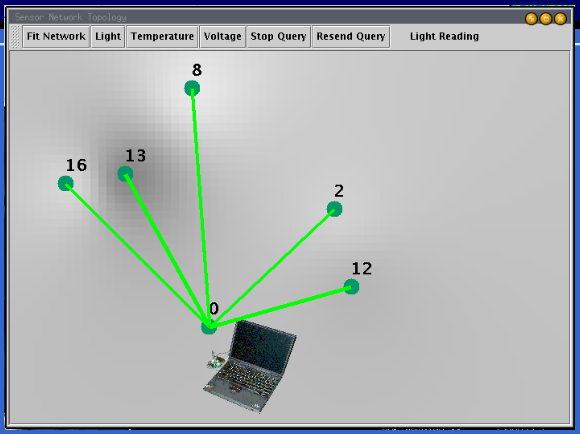
- Sensors, TinyDB, TinyOS
  - This Talk:
    - » TAG (OSDI 2002)
    - » ACQP (SIGMOD 2003)
    - » WMCSA 2002
    - » IPSN 2003
  - TOSSIM. Levis, Lee, Woo, Madden, & Culler. (In submission)
  - TinyOS contributions: memory allocator, catalog, network reprogramming, OS support, releases, TinyDB

Other Research (Cont)

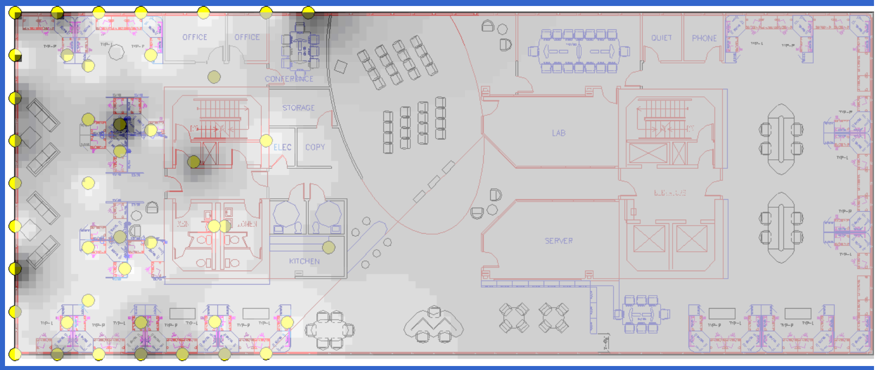
- Stream Query Processing
  - CACQ (SIGMOD 2002)
    - » Madden, Shah, Hellerstein, & Raman
  - Fjords (ICDE 2002)
    - » Madden & Franklin
  - Java Experiences Paper (SIGMOD Record, December 2001)
    - » Shah, Madden, Franklin, and Hellerstein
  - Telegraph Project, FFF & ACM1 Demos
    - » Telegraph Team

TinyDB Deployments

- Initial efforts:
  - Network monitoring
  - Vehicle tracking



- Ongoing deployments:
  - Environmental monitoring
  - Generic Sensor Kit
  - Building Monitoring
  - Golden Gate Bridge

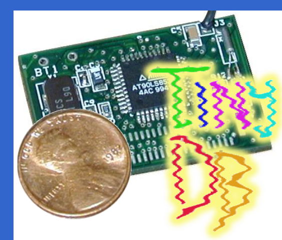


Overview

- TinyDB: Queries for Sensor Nets
- Processing Aggregate Queries (TAG)
- Taxonomy & Experiments
- Acquisitional Query Processing
- Other Research
- Future Directions

TinyDB Future Directions

- Expressing lossiness
  - No longer a closed world!
- Additional Operations
  - Joins
  - Signal Processing



- Integration with Streaming DBMS
  - In-network vs. external operations
- Heterogeneous Nodes and Operators
- Real Deployments

Contributions & Summary

- Declarative Queries via TinyDB
  - Simple, data-centric programming abstraction
  - Known to work for monitoring, tracking, mapping
- Sensor network contributions
  - Network as a single queryable entity
  - Power-aware, in-network query processing
  - Taxonomy: Extensible aggregate optimizations
- Query processing contributions
  - Acquisitional Query Processing
  - Framework for new issues in *acquisitional systems*, e.g.:
    - » Sampling as an operator
    - » Languages, indices, approximations to control

  when, where, and what data is acquired + processed by the system

http://telegraph.cs.berkeley.edu/tinydb

Questions?