15-744: Computer Networking

L-13 Sensor Networks

Sensor Networks

- Directed Diffusion
- Aggregation
- Assigned reading
  - TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks
  - Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks

Outline

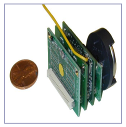
- Sensor Networks
- Directed Diffusion
- TAG
- Synopsis Diffusion

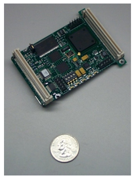
Smart-Dust/Motes

- First introduced in late 90's by groups at UCB/UCLA/USC
  - Published at Mobicom/SOSP conferences
- Small, resource limited devices
  - CPU, disk, power, bandwidth, etc.
- Simple scalar sensors – temperature, motion
- Single domain of deployment (e.g. farm, battlefield, etc.) for a targeted task (find the tanks)
- Ad-hoc wireless network

Smart-Dust/Motes

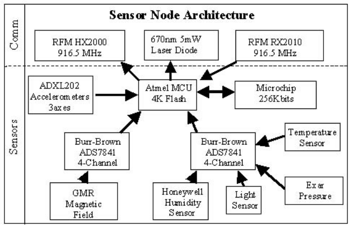
- Hardware
  - UCB motes
- Programming
  - TinyOS
- Query processing
  - TinyDB
  - Directed diffusion
  - Geographic hash tables
- Power management
  - MAC protocols
  - Adaptive topologies

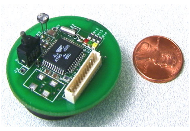




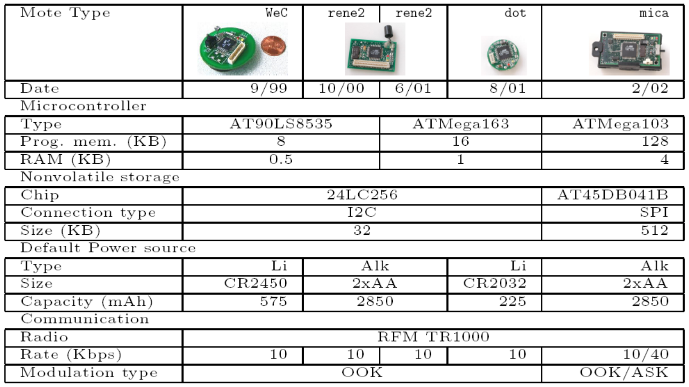
Berkeley Motes

- Devices that incorporate communications, processing, sensors, and batteries into a small package
- Atmel microcontroller with sensors and a communication unit
  - RF transceiver, laser module, or a corner cube reflector
  - Temperature, light, humidity, pressure, 3 axis magnetometers, 3 axis accelerometers





Berkeley Motes (Levis & Culler, ASPLOS 02)



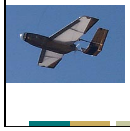
Mote Type	WeC	rene2	rene2	dot	mica
Date	9/99	10/00	6/01	8/01	2/02
Microcontroller					
Type	AT90LS8535		ATMega163		ATMega103
Prog. mem. (KB)	8		16		128
RAM (KB)	0.5		1		4
Nonvolatile storage					
Chip	24LC256				AT45DB041B
Connection type	I2C				SPI
Size (KB)	32				512
Default Power source					
Type	Li	Alk		Li	Alk
Size	CR2450	2xAA		CR2032	2xAA
Capacity (mAh)	575	2850		225	2850
Communication					
Radio	RFM TR1000				
Rate (Kbps)	10	10	10	10	10/40
Modulation type	OOK				OOK/ASK

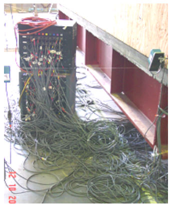
Sensor Net Sample Apps

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.

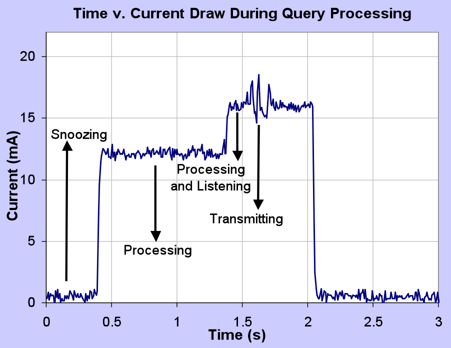




Traditional monitoring apparatus.

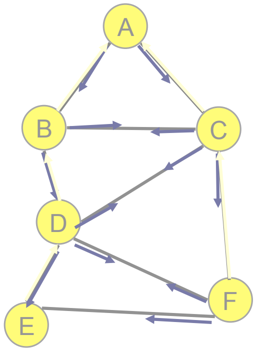
Metric: Communication

- 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



Communication In Sensor Nets

- Radio communication has high link-level losses
  - typically about 20% @ 5m
- Ad-hoc neighbor discovery
- Tree-based routing

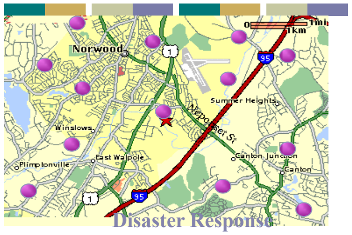


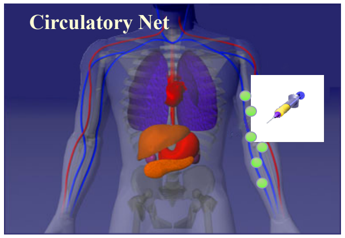
Outline

- Sensor Networks
- Directed Diffusion
- TAG
- Synopsis Diffusion

The long term goal

Embed numerous distributed devices to monitor and interact with physical world: in work-spaces, hospitals, homes, vehicles, and "the environment" (water, soil, air…)





Network these devices so that they can coordinate to perform higher-level tasks.

Requires robust distributed systems of tens of thousands of devices.

Motivation

- Properties of Sensor Networks
  - Data centric, but not node centric
  - Have no notion of central authority
  - Are often resource constrained
- Nodes are tied to physical locations, but:
  - They may not know the topology
  - They may fail or move arbitrarily
- Problem: How can we get data from the sensors?

Directed Diffusion

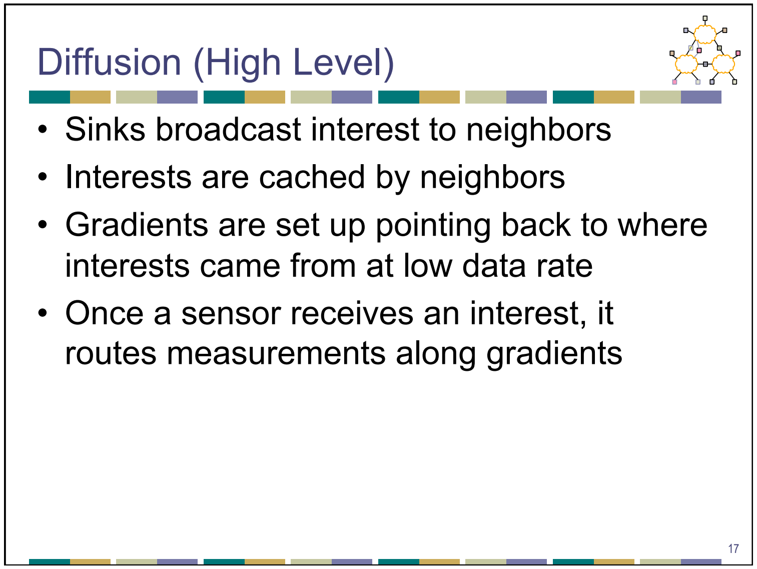
- Data centric – nodes are unimportant
- Request driven:
  - Sinks place requests as interests
  - Sources are eventually found and satisfy interests
  - Intermediate nodes route data toward sinks
- Localized repair and reinforcement
- Multi-path delivery for multiple sources, sinks, and queries

Motivating Example

- Sensor nodes are monitoring a flat space for animals
- We are interested in receiving data for all 4-legged creatures seen in a rectangle
- We want to specify the data rate

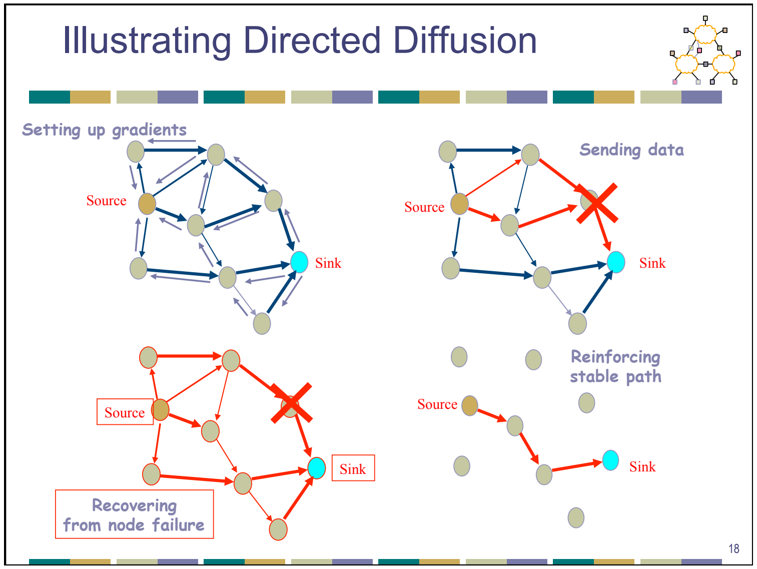
Interest and Event Naming

- Query/interest:
  1. Type=four-legged animal
  2. Interval=20ms (event data rate)
  3. Duration=10 seconds (time to cache)
  4. Rect=[-100, 100, 200, 400]
- Reply:
  1. Type=four-legged animal
  2. Instance = elephant
  3. Location = [125, 220]
  4. Intensity = 0.6
  5. Confidence = 0.85
  6. Timestamp = 01:20:40
- Attribute-Value pairs, no advanced naming scheme

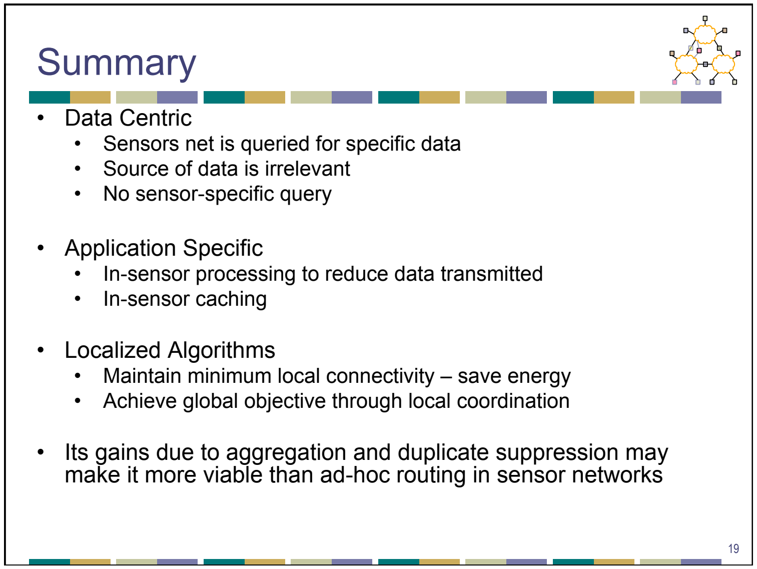


Diffusion (High Level)

- Sinks broadcast interest to neighbors
- Interests are cached by neighbors
- Gradients are set up pointing back to where interests came from at low data rate
- Once a sensor receives an interest, it routes measurements along gradients

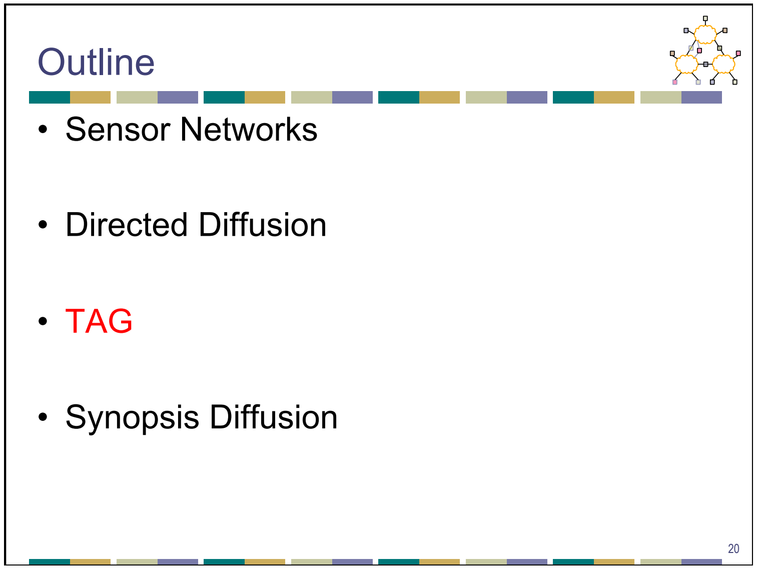


Illustrating Directed Diffusion



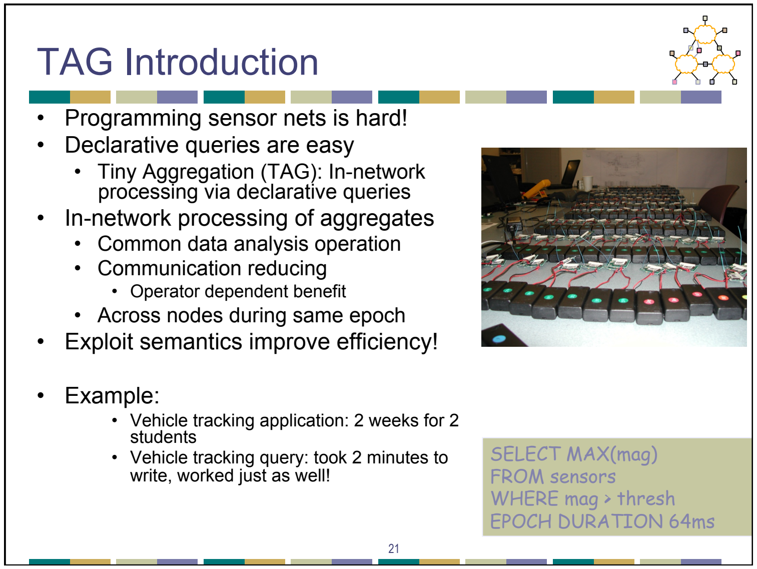
Summary

- Data Centric
  - Sensors net is queried for specific data
  - Source of data is irrelevant
  - No sensor-specific query
- Application Specific
  - In-sensor processing to reduce data transmitted
  - In-sensor caching
- Localized Algorithms
  - Maintain minimum local connectivity – save energy
  - Achieve global objective through local coordination
- Its gains due to aggregation and duplicate suppression may make it more viable than ad-hoc routing in sensor networks



Outline

- Sensor Networks
- Directed Diffusion
- TAG
- Synopsis Diffusion

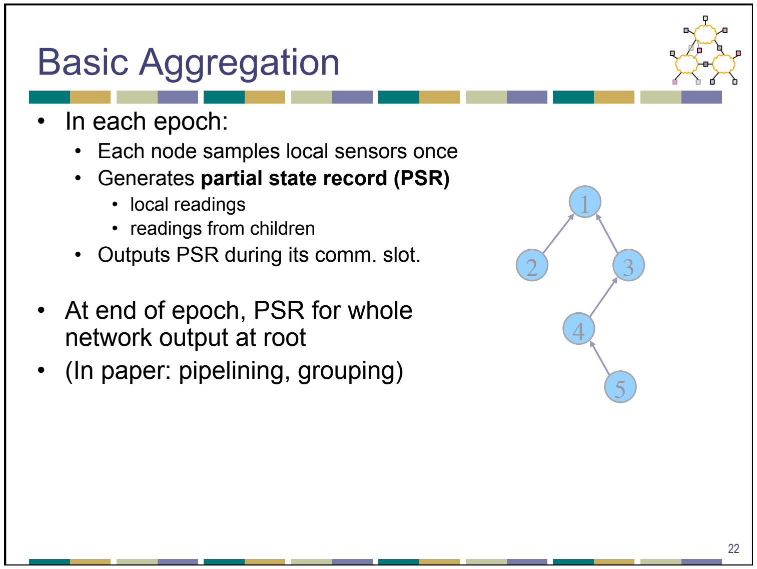


TAG Introduction

- Programming sensor nets is hard!
- Declarative queries are easy
  - Tiny Aggregation (TAG): In-network processing via declarative queries
- In-network processing of aggregates
  - Common data analysis operation
  - Communication reducing
    - Operator dependent benefit
  - Across nodes during same epoch
- Exploit semantics improve efficiency!

- 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
```



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)

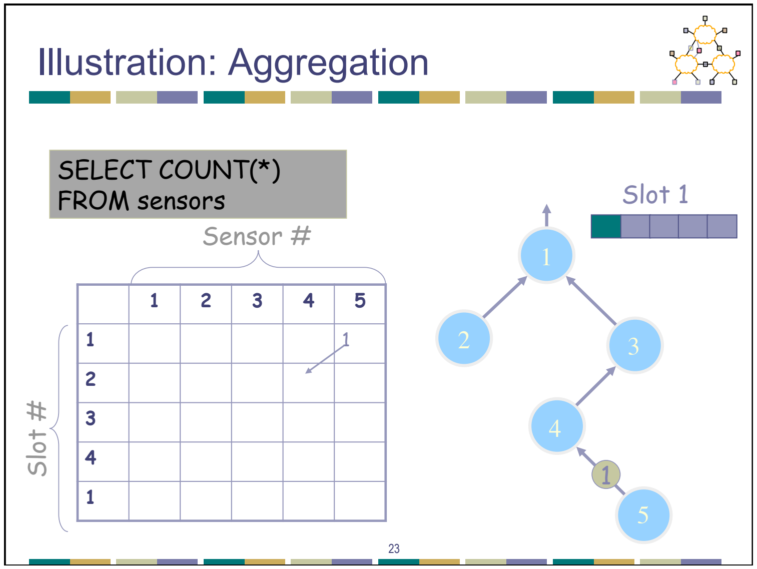


Illustration: Aggregation

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

Sensor #

Slot #

	1	2	3	4	5
1					1
2					
3					
4					
1					

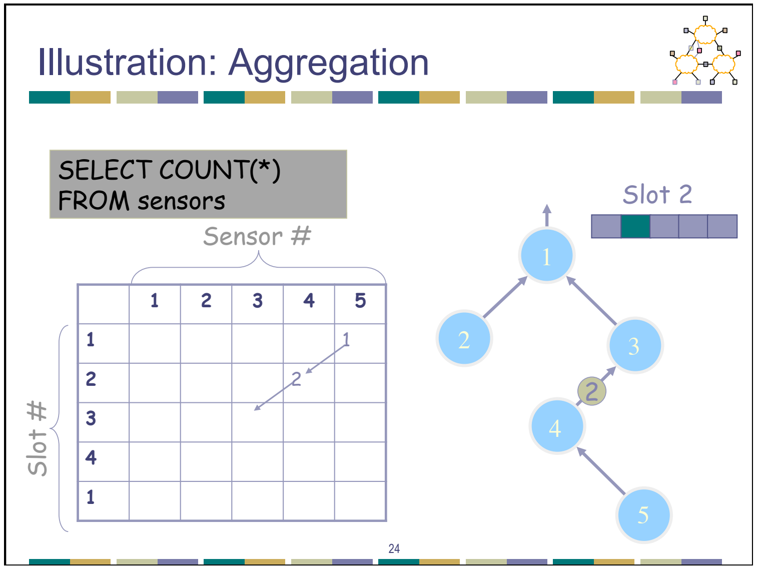


Illustration: Aggregation

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

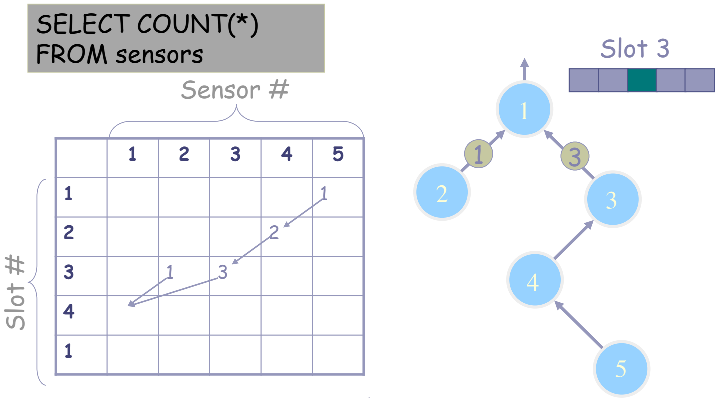
Sensor #

Slot #

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

Illustration: Aggregation

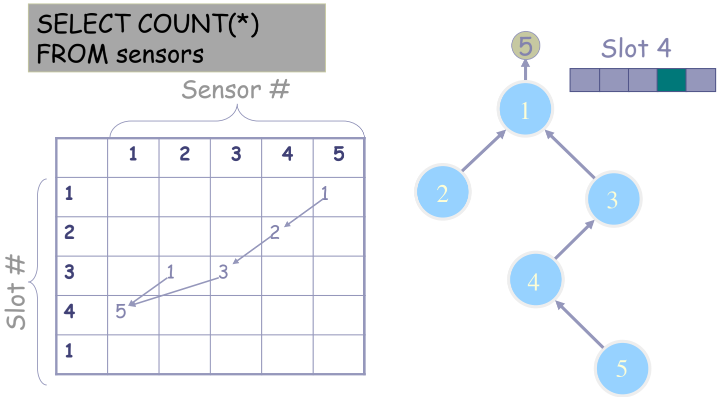
```
SELECT COUNT(*)
FROM sensors
```



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

Illustration: Aggregation

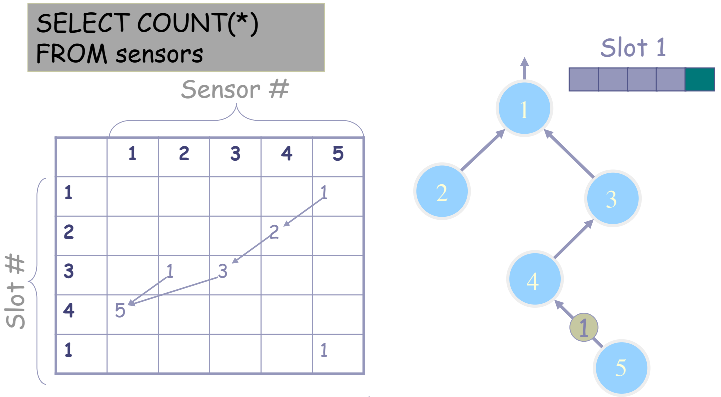
```
SELECT COUNT(*)
FROM sensors
```



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

Illustration: Aggregation

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



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

Types of Aggregates

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

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	COUNT : monotonic AVG : non-monotonic	Hypothesis Testing, Snooping

Benefit of In-Network Processing

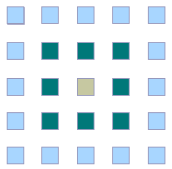
Simulation Results

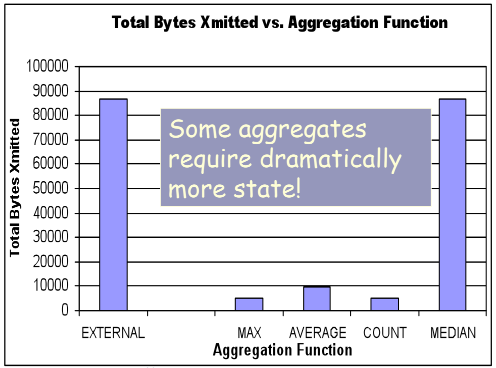
2500 Nodes

50x50 Grid

Depth = ~10

Neighbors = ~20





Some aggregates require dramatically more state!

Optimization: Channel Sharing (“Snooping”)

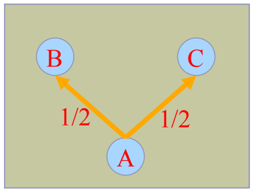
- Insight: Shared channel enables optimizations
- Suppress messages that won’t affect aggregate
  - E.g., MAX
  - Applies to all exemplary, monotonic aggregates

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

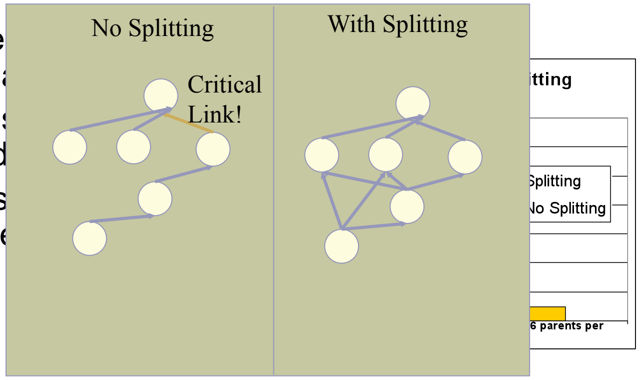
Optimization: Use Multiple Parents

- 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



Multiple Parents Results

- Be
ana
- Los
ind
- Ins
ove



Outline

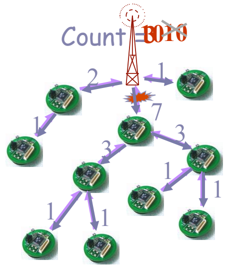
- Sensor Networks
- Directed Diffusion
- TAG
- Synopsis Diffusion

Aggregation in Wireless Sensors

Aggregate data is often more important

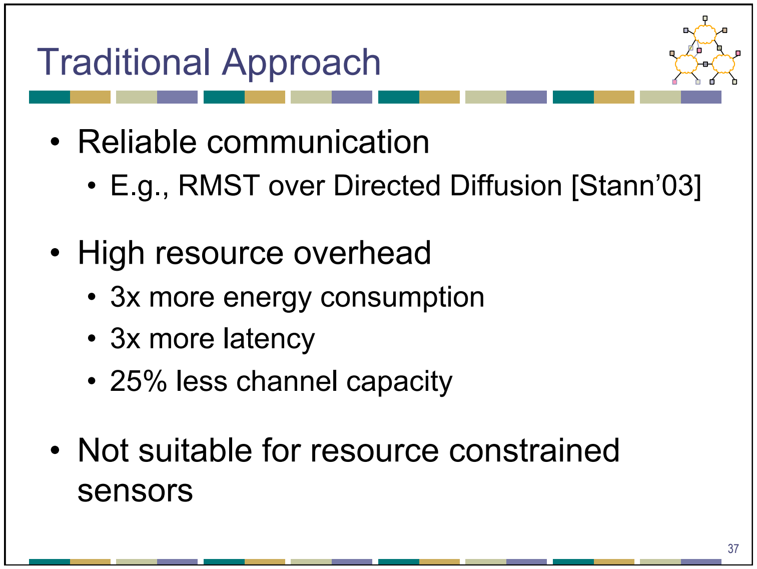
In-network aggregation

over tree with unreliable communication



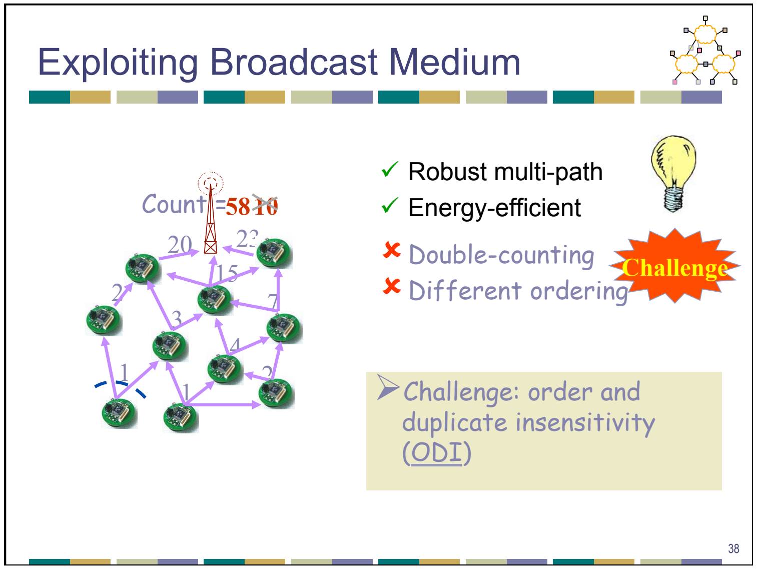
Used by current systems,
TinyDB [Madden et al. OSDI'02]
Cougar [Bonnet et al. MDM'01]

Not robust against
node- or link-failures



Traditional Approach

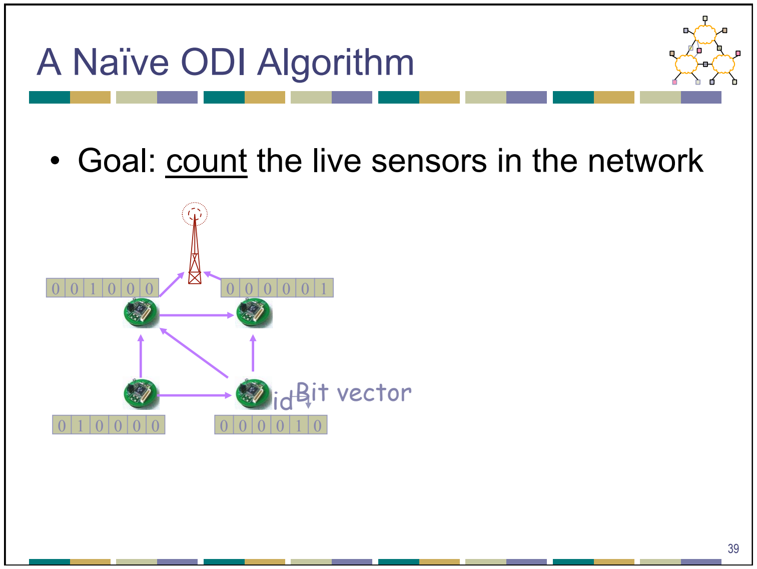
- Reliable communication
  - E.g., RMST over Directed Diffusion [Stann'03]
- High resource overhead
  - 3x more energy consumption
  - 3x more latency
  - 25% less channel capacity
- Not suitable for resource constrained sensors



Exploiting Broadcast Medium

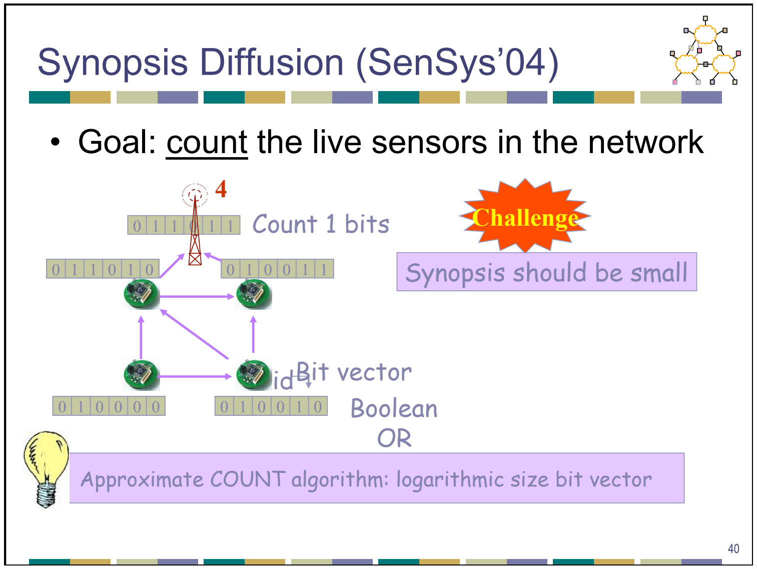
- ✓ Robust multi-path
- ✓ Energy-efficient
- ✗ Double-counting
- ✗ Different ordering

➢ Challenge: order and duplicate insensitivity (ODI)



A Naïve ODI Algorithm

- Goal: count the live sensors in the network



Synopsis Diffusion (SenSys'04)

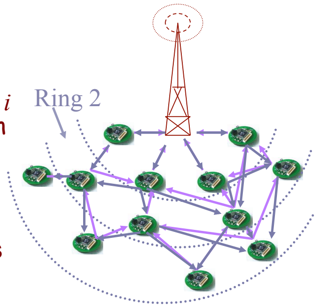
- Goal: count the live sensors in the network

Synopsis should be small

Approximate COUNT algorithm: logarithmic size bit vector

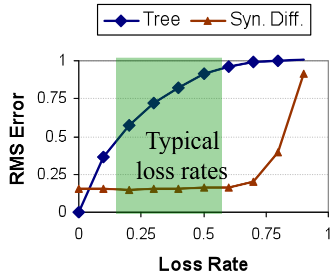
Synopsis Diffusion over Rings

- A node is in ring *i* if it is *i* hops away from the base-station
- Broadcasts by nodes in ring *i* are received by neighbors in ring *i*-1
- Each node transmits once = optimal energy cost (same as Tree)



Evaluation

Approximate COUNT with Synopsis Diffusion



Scheme	Energy
Tree	41.8 mJ
Syn. Diff.	42.1 mJ

Per node energy

More robust than Tree

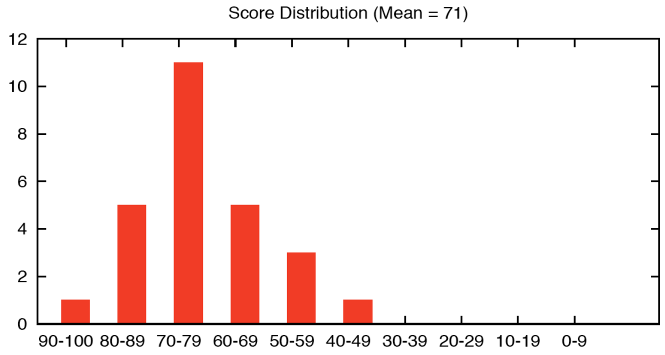
Almost as energy efficient as Tree

Next Lecture

- No lecture on Wednesday
- Data center networks on Friday
- Required readings
  - PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric [Sigcomm09]
  - Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication [Sigcomm09]

Midterm

- Below 50 – cause for concern



Particular Issues

- A – 26/40
- B – 4/13
- C – 11.5/15
- D – 13/14
- E – 15/17

Feedback (more of)

- Routing, Security (2), Wireless, Supercomputer, Machine learning
- Discussion boards/paper review (2)
- Student presentation (2)
- Historic/modern mix
- Hands-on questions in HW, exam practice, add references (4)
- In-class demos
- Video (4)

Feedback (change)

- Early overview (lecture, or student) (2)
- Student presentation feed redundant, too long, not useful (4)
- More discussion in student presentation
- Forum interaction
- Project help (11/3 and 11/4)
- Slide printouts in advance (will post by noon)
- Optional readings – clarify who they are for
- Project status page
- More HW/proj and less exam (2)
- Background recitations

Design Considerations

Diffusion element	Design Choices
Interest Propagation	• Flooding • Constrained or directional flooding based on location • Directional propagation based on previously cached data
Data Propagation	• Reinforcement to single path delivery • Multipath delivery with selective quality along different paths • Multipath delivery with probabilistic forwarding
Data caching and aggregation	• For robust data delivery in the face of node failure • For coordinated sensing and data reduction • For directing interests
Reinforcement	• Rules for deciding when to reinforce • Rules for how many neighbors to reinforce • Negative reinforcement mechanisms and rules

Figure 3: Design Space for Diffusion

Directed Diffusion (Data)

- Sensors match signature waveforms from codebook against observations
- Sensors match data against interest cache, compute highest event rate request from all gradients, and (re) sample events at this rate
- Receiving node:
  - Finds matching entry in interest cache, no match – silent drop
  - Checks and updates data cache (loop prevention, aggregation)
  - Retrieve all gradients, and resend message, doing frequency conversion if necessary

Directed Diffusion (Reinforcement)

- Reinforcement:
  - Data-driven rules unseen msg. from neighbor → resend original with smaller interval
  - This neighbor, in turn, reinforces upstream nodes
  - Passive reinforcement handling (timeout) or active (weights)

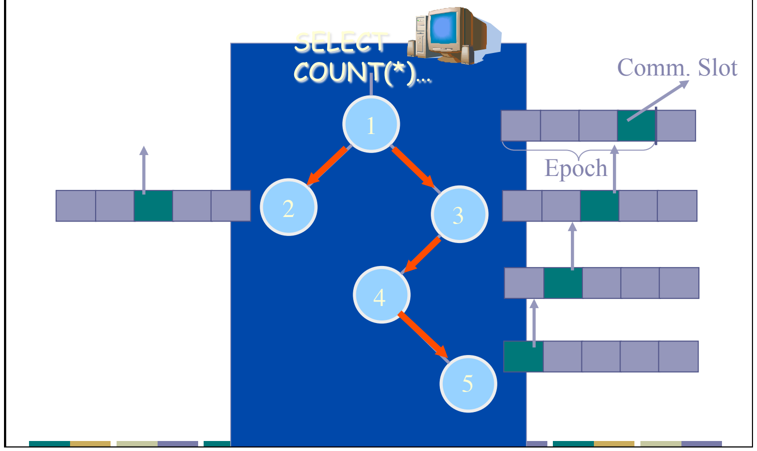
Approach

- Energy is the bottleneck resource
  - And communication is a major consumer--avoid communication over long distances
- Pre-configuration and global knowledge are not applicable
  - Achieve desired global behavior through localized interactions
  - Empirically adapt to observed environment
- Leverage points
  - Small-form-factor nodes, densely distributed to achieve Physical locality to sensed phenomena
  - Application-specific, data-centric networks
  - Data processing/aggregation inside the network

Directed Diffusion Concepts

- Application-aware communication primitives
  - expressed in terms of named data *(not in terms of the nodes generating or requesting data)*
- Consumer of data initiates interest in data with certain attributes
- Nodes diffuse the interest towards producers via a sequence of local interactions
- This process sets up gradients in the network which channel the delivery of data
- Reinforcement and negative reinforcement used to converge to efficient distribution
- Intermediate nodes opportunistically fuse interests, aggregate, correlate or cache data

Query Propagation



Synopsis Diffusion (SenSys'04)

- Synopsis Diffusion: a general framework

Synopsis Diffusion algorithms ⇒

Count	Uniform Sample
Sum (Average)	Iceberg queries
Distinct count	Top-k items

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**

(Merge associative, commutative!)

Example: Average

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

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

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