Directed Diffusion
Query Forwarding and Aggregation in an Ideal Sensor Network

- **Query forwarding**
  - Query gets routed exactly to nodes that have required information.

- **Aggregation**
  - Nodes process data as much as they can
  - Routing structures enable optimal aggregation of spatially distributed but correlated responses.
  - Intermediate nodes aggregate data maximally within the network.
TinyDB Review

- Query Language
  - SQL-based

- Networking Structure:
  - Tree-based routing

- Placement of Aggregation Operators
  - Very simple in-network aggregation at tree junctions. Most intelligence is outside the network

- Platform
  - Intended for motes

*TinyDB Philosophy: Restrict routing dynamics to the class of sink-rooted trees. Study problem of processing over trees and query language issues*
When is TinyDB insufficient?

- Applications that involve a distributed-signal processing component are difficult to represent in TinyDB.
- Example: Consider a DSP application, where a clusterhead pulls data from local nodes that detect an event and push the processed data to a data sink.
- TinyDB assumes a fixed tree, hence individual nodes are limited in capability.
Diffusion Overview

- **Query Language**
  - SQL-based
  - Custom predicates (rules that are matched)

- **Networking Structure:**
  - Tree-based routing
  - Multi-source, multi-sink (geographic, flooding, tree-based) routing using publish-subscribe

- **Placement of Aggregation Operators**
  - Most intelligence is outside the network
  - Keep intelligence within the network.

- **Platform**
  - Intended solely for motes
  - Different platform classes

*Diffusion Philosophy:* Consider the broad problem of correlated sources of data and multiple sinks. Construct application-specific aggregation structures on-the-fly.
Directed Diffusion [1997]

- Pre-cursor to Motes and TinyDB
  - Together with Cougar [Cornell], earliest attempts at data management in WSN

- Initial Goals:
  - Propose an application-aware paradigm to facilitate efficient aggregation, and delivery of sensed data to inquiring destination
The Problem

- A region requires event-monitoring (harmful gas, vehicle motion, seismic vibration, temperature, etc.)
- Deploy sensors forming a distributed network
- On event, sensed and/or processed information delivered to the inquiring destination
Where should the data be stored?

How should queries be routed to the stored data?

Where and how should aggregation be performed?

How should queries for sensor networks be expressed?
Directed Diffusion

- What does Diffusion offer?
  - Publish and subscribe
  - Named data
  - Diffusion protocols

slides borrowed from Deborah Estrin, John Heidemann, Fabio Silva, Eric Osterweil
Directed Diffusion

“What sharp temperature variations (delta T > 10 degrees in the course of 1 minute) have been observed in the southeast quadrant?”

- Robust, efficient data distribution in sensor networks
  - name data (not nodes), use physicality
  - diffuse requests and responses across network
  - optimize path with gradient-based feedback
  - additional data causes in-network aggregation
Information-Centric Applications

- Software and Antivirus Updates
- Consumer Alerts
- Location-Based Services for Mobile Wireless
- Multiplayer Online Games
- Web Search Engines
- e-Business (e.g., Supply Chain Mgmt)
- Distributed Sensor Networks

Publish/subscribe is a natural fit!
Publish/Subscribe Systems

- Push model (a.k.a. event-notification)
  - Producer publishes messages
  - Consumer waits for certain types of events by placing subscriptions
  - subscribe → publish → match

- 2 types
  - topic-based: ≈ Usenet newsgroup topics
  - content-based: attribute-value pairs
    - e.g. (attr1 = value1) ∧ (attr2 = value2) ∧ (attr3 > value3)
Example: Brokerage Trading Floor

- Publisher: Publishes keywords as property objects
- Subscriber: Subscribes and accepts stories
- Matching: Match based on Keyword Generator
- Monitors interprets & displays the property objects
Data Naming

- Expressing an Interest
  - Using attribute-value pairs
  - E.g.,
    - Type = Wheeled vehicle  // detect vehicle location
    - Interval = 20 ms      // send events every 20ms
    - Duration = 10 s      // Send for next 10 s
    - Field = [x1, y1, x2, y2]  // from sensors in this area

- Other interest-expressing schemes possible
  - E.g., hierarchical (different problem)
Directed Diffusion

- Typical IP based networks
  - Requires unique host ID addressing
  - Application is end-to-end, routers unaware

- Directed diffusion – uses publish/subscribe
  - Inquirer expresses an interest, $I$, using attribute values
  - Sensor sources that can service $I$, reply with data
Content-based Matching

- Keyword/Subject-based matching is restrictive since sensors generate continuous-valued data.
- Content based subscription systems more generic
  - Can deal with continuous-valued sensor data
  - Can deal with multiple sensors that correspond to an event.
- How to match?
  - Match each subscription to all publishers
  - Reduce the search space by including some easily searchable attributes (eg: geography)
  - Combine different subscriptions - (~Multi-query optimization): Hard problem
Pros and Cons of PubSub for sensor networks

- **Advantages:**
  - No need to know where sinks are relative to sources.
  - Dynamic Matching between sinks and sources:
    - Routing structure can be dynamically built between wherever sinks are located to wherever sources are located.
    - Data can be routed dynamically, according to its value

- **Difficulties:**
  - More complex problem than with a fixed routing structure (TinyDB)
  - Matching can be an expensive operation
Named Data in Directed Diffusion

- Data names are attributes
  - Key
  - Operation
  - Value
- Routes are known as Gradients
- Attributes (data) is matched along gradients as it flows through the network
Query language in Directed Diffusion vs TinyDB

- Different *specifics* but same general approach
- Declarative Language
  - TinyDB: SQL-based
  - Directed Diffusion: Rule-matching
- Naming data rather than nodes
- Routing support for data names
  - Directed Diffusion: Geographic, Flooding, Scoped, Tree-based, Multipath etc
  - TinyDB: Tree-based
Attributes Implementation

- Each attribute implemented as a key-type-value-operator tuple
- Key
  - Indicates the semantics of the attribute
    - Latitude, frequency, etc
- Operator
  - Describes how the attribute will match when two attributes with the same key and type are compared
  - Available operators are: IS, EQ, NE, GE, GT, LE, LT, EQ_ANY
  - IS operator specifies a known, actual value
  - Other operators specify a condition that must be satisfied
- Examples
  - LATITUDE IS 12.3
  - LONGITUDE LE 45.2
  - TEMPERATURE IS 72.7
  - CONFIDENCE GT 0.80
Matching Example

- Simple matching rules
  - Small code footprint
  - Not general purpose language
- Each **formal** must match some **actual**:

  type EQ detection  
  x GE -100  
  x LE 200  
  y GE 100  
  y LE 400  
  confidence GT 0.5

  type IS detection  
  x IS 10  
  y IS 150  
  confidence IS 0.7

  **valid match**
Matching Example (2)

Each formal must match some actual:

- **type EQ detection**
  - $x \geq -100$
  - $x \leq 200$
  - $y \geq 100$
  - $y \leq 400$
  - confidence $\gt 0.5$

- **type IS detection**
  - $x \ IS \ 10$
  - $y \ IS \ 10$
  - confidence IS $0.7$

**no match**
Matching Example (3)

Each formal must match *some actual*, both ways:

- **type EQ detection**
  - \( x \geq -100 \)
  - \( x \leq 200 \)
  - \( y \geq 100 \)
  - \( y \leq 400 \)
  - confidence \( \gt 0.5 \)

- **type IS detection**
  - \( x \text{ IS } 10 \)
  - \( y \text{ IS } 150 \)
  - confidence \( \lt 0.7 \)

*no match*
Filters

- Application-provided software modules
  - Allow applications to influence diffusion and data processing

- Uses of filters include
  - Routing
  - In-network aggregation
  - Collaborative signal processing
  - Caching
  - Debugging
  - Monitoring
Filters Details (1)

- Filters use only one-way matching
  - A message entering a node triggers a filter if attributes specified by the filter match the attributes in the message
  - Does not require matching in the other direction
- Allows filter to process data more generally

Filter attributes

- class EQ data
- confidence GE 0.5

Packet attributes

- class IS data
- type EQ detection
- x IS 10
- y IS 10
- confidence IS 0.7

Valid one-way match
Diffusion Protocols

- How are Gradients initiated
  - Interests

- How Gradients are established
  - Classic
  - Push
  - One Phase Pull
Diffusion family of Push-Pull protocols

- Diffusion family of protocols includes variants optimized for specific scenarios

- Basic (Mobicom 2000) Pull Diffusion
  - Uses Publish/Subscribe API where
    - Interests have global scope (are sent to the network)
    - Data has local scope (does not leave a node if there is no matching interest)
    - Local subscriptions (interests for interests) also possible with local scope
      - Allow application to be notified when there are sinks interested in receiving their data

- PUSH Diffusion
  - Uses Publish/Subscribe API, where
    - Interests have local scope (do not leave the sink node)
    - Data has global scope (is flooded to the network)
    - After rendezvous at a sink node, a positive reinforcement is sent on the reverse path, establishing a reinforced path
    - Local subscriptions not possible (as interest messages do not leave sinks)

- Other rendezvous techniques for intermediate push-pull schemes.
Diffusion Routing Protocols

- Simple Flooding
- GEAR
  - Interests (or Data messages) include geographic coordinates
  - Supports both regions and points for describing the intended geographic scope
  - Allows both Pull and Push semantics
    - Pull: Interests include geographic information, flooding is limited to specified geographic region
    - Push: Data contains geographic information, used to specify the geographic region in which we are looking for sinks

```plaintext
class IS data

class IS data
lat GE 10
lat EQ 10
lat LT 50
lon EQ 40.3
lon GT 10
lon LE 70

Specifies a point  Specifies a region
```
Gradient-based Routing

- Inquirer (sink) broadcasts exploratory interest, $i1$
  - Intended to discover routes between source and sink

- Neighbors update interest-cache and forwards $i1$

- Gradient for $i1$ set up to upstream neighbor
  - No source routes
  - Gradient – a weighted reverse link
  - Low gradient $\rightarrow$ Few packets per unit time needed
Exploratory Gradient

Bidirectional gradients established on all links through flooding
Event-data propagation

- Event $e_1$ occurs, matches $i_1$ in sensor cache
  - $e_1$ identified based on waveform pattern matching

- Interest reply diffused down gradient (unicast)
  - Diffusion initially exploratory (low packet-rate)

- Cache filters suppress previously seen data
  - Problem of bidirectional gradient avoided
Reinforcement

- From exploratory gradients, reinforce optimal path for high-rate data download → **Unicast**
  - By requesting higher-rate-\(i_1\) on the optimal path
  - Exploratory gradients still exist – useful for faults
Path Failure / Recovery

- Link failure detected by reduced rate, data loss
  - Choose next best link (i.e., compare links based on infrequent exploratory downloads)
- Negatively reinforce lossy link
  - Either send $i_1$ with base (exploratory) data rate
  - Or, allow neighbor’s cache to expire over time

![Diagram showing event flow and link lossiness](Image)
M gets same data from both D and P, but P always delivers late due to looping

- M negatively-reinforces (nr) P, P nr Q, Q nr M
- Loop \{M \rightarrow Q \rightarrow P\} eliminated

- Conservative nr useful for fault resilience
Conclusion

- Directed diffusion, a paradigm proposed for event monitoring sensor networks
- Energy efficiency achievable
- Diffusion mechanism resilient to fault tolerance
  - Conservative negative reinforcements proves useful

- A careful MAC protocol, designed for such specifics, can yield further performance gains
Nested Queries

a)

b)
Contribution

- Application-awareness – a beneficial tradeoff
  - Data aggregation can improve energy efficiency
  - Better bandwidth utilization

- Network addressing is data centric
  - Probably correct approach for sensor type applications

- Notion of gradient (exploratory and reinforced)
  - Flexible architecture – enables configuration based on application requirements, tradeoffs

- Implementation on Berkeley motes
  - Network API, Filter API