GeoSensor Network Querying

Dina Q Goldin
CS Dept., Brown University

November 27, 2007

The Disappearing Computer

- More and more processors are not on desktops
- Processors in cars, in cellular telephones, in appliances
- Even the computer itself is “dissolving” into an entertainment system
    - digital TV screen and speakers
    - CPU on shelf (soon in pocket)
    - wireless keyboard on lap

Pervasive (Ubiquitous) Computing:
the paradigm for the 21st century.

Outline

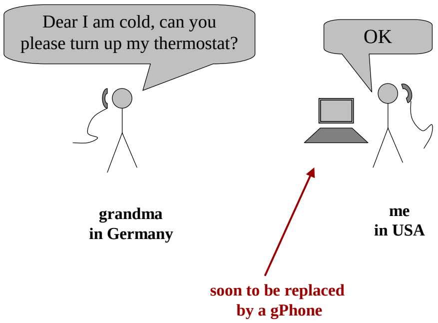
- **Pervasive Computing: a vision**
- **Sensors & Sensor Networks: introduction**
- **Sensor Network Querying**
- **Spatial Aggregation**
- **Algorithms for improving SA evaluation**
- **Other Results**
- **Conclusion: SN Querying + Spatial SQL**

Cars of Tomorrow

- **GPS** to know position
- **Wireless connection** to get traffic conditions
- **Sensors**:
    - distance to cars / people / obstacles
    - indoor/outdoor temperatures
    - road traction
- **Screen** to show sensor readings / maps
- **Speaker** used for warnings / directions
- **Automatic controls** based on sensor readings

House as a Web Site

- **Processors in various appliances, sensors on doors & windows, cameras in rooms**
    - All networked (locally, and to wireless hub)
- **House communicates with outside world**
    - Security system calls you or police
    - “Smart recycling bin” orders more food
    - Oil tank alerts company that level is low
- **You can log onto your house site to control it**
    - See map of which windows are open
    - Remote camera to see what your child is up to
    - Turn coffeemaker on, turn heat up



Sensors for / in the Body

- **Embedded microsensors**
  - Track vital signs, blood levels (for at-risk people - sick, old, mountain climbers)
  - Enabling hearing, vision (for deaf/blind)
- **Digital jewelry**
  - DCPU in watch, speaker in an earring, camera in glasses
- **Scenarios:**
  - (salesmen) Identifies person approaching, whispers their name, position to you
  - (repair trainee) Identifies machine parts, projects visual instructions on glasses

---

Ambient Intelligence

Intelligent environments of all kinds:

- **Highways**
  - Where are the traffic jams?
- **Airports**
  - Who is entering/leaving high-risk areas?
- **Large high-rise office complexes**
  - Are there problems with heat/AC anywhere?
- **Oceans**
  - Is a Tzunami on its way?
- **Forests**

Pervasive Computing

Computation in service of our needs:

- **Personal**: Entertainment, daily activities, travel, house monitoring
- **Companies**: Work efficiency, building monitoring
- **Governments**: security, automatic gathering of statistics
- **Scientific / medical**: remote training / diagnosis, monitoring oceans

spatial data everywhere

Pervasive Computing

- **Computing made easy**
  - Interaction through natural modalities
  - Interaction during natural activities
- **Computing made invisible**
  - Hidden in objects of everyday use
  - Embedded in environments

The computing paradigm for 21st century

Outline

- **Pervasive Computing: a vision**
- **Sensors & Sensor Networks: introduction**
- **Sensor Network Querying**
- **Spatial Aggregation**
- **Algorithms for improving SA evaluation**
- **Other Results**
- **Conclusion: SN Querying + Spatial SQL**

Sensors

Essential part of the vision

- **Computation**
  - A small embedded computer with limited processing power and memory
- **Communication**
  - LAN, Wireless, Infrared / sound
- **Sensing**
  - Temperature, pressure, magnetic field, noise levels, chemicals, etc.
- **GeoSensors**
  - Aware of their location
  - Location may affect their computation

Sensor Constraints

- **A race to increase:**
  - ✓Sensing / transmitting abilities
  - ✓Computation power
- **A race to decrease:**
  - ✓Size
  - ✓Price
  - ✓Energy consumption
- **Applications constrained by this tradeoff**

Sensor Networks

- **Many sensors distributed in a region**
  - Performing a common task
- **Communication is local**
  - between neighbors
- **Frequent failures**
  - power loss
  - an elephant stepped on a sensor

Fault-tolerant distributed computing

- **Monitoring tasks:**
  "killer application" for sensor networks

Wish Lists

Sensor Networks

- **Robust performance**
  - Failed sensors do not bring down the network
- **Ad-hoc routing**
  - New sensors join the network on their own
- **Established research area**

Monitoring tasks

- **Ad-hoc querying**
  - New monitoring tasks specified and initiated by user
- **Impossible while each task is custom-engineered**
  - Which has been the case
- **New approach is needed**
  - Sensor Network Querying

Outline

- **Pervasive Computing: a vision**
- **Sensors & Sensor Networks: introduction**
- **Sensor Network Querying**
- **Spatial Aggregation**
- **Algorithms for improving SA evaluation**
- **Other Results**
- **Conclusion: SN Querying + Spatial SQL**

Sensor Network Querying

- **Treat sensor network as single general-purpose platform enabling users to perform all monitoring activities**
  - A single (extensible) query language
  - A single (extensible) OS/DB engine
  - No more custom engineering
- **New & exciting research area**

Axioms of SN Querying

- **User sees network as a single intelligent information system**
  - Sensors as sources of data
  - Monitoring tasks as data processing
- **Ad-hoc querying of sensor networks**
  - Each task specified by user, not custom-engineered
  - Multiple tasks can be present at once
- **Separation of engineering concerns**
  - physical level (routing, communication)
  - logical level (data processing)

Sensors As Data

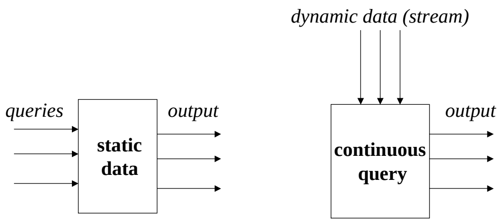
- **Sensors form a database relation**
  - Sensor(NodeID, locn, temp, pressure, ….)
- **Syntax as for regular relations**
  - Employee(EmpID, birthdate, salary, …)
- **Data semantics is *dynamic***
  - Sensors generate *streams* of continuously changing values
  - e.g. temperature and pressure

Example

Find the average temperature in each patch of old forest

```
SELECT p.id, AVG(s.temp)
FROM sensor s, forestpatch p
WHERE p.type = OLD, s.locn IN p.geom
GROUPBY p.id
EPOCH DURATION 30 s
```

Traditional DBMS
vs. Sensor Network Querying



Distributed DB Engine

- **Each sensor has an OS**
  - for managing routing, communication, etc
  - for controlling sensors
  - such as TinyOS (UC Berkeley)
- **Each sensor has a DB processor**
  - remembers all queries "alive" in the network
  - evaluates each of them continuously
  - such as TinyDB (UC Berkeley)
- **New sensors join the network seamlessly**
- **Fault tolerance when old sensors fail**

Coupled to Central Processor

- **User interacts with network via a CP**
  - Entry point into sensor network
- **Additional (static) data stored at CP**
  - Including spatial data
- **Distributed data processing**
  - some *centralized* (at the CP)
  - other *localized* (at the sensors)
- **Sensors routed in a single routing tree**
  - its root is connected to CP

Routing Tree

- **Standard assumption for sensor network querying**
- **Maintained over the sensor network throughout query evaluation**
- **Each sensor communicates locally**
  - with children and parent
- **Information travels in both directions**
  - Queries are propagated down (broadcast)
  - Answers are propagated up
- **Root coupled to central processor**

Query Optimization

- **Traditionally:**
  - minimize computation time / disk accesses
- **In sensor networks:**
  - minimize power consumption
- **Sensor power consumption:**
  - Sensing (various modalities)
  - Communication (receiving, transmitting)
  - Computation (the least of it)
- **In-network processing**
  - Essential strategy for SN Querying
  - Decreases need for communication
  - Communication is expensive!

Outline

- **Pervasive Computing: a vision**
- **Sensors & Sensor Networks: introduction**
- **Sensor Network Querying**
- **Spatial Aggregation**
- **Algorithms for improving SA evaluation**
- **Other Results**
- **Conclusion: SN Querying + Spatial SQL**

Aggregation

- **Aggregation: family of operators to summarize data**
  - Min, max, average
- **Impossible to continuously collect raw sensor data (information overload)**
  - Most queries expected to use aggregation
- **In-network aggregation for optimal query evaluation**
  - Aggregate computed gradually, as values are routed back to central processor

In-network Aggregation

- **Initializer i**: sensor ☐ state record
  - applied at the leaves of the routing tree
  - produces a **state record** for a single sensor value
- **Merger f**: state records ☐ state record
  - Computes new state record at each internal node Y, by combining the state records of all its children; the result is transmitted to Y's parent.
- **Evaluator e**: state record ☐ aggregate value
  - Applied at the root
  - Takes the state record obtained at the root and computes the aggregate value
- **Example**: computing average
  - State record <**cnt**, **avg**>; initially **cnt**=1, **avg**=value
  - new cnt = sum(cnt_i)
  - new avg = sum(avg_i * cnt_i) / new cnt

Spatial Data

Spatial data will play large role in SN queries

- **Spatial Database at Central Processor**
  - Locations (of fire stations)
  - Regions (towns, lakes)
  - Lines (roads, rivers)
- **Challenge: querying over spatial data**
- **Dynamic spatial data**
  - Contour maps for sensor readings
  - Tracking paths
  - Sensor locations (for mobile sensors)
  - *Even more challenging*

Spatial Aggregation

- **The computation of some arbitrary aggregate function for each spatial region:**
  - Find *average* rainfall for each forest patch.
  - Find *maximum* humidity for each room in building.
- **Assumptions**:
  - Regions are co-located with the sensor network; every sensor is located within some region.
  - There may be multiple sets of regions in same area (i.e. forest patches vs. land ownership).
  - Multiple queries co-exist, even over different sets of regions.
  - Sensors are **geoaware;** they know their location in some local or global coordinate system.

Spatial Aggregation: Formal Semantics

- Given some aggregate function ***f*** and the relations

  region (RegionID, RegionGeometry)
  sensor (SensorID, SensorLocn, Value)

  where Value represents the sensor reading of some value of interest (e.g. temperature or humidity)
- Spatial aggregation of ***f*** over *region* computes the relation

  agg (RegionID, AggregateValue)
- A tuple (*R*, *v*) belongs to *agg* if and only if:

  given the set S_R of tuples in *sensors* which represents sensors lying inside the region *R*, *v* is the aggregate of Value's in S_R, with ***f*** as the aggregate function.

Nested Spatial Aggregation

- **Two levels of aggregation, with spatial aggrega-tion at the lower level:**
  - Which region has the *most* working sensors per sq. km.?
  - Which region has the *highest* average temperature?
  - Which region has the *lowest* maximum ozone value?
- **Spatial aggregation is performed first, to find the aggregate value**
  - COUNT, AVG, MAX in examples above
- **Then compute an aggregate over the results**
- **Example: evaluating query 1**
  - apply COUNT to the sensor readings, grouped by region, divided by region area;
  - apply MAX to the result.

Outline

- Pervasive Computing: a vision
- Sensors & Sensor Networks: introduction
- Sensor Network Querying
- Spatial Aggregation
- **Algorithms for improving SA evaluation**
- **Other Results**
- **Conclusion: SN Querying + Spatial SQL**

In-network Evaluation of Spatial Aggregation

- **Naïve approach**:

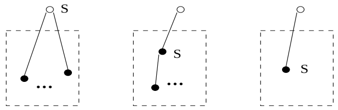
  simultaneously apply in-network aggregation approach to all the regions
- **Compute an aggregate value for each region**
- **Only merge together records from same region**
  - records carry ID of their region
- **The root receives (and evaluates) one state record per region**

Early Evaluation in Spatial Aggregation

- **Often state records are ready for evaluation long before they reach the root**
  - i.e. if that region is far from root
- **Earlier evaluation saves on communication**
  - No need to carry full state record
  - Can push the second step in nested spatial aggregation into the network
  - Can prune values
- **Also saves on computation**
  - No need for further merging up the tree
  - Record can go straight from input to output buffers
- **How can we know when to evaluate early?**

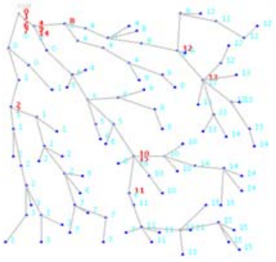
Region Leaders

- Each region is associated with one node – its **region leader**.
- A sensor node S may be the **region leader** for a region *R* only if all sensors that lie within *R* are descendants of *S*
  - S may or may not be in R
- **Exact region leader**: least common ancestor



Region Leader Lists

- Each node S is associated with a (possibly empty) list of regions for which it is the leader (its **region leader list**).
- If R is on the region leader list of S, then the aggregation record for R is **complete** once it is processed at S
  - i.e, will not be modified (by merging) by any of the ancestors of S.
  - so it can be evaluated right away



Computing Exact Region Leaders

1. Bottom-up traversal

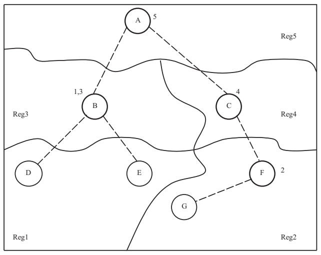
- each node X with ID X.id, located in region X.r, sends its parent a message <X.id, X.r>;
- if X is an internal node, it checks messages of the form <Y.id, Y.r> coming from its children:
  - if Y.r = X.r, the message is suppressed;
  - if Y.r appears in only one message, it is passed up;
  - If two or more messages have the same region ID Y.r, where Y.r ≠ X.r, X suppresses all such messages, sending a message <X.id, Y.r> instead.

2. Broadcast (top-down)

- at the end of pass 1, the root receives one message for every region ID, indicating which sensor node is its leader;
- this information is broadcast back into the network to notify the nodes of their region leader lists.

Example

1. Each node sends the pair <nodeID, RegID> to its parent; for example, G sends <G,2> to F.



2. Since F is in the same region as G, F sends only <F,2> to C, rather than both records.
3. In turn, C sends to A two records, <C,4> and <F,2>.
4. Similarly, B receives <D,1> and <E,1> from its children D and E, and sends <B,1> and <B,3> to A.
5. The resulting assignments are then broadcast: (B,1), (F,2), (B,3), (C,4), (A,5).

Algorithm Discussion

- Distributed network algorithm
- Bottom-up traversal requires synchronization; nodes must wait for all messages from their children, so they can know how many of them have the same region ID.
- All sensor nodes must know their own region(s) at the start.
- This algorithm will take O(h) time, where h is the height of the routing tree.
- This is optimal for *exact* leaders.
- Can *approximate* leader computation be faster?

GeoRouting Trees

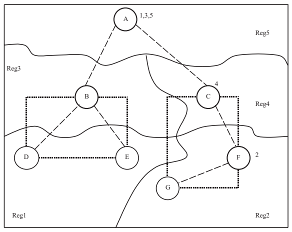
- Associate with each node a **bounding box** (or convex hull) that includes locations of all its descendants.
- **GeoRouting trees**: routing trees augmented with such bounding boxes or convex hulls.
- Can use a GeoRouting tree to find approximate region leaders.
- GeoRouting trees can also be used during **broadcasting** of spatial data (or query)
  - route spatial data only to those sensors that lie in the relevant region
  - allows significant savings in communication
  - Later in the talk

Computing Approximate Region Leaders

- Approximate region leader lists are computed via region broadcasting, when the IDs and the spatial extents for the regions $\{R_1, \ldots, R_k\}$ are broadcast.
- The computation is in parallel for all regions.
- Before broadcasting a region R, we associate with it a region leader flag R.b, initially set to FALSE.
- As R is propagated down the tree with a FALSE R.b, it will eventually reach some node X such that either
  - X is inside R, or
  - X is outside R, but it has more than one child whose bounding box (or convex hull) intersects R.
- At this point, R.b is set to TRUE and R is added to X's region leader list.

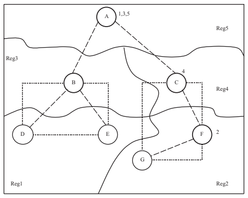
Example

- Consider computing the leader for region 1. Start with root A.
- Since the bounding boxes of A's children B and C both intersect with region 1, A is the leader of region 1.
- Now, consider computing the leader for region 2. Again, start with A.
- Since only one of A’s children (C) has a bounding box intersecting with region 2, we send this region's ID and bounding box to C.
- C is not in region 2, so it forwards this information to F, its only child.
- F is in region 2, so it becomes the leader for this region.



Algorithm Discussion

- This algorithm always finds a common ancestor of all sensor nodes in R, but not necessarily the LCA.
- This is because the intersection of R and some bounding box may contain no sensor nodes, even though it is not empty.
- Example: the intersection of C's bounding box and B's region is not empty, but there are no sensors in it. As a result, A is the leader of regions 1 and 3, instead of B.



Dynamic Maintenance

If the topology of the network changes, the region leader lists may have to be updated as well.

The following cases necessitate an update of the region leader of a region R:

1. The region leader of R fails. A new region leader needs to be computed and labelled.
2. A node in R fails. In this case, the region leader for R may have changed.
3. A node lying in R changes parents, so the new one is neither the region leader of R nor its a descendant.

While the cases 1 and 2 affect only the efficiency of the spatial aggregation procedure, case 3 also affects correctness.

Application to
Nested Spatial Aggregation

Before

- Root receives (and evaluates) one state record per region
- Second level of aggregation is in CP

Now

- Once a record has been aggregated over a region, it is ready for the 2nd level of aggregation.
- Ready records can now be merged in-network rather than at root.

Outline

- Pervasive Computing: a vision
- Sensors & Sensor Networks: introduction
- Sensor Network Querying
- Spatial Aggregation
- Algorithms for improving SA evaluation
  - Algorithms
  - **Experiments**
- **Other Results**
- **Conclusion: SN Querying + Spatial SQL**

Simulating the Sensor Network

- We varied our network size from 100 up to 1600 nodes, arranged in a grid with a cell side length of 5.
- Each cell contains a node at a random position.
- We choose the node that is closest to the top left corner of the network to be the root node.
- The communication range was set to 10.

- Network area partitioned into rectangular non-overlapping regions.
- The regions can be uniform (same width and height) or non-uniform (different widths and heights).

Simulating Sensor Readings

- We used real data from environmental monitoring.
- Three datasets, temperature, ozone and wind speed, all with different data variations and value ranges, were used.

Performance Evaluation

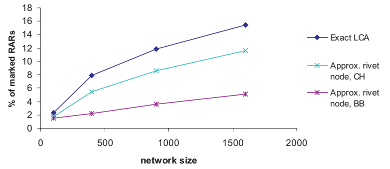
- We measured the amount of completed RARs that are forwarded in-network per epoch. This amount is given as a percentage:

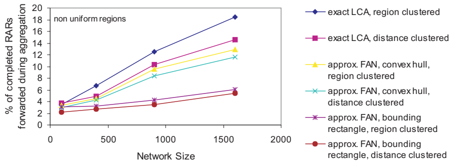
  completed RARs sent per epoch / all RARs sent per epoch * 100
- We take this amount as our “communication savings potential”.
- We plotted it for different network settings, varying the network size (number of sensors) while keeping the average number of sensors per region constant.

Performance Evaluation (cont)

- In our experiments, we considered different algorithm designs, with the following options:
  - Region leader: either approximate or exact region leaders.
  - Bounding region used: Does the approximate region leader algorithm use convex hulls (CH) or bounding boxes (BB)?
  - Parent selection: During construction of routing tree, do nodes select ancestors that are close to them (distance-clustered) or ancestors that are in the same region (region clustered)?

Results





Outline

- Pervasive Computing: a vision
- Sensors & Sensor Networks: introduction
- Sensor Network Querying
- Spatial Aggregation
- Algorithms for improving SA evaluation
- **Other Results**
  - **Region leader lists**
  - **Georouting trees**
  - **Sensor Terrains**
- **Conclusion: SN Querying + Spatial SQL**

Other Applications of Region Leader Lists

- **Filtering predicates:**

  discard aggregate records that do not satisfy predicate

  SELECT avg(volume) FROM sensors
  GROUP BY region
  HAVING avg(volume) > threshhold
- **Data pruning:**

  discard aggregate records whose value is same as in last transmission

Other applications of GeoRouting Trees

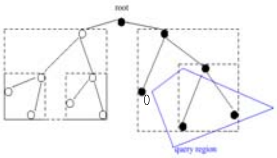
- **GeoRouting trees**: routing trees where we associate with each node a **bounding box** (or convex hull) that includes locations of all its descendants.
- **Context**: querying of geosensor networks over geo-aware sensors
- **Issue**: techniques for efficient propagation of queries and data, to reduce energy consumption by minimizing communication

---

Georouting

Optimization technique for broadcasting geospatial queries.

How it works: Selective filtering



- ➢When building routing tree, compute the bounding boxes of all subtrees, store them at parent nodes
- ➢When broadcasting geo-spatial data and queries, transmit only the information whose geospatial extent overlaps the bounding box.

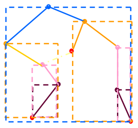
Building the GeoRouting Tree

- ➢ **Assign levels down**

  Assign a level to each node according to its distance from the root. Given a current node A at level k in the tree, any node B within A's sensing range is assigned level k+1 and added to the list of A's *candidate children*, unless it has already been assigned level k or less. A node may the candidate child of several nodes, each of which will be its *candidate parent*.
- ➢ **Select the parents**

  Start from the leaf nodes, select the nearest parent for each node from its candidate parents. Remove this node from the candidate children list of all other candidate parents.
- ➢ **Assign the bounding box**

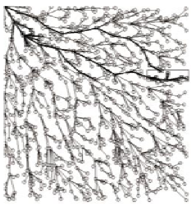
  The bounding box of all leaf nodes is their coordinates point. Then go up to the root, calculating the bounding box of each node as the minimum rectangle which includes the bounding boxes of all its children. Store the bounding boxes of the children in the parents.



Experiment Design

- to evaluate the communication savings with geo-routing;
- number of hops was measured for many broadcasts;
- plotted against the number of sensors in the query box.

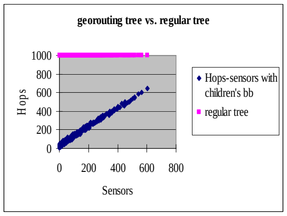
Experiment Details



- **Sensor distribution Region**
  - A fixed range of (0; 100) in both x and y.
- **Total Sensor Numbers**
  - Randomly generated 1000 pairs of values to simulate the positions of sensors.
- **Root**
  - Let the origin (0,0) acts as the root node, and the sensing range be 10 units.
- **Querying setting**
  - A rectangle query box was generated randomly and propagated down the georouting tree.
- **Evaluation**
  - Simulated 500 localized broadcasts over the sensor network.

Results

The number of sensors in the graph refers to the number of sensors that fall within the query region.



Without georouting, broadcasting always makes number of hops equal to number of sensors (999 in our experiment) minus 1 , no matter how many sensors that fall within the query region.

Analysis of Results

Georouting Efficiency: **the ratio between the minimum number of necessary hops from the root to all sensors in the query box and the number of hops for georouting.**

Over 500 queries, the average number of actual hops was 137, and the average number of necessary hops was 118. Therefore, the efficiency is:

118/137 = 86%.

Saved hops: the ratio between the average number of hops necessary to broadcast a query in georouting tree and the number of hops necessary to broadcast a query in a regular routing tree

Over 500 queries, the average number of hops was 137. Regular tree routing always results in 999 hops (one for every edge in the routing tree). Therefore, the percentage of hops saved is:

(999-137)/999 * 100 = 86.3%

Outline

- Pervasive Computing: a vision
- Sensors & Sensor Networks: introduction
- Sensor Network Querying
- Spatial Aggregation
- Algorithms for improving SA evaluation
- Other Results
  - Georouting
  - **Sensor Terrains**
- **Conclusion: SN Querying + Spatial SQL**

Sensor Terrains

- **For visualization of sensor data .**
- **Represented by *triangulated irregular networks* (TINs).**
- **TIN is a set of contiguous triangles without overlap. Its vertices are 3D points (x,y,z) where (x,y) is the location of a sensor and z is the reading at that sensor.**
- **The TIN representation is popular in *terrain mapping* because of its capacity to represent terrains over irregularly scattered data points, such as the case here.**

Why Sensor Terrains?

Several reasons to prefer sensor terrains to contours as the means of sensor data visualization:

More intuitive: 3D surfaces are cognitively easier than contour maps; for example, differences in height are directly recognizable whereas in isolines, values have to be interpreted.

Less lossy: we can compute a contour map from the sensor terrain, but not vice-versa.

Greater manipulability: Graphic manipulations of sensor terrains, such as rotations or changes to shading, can further enhance our understanding of the data; this is not possible with isolines.

Easier updates: If one sensor changes value, then only the z-coordinate of that point changes; by contrast the contour map requires more change.

Efficient Dynamic TIN Generation

- **For sensor data visualization, we must continuously regenerate the TIN corresponding to the dynamic sensor terrain.**
- ***Efficient dynamic TIN generation* is a new computational geometry problem for which we present an incremental O(log n) algorithm.**
- **We also present a new efficient algorithm for dynamically generating isolines from the TIN.**

Outline

- Pervasive Computing: a vision
- Sensors & Sensor Networks: introduction
- Sensor Network Querying
- Spatial Aggregation
- Algorithms for improving SA evaluation
- Other Results
- **Conclusion: SN Querying + Spatial SQL**

Extending SQL for Spatial Data

- **SQL 3 allows user-defined data types and operations**
  - spatial data can be added as atomic data types
  - operations over spatial data to manipulate it
  - Introduction of SQL3 started the work on SpatialSQL
- **Open Geodata Interchange Standard (OGIS)**
  - Spatial data types and operations
  - Supported by major vendors, e.g. ESRI, Intergraph, Oracle, IBM,…
  - Led to standartization of SpatialSQL

Examples

- **For all the rivers listed in the River table, find the countries through which they pass**

```
SELECT r.name,c.cntry_name
FROM river r, country c
WHERE crosses(r.the_geom, c.the_geom) = 'T' ;
```

- **Also show length of the rivers in each of the countries they pass through**

```
SELECT r.name, c.cntry_name,
   length(intersection(r.the_geom, c.the_geom))
FROM river r, country c
WHERE crosses(r.the_geom, c.the_geom) = 'T';
```

Queries can involve multiple tables (datasets)
Queries can return both traditional and spatial data

PostgreSQL & PostGIS

- **PostgreSQL - open source database**
  - Featureful – as good as commercial DBs
  - Highly customizable – with user-defined data types and methods
- **PostGIS – spatial extention to PostgreSQL**
  - Equivalent to Oracle Spatial or ArcSDE
  - OGC Standards compliant
- **Great for experimenting with further extensions**

Pairing SpatialDBs with Sensor Network Querying

Find the average temperature in each patch of old forest

```
SELECT p.id, AVG(s.temp)
FROM sensor s, forestpatch p
WHERE p.type = OLD, s.locn IN p.geom
GROUPBY p.id
EPOCH DURATION 30 s
```

A great research challenge.

Thank You

I thank the organizers of GeoInfo 2007
for giving me a chance to present this work,
and to finally visit
the wonderful country of Brazil!

References

D. Goldin, M. Song, A. Kutlu, H. Gao, H. Dave. **Georouting and Delta-gathering: Efficient Data Propagation Techniques for GeoSensor Networks**. In *Geosensor Networks: Sensor-Based Distributed Geocomputing*, eds. A. Stefanidis & S. Nittel, CRC Press, 2004, pp. 73-95

D. Goldin, H. Gao. **Dynamic Isoline Extraction for Visualization of Streaming Data**. *LNCS 3074*, Springer (presented at CSR 2006, June 2006, St. Petersburg, Russia)

D. Goldin. **Faster In-Network Evaluation of Spatial Aggregation in Sensor Networks**. *ICDE 2006*, April 2006, Atlanta GA (logner version in UConn BECAT/CSE Technical Report TR-05-7, Nov. 2005)

S. Madden, M. Franklin, J. Hellerstein, W. Hong. **TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks**. *OSDI*, Dec. 2002.

Y. Yao and J. Gehrke. **Query Processing in Sensor Networks**. *CIDR 2003*, Jan. 2003.

http://www.cs.brown.edu/people/dqg/papers