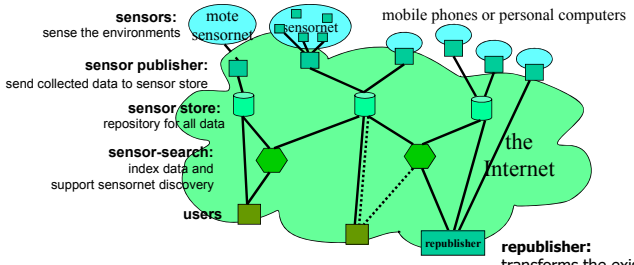
SISS: Sensor Internet Sharing and Search

PIs : Junghoo Cho, Mark Hansen, John Heidemann
Post-doc: Sung Jin Kim; Students: Gong Chen, Brian Fulkerson, Unkyu Park, Sasank Reddy, Nathan Yau

Sensor-Internet Sharing and Search

• **Sensor-Internet** [S. Reddy, G. Chen, B. Fulkerson, S. J. Kim, U. Park, N. Yau, J. Cho, M. Hansen, and J. Heidemann. Sensor-Internet Share and Search: Enabling Collaboration of Citizen Scientists. in Data Sharing and Interoperability on the World-wide Sensor Web, IPSN 2007, April 2007]

- •Goal: Share and Search across many independently run sensor networks
- •Basic Components:



In deployments, sensors pass data through publisher to store, search engine support discovery of streams and features, and users and republishers access and process data.

•**SensorBase.org** [http://sensorbase.org] ongoing work by Reddy, Yau, Hansen

- •A centralized sensor data repository: our sensor store implementation
- •Store the sensor data and support queries and visualization

What kind of software infrastructure is required for sensor-Internet?

•**Data Management:** Support to take source data and publish back the transformed data. Unkyu Park and John Heidemann

• **Sensornet discovery and cross-schema aggregation :** Discovering and integrating independently run sensor streams sources. Sung Jin Kim and Junghoo Cho

• **Sensor data-centric search:** Detection of statistical patterns in data streams (e.g., sudden temperature changes). Gong Chen, Junghoo Cho, Mark Hansen

Sensor Data Republishing Unkyu Park, John Heidemann

Republishing: Transformation of Sensor Data

- •**Beyond just sharing the sensor data**
- •**Usually require further processing on the data to user needs**
  aggregation, filtering, statistical estimation, vetting and error suppression, etc.
- •**Example: Temperature data collection**
  Read off-the-shelf wireless temperature sensors with standard web cameras (image processing done by Brian Fulkerson)
  Publishing: Web Camera, Wireless Temperature Sensors, USB LED Light
  Republishing: Fix unrecognized digits of published temperatures based on temporal-spatial correlation

Research Challenges in Republishing

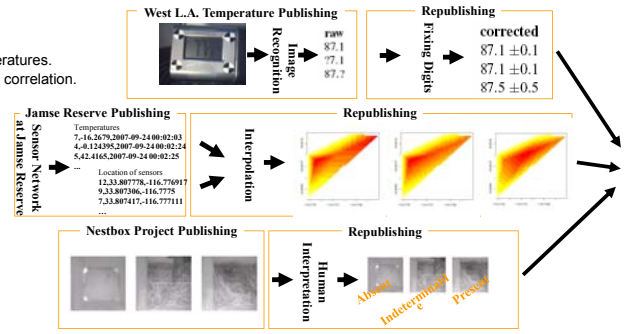
- **Data Provenance**: Tracking republishing steps back to original data source.
- **Robustness**: Coping with network outages when publishing and republishing.
- **Synchronization**: Coordinating republishing when sources have variable or different publication rates.

Examples of Republishing

Digit Repair in Temperature Capture
Sources: An image-captured temperature and preceding temperatures.
Process: Fix the unrecognized digits based on temporal-spatial correlation.
Output: Corrected temperature (if possible)

Temperature Contour Map at James Reserve
Sources: Temperatures and geographical location of sensors.
Process: Interpolate point data into a complete contour map.
Output: A temperature contour map

Human Interpretation of Nestbox Images
Source: An image of nestbox
Process: Human interpretation on the indeterminable image
Output: An annotation on the image



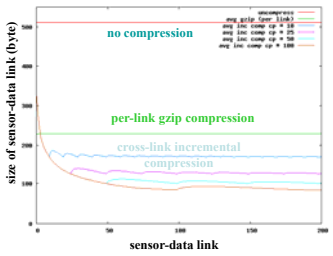
SensorPress:
A new Sensor Data Framework that provides common components required for republishing composition.
Republisher saves time and effort to build a republishing.
[Work in progress]

Data Provenance: *Tracking Sources using Sensor-data Link*

- •**Problem:** How to trace back the source of transformed data?
- •**Solution:** Use sensor-data links to locate the source data
  - •Sensor-data Link Template:
  - •sb://<location of ws0l file>?s=<servicename>&a1=<arg1>&a2=<arg2>…@TIMESTAMP
  - •e.g.) Sensor-data Link of Digit Repair Republishing:
  - •sb://sensorbase.org/soap/sensorbase2.wsdl?s=getData&a1=datetime,sensorid,rawtemperature,temperature&a2=p_97_temperature&a3=sensorid="sum-in" and datetime > "2007-10-10 00:00:00"&a4=0&a5=2
  - •Easy to extend the capability by defining new webservice.
- •**Problem:** The sensor-data link may not reproduce same sources after the republishing.
- •**Solution:** Embedded the timestamp when the republishing happens.

Data Provenance: *Efficient Link Storage*

- •**Problem:** A simple solution of data providence is just storing source links. But it is storage-intensive to provide tuple-level data provenance.
- •**Solution:** Use an incremental compression to store sensor-data links efficiently, exploiting patterns in the previous data.
- •Reduces storage cost of sensor-data links to 15% of uncompressed.



Robustness to network outages

- •**Problem:** occasional network outages
- •**Solution:** Delay Tolerant Networking techniques to tolerate communication outages
- •**Work in progress**

Synchronizing Republishing

- **Problem:** How can republishers track data streaming from multiple publishers at different rates and phases
- **Solution:** SensorPress tracks rates and phases and predicts next rendezvous times
- **Work in progress**

28 November 2007

SISS: Sensor Internet Sharing and Search

PIs : Junghoo Cho, Mark Hansen, John Heidemann
Post-doc: Sung Jin Kim; Students: Gong Chen, Brian Fulkerson, Unkyu Park, Sasank Reddy, Nathan Yau

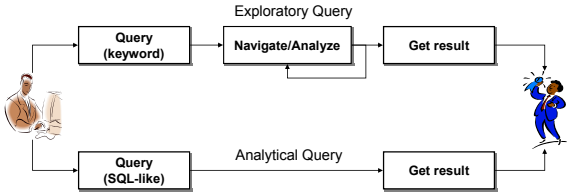
Sensornet Discovery and Cross-Schema Aggregation

Sung Jin Kim, Junghoo Cho

• Search over a Sensor Network

Exploratory query: users have vague informational need that is difficult to formulate exactly or they may lack the technical expertise to write a precise query. (e.g., how is the LA weather in summer?)

Analytical query: users have informational need that can be expressed precisely. (e.g., what is the average yearly rainfall in LA?)



• Analytical Query

Sensor Data
- Each sensor is modeled as a relational table
- There are a large number of tables

Necessity to Query
- A way to declaratively specify tables of interest
- A way to query over multiple tables

Limiataion of RDB/SQL

Query: what is the average temperature value of LA?

```
CREATE VIEW All_Temperatures_LA AS
   SELECT temperature
   FROM Sensor_A
   WHERE Location = "LA"
   UNION
   SELECT temperature
   FROM Sensor_B
   WHERE Location = "LA"
   ...;
SELECT   avg (temperature)
FROM     All_Temperatures_LA;
```

How can users specify all the tables that they want to get data from?

Extended version of RDB/SQL

A property of a table is an object that describes the table.

Syntax: CREATE TABLE tbl_name (...) WITH PROPERTY (...);

Example:
```
CREATE TABLE SensorB (time datetime, value float)
        WITH PROPERTY (type varchar(20) default "temperature",
                       location varchar(20) default "LA"));
```

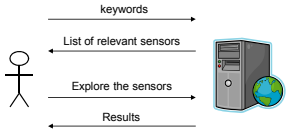
A table set is a set of the tables that users are interested in.

Syntax: CREATE TABLE SET tableset_name AS ...;

Example:
```
CREATE TABLE SET SensorSet_LA AS
   SELECT *
   FROM ALLTABLES AS TABLE table
   WHERE type = temperature and location = "LA";
SELECT   avg(temperature)
FROM     SensorSet_LA;
```

Query over a tableset

• Exploratory Query



Three ways of helping the exploration

Grouping relevant sensors by interest dimensions, e.g., project, time (year, month, day), sensor type, location (country, state, city)

Filtering out irrelevant sensors by giving conditions on each interest dimensions

Presenting values in different ways, e.g., map, bar chart, pie chart

Properties of Measuring the Relevance between a Sensor and a Query

Keyword similarity: the closeness between descriptions of the sensor and keywords of the query
Temporal similarity: the closeness between measurement times and specified times
Spatial similarity: the closeness between measurement locations and specified locations

Current Relevance Computation

$$Relevance\ (s,q) = \sum_{i=1}^{N} w_i \cdot SF_i(s,q)$$

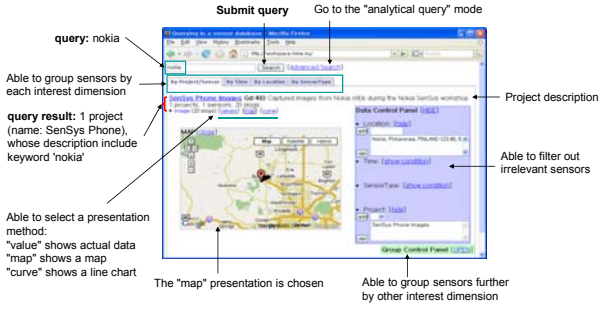
s: *a sensor* (a set of slogs, description, etc)
q: *a query* (a set of keywords)
N: # of similarity factors (currently, 3)
w_i: a weight of the *i*'th factor (sum of all w_i is 1)
SF_i: similarity in terms of the *i*'th factor ($0 \le SF_i \le 1$)

SF_1 (keyword similarity): tf.idf cosine similarity
SF_2 (temporal similarity), SF_3 (spatial similarity)

$$SF_2(s, q=t) = \frac{(\#\text{ of slogs belonging to time } t)}{(\text{total } \#\text{ of slogs of sensor } s)}$$

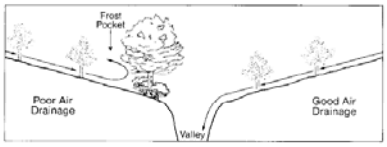
$$SF_3(s, q=l) = \frac{(\#\text{ of slogs belonging to location } l)}{(\text{total } \#\text{ of slogs of sensor } s)}$$

Exploratory Query (prototype)



Sensor Data-Centric Search

Gong Chen, Junghoo Cho, Mark H. Hansen



- Background and problem
  - Cold Air Drainage (CAD) events
    - Affect the spread of disease
    - Set micro-geographic limits on plant distribution
  - CAD transect
    - At James Reserve in the San Jacinto mountains, a network of wireless sensors records air temperature
  - Exploratory queries
    - Allow biologists to search for periods that experience a certain amount of drop in temperature within a time span in a large collection of recorded data

- Challenges
  - Difficult to operate on underlying continuous signal input
  - Exhaustive search on all combinations of drops and time spans
    - Has very large space requirements
    - Incur very long query processing time for a single search

- Proposed solution: SegDiff

1. Approximate signals by Piecewise Linear Representation (PLR)
2. Extract features from PLR
3. Compress features into parallelograms
4. Map search to queries on boundary points of parallelograms
5. Guarantee no true event missed and false positives within a user specified error tolerance

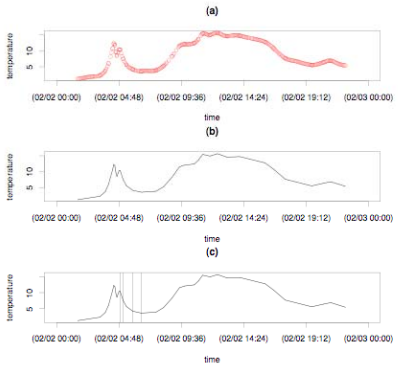


Figure 1: (a) Data; (b) segments: piecewise linear approximation of data; (c) a search result overlaid with segments

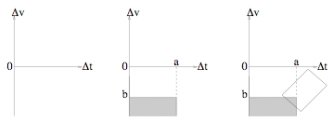


Figure 2: Feature space (left); a query region (middle); intersection between a query region and a parallelogram(right)

Figure 3: Two sub-series (left) and the corresponding segments (right)

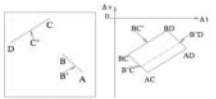


Figure 4: Two segments (left) and the corresponding parallelogram in feature space (right)

- Experimental results

The proposed solution saves one order magnitude of space usage compared to the exhaustive search and can be one order magnitude faster than the exhaustive search under a reasonable error tolerance