DIFS: A Distributed Index for Features in Sensor Networks

B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, S. Shenker

WSNA 2003
’Naive’ use of a Sensor Network

Sensor Network ≡ Pure Data Collection Device

- Individual nodes gather low-level data and send it to central instance for storage, processing, interpretation.
- Huge amounts of data induce communication bottlenecks.
- Energy required to transfer data enormous, even if no ongoing queries.
’Less Naive’ use of a Sensor Network

Sensor Network ≡ Data Collection and Storage Device

- each sensor stores its collected data locally
- a query is flooded to all nodes of the network, nodes with matching information answer
- no effort when no queries
- bad if a lot of queries arrive
- Possible: data aggregation/pruning while answering query
- . . . still bad for many queries
Data Centric Storage (1)

- define a map $f : \text{Event Type} \rightarrow \text{Network Location}$
- store all occurrences of an event at location defined by $f$
- e.g. all ’elephant sightings’ are stored at node $f(\text{elephant sighting})$
- well chosen hash function $f \Rightarrow$ load is well distributed for many different event types
- query for ’elephant sightings’ only needs to inquire at location $f(\text{elephant sightings})$
- examples: GHT + extensions
Data Centric Storage (2)

- load problem when many events of same type are occurring
- ⇒ create ‘well-spread’ set of locations responsible for one data type
- event is always stored at closest of these locations

Upon query: check all responsible locations
Data Centric Storage (3)

- DCS developed for *discrete* events (’elephant sightings’)
- Problem with continuous attributes like ’temperature’, ’time’
- e.g. ’elephant sightings between 7pm and 9pm’; probably only few ’elephant’ locations actually have matching data, still all have to be inspected
A quadtree-based approach

- use hierarchical decomp., e.g. Quadtree
- define for each event type and square a unique responsible location
- responsible location knows histogram of event times of resp. locs. of its children
- query starts at root and only descends into subtrees where histogram indicates matches
- Problem: high load on responsible root node
DIFS: Relieving the Root node

1st Idea: hash not only by event type and square but also by time range to a particular node, i.e.

\[ f : \text{Square} \times \text{Type} \times \text{Time Range} \rightarrow \text{Location} \]

if time is \([0, 255]\), we could hash \(f(s, \text{eleph}, 0), f(s, \text{eleph}, 1), \ldots\)

in the lowest level of the Quadtree where there are a lot of squares, we’d get 255 as many resp. locations!

2nd Idea: while the relevant region for a resp. location shrinks when going down the tree, we want the range to increase

at level 0 hash only with \((ES, [0, 255])\),

at level 1 hash for \((ES, [0, 63]), (ES, [64, 127]), (ES, [128, 191]), (ES, [192, 255])\)

\(\ldots\)
DIFS: Event registration / Queries

Registration:

- \((ES, 155)\) is stored at \(f(cell, ES, (0, 255))\)
- \(f(cell, ES, (0, 255))\) updates its histogram and sends the changed part \((128, 191)\) to \(f(par(cell), ES, (128, 191))\)

Query: all ES in \((47, 68)\)

- decompose into \((47, 47), (48, 63), (64, 67), (68, 68)\) (essentially \(\log|\text{range}|\) pieces)
- inspect all responsible locations determined by ES and decomposed ranges
- Queries do not always start at a root node!
Experimental Evaluation

- 1024m \times 1024m area
- 2048 nodes with comm. radius of 25m (sparse)
- generated 2048 events at random locations
- Uniform: scalar value random
- HotSpot: inversely proportional to distance to closest of 5 ’hot spots’
- For comparison: simple DCS, QuadTree, Directed Diffusion
- Not clear: Quadtree/DIFS refined to bottom?
Query Costs

- Pruning not very effective for uniform case
- for small ranges and non-uniform case QT and DIFS good
Storage Communication Costs

- Registration order of magnitude worse than for DCS
- No update intervals for QT ??
Bottleneck nodes during queries

- load on individual nodes much lower for DIFS
Summary

- #queries ≤ # events, uncorrelated events, or storage limitations ⇒ standard DCS
- # queries ≫ # events and correlated events make additional in-network organization worthwhile ⇒ QT/DIFS
- if balancing load over network nodes is important ⇒ DIFS