TSAR: A Two Tier Sensor Storage Architecture using Interval Skip Graphs

Authors: Peter Desnoyers, Deepak Ganesan, Prashant Shenoy
(University of Massachusetts, Amherst, MA)

Presented by: Nikhil Raghavan
Index

• Abstract
• Context and Motivation
• Design Considerations
• Design Principles
• Data Structures
• Data Storage and Summarization
• Implementation and Experimental Data
• Open Questions/Future Work
Abstract

• Archival storage of data is necessary for applications that query, mine and analyze such data for interesting features, trends.

• Existing sensor storage systems do not fully exploit the multi-tier nature of emerging sensor networks.
Context

• Need a system for archival storage of past data for applications that require this in an energy efficient manner with low latency.

• Existing approaches
  – Centralized Storage (reads cheap, writes expensive)
  – Distributed local storage without indexing
  – GHT (Geographic hash table) but flat hierarchy with homogeneous nodes.
Motivation

• Need to exploit the resource rich nature of tethered proxies and respect resource constraints at remote sensors

• Newer NAND flash memories offer very low write and erase energy costs. For example, a 1GB NAND flash storage when compared with an 802.15.4 shows a 1:100 per-byte energy saving

• Hence local storage is an energy efficient alternative to communication in sensor networks.
Architecture of a multi-tier sensor network
System Model

- Envision a multi-tier framework
  - Bottom tier, untethered remote sensors
  - Middle tier of tethered proxies
  - Upper tier of applications/user terminals
- Proxies have 2 radios: 802.11b for communicating with proxy tier and 802.15.4 for communicating with sensors.
- **Focus**: Applications that require access to past sensor data.
Usage Model | Query Examples

- Queries about time and location of intruder
- Range queries of one or more attributes
- Value queries: Determine if a particular value was observed at a sensor
- Value Ranges
- Hybrid value, spatio - temporal
Design Principles

• Store locally, access globally
• Distinguish data from metadata
• Provide data-centric query support [expect that applications will be best served by a query interface that allows them to locate data by value or attribute]
Data Structures

- Interval Skip Graphs
- Sparse Interval Skip Graphs
Skip Graph – (1)

• What are Skip Graphs?
• They are ordered, distributed data structures for peer to peer systems.
• Contain a set of keys and associated pointers
• Log n height
• Probabilistic balance
• Redundancy/Resiliency
Skip Graph - Advantages

- Resource location, dynamic node addition/deletion can be done in logarithmic time
- Each node only requires logarithmic space to store information about its neighbors
- Resource keys are not hashed, hence related resources can be located near each other. Application -> Prefetching of webpages, enhanced browsing/searching.
- Support complex queries like “Range Queries”
- Resilient to node failures without becoming disconnected. Can even tolerate $O(1/\log(n))$ node failures.
- Does not require a priori knowledge about number of nodes
Skip Graph – (Continued)

• Pointer traversal at the highest level skips over n/2 elements, n/4 at the next level and so on

• Search: descend tree from highest level to level 0, at each level comparing the target key with the next element at that level and deciding whether or not to traverse.

• In a perfectly balanced tree with $\log_2 n$ levels of pointers, search cost (in terms of no. of msgs = no. of pointers) = $O (\log n)$. 
Find (21)

Skip Graph of 8 Elements

Key
1  7  9  13  17  21  25  31

Level 0

Level 1

Level 2

Node-to-node messages
Interval Skip Graph

- Is a combination of Interval Trees and Skip Graphs
- Given a set of intervals \([low_i, high_i]\) sorted by lower bound \(low_i < low_{i+1}\) we introduce a secondary key \(max_i = \max_{k=0..i}(high_k)\).
- The set of intervals intersecting a value \(v\) maybe found by searching for the 1\(^{st}\) interval where \(max_i \geq v\).
- Traverse intervals in increasing order lower bound till we find an interval with \(low_i > v\), selecting those intervals which intersect \(v\).
Interval Skip Graph

Lookup complexity: $O(\log n)$
Worst case insert: $O(n)$

contains (13)
Distributed Interval Skip Graph
Sparse Interval Skip Graph

- A further improvisation!
- Arrange data structures such that most traversals occur locally within a single node thus incurring ZERO! Network cost.
- Since both congestion and failure only occur on a per-node basis we may eliminate links if they only contribute to load distribution or resiliency within a single node.
- These 2 modifications greatly improve update and search.
Constructing a Sparse Interval Skip Graph

• Ensure that there is only a single distinguished element per system->Root.
• All searches start at one of these root elements.
• New element insertion stops when it is in a list containing no root element rather than splitting lists at increasing levels L until the element is in a list with no others.
Sparse Interval Skip Graphs – Performance Gains

• In a non-interval skip graph, expected insert complexity = $O(\log N_p)$, rather than $O(\log n)$

• Since there are only $N_p$ root elements, updates only have to propagate to these, for a worst case cost of $N_p \log_2 n$ for interval sparse skip graphs.
Data Storage and Summarization

- Local Storage at Sensors
- Adaptive Summarization
Local Storage at Sensors

- Sensors produce time-series data and hence temporal ordering of data is natural.
- Local store is a collection of records designed as an append only circular buffer where new records are appended to tail.
- Support for create, read, delete operations
- Due to resource constraints no index is maintained about data in the local archive, but data’s location in terms of offsets is stored in the proxy thus enabling random access.
Local Storage at Sensors

1) Single Storage Record

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Calibration Parameters</th>
<th>Data/Event Attributes</th>
<th>size</th>
<th>Opaque Data</th>
</tr>
</thead>
</table>

2) Sensor Summarization
Adaptive Summarization

- This is used to address the existing tradeoff between energy used to send summaries and overhead/cost of false hits.
- The coarser, less frequent the summary, lower energy involved in sending, however this causes a higher rate of false hits which waste energy on requests for non-existent data.
- TSAR solves this by computing the ratio of false and true hits and varies the intervals between summaries (the authors are non-committal about varying size of the summary).
• Implementation
• Experimental Evaluation
Implementation - Hardware

- TSAR prototype testbed:
- **Proxies** -> Crossbow Stargate [400MHz Intel XScale Processor, 64 MB RAM, Linux 2.4.19 kernel, 802.11b radio +host mote bridge]
- **Sensors** -> Mica2/Mica2dots [ATMega 128L Processor, 4MB onboard Flash, 915CC1000 radio, TinyOS1.1.8]
• Motes periodically obtain sensor reading and log them to flash memory.
• Sensor updates are used to construct a sparse interval skip graph
• Current implementation supports queries based on time interval \((t_1, t_2)\) or value range \((v_1, v_2)\).
• Support for 2 types of query messages
  - Lookup
  - Fetch
Experimental Evaluation
Sparse Interval Skip Graph
Performance

a) Skip Graph Insert Performance

b) Skip Graph Lookup Performance
Experimental Evaluation
Skip Graph Overheads

a) Impact of number of proxies

b) Impact of redundant summaries
Experimental Evaluation
Adaptive Summarization

(a) Impact of summary size

(b) Adaptation to query rate
Open Questions

• How to reduce the number of pointers in a skip graph and yet maintain its locality?
• While TSAR is a viable solution for multi-tiered sensor networks, what can we learn /take from this approach in trying to improvise on existing homogeneous, flat hierarchy systems?
• What if proxies/any node cannot be tethered in some deployment/case? Solar cells?
Conclusions

• TSAR is fundamentally different from conventional sensor storage systems in that it exploits the multi-tiered nature of emerging sensor networks.

• Envisions the separation of data from metadata by employing local storage at sensors, distributed indexing at proxies.

• Adaptive Summarization trading off energy cost of transmitting metadata to proxies against overhead of false hits.
Thank You

Any Questions?