Energy-Efficient Monitoring of Extreme Values in Sensor Networks (ACM SIGMOD,2006)

Presented by,
Wiwek Deshmukh
Csc 8910
Apr 3rd 2007
Instructor: Dr. Yingshu Li

Introduction

- Lifetime of a sensor network can be extended by developing query specific plans that limit the data transmitted by the nodes.

- We consider MAX (or MIN) queries only.

  This query is an example of a more general *exemplary query*:

  where solution consists of one or more representative values as opposed to a summary, where solution is computed over all values.

- TYPE of query considered:

  Continuous MAX Query

Introduction (Contd …)

- Example Scenarios:

1. Maintaining maximum temperature in a factory.
2. Tracking the locations and amounts of highest rainfall across geographic regions.

- Distinction between MAX and Selection Queries is that nodes cannot decide their inclusion in the query result by themselves.

Previous Approaches:

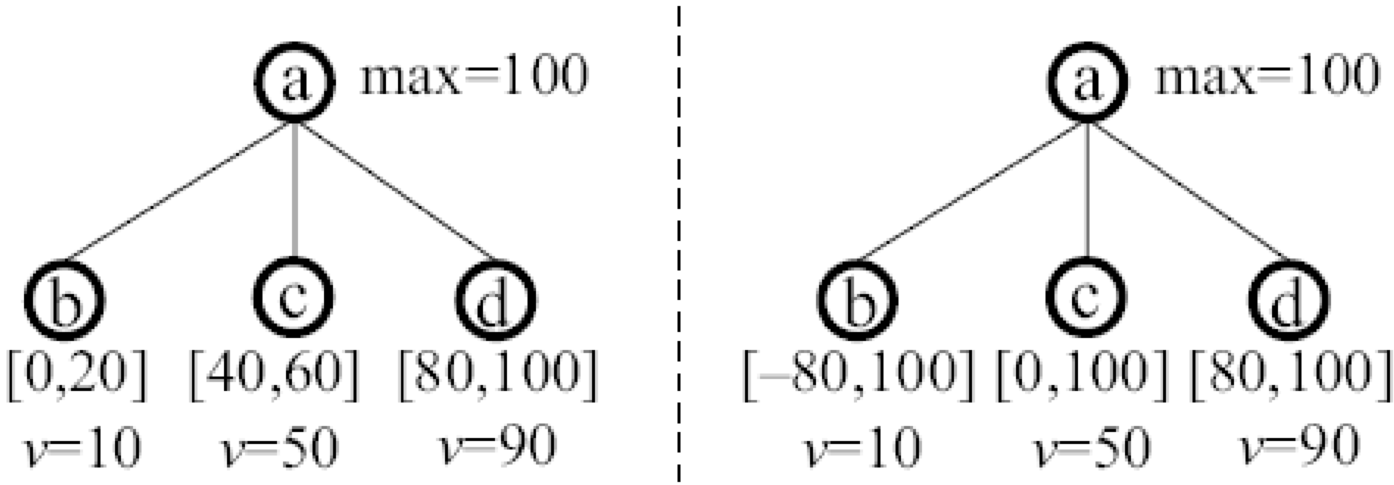
1. **Temporal Suppression**:

   A node transmits its value only if it has changed since the last transmission.

2. **Range Caching**:

- The root caches a range for each value at a remote node.
- The remote node synchronously maintains the same range stored at the root.
- The node only reports its value if it violates the range.
- The length of the range controls the trade-off between accuracy and amount of communication between root and the remote node.

Range Caching (Contd …)



Here max = 100.

Node with v=90 should have smaller range.

Topology-Oblivious Algorithms

- Here we assume all nodes are connected directly to the root in a one-level tree.
- Algorithm-1: Temporal Suppression.
- Algorithm-2: Range Caching
- Algorithm-3: SLAT (Single Layer Adaptive Threshold)

Packet Types used:

Type	Description
Boot	Initial application installation and query
Trigger	Node sending own value
Query	Root initiating fetching of values
Reply	Response to *Query*
ThresholdUpdate	Update to node constraints
MaxDesignate	Designate node as current max
MaxOff	Notify node that it is no longer max

Range Caching (Details):

- After receiving Boot packets from root, each node u_i transmits to the root its value v_i and a range $[lb_i, ub_i]$ around v_i : $v_i = \frac{(lb_i + ub_i)}{2}$
- The root maintains the max. value v_{max}.
- In subsequent rounds, u_i sends a Trigger packet listing v_i along with a new range if v_i falls outside $[lb_i, ub_i]$.
- After root gets all the reports, it finds v_{max} as follows:

Range Caching (Details):

- Let v^* = highest reported value.
- Let ub^* = highest upper-bound of unreported value.
- Let lb^* = highest lower-bound of unreported value.

- If $v^* > ub^*$: set v_{max} to v^*.

- If $ub^* > v^*$ or if no nodes reported:
  For each unreported node u_i with $ub_i > lb^*$, root sends query packet requesting v_i.

- The solution is the max. of all reply packets.

Range Caching (Details):

Range Adjustment:

- Whenever a node reports its value, it expands the length of its range by a factor α > 1.
- The new range is then centered at v_i.
- Whenever a node u_i is queried by the root:

  It contracts the length of its range by a factor α with 50% probability.
- In both cases, it transmits lb_i and ub_i to synchronize with the root.
- For the node u_{max} the range is always set to zero since its value is always required.

Range Adjustment (Contd…)

- Smaller range => Lesser chance of being queried.
- Larger range => Lesser chance of being required to report.
- Important nodes contract their range.
- Unimportant nodes expand their range.

SLAT (Single Layer Adaptive Threshold):

Initialization:

- Each node u_i is assigned a threshold T_i known to both u_i and root.
- After recv. Boot packet, each u_i sends v_i to root and sets $T_i = v_i$.
- Root finds the highest reported value = v*.
- The one who reported v* is u_{max}.
- Root sends "MaxDesignate" packet to u_{max} and instructs to perform temporal suppression.

SLAT (Contd …)

Invariant-1:

In a particular round, thresholds are set such that for each node u_i ($\neq u_{max}$): $T_i \leqslant u_{max}$.

Node-initiated reporting (1st stage):

- Max. node does temporal suppression sending Trigger packets when required.
- All other nodes transmit a Trigger packet listing v_i if $v_i > T_i$.

SLAT (Contd …)

Root-initiated querying (2nd stage):

- After all nodes have reported, root determines v* using all returned values and stored value of u_{max} if u_{max} did not report.
- Let u* be the node with value v*.
- If v* $\geqslant$ thresholds of all sensors: set v_{max} = v* and u_{max} = u*.
- Or else send Query packet to each u_i for which T_i > v*. The query contains v*.

SLAT (Contd …)

Root-initiated querying (2nd stage contd …):

- Each u_i receiving a Query sends a Reply with its value v_i, only if $v_i > v^*$.
- At root, v_{max} is set to max. of all reply packets.
- If no nodes reply, v_{max} stays at v^*.
- If u_{max} designation changes, “MaxOff” and “MaxDesignate” packets are sent to old u_{max} and new u_{max} respectively.

SLAT (Contd …)

Threshold Setting:

- To maintain the invariant, each T_i must be updated and stored at u_i and root.
- Whenever u_i breaks its threshold and sends Trigger to root, it waits for a ThresholdUpdate packet.
- The root transmits v_{max} (found in 2^{nd} stage) to all u_i waiting for threshold update.
- u_i updates its T_i to be halfway between its own value v_i and v_{max}.
- The root does the same thing and updates its own copy of T_i using v_i and v_{max}. (both are available at the root).

SLAT (Contd …)

Threshold Setting (Contd…):

When a node is queried (in 2nd stage), there are 2 cases:

Case 1: If v_i > Query Value: T_i is set to v_i.

Case 2: If v_i < Query Value:

T_i is lowered and set between v_i and Query Value.

This is also sent to the root in a reply packet.

Optimization:

If doing so does not lower Ti much below query value, u_i has option of setting T_i to the Query Value and not replying.

SLAT (Contd …)

Some Observations:

- Higher Thresholds =>

  Lower reporting + Higher Querying.

- Lower Thresholds =>

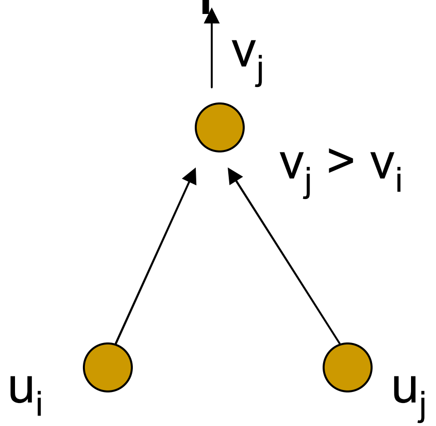
  Higher reporting + Lower Querying.

Topology-Aware Algorithms

- These algorithms take into account the routing tree.
- Algorithm-1: SLAT-A (SLAT with Aggregation)
- Algorithm-2: HAT (Hierarchical Adaptive Thresholds)

SLAT-A (Single Layer Adaptive Threshold with Aggregation):

- SLAT transmits all Trigger and Reply packets to the root.
- This can be improved.



- If $v_j > v_i$: we drop v_i and forward only v_j. (it is impossible for v_i to be max.)

SLAT-A (Contd …)

- How to update thresholds in this case?
- Any node that previously received a Trigger packet from a child must now remember that child.
- When root sends "ThresholdUpdate" with v_{max} to a child, it is sent down the tree recursively to all descendants who had sent a Trigger.

SLAT-A (Contd …)

ADVANTAGE:

No. of Triggers are reduced.

DISADVANTAGE:

Querying by the root is increased.

(The root does not have information of all thresholds. If the max. value suddenly falls, it has to query many nodes).

This problem is solved by HAT.

HAT (Hierarchical Adaptive Threshold):

- Values are only propagated upward until they reach an ancestor with threshold higher than it.
- Queries do not propagate as far downward, and stop at nodes with thresholds below the fallen max. value.

Invariant-2:

For each node u having a threshold T with parent u_p having threshold Tp, $T \leqslant Tp$.

HAT (Contd …)

Node-initiated reporting:

- When a node u_i recv. a trigger from a child node u_c , it sets a flag indicating u_c needs threshold update.

- If u_i either breaks T_i or recv. from a child a trigger with value v_{Tr} such that

  $\max\{v_i, \text{all}(v_{Tr})\} > T_i$

* It sends trigger packet to parent u_p.
* Also sends threshold update listing its own value v_i back to u_c.

- Here HAT performs additional local filtering compared to SLAT-A based on Ti.

HAT (Contd …)

- When node u_i receives a ThresholdUpdate from its parent listing a value v_{TU} :
  it modifies its threshold T_i to within the range
  $[\max\{v_i, v_{Tr}\}, v_{TU}]$ i.e. somewhere below the ThresholdUpdate value and above the value that previously caused it to trigger.

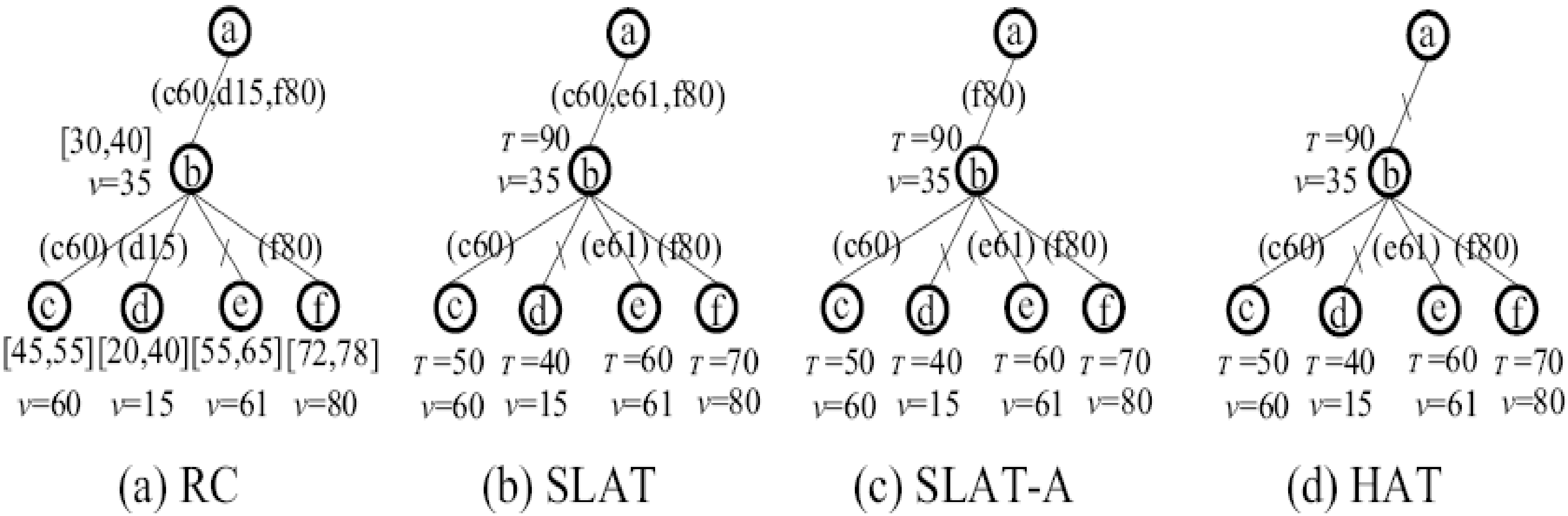
- u_i then sends ThresholdUpdate to all child nodes it has flagged earlier.

HAT (Contd …)

Root-initiated querying:

- Root first finds v*.
- It sends query packets to only those of its child nodes having thresholds > v*.
- Each node receving a query forwards it only to its children having threshold > v*.
- Any nodes having value > v* sends it in reply packets as before.
- If reply is sent, all nodes from source to the root set their thresholds to the reply value.
- Replies are aggregated just like triggers.

Comparitive Example:



Some Results:

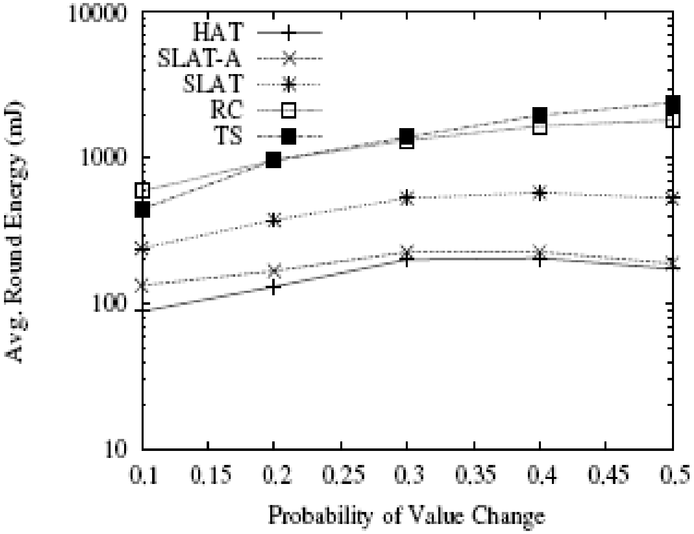


Figure 5: Random behavior.

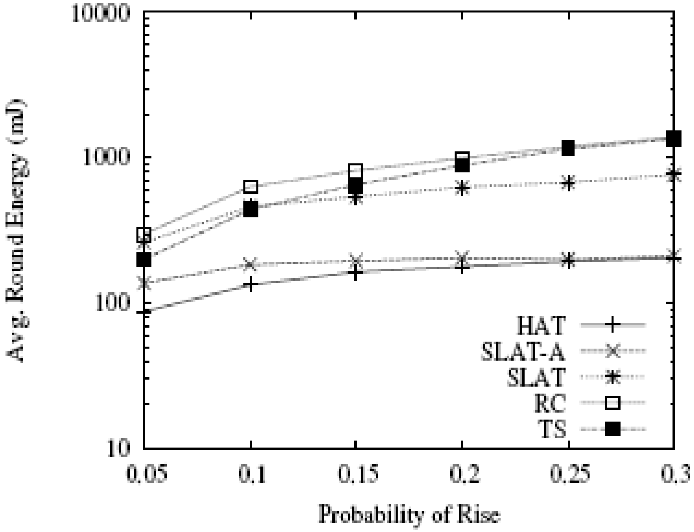


Figure 6: Randomly rising values.

References

- Energy-Efficient Monitoring of Extreme Values in Sensor Networks
  Adam Silberstein, Kamesh Munagala & Jun Yang
  Proceedings of ACM SIGMOD-PODS 2006, Chicago, IL, USA,
  June 26-29, 2006.

THANK YOU