Local Monitoring and Maintenance for Operational Wireless Sensor Networks

Md Zakirul Alam Bhuiyan, Guojun Wang, Jiannong Cao, and Jie Wu

The 11th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA-13) Melbourne, Australia, 16-18 July, 2013

Contact email: zakirulalam@gmail.com; csgjwang@gmail.com
Outline

- Motivation
- Our Approach (LoMoM)
- Two-part Monitoring Architecture
- Local Monitoring and Maintenance
- Evaluation
- Conclusions
Motivation

- Wireless sensor networks are increasingly being designed and deployed for event monitoring and surveillance applications.
  - The single most important mechanism underlying such systems is the monitoring of the network itself, that is, the monitoring/control center (MC) needs to be constantly made aware of the existence/health of all the sensors in the network.
    - Fault monitoring: detection and localization
    - Fault removal/avoidance/recovery
Difficulties in Existing Solutions

Each node is with sympathy

Routing specific
Plug-in
Other metrics
Data collection and analysis for fault monitoring
Sink metrics
Sink components
The Sink

Observer Node X
Internal state
Aggregation operator
Cached input

Fault diagnosis
Network topology

Mark parsing

Centralized Trace file

Sniffer

Poller
S_{1}
S_{2}
S_{3}
S_{4}
S_{5}
S_{6}
S_{7}
...
S_{N}

Poller
S_{1}
S_{2}
S_{3}
S_{4}
S_{5}
S_{6}
S_{7}
...
S_{N}

Cluster-based

Difficulties in Existing Solutions

- Many are centralized
  - The sink is responsible for fault monitoring
    - It actively sends requests to the network for retrieving states of the network

- Some are partially distributed
  - Add-on modules, evaluation tools, etc.
  - Need extra hardware for node monitoring
    - Pros: Accurate
    - Cons: Too much energy consumption, poor scalability, message implosion, not real-time.
More Importantly

- Those approaches are not feasible in practice for a large scale resource-constrained WSN
  
  - First, they require a large number of active nodes for the monitoring function.
  
  - Second, the monitoring function is carried out separately, which should be, arguably, performed in conjunction with the normal operation of an application.
  
  - Third, monitoring link behaviors are not seriously focused.
  
  - Fourth, it is commonly assumed to deploy a PC close to the sink; it is, however, infeasible in practice

Need a fully distributed approach
Our Approach: LoMoM

- **Local Monitoring and Maintenance for a WSN**
  - It performs monitoring operations, together with the operations of a mobile event monitoring application.
    - Suppose that a subset $E_i$ of engaged nodes, which are engaged in detecting an event of interest as it appears in a region $P_i$ of the WSN, will monitor themselves (node/link failure, node faults, H/S/W faults, application flaws, etc.).
Our Approach: LoMoM

- Local Monitoring and Maintenance for a WSN
  - It performs monitoring operations, together with the operations of a mobile event monitoring application.
    - A small subset $C_i$ (of two or more nodes) from $E_i$, which has higher detection probability are called coordinators that collected both the event and node monitoring status. They repair the faults locally, also report to the sink if needed.
The Problem

- **Given:**
  - A resource-constrained WSN with \( N \) identical nodes, a sink, a MC (located at a remote placed)

- **Find:**
  - **Coordinators** (\( C_i \)) from the engaged nodes (\( E_i \)) that monitor all the engaged nodes and themselves and the links such that the coordinator gather monitoring status and repair the network if there is any fault status and reports to the sink, and the sink then reports to the MC.

- **Objectives:**
  - Reducing the false alarm rate, \( F_c(h,T_d) \)
  - Reducing the detection latency
  - Increasing lifetime, \( T = \frac{w_r}{w_m} \)
Monitoring Architecture

- **Two-part monitoring architecture**
  - Lower part is the WSN
    - It is built on planar graphs. We apply related neighborhood graph (RNG) for our purpose [7-11]. A connected planar subgraph $G' \subseteq G$ is achieved without crossing edges, is neither unidirectional nor disconnected.
  - Each subgraph contain one or more polygonal regions (faces in face routing techniques) [7-11]

Fig. 2. Planarization techniques: a planar graph decomposed by polygons, showing the connected nodes and the edges between them that are associated to $P_2$; (a) event detection in a rectangular spot; (b) the entry of an event into $P_i$; both $P_i$ and its edge intersection leaves a trail of edges.

- **Edge Intersection Probability**
  - It is computed as an event goes across the common edge between $P_i$ and $P_j$
Local Monitoring: Node Self-Monitoring

- We trace the faulty reasons by characterizing their fault patterns.

- Generally, a node usually has different modes of operation (e.g., active, waking, and sleeping).

> We divide the active mode into three consecutive process states.  
> Each such process state consists of a lot of components. This node can be monitored by the monitoring of its component run-time behaviors (e.g., “1” for a successful run)

Fig. 3. (a) Some event detection tasks in the three process states, including the CheckCondition() function; (b) a finite set of processes of such a state, as long as an event exists.
Local Monitoring: Node Self-Monitoring

- The process state automation is modeled as a DMC (discrete time semi-Markov chain), by which, a node estimates the fault detection probability ($\beta = \Lambda, Q$).

- Node status—An engaged node $v_i \in E_i$ is said to be faulty if at least a process state of $v_i$ is altered and the node transmits inconsistent values. In such a case, $\beta \leq 0.5$
Local Monitoring: Link Monitoring

- **Link failure:**
  - A node itself may fail before a report transmission.
  - A node may transmit the report, but there is an incident of consecutive report packet loss.
  - A node is unreachable (the coordinators or the sink does not receive the status in a given time bound).
Local Monitoring: Link Monitoring

- **Techniques:**
  - We apply CMC (Continuous-time Markov Chain) technique.
    - Each link is a part of a chain between two nodes. Each coordinator has the **start** and the **end** of the chain of links.
    - Each node also individually monitors the links to its 1-hop neighbors, including the all adjacent neighbors in a \( P_i \).
  - **State transition:**

![Diagrams](image)
Local Monitoring: Link Monitoring

- We assign a transition status for each chain between node $v_i$ to $v_j$, and node $v_j$ to $v_i$, respectively.
  - For a single hop communication, “1” is for a direct successful chain from $v_i$ to $v_j$ and “1” is for $v_j$ to $v_i$, otherwise, it is “0”.
  - For a two-hop communication, “11” is for a successful chain from $v_i$ to $v_j$; otherwise, it is “01” or “10” for a broken link.
  - The link failure detection probability $\alpha = \mu(\lambda + \mu)^{-1}$ is set to calculate the false alarm rate $F_c(h,T_d)$.
Local Monitoring and Maintenance for Operational Wireless Sensor Networks

LoMoM is adaptive to network dynamics, and supports local maintenance as a link/node fault occurs.

- If one or more nodes fail, or even if all of the nodes in \(i\)-th \(P_i\) fail, that does not significantly affect the correctness of the performance of the mobile event application.
- Sensor fault and removal, connectivity hole, obstacle, etc., can be tackled with local operations, which do not require the sink’s mediation.

Fig. 5. An example of the local maintenance: single link failure; (b) single node failure.
Evaluation

- Simulations are performed with OmNet++ platform in two scenarios
  - Scenario I and Scenario II are comprised of 200 nodes and 800 nodes that are randomly distributed in 2D planar fields of 200m*200m for Scenario I and 800m * 800m for Scenario II, respectively.
    - We inject different types of faults (up to 20% of the nodes) randomly into the WSN. This invalidates the sensing and radio capabilities of 10% (5%+5%) of the nodes, and provides minimum power to 10% of the nodes so that they fail during runtime.
  - More schemes are implemented: Memento, PP (poller-pollee), and DiMo (distributed monitoring), which are proposed only for monitoring WSNs.
Results (1)

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>$F_c(h, T_d)$, $h = 3$ TO $5$, DIFFERENT PPRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRR</td>
<td>80%</td>
</tr>
<tr>
<td>PP</td>
<td>0.061</td>
</tr>
<tr>
<td>DiMo</td>
<td>0.113</td>
</tr>
<tr>
<td>Memento</td>
<td>0.151</td>
</tr>
<tr>
<td>LoMoM-C</td>
<td>0.043</td>
</tr>
<tr>
<td>LoMoM</td>
<td>0.025</td>
</tr>
</tbody>
</table>
Results (2)

![Graph showing system lifetime and latency in detection over number of sensors and number of detected faults. The graph compares Memento, PP, and LoMoM with different values of parameter c.]
Real Implementation

- A proof-of-concept system is implemented using the TinyOS on Imote2 sensor platforms.
  - We deploy 20 Imote2, plus 1 sink Imote2.
We inject 4 types of faults into four sensors:

> The communication fault (restarting radio) and sensing faults (invalidating sensing module) are injected every two minutes into two sensor nodes (N1C and N2S).
> One sensor (N3R) is set to restart every minute.
> Another sensor (N4D), preloaded with a local decision program, is set in a way in which it makes false decisions every three decisions.

Fig. 9. (a) $F_c(h, T_d)$ vs. $\beta$; (b) achieved $\alpha$ (the link failure detection probability) at different times.
Conclusions

• An interesting research area since it is a practical issue for WSNs.
  – Papers on distributed network monitoring on this issue are still lacking
  – Instead of maintenance operation by the sink, software/system level maintenance by using neighbor nodes’ cooperation in a distributed manner should receive more attention.

• The performance of event monitoring and the cost of local maintenance in WSNs can be further analyzed
Our previous work


Thank you for your attention

Q & A

Contact Email: zakirulalam@gmail.com; csgjwang@gmail.com;
Approaches

- **Offline Approaches**
  - Collect the data of system states so that later we can perform fault analysis.
    (not suitable for a resource-constrained WSN)

- **Online approaches**
  (Received more attention)
  - Find a fault during system runtime
    - Component/module faults
    - Link failure
    - The energy level of some node are much lower...
    - Sensor board temperature is larger...