Sympathy for the Sensor Network Debugger

Nithya Ramanathan
Kevin Chang
Eddie Kohler
Deborah Estrin
Sports

Posted 5/12/2005 9:53 PM

Sports Television
Michael Hiestand

Stern sees technology, global reach as NBA's challenge of TV future

Just about all pro sports realize they help themselves when they help TV.

But NBA Commissioner David Stern is unusually open-minded. Asked if TV has gone as far as it can in getting inside NBA games, Stern says, "We're far from seeing the end of it."

After all, he says, "what if they used sensors, dust-like particles, reporting (on-court) activity? I'm not sure where we'd sprinkle them. But imagine if a sensor, the weight of a penny and sewn into uniforms, could transmit broadcast quality images wirelessly."
Some Debugging Challenges

- Minimal resource sob story
  - Cannot remotely log on to nodes
- Bugs are hard to track down
- Application behavior changes after deployment
- Extracting debugging information
- Existing fault-tolerance techniques (i.e. rebooting) don’t necessarily apply; and
- Ensuring system health
After Deploying a Sensor Network…

- No data arrives at the sink, could be anything!

- The sink is receiving fluctuating averages from a region – could be caused by
  - Environmental fluctuations
  - Bad sensors
  - Channel drops the data
  - Calculation / algorithmic errors; and
  - Bad nodes
Related Work

- **Simulators / Visualizers**
  - E.g. EmTOS, EmView, and Tossim
    - Minimal historical context/ event detection
    - Not designed to discern “why” something is happening

- **SNMS**
  - Interactive health monitoring
  - Model-based calibration
  - Modeling For System Monitoring
Our Contributions

- Working, deployed system that aids in **debugging** by **identifying** and **localizing** failures
  - Debugging – an **iterative** process of detecting and discovering the root-cause of failures
- Low overhead system that runs in pre- or post-deployment environments
Failure Identification

- **Application Model**
  - Applications that collect data from distributed nodes at a sink
  - “Regular” data exchange required, and interruptions are unexpected

- **Insufficient data => Existence of a problem**
  - “Insufficient data” – defined by components

- Does NOT identify all failures or debug failures to line of code
Failure Localization

- Determining *why* data is missing
- Physically narrow down cause
  - E.g. Where is the data lost
Outline

- Sympathy’s Approach
- Architecture
- Results
Sympathy Approach

Sink collects stats passively & actively

Sink

Monitors data flow from nodes / components

- Highlights failure dependencies and event correlations

Identifies and localizes failures
Architecture Definitions

- Network: a sink and distributed nodes
- Component
  - Node components
  - Sink components
- Sympathy-sink
  - Communicates with sink components
  - Understands all packet formats sent to the sink
  - Non resource constrained node
- Sympathy-node
- Statistics period
- Epoch
Node Statistics

- Passive (in sink’s broadcast domain) and actively transmitted by nodes

<table>
<thead>
<tr>
<th>Statistic Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing Table</td>
<td>(Sink, next hop, quality) tuples.</td>
</tr>
<tr>
<td>Neighbor Lists</td>
<td>Neighbors and associated ingress/egress</td>
</tr>
<tr>
<td>Time awake</td>
<td>Time node is awake</td>
</tr>
<tr>
<td>#Statistics tx</td>
<td>Number of statistics packets transmitted to the sink</td>
</tr>
<tr>
<td>#pkts routed</td>
<td>Number of packets routed by the node</td>
</tr>
</tbody>
</table>
Component Statistics

- Actively transmitted by a node to the sink, for each instrumented component

<table>
<thead>
<tr>
<th>Statistic Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Reqs comp rx</td>
<td>Number of packets component received from sink</td>
</tr>
<tr>
<td>#Pkts tx</td>
<td>Number of packets component transmitted to sink</td>
</tr>
<tr>
<td>Last timestamp</td>
<td>Timestamp of last data stored by component</td>
</tr>
</tbody>
</table>
Sympathy System

- Collect Stats
- Perform Diagnostic
  - If Insufficient data
  - Run Tests
  - Run Fault Localization Algorithm

Comp 1

... Routing

Nodes

Sink Components

SYMPATHY

USER

SINK
Sympathy System

Comp 1
...
Routing

Sympathy

1

SINK
Network Node

- Each component is monitored independently
- Return generic or app-specific statistics
Sympathy System

Comp 1
...
Routing

Sympathy

Sink Components

Comp 1

Collect Stats

SYMPATHY

SINK

Comp 1

2
**Sink Interface**

- Sympathy passes comp-specific statistics using a packet queue
- Components return ascii translations for Sympathy to print to the log file
Sympathy System

- **Comp 1**
  - ... Routing

Sympathy System Diagram:

- **Collect Stats** → **Perform Diagnostic**
  - **If No / Insufficient data**
    - **Run Tests**
    - **Run Failure Localization Algorithm**

- **Sink Components**

- **Comp 1** connected to Sympathy System

Legend:
- **SINK**
- **SYMPATHY**
Failure Localization Algorithm

- Node Rebooted
  - Yes
  - No
    - Rx a Pkt from node
      - Yes
      - No
        - Rx Statistics
          - Yes
          - No
            - Rx all Comp's Data
              - Yes
              - No
                - No stats
                  - Yes
                  - No
                    - Comp Rx Reqs
                      - Yes
                      - No
                        - Node not Rx Reqs
                          - Yes
                          - No
                            - Comp Tx Resps
                              - Yes
                              - No
                                - Node not Tx Resps
                                  - Yes
                                  - No
                                    - Sink Rx Resps
                                      - Yes
                                      - No
                                        - Insufficient Data
                                          - Sink not Rx Resps
                                            - Yes
                                            - No
                                              - Insufficient Data

- Some node has heard this node
  - Yes
  - No
    - Some node has route to sink
      - Yes
      - No
        - No Data
          - Yes
          - No
            - No node has a Route to sink
              - Yes
              - No
                - No node has sink as neighbor
                  - Yes
                  - No
                    - No node has sink on their neighbor list
                      - Yes
                      - No
                        - No Data
                          - Yes
                          - No
                            - DIAGNOSTIC
                              - Insufficient Data
                                - No Data
<table>
<thead>
<tr>
<th>Failure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Crash</td>
<td>Node has crashed and not come back</td>
</tr>
<tr>
<td>No Route to Sink</td>
<td>No valid route exists to the sink from a node</td>
</tr>
<tr>
<td>No Data</td>
<td>No data received from a node, and Sympathy cannot localize the failure</td>
</tr>
</tbody>
</table>
# Performance “Insufficient Data” Failure Localization

<table>
<thead>
<tr>
<th>Failure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Reboot</td>
<td>Node has rebooted</td>
</tr>
<tr>
<td>Congestion</td>
<td>Correlated failures on packet reception</td>
</tr>
<tr>
<td>No reqs rx</td>
<td>Component is not receiving requests from sink</td>
</tr>
<tr>
<td>No rsps tx</td>
<td>Component is not transmitting data in response to requests</td>
</tr>
<tr>
<td>No rsps rx</td>
<td>Sink is not receiving data transmitted by a component</td>
</tr>
<tr>
<td>No stats rx</td>
<td>Sink has not received Sympathy statistics on the component</td>
</tr>
</tbody>
</table>
Node 25, Time: Node awake(mins): 78 Sink awake: 78(mins)
Route: 25 -> 18 -> 15 -> 12 -> 10 -> 8 -> 6 -> 2
node 27, are children
Num neighbors heard this node: 6

Pkt-type           #Rx Mins-since-last               #Rx-errors      Mins-since-last
1:Beacon          15(2)     0 mins          1(0)    52 mins
3:Route             3(0)     37 mins        0(0)           INF
Symp-stats      12(2)     1 mins

Reported Stats from Components
------------------------------------**Sympathy:
#metrics tx/#stats tx/#metrics expected/#pkts routed: 13(2)/12(2)/13(1)/0(0)

Node-ID  Egress  Ingress
-----------------------------
 8       128     71
 13      128     121
 24      249     254
Failure Log File

Node 18, Time: Node awake(mins): 0 Sink awake: 3(mins)
Node Failure Category: Node Failed!

TESTS
Received stats from module [FAILED]
Received data this period [FAILED]
Node thinks it is transmitting data [FAILED]
Node has been claimed by other nodes as a neighbor [FAILED]
Sink has heard some packets from node [FAILED]
   Received data this period: Num pkts rx: 0(0)
   Received stats from module: Num pkts rx: 0(0)

Node’s next-hop has no failures
Spurious Failures

- An artifact of another failure
- Sympathy highlights failure dependencies in order to distinguish spurious failures
Testing Methodology

- **Application**
  - Run in Sympathy with ESS
  - In simulation, emulation and deployment

- **Traffic conditions:** no traffic, application traffic, congestion

- **Node failures**
  - Node reboot – only requires information from the node
  - Node crash – requires spatial information from neighboring nodes to diagnose

- **Failure injected in one node per run, for each node**

- **18 node network, with maximum 7 hops to the sink**
Time to Detect Node Crash/Reboot

CDF of Time to Detect Node Reboot/Crash in Simulation

CDF

Time [minutes]

0 2 4 6 8 10 12 14 16

reboot

crash
Spurious Failure Notifications

Simulation and emulation are similar.

Reboot is easy to detect, thus few spurious failures.
Time to Detect Node Crash

CDF Fault=Die, All nodes

“Congestion” cases may take longer
Spurious Failure Notifications w/ Congestion

Simulation and emulation are similar.

Congestion results in more spurious failure notifications.
Sympathy Packet Overhead

CDF Ratio of Sympathy/Total Traffic

Ratio for a node (Bytes Sympathy Tx/Total B Tx)
Varying Epoch Window Size, No Traffic

- Window size: Number of statistics periods in the epoch
<table>
<thead>
<tr>
<th>Binary</th>
<th>RAM</th>
<th>ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESS w/o Sympathy</td>
<td>3089 B</td>
<td>96094 B</td>
</tr>
<tr>
<td>ESS w/ Sympathy</td>
<td>3160 B</td>
<td>104802 B</td>
</tr>
<tr>
<td>Difference</td>
<td>71 B</td>
<td>8708 B</td>
</tr>
</tbody>
</table>
Another Real World Example

- Temporal sink presence
Ongoing Work

- Using a Bayes engine to reduce the number of spurious failure notifications
- More deployments
Conclusion

- A deployed system that aids in debugging by detecting and localizing failures
- Small list of statistics that are effective in localizing failures
- Behavioral model for a certain application class that provides a simple diagnostic to measure system health
Thank You!
Iter_fail Variable

- For some failures, Sympathy must get information from all nodes within the epoch

  OR

- Sympathy should not have heard from that node for iter_fail statistics periods in order to ignore the node
Sympathy System

Comp 1
... Routing

Sympathy

Collect Stats
Perform Diagnostic
If Insufficient data
Run Tests
Run Fault Localization Algorithm

Sympathy

Collect Stats
Perform Diagnostic
Run Tests
Run Fault Localization Algorithm

Sink Components

USER

SINK

1

2

3
4
Failures Sympathy Detects$^{1,2}$

- System Design / algorithm / protocol bugs
- Connectivity / topology

---

$^1$ R. Szewczyk, J. Polastre, A. Mainwaring, D. Culler “Lessons from a Sensor Network Expedition”. In EWSN, 2004

Sink Application

Sympathy-Sink

Ring Buffer

Request State & Stats Recorder

Event Analysis

Node 1 process

Node 3 process

Node 3 process

... Node n process

Update stats using Emstar IPC

ETHERNET BACK CHANNEL
Regular Sympathy Peon

- Self-tests and probes can also be externally specified (e.g. by a neighbor)
- External visible interfaces

Diagram:
- Collect statistics
- ID Events
- Inject Probe/Self-Test
- Record tests/Probes injected
- Record Events/Return buffer
- Send Event
- Send Statistics
- Return Debug Info upon request
- Specify self-test or Probe to inject
- Externally visible interfaces
SNMS/ Nucleus Management System

- Enables interactive health monitoring of WSN in the field
- 3 Pieces
  - Parallel dissemination and collection
  - Query system for exported attributes
  - Logging system for asynchronous events
- Small footprint / low overhead
  - Introduces overhead only with human querying

---

1 Gilman Tolle, David Culler, “Design of an Application-Cooperative Management System for WSN”
Second EWSN, Istanbul, Turkey, January 31 - February 2, 2005
Model-Based Calibration\textsuperscript{1,2}

- Use models of the physical environment to identify faulty sensors, e.g.:
  - Assume values from neighboring sensors in a dense deployment should be “similar”\textsuperscript{2}
  - Plug sensor data into a pre-defined physical model; identify sensors that make the model inconsistent\textsuperscript{1}

\textsuperscript{1} Jessica Feng, S. Megerian, M. Potkonjak “Model-based calibration for Sensor Networks”. IEEE International Conference on Sensors, Oct 2003
\textsuperscript{2} A Collaborative Approach to In-Place Sensor Calibration – Vladimir Bychovskiy Seapahn Megerian et al
Modeling For System Monitoring

- Identify “anomalous” behavior based on externally observed statistics
  - Statistical analysis and Bayesian networks used to identify faults

1 E. Kiciman, A. Fox “Detecting application-level failures in component-based internet services”. In IEEE Transactions on Neural Networks, Spring 2004
3 E. Kiciman, L Subramanian. “Root cause localization in large scale systems”