Analyzing WSN-based IoT Systems using MDE Techniques and Petri-net Models

Burak Karaduman, Moharram Challenger, Raheleh Eslampanah, Joachim Denil, & Hans Vangheluwe

4th International Workshop on Model-Driven Engineering for the Internet-of-Things (MDE4IoT)
1. Introduction

IoT and WSN relation:

- Home appliances, smart buildings, Industry 4.0 applications, and Digital Twin systems.

- Benefit from (WSN) to make their communication topology more flexible

- Increase the coverage of the resulting IoT system

- Without a need for a direct Internet connection.

This causes Complexity, because:

- Require more components such as source nodes, sink nodes, and gateways

- The resulting system is complex with these additional components

- Complexity makes the design and analyses of these systems time-consuming, costly, and cumbersome.

- It can be addressed with MDE by automatically synthesize and transform the system artifacts.
1. Introduction - 2

What should we do?
- Design models can be used for the early analyses and validation of the system to reduce the number of errors in the SUD.
- Fast development process, saves development cost and effort.

What should we use?
- As these multi-component systems work based on message passing to fulfill their tasks,
- Their behavior can be represented by the Petri-net models (M2M Transformation).

Which feature can be applied?
- K-boundedness can be used for power consumption, bottleneck, and first crashing node.
2. Analyzing IoT systems with Petri-net models

How to map the elements?

- Generation rules both for LoLA and PIPE
- PIPE for GUI
- LoLA for k-boundedness

<table>
<thead>
<tr>
<th>DSML Element</th>
<th>Petri-Net Element</th>
<th>DSML Element</th>
<th>Petri-Net Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag (Sensor Nodes)</td>
<td>Place</td>
<td>Node messages</td>
<td>Token</td>
</tr>
<tr>
<td>ESP8266</td>
<td>Place</td>
<td>ESP Messages</td>
<td>Token</td>
</tr>
<tr>
<td>IoT Log Manager</td>
<td>Place</td>
<td>Element Relations</td>
<td>Transition</td>
</tr>
<tr>
<td>RaspberryPi</td>
<td>Place</td>
<td>Element Relations</td>
<td>Arcs</td>
</tr>
</tbody>
</table>

Mapping table: DSML4Contiki elements to Petri-net model elements,
2. Analyzing IoT systems with Petri-net models - 2

[for (l:LogMan | IoTSys.logman)]
  ◦ [for (e:ESP | IoTSys.esp)]
    ◦ [e.Name/],
  ◦ [/for]
  ◦ [for (t : Tag | tag)]
    ◦ [t.Name/],
  ◦ [/for]
[/for];

Code generation codes are written based on LoLA’s and PIPE’s syntax.
3. Topology viewpoint to Petri-net model

- User designs a topology.
- Inputs distance parameter for each node.
- Token is delivered to neighbor node.
- Only one neighbor (Routing Protocol Constraint)

- The generated BOM represents components that are used in the design and their total cost.

- It is easily made using the Acceleo.

- It is important when you try to ease the bottleneck.

- It is important when you try to increase battery cap.

```plaintext
[for(t:String|calcDist)] TmoteSky x [aloTSystem.logman.tag->size()]/= [100*aloTSystem.logman.tag->size()]/$
```

...

- For the time and power constraint systems:
  - Message transmission delays.
  - What happens if the distance to the sink node is too far in a multi-hop network,
    - Propagation delay for may cause time-violation.
  - If the distance increases, the propagation delay increases.
  - Processing delay trivial amount of time comparing to the message transmission time so can be ignored

- Propagation delay and power and time constraints
  - Worst case needs to be calculated with respect to the node which has the furthest distance
  - Trade-off analyses can be done by the user using
  - Automatically calculated propagation delays help the user to handle trade-offs
4. Additional Reports: 
Bills of Materials & Propagation Delays - 3

```java
calcDist(Collection tag)
{
for(int i=0; i<aTag.size(); i++)
{
    for(int j=0; j<aTag.get(i).getSelftag().size(); j++)
    {
        aTag.get(i).getSelftag().get(j).setDistance(aTag.get(i).getSelftag().get(j).getDistance() + aTag.get(i).getDistance());
    }
}
```

- Java Service used to calculate total distances
- Travels all Tag elements and sums the distances (calculates using speed of light formula).

As it can be seen that for the node named sensor123 three information are given

1-) It’s parent is node named sensor122.
2-) It is 258 meter away from the sink node.
3-) Propagation latency is 0.86 micro-seconds
5. How to use k-boundedness?

• What is k-boundedness?

K-boundedness feature checks a place in a Petri-net compares tokens that are passed in that place and given k value.

• k increases bottleneck gets intense and power consumption increases at a node.

• If the power is depleted in a node, a part of the network may be disconnected.

• What should be done?

• Optimal k value must be found by the designer.

• The topology design must be made considering this value.

• The number of the tokens pass through a place must be below this k value.

• One way to reduce the k value is by adding extra nodes to decrease message traffic (extra cost)

• Another way is increasing battery capacity (also extra cost)
6. Analyzing the Fire Detection System with Petri-net

- To keep the planned life-time of the network, the bottleneck problem must be analyzed before development.

- The k value must be found to provide the desired life-time with planned battery capacity.

- Messages which are received and transmitted in a node should be considered.

- Send and receive two operations.

- Adds 1 more operation (node’s own sampling data).

\[
\text{NumberOfOperations} = (2 \times k) + 1
\]
7. Power Consumption

- Power consumption depends on the number of operations for each node.
- The network must be bounded by k value.
- The user decides node lifetime, the battery capacity, and message sending period.
- The power consumption of a send and receive operation is averaged \((R \times T \times \text{AvgmA})\).
- \(T\) represents the number of using antenna in an hour (unit of battery capacity is mAh).
- If the system’s lifetime is determined then k value can be found to analyze the Petri-net.

\[
\text{NumberOfOperations} = (2 \times k) + 1
\]

\[
\text{LifeTime}_{\text{hour}} = \frac{\text{BatteryCapacity}_{\text{mAh}}}{(\text{NumberOfOperations} \times (R \times T \times \text{AvgmA}) \times T)}
\]

\[
T = \frac{3600}{\text{SamplingPeriod}_{\text{seconds}}}
\]
8. Conclusion & Future Works

- The current study extends DSML4Contiki by automatic generation of the Petri-net using k-boundedness.
- Network lifetime (first crashing nodes), bottleneck, and power consumption analyses in the early design phase.

In the future;
- Raise the level of abstraction to PIM. (including RiotOS and TinyOS)
- This will be achieved using the M2M.
- Moreover, we aim to use Multi-agent Systems in the modeling, analysis, and implementation of IoT systems.
- **PIM -> MDE4IoT special issue in SoSyM journal**
Thanks for your attention.