Formal Analysis of Policies in Wireless Sensor Network Applications

Marco Patrignani      Nelson Matthys      José Proença      Danny Hughes
                        Dave Clarke

K.U.Leuven

May 31, 2012
Outline

WSN Applications with CaPI

Policies

Analysis with mCRL2

Prototype Tool

Recap & Future Work
Outline

WSN Applications with CaPI

Policies

Analysis with mCRL2

Prototype Tool

Recap & Future Work
Wireless Sensor Network Applications with CaPI

- Application: high level of abstraction (no routing, no low level communication).

Marco Patrignani, Nelson Matthys, José Proença, Danny Hughes, Dave Clarke

Formal Analysis of Policies in Wireless Sensor Network Applications
Wireless Sensor Network Applications with CaPI

- Application: high level of abstraction (no routing, no low level communication).
- Key elements of CaPI middleware:
  - Components
  - Wires
  - Policies
Graphical Representation of Components

\[ N_1 \quad N_2 \quad N_3 \]
Graphical Representation of Components

\[ N_1 \quad C_{a1} \quad \cdots \quad C_{an} \quad N_2 \quad C_{b1} \quad \cdots \quad C_{bn} \quad N_3 \quad C_{c1} \quad \cdots \quad C_{cn} \]
Graphical Representation of Wires
Graphical Representation of Wires
Graphical Representation of Wires
Graphical Representation of Wires

Graph showing the connections between nodes and policies.
Policies

- Take care of non functional requirements via domain-specific actions.
Policies

- Take care of non functional requirements via domain-specific actions.
- Triggered by messages when messages leave and enter a node.
Policies

- Take care of non functional requirements via domain-specific actions.
- Triggered by messages when messages leave and enter a node.
- Executed at 4 entry points: from components, to components, from network, to network.
Policies

- Take care of non functional requirements via domain-specific actions.
- Triggered by messages when messages leave and enter a node.
- Executed at 4 entry points: from components, to components, from network, to network.
Policies

- Take care of non functional requirements via domain-specific actions.
- Triggered by messages when messages leave and enter a node.
- Executed at 4 entry points: from components, to components, from network, to network.
Policies

- Take care of non functional requirements via domain-specific actions.
- Triggered by messages when messages leave and enter a node.
- Executed at 4 entry points: from components, to components, from network, to network.
Policies

- Take care of non-functional requirements via domain-specific actions.
- Triggered by messages when messages leave and enter a node.
- Executed at 4 entry points: from components, to components, from network, to network.
Policies

- Take care of non functional requirements via domain-specific actions.
- Triggered by messages when messages leave and enter a node.
- Executed at 4 entry points: from components, to components, from network, to network.
Problems Related to Policies

Policies generate problems like:

- security holes;
- wastage of resources;
- other (logging encrypted data).

because:

- they are deployed over time;
- they change over time;
- their semantics can conflict.
Problems Related to Policies

Policies generate problems like:

- security holes;
Problems Related to Policies

Policies generate problems like:

► security holes;
► wastage of resources;
Problems Related to Policies

Policies generate problems like:

- security holes;
- wastage of resources;
- other (logging encrypted data).
Problems Related to Policies

Policies generate problems like:

- security holes;
- wastage of resources;
- other (logging encrypted data).

because:

- they are deployed over time;
Problems Related to Policies

Policies generate problems like:

- security holes;
- wastage of resources;
- other (logging encrypted data).

because:

- they are deployed over time;
- they change over time;
Problems Related to Policies

Policies generate problems like:

▶ security holes;
▶ wastage of resources;
▶ other (logging encrypted data).

because:

▶ they are deployed over time;
▶ they change over time;
▶ their semantics can conflict.
Outline

WSN Applications with CaPI

Policies

Analysis with mCRL2

Prototype Tool

Recap & Future Work
Policies Example

\[ P_1 @ N_1 \]
\[
\begin{align*}
on \text{event (T) as } e \\
\quad \text{if (true) then} \\
\quad \text{encrypt } e;
\end{align*}
\]
Policies Example

$$P_1 \otimes N_1$$

- on event (T) as e
- if (true) then
- encrypt e;

Diagram:

- Transition labeled T
- Transition labeled $\neg T$
- Node labeled $true$
- Node labeled $encrypt$
- Initial state $P_1$
### Policies Composition Example

<table>
<thead>
<tr>
<th>$P_1@N_1$</th>
<th>$P_2@N_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>on event (T) as e if (true) then encrypt e;</td>
<td>on event ($^\circ$C) as e if (lowBatt()) then deny e;</td>
</tr>
</tbody>
</table>
Policies Composition Example

<table>
<thead>
<tr>
<th>$P_1 \circ N_1$</th>
<th>$P_2 \circ N_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>on event (T) as e</td>
<td>on event (°C) as e</td>
</tr>
<tr>
<td>if (true) then</td>
<td>if (lowBatt()) then</td>
</tr>
<tr>
<td>encrypt e;</td>
<td>deny e;</td>
</tr>
</tbody>
</table>

$\neg T \quad \text{true} \quad \text{encrypt} \quad \neg \text{lowBatt}() \quad \neg \text{°C} \quad \text{deny} \quad \text{lowBatt}() \quad \text{°C}$
### Policies Network Composition Example

<table>
<thead>
<tr>
<th>$P_1@N_1$</th>
<th>$P_2@N_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>on event (T) as e</td>
<td>on event (Any) as e</td>
</tr>
<tr>
<td>if (true) then</td>
<td>if (true) then</td>
</tr>
<tr>
<td>encrypt e;</td>
<td>decrypt e;</td>
</tr>
</tbody>
</table>

Wires on Any messages.
Policies Network Composition Example

<table>
<thead>
<tr>
<th>$P_1@N_1$</th>
<th>$P_2@N_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>on event (T) as e</td>
<td>on event (Any) as e</td>
</tr>
<tr>
<td>if (true) then encrypt e;</td>
<td>if (true) then decrypt e;</td>
</tr>
</tbody>
</table>

Wires on Any messages.
Policy Programs

\[ Policy \; Program \; P ::= \text{on}(t)\{P\} \quad \text{Type} \; t \in T \]
\[ \quad | \text{if}(g)\{P\} \quad \text{If-Guard} \; g \in \mathcal{L} \]
\[ \quad | a \quad \text{Action} \; a \in A \]
\[ \quad | P; P' \]

\[ A ::= \{\text{allow, deny, encrypt, decrypt, sign, verify, persist, delete}\}. \]
Semantics Function

The idea is to capture what was drawn before: all paths of a tree of execution.

\[
\begin{align*}
\text{on}(t\{P\})_c &= \{ \tau \mid \tau \in [P]_c \cap \{t\}, C \models t \} \cup \{(\epsilon; C \cap \{\neg t\}) \mid C \models \neg t\} \\
\text{if}(g\{P\})_c &= \{ \tau \mid \tau \in [P]_c \cup \{g\}, C \models g \} \cup \{(\epsilon; C \cup \{\neg g\}) \mid C \models \neg g\} \\
[a]_c &= \{(a; C)\} \\
[P; P']_c &= \{(\bar{a} \bar{a'}; C') \mid (\bar{a}; C'') \in [P]_c, (\bar{a'}; C') \in [P']_{c'}\}
\end{align*}
\]
Outline

WSN Applications with CaPI

Policies

Analysis with mCRL2

Prototype Tool

Recap & Future Work
Analysis

Specification of undesired/desired properties via modal logic:

\[ [\text{true}^\ast . \text{encrypt} . \text{true}^\ast . \frac{1}{2} . (!\text{decrypt})^\ast ] < \text{true}^\ast . \text{decrypt} > \text{true}; \quad \text{(dec-after-enc)} \]
\[ [\text{true}^\ast . \text{encrypt} . \text{true}^\ast . \text{encrypt} ] \text{false}; \quad \text{(enc-most-once)} \]
\[ [\text{true}^\ast . \text{encrypt}] < \text{true}^\ast . \frac{1}{2} > \text{true}; \quad \text{(snd-after-enc)} \]
Outline

WSN Applications with CaPI

Policies

Analysis with mCRL2

Prototype Tool

Recap & Future Work
We developed a tool in Scala that:

- extracts the formalization of policies;
- calculates the semantics of policy programs;
- analyzes the constrained trace semantics against some user-defined properties via mCRL2;
- scales linearly in terms of connections between nodes and number of policies (with similar logical conditions).
Tool

We developed a tool in Scala that:

▶ extracts the formalization of policies;

Marco Patrignani, Nelson Matthys, José Proença, Danny Hughes, Dave Clarke
K.U.Leuven
Formal Analysis of Policies in Wireless Sensor Network Applications
We developed a tool in Scala that:

- extracts the formalization of policies;
- calculates the semantics of policy programs;
Tool

We developed a tool in Scala that:

▶ extracts the formalization of policies;
▶ calculates the semantics of policy programs;
▶ analyzes the constrained trace semantics against some user-defined properties via mCRL2;
We developed a tool in Scala that:

- extracts the formalization of policies;
- calculates the semantics of policy programs;
- analyzes the constrained trace semantics against some user-defined properties via mCRL2;
- scales linearly in terms of connections between nodes and number of policies (with similar logical conditions).
Tool- feedback

Extraction of constrained traces: ...

Fast analysis.

- Checking formula generalWastefulDeny:Failed
- Checking formula signBeforeverify:OK
- Checking formula encryptBeforeDecrypt:OK
- Checking formula verifyAtmostOnce:OK
- Checking formula decryptAfterEncrypt:Failed
- Checking formula decryptAtmostOnce:OK
- Checking formula signAtmostOnce:OK
- Checking formula persistBeforeDelete:OK
- Checking formula sendAfterEncrypt:Failed
- Checking formula networkAtmostOnce:OK
- Checking formula sendAfterSign:OK
- Checking formula generalWastefulDenyAfter:OK
- Checking formula verifyAfterSign:OK
- Checking formula encryptAtmostOnce:OK

Total time: 1353 milliseconds.

### WARNING ### errors discovered ### WARNING ###

Running in-depth analysis...

Analyzing node n6
Analyzing node n7
Analyzing node n0
Analyzing trace n0 encrypt.allw.decrypt.allw.decrypt n0
Tool feedback

Analyzing trace n0 encrypt.allw.NW.decrypt.allw n6
Analyzing trace n0 encrypt.allw.deny n6
    Violation of sendAfterEncrypt by trace: Trace: n0 "encrypt.allw.deny" n6
Assumptions: types: 6
guard: true

Violation of generalWastefulDeny by trace: Trace: n0 "encrypt.allw.deny" n6
Assumptions: types: 6
guard: true

Analyzing trace n0 encrypt.allw.NW n7
    Violation of decryptAfterEncrypt by trace: Trace: n0 "encrypt.allw.NW" n7
Assumptions: types: 2
guard: true
Outline

WSN Applications with CaPI

Policies

Analysis with mCRL2

Prototype Tool

Recap & Future Work
Recap & Future Work

Recap

- Formalized policies and their semantics in CaPI-driven WSN applications;
- Created a prototype for the analysis of policies in real-world deployment.

Future Work

- Model dynamic actions of policies;
- Provide a concurrency model for the modeled application.
Recap & Future Work

Recap

▶ Formalized policies and their semantics in CaPI-driven WSN applications;
▶ Created a prototype for the analysis of policies in real-world deployment.

Future Work

▶ Model dynamic actions of Policies;
Recap & Future Work

Recap

▶ Formalized policies and their semantics in CaPI-driven WSN applications;
▶ Created a prototype for the analysis of policies in real-world deployment.

Future Work

▶ Model dynamic actions of Policies;
▶ Provide a concurrency model for the modeled application.