Session 5c: WiseML

An XML dialect

Ioannis Chatzigiannakis

Research Academic Computer Technology Institute

October 14th, 2009

Outline

What is WiseML?

- XML format for expressing traces collected from WSNs
- Descendant from:
  1. GraphML – describing topologies
  2. KML – describing temporal relationships
  3. Our imagination – sections and organization for easy parsing
- "Standard language" between
  1. Wireless sensor network testbeds
  2. Simulators
  3. Management tools
  4. Users

Why WiseML?

- Data collected by WSNs need to be replaced in simulators/testbeds
- GraphML & KML
  - too focused on their application area
- SensorML, TransducerML, "Observations and Measurements" ML
  - way too large...
- One standard "language" was needed to bridge the tools used in developing algorithms for WSN
- Good moment to standardize the data transfer

WiseML Sections

```
<wiseml version="1.0" xmlns="http://wisebed.eu/ns/wiseml/1.0">

  <setup>
    <!-- static information section -->
  </setup>

  <scenario id="...">
    <!-- scenario information section -->
  </scenario>

  <trace id="...">
    <!-- dynamic information section -->
  </trace>

</wiseml>
```

Setup information section

- Describes the setup and the data collected with it
- Two concepts:
  1. Attributes – compulsory features of elements (such as "name")
  2. Capabilities – user defined characteristics (such as "pressure gradient")
- Four sub-sections:
  1. Setup description
     - Geographical placement, starting time information, mobility interpolation, etc.
  2. Node description
     - Attributes (position, gateway,...) + capabilities & default values (sensors,...)
  3. Link description
     - Attributes (default RSSI, is virtual, etc.) + capabilities & default values (LQI)
  4. Topology description
     - List of nodes and links

Example 1/2

```
1 <defaults for="node">
2   <gateway>false</>
3   <capability>
4     <name>urn:wisebed:node:capability:temperature</name>
5     <datatype>integer</datatype>
6     <unit>lux</unit>
7     <default>0</default>
8   </capability>
9 </defaults>
```

Example 2/2

```
1  <link source="urn:wisebed:node:tud:3" target="urn:wisebed:node:tud:4">
2    <encrypted>true</encrypted>
3    <virtual>false</virtual>
4    <rssi datatype="decimal" unit="dBm" default="-120" />
5    <capability>
6      <name> urn:wisebed:node:capability:noise</name>
7      <datatype>decimal</datatype>
8      <unit>dBm</unit>
9      <default>-120</default>
10   </capability>
11 </link>
```

Scenario information section

- Describes changes to applied on top of real data
- Simulated node failure
  - enableNode, disableNode
- Simulated interference at link level
  - enableLink, disableLink
- Simulated sensor noise level or failure
  - Modifications of default sensor values

Example

```
<scenario id="...">

  <timestamp>0</timestamp>
  <enableNode id="..." />
  <disableNode id="..." />
  <enableLink source="..." target="..." />
  <disableLink source="..." target="..." />

  <node id="...">
    <position>
      <x>0</x>
      <y>1</y>
      <z>2</z>
    </position>
    <data key="lqi">50</data>
  </node>
</scenario>
```

Dynamic information section

- Expresses actual collected data
  - Sensor readings
  - Network layer information (links + rssi values)
  - Mobility information
- Time description
  - Makes use of "timestamp" tag marking moments in time
  - Intervals can be emulated using the above tag
  - If data has a continuous characteristic (such as a sine wave) user can define interpolation type in the setup section
- Mobility description
  - Positions at successive points given
  - User specifies interpolation type between points
  - If granularity is not enough/too fine grained simulators/other tools will resample the data

Example

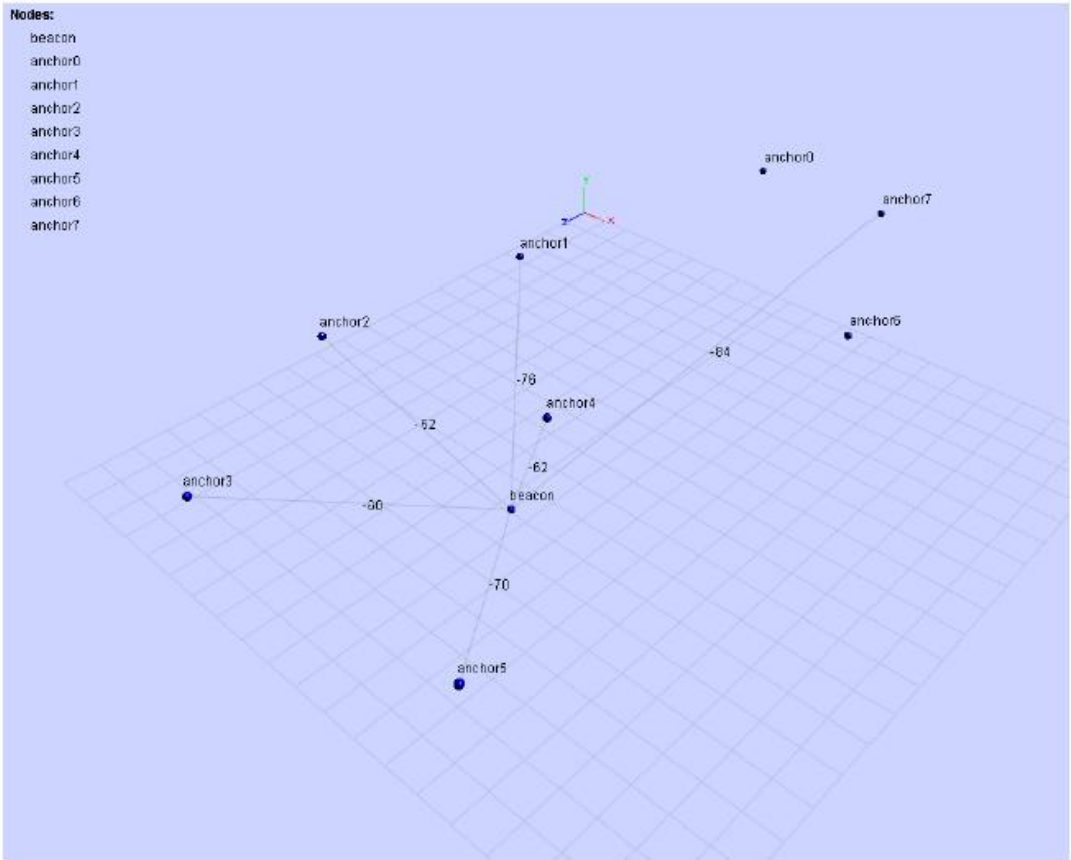
```
<trace id="...">
  <timestamp>2</timestamp>

  <node id="...">
    <position>
      <x>2</x><y>4</y><z>6</z>
    </position>

    <data key="ranger">12</data>

    <data key="...">...</data>
  </node>

  <link source="..." target="...">
    <rssi>12</rssi>
    <data key="lqi">100</data>
  </link>

  <timestamp>4</timestamp>
</trace>
```

Schema Validation

- Automatic checking of WiseML files
- Two levels:
  1. WiseML schema (check for the tags and presence of all values)
     - W3C XML Schema available (spots missing/incorrect tags)
     - Relax NG schema (checks also the correct tags nesting and attributes)
  2. Temporal ordering of events in the file
     - Schematron script (in progress...)

Visualisation Tool

- It is useful to visualise the topology of the network and the evolution of the network
- A collection of TCL/TK scripts that parse WiseML files



Programming Tool 1/8

- A Java Tool for parsing WiseML files – called EasySense
- Uses SimpleXML framework to manipulate XML files
- Uses Hibernate framework to store/load data to RDBMS
- Uses JAX-WS to expose the basic functinality defined by Wisebed API as web services

Programming Tool 2/8

- Models the WiseML with Objects
  1. Setup
  2. Scenario
  3. Trace
  4. Node
  5. Link
  6. Capability
  7. Position
  8. Timeinfo
  9. ...

Programming Tool 3/8

- Uses the concept of a "Store" to hold information for each section
  1. NodeStore
  2. EdgeStore
  3. CapabilityStore

Programming Tool 4/8

```
1 final ParseXML parse = new ParseXML();
2 parse.readFile("experiment.wiseml");
3
4 for (Node node : NodeStore.getInstance().list()) {
5   System.out.println(node.getID());
6 }
```

Programming Tool 5/8

- Uses Hibernate so that data can persist

```
1 final ParseXML parse = new ParseXML();
2 parse.readFile("experiment.wiseml");
3
4 final StoreToDatabase storeDb = new StoreToDatabase();
5
6 for (Node node : NodeStore.getInstance().list()) {
7   System.out.println(node.getID());
8   storeDb.storeNode(node);
9 }
```

Programming Tool 6/8

```
1  CreateXML create = new CreateXML();
2
3  List<Node> nodeList = new ArrayList<Node>();
4  List<Link> edgeList = new ArrayList<Link>();
5  List<Capability> capList = new ArrayList<Capability>();
6
7  Capability cap1 = new Capability("urn:wisebed:node:capability:temp",
8  capList.add(cap1);
9
10 Position position = new Position(1.23, 1.56, 1.77);
11
12 Data data = new Data("urn:wisebed:node:capability:temp");
13
14 Node node1 = new Node("urn:wisebed:node:tud:M4FTR",
15                        position,
16                        "gw1", "blinkfast.tnode",
17                        "fast blinking node", "TNide v4", capList, data
18
19 nodeList.add(node1);
```

Programming Tool 7/8

```
1  List<Capability> capList2 = new ArrayList<Capability>();
2
3  Capability cap2 = new Capability("urn:wisebed:link:capability:encrypt
4  capList2.add(cap2);
5
6  Rssi rssi = new Rssi("decimal", "dBam", "-120");
7
8  Link edge = new Link("urn:wisebed:node:tud:330006",
9                       "urn:wisebed:node:tud:330009",
10                      "true", "false", rssi, capList2);
11
12 edgeList.add(edge);
```

Programming Tool 8/8

```
1  Origin origin = new Origin(position, 5, 0);
2
3  Timeinfo time = new Timeinfo("9/7/2009", "19/12/2010", "seconds");
4
5  NodeDefaults ndefault = new NodeDefaults("node", node1);
6
7  LinkDefaults ldefault = new LinkDefaults("link", edge);
8
9  Setup setup = new Setup(origin, time, "cubic", "wiseml example",
10                          ndefault, ldefault, nodeList, edgeList);
11
12 // ...
13
14 create.writeXML("sample.wiseml", setup, scenario, trace);
```

Conclusions

- WiseML = common data format to be used between tools and real testbeds
- In progress
  - Website where people can upload their data sets
  - Representative data sets for problems ranging from temperature monitoring to target tracking