



ANDES: an Analysis based DEsign tool for wireless Sensor networks

Vibha Prasad, Sang H. Son, John A. Stankovic,
Jörgen Hansson

{vibha, son, stankovic}@cs.virginia.edu,
hansson@sei.cmu.edu

Computer Science
University of Virginia

University of Virginia



Outline

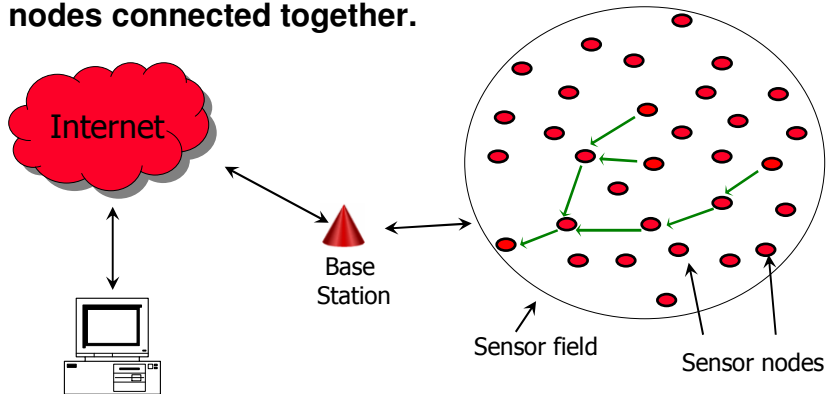
- Introduction
- Analysis based design: Using AADL/OSATE
- Analysis of wireless sensor networks (WSNs)
- Modeling WSNs: Implementation of ANDES
- Case Study: Tracking Analysis
- Summary
- Future Work

University of Virginia



What is a Wireless Sensor Network (WSN)?

- A large no. of low cost, low power, small sensor nodes connected together.

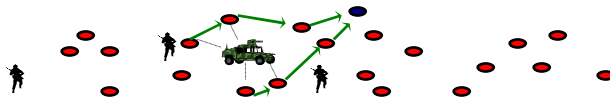


University of Virginia



Applications

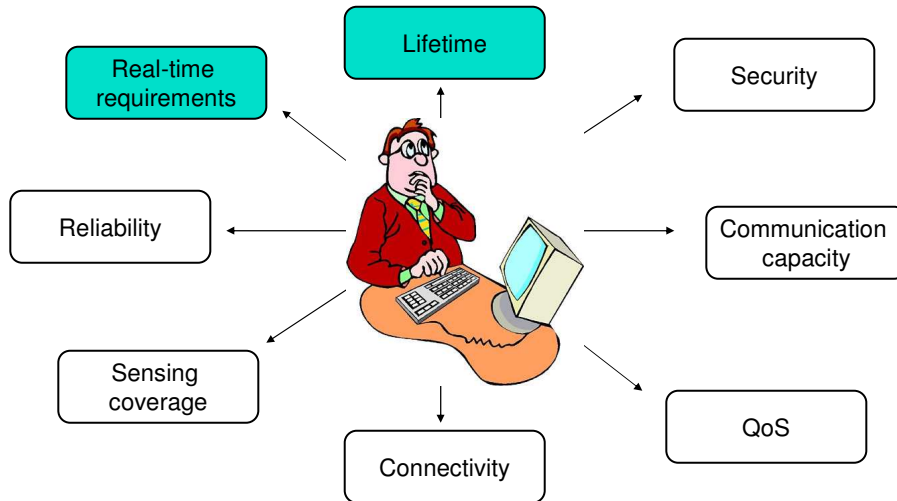
- Military Surveillance
- Smart Home
- Seismic Structural Monitoring
- Industrial Process Monitoring
- Transportation (Traffic Monitoring)
- Habitat and Ecosystem Monitoring
- Monitoring groundwater contamination



University of Virginia



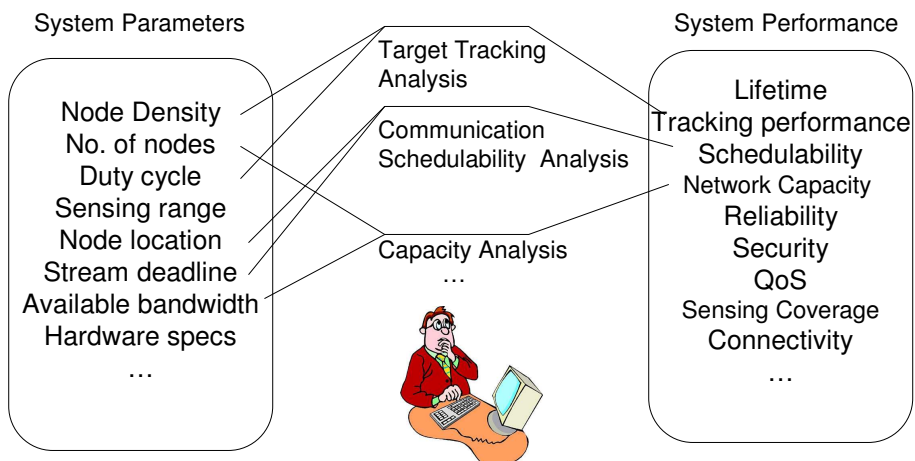
System Designer: design decisions



University of Virginia



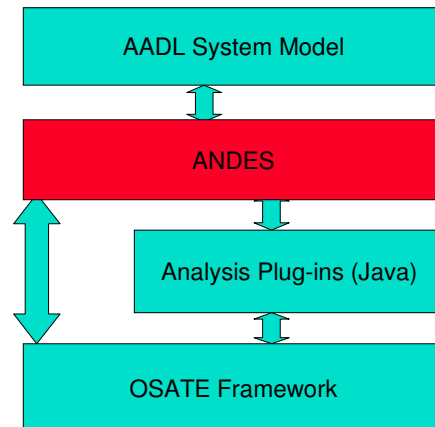
Analysis based Design



University of Virginia



Using AADL/OSATE framework for Analysis



University of Virginia



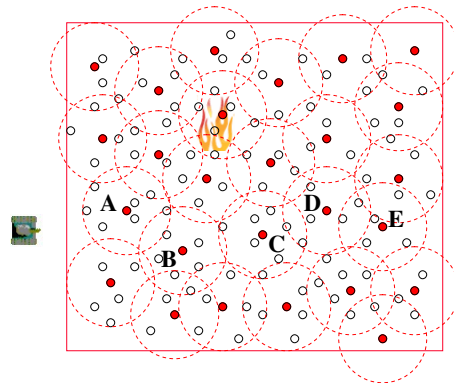
Analysis of WSNs: Current plug-ins

- **Target detection analysis: high-level model**
 - System parameters: node density, duty cycle
 - System performance: probability of detection, average detection delay
- **Communication Schedulability analysis: detailed system-level model**
 - System performance: path of streams, stream deadlines, period, latency.
 - Result: if the streams are schedulable within their deadlines

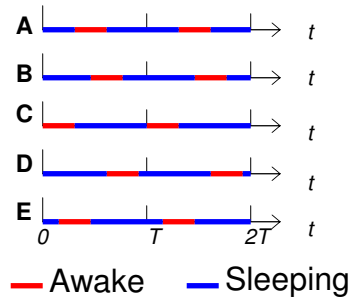
University of Virginia



Tracking Analysis



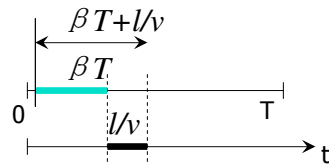
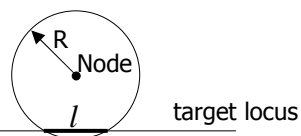
Same period,
random wakeup times



University of Virginia



Probability of Detection



Probability of detection
 $(\beta T + l/v) / T$, or $\beta + l/vT$

l/v Time interval when the target is in the sensing area

βT Time interval when the node is awake in one period

University of Virginia



Results

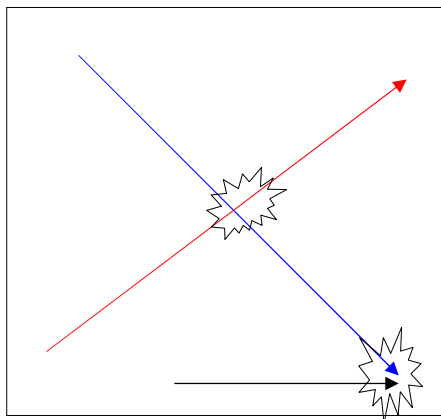
Speed of the target	From	Probability of Detection	Average Detection Delay
Fast	Outside	$1 - e^{-2RLd(\beta + \frac{\pi R}{2vT})}$	$\frac{e^{-\beta\pi R^2 d}}{d(2R\beta v + \frac{\pi R^2}{T})}$
	Inside	$1 - e^{-(2RL + \pi R^2/2)d(\beta + \frac{\pi RL}{vT(2L + \pi R/2)})}$	$\frac{e^{-\beta\pi R^2 d/2}}{d(2R\beta v + \frac{\pi R^2}{T})}$
Slow	Outside	$1 - e^{-2RLd(\beta + \frac{\pi R^2 + k}{2RvT})}$	$\frac{e^{-\beta\pi R^2 d/2}}{d(2R\beta v + \frac{\pi R^2}{T})} [1 - \frac{ke^{(-2R\beta vT + \pi R^2)(1-\beta)d/2}}{2R\beta vT + \pi R^2 + k}]$
	Inside	$1 - e^{-(2RL + \pi R^2/2)d(\beta + \frac{\pi R^2 L + m}{vTR(2L + \pi R/2)})}$	$\frac{e^{-\beta\pi R^2 d}}{d(2R\beta v + \frac{\pi R^2}{T})} [1 - \frac{ke^{(-2R\beta vT + \pi R^2)(1-\beta)d}}{2R\beta vT + \pi R^2 + k}]$

where $a = (1 - \beta)vT/2$, $k = 2a\sqrt{R^2 - a^2} - 2R^2 \cos^{-1}(\frac{a}{R})$, $m = (L - a)k$

Reference: Qing Cao, Ting Yan, John Stankovic, Tarek Abdelzaher, Analysis of Target Detection Performance for Wireless Sensor Networks, International Conference on Distributed Computing in Sensor Networks (DCOSS 2005), June 2005.



Communication Schedulability Analysis



- Period: 0.2 second
Delay per hop: 4ms
Deadline: 20 s
- Period: 0.4 second
Delay per hop: 3 ms
Deadline: 30 s
- Period: 0.8 second
Delay per hop: 2ms
Deadline: 20 s



Scheduling algorithm

- **NP Hard Problem**
 - Equivalent to the cylinder packing problem.
- **Urgency heuristics are used to choose the most important stream for scheduling.**
- **Scheduling strategy:**
 - Find and choose the most *important* stream.
 - Find and allocate free link entries to the stream.
 - Update every streams' importance based on the new allocation.
 - If there are streams waiting to be scheduled, repeat.

University of Virginia



Modeling WSNs

- **At application level:**
 - WSN: System
 - Model external factors like tanks moving in the battlefield.
- **At system level:**
 - WSN: System
 - Sensor nodes: Devices
 - Wireless communication? Routing protocols?
- **At a lower level:**
 - WSN: System
 - Sensor nodes: System
 - Contain sensors, memory, processor, radio.

University of Virginia

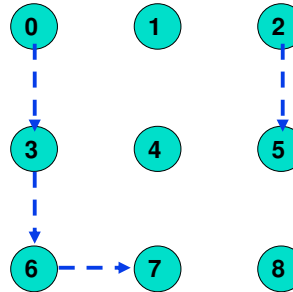


Example AADL Model: WSN System

```

device Node
features
  in_port: in data port sensor_data;
  out_port: out data port sensor_data;
flows
  flow_source: flow source out_port ;
  flow_out: flow path in_port -> out_port;;
  flow_sink: flow sink in_port;
end Node ;
:
conn03: data port node0.out_port -> node3.in_port ;
:
flows
  stream1 : end to end flow node0.flow_source ->
  conn03 -> node3.flow_out -> conn36 ->
  node6.flow_out -> conn67 -> node7.flow_sink ;
  stream2 : end to end flow node2.flow_source ->
  conn25 -> node5.flow_sink ;

```



WSN model with data streams

```

data sensor_data
end sensor_data
property set wsn, property is
  qm1, auto, auditranger applies to (system);
  qm2, deployment, auditranger applies to (system);
  md1, range, auditranger applies to (system);
  md2, range, auditranger applies to (system);
  active, nodes, auditranger applies to (system);
  node, no, auditranger applies to (device);
end wsn, property;

device Node
features
  in_port: in data port sensor_data;
  out_port: out data port sensor_data;
flows
  flow_source: flow source out_port ;
  flow_out: flow path in_port -> out_port;
  flow_sink: flow sink in_port;
end Node ;

device Implementation Node.n0
properties
  wsn, property:node, no == 0;
end Node.n0;

device Implementation Node.n1
properties
  wsn, property:node, no == 1;
end Node.n1;

device Implementation Node.n2
properties
  wsn, property:node, no == 2;
end Node.n2;

device Implementation Node.n3
properties
  wsn, property:node, no == 3;
end Node.n3;

device Implementation Node.n4
properties
  wsn, property:node, no == 4;
end Node.n4;

device Implementation Node.n5
properties
  wsn, property:node, no == 5;
end Node.n5;

device Implementation Node.n6
properties
  wsn, property:node, no == 6;
end Node.n6;

device Implementation Node.n7
properties
  wsn, property:node, no == 7;
end Node.n7;

device Implementation Node.n8
properties
  wsn, property:node, no == 8;
end Node.n8;

system wsn
connections
  conn03: data port node0.out_port -> node3.in_port;
  conn05: data port node0.out_port -> node5.in_port;
  conn07: data port node0.out_port -> node7.in_port;
  conn25: data port node2.out_port -> node5.in_port;
flows
  stream1 : end to end flow node0.flow_source => conn03 -> node3.flow_out => conn05 -> node5.flow_out => conn07 -> node7.flow_sink ;
  stream2 : end to end flow node2.flow_source => conn25 -> node5.flow_sink ;
properties
  wsn, property:q1, no == 2;
  wsn, property:active, nodes == 4;
  wsn, property:qm1, deployment == false;
  wsn, property:md1, range == 1.2;
  wsn, property:md2, range == 1.2;
end wsn, system;

```

Complexity increases for the System designer!!!

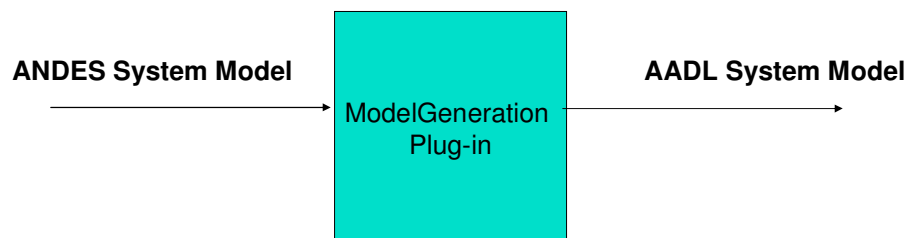


Initial issues with AADL for WSN modeling

- **Scalability:**
 - Define a large-scale WSN without explicitly defining the implementation of each of them separately
- **Connectivity:**
 - Support the semantics of radio range and interference range in wireless communication
- **WSN coverage shape:**
 - Defining a high-level deployment coverage shape rather than define the location of each node



ModelGeneration Plug-in





Nodes/Scalability

```
System wsn
System wsn;
```

```
system implementation wsn.wsn1
properties
```

```
wsn_property::grid_size => 3;
wsn_property::active_nodes => 9;
wsn_property::grid_deployment => true;
wsn_property::radio_range => 1.2;
wsn_property::interference_range => 2.5;
```

```
end wsn.wsn1;
```

```
device Node
end Node ;
```

```
device implementation Node.node0
properties
```

```
wsn_property::node_no => 0;
end Node.node0 ;
...
end Node.node8 ;
```

```
system implementation wsn.wsn1
subcomponents
```

```
node0: device Node.node0;
...
node8: device Node.node8;
```

University of Virginia



Data Streams/connectivity

```
property set stream_property is
source_node_no: aadlinteger applies to
(device);
sink_node_no: aadlinteger applies to
(device);
end stream_property;
```

```
device Stream
end Stream;
```

```
device implementation Stream.stream1
properties
```

```
stream_property::source_node_no =>
0;
stream_property::sink_node_no => 2;
```

```
end Stream.stream1;
```

```
system implementation wsn.wsn1
```

```
...
```

```
connections
```

```
conn01: data port node0.out_port ->
node1.in_port;
conn12: data port node1.out_port ->
node2.in_port;
```

```
...
```

```
flows
```

```
stream1 : end to end flow
node0.flow_source -> conn01 ->
node1.flow_out -> conn12 ->
node2.flow_sink;
```

```
...
end wsn.wsn1;
```

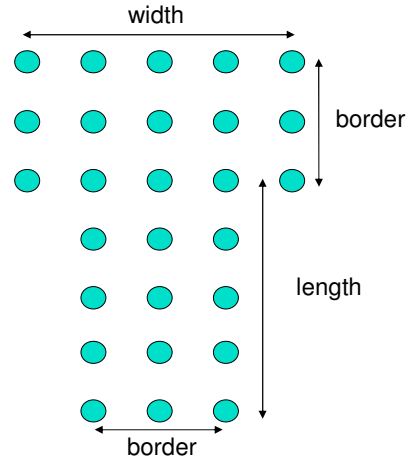
University of Virginia



Coverage Shape

```
property set grid_property is
  topology : aadlinteger applies to (system);
  length : aadlinteger applies to (system);
  width : aadlinteger applies to (system);
  border : aadlinteger applies to (system);
end grid_property ;
```

```
system implementation wsn.wsn1
  properties
  ...
  wsn_property::grid_size => 5;
  wsn_property::grid_deployment => false;
  grid_property::topology => 3;
  grid_property::length => 5;
  grid_property::width => 5;
  grid_property::border => 3;
end wsn.wsn1 ;
```



University of Virginia



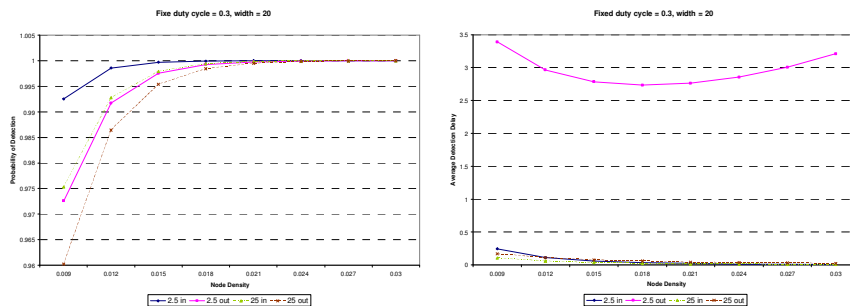
Case Study: Tracking Analysis

- **Area: 100m X 100m**
- **Velocity:**
 - Slow: 2.5 m/s
 - Fast: 25 m/s
- **System Performance Requirements**
 - Probability of Detection: $\geq 98\%$ for all cases
 - Average Detection Delay: Slow: 3s, Fast: 1s
- **System Parameters**
 - Node Density: 0.0090 – 0.030 (90 – 300 nodes)
 - Duty Cycle: 0.1 - 1

University of Virginia



Fixed Duty cycle at 0.3



University of Virginia



Summary

- We developed an analysis-based design tool, ANDES, for WSNs using the AADL/OSATE framework.
- AADL/OSATE can be successfully used for WSNs with some additional support for WSNs.
- ANDES currently has analysis plug-ins for target detection analysis and communication schedulability analysis.
- ModelGeneration can be extended to support additional features.

University of Virginia



Future Work

- **Incorporate analysis plug-ins in ANDES to make it useful for the designers.**
- **Build a generic WSN model for all analyses.**
- **Intelligent analysis: iteratively refine the input to the analysis based on performance results.**



Questions?

Thank You