SIMULATION TOOLS

OVERVIEW

Wireless Sensor Network Simulators can be categorized into two groups, including the network simulator and emulator.

Network Simulator is to evaluate the protocol performance in WSN.

Network Simulator: OMNET++, NS-2, Javasim, J-Sim, Glomosim, SensorSim (withdrawn), SENS, EmStar, etc.

Emulator is to simulate the performance on the real hardware platform of WSN nodes.

Emulator: TOSSIM, ATEMU, and AVRORA.

OMNET++

What is OMNeT++?

- OMNeT++ is a Simulator
- For **discrete event** network
- It is **object oriented** and **modular**
- Used to simulate
  - Modeling of **wired and wireless** communication networks
  - Protocol modeling
  - Modeling of queuing networks etc.

- Modules are connected using **gates** to form **compound module**
  - In other system sometimes gates are called **port**

Modeling concepts

- **Modules** are **communicate** with **message** passing
- **Active modules** are called **simple modules**
- **Message** are sent through **output gates** and receive through **input gates**
- **Input and output gates** are linked through **connection**
- Parameters such as **propagation delay, data rate and bit error rate**, can be assigned to **connections**

MODULES

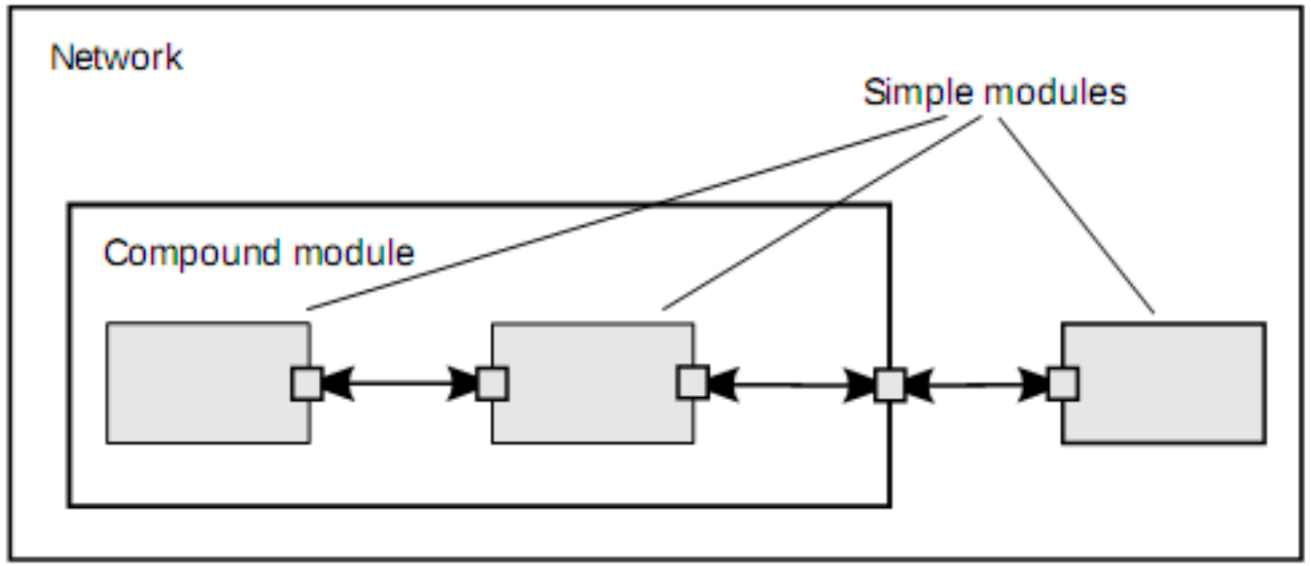


Fig – simple and compound module

Module

- In hierarchical module, the **top level module is system module**
- System module contains **sub modules**
- Sub modules contains sub modules themselves
- Both simple and compound modules are **instance** of module type

Message, gets and links

- **Module** communicate by exchanging **message**
- **Message** can represent **frames or packets**
- **Gates** are the input and output **interface of modules**
- **Two sub modules** can be connected by **links** with **gates**
- **Links = connections**
- **Connections** support the following parameter
    - **Data rate, propagation delay, bit error rate, packet error rate**

Parameters

- Modules parameters can be assigned
    - in either the **NED files** or
    - the configuration file **omnetpp.ini.**
- Parameter can take string, numeric or Boolean data values or can contains XML data trees

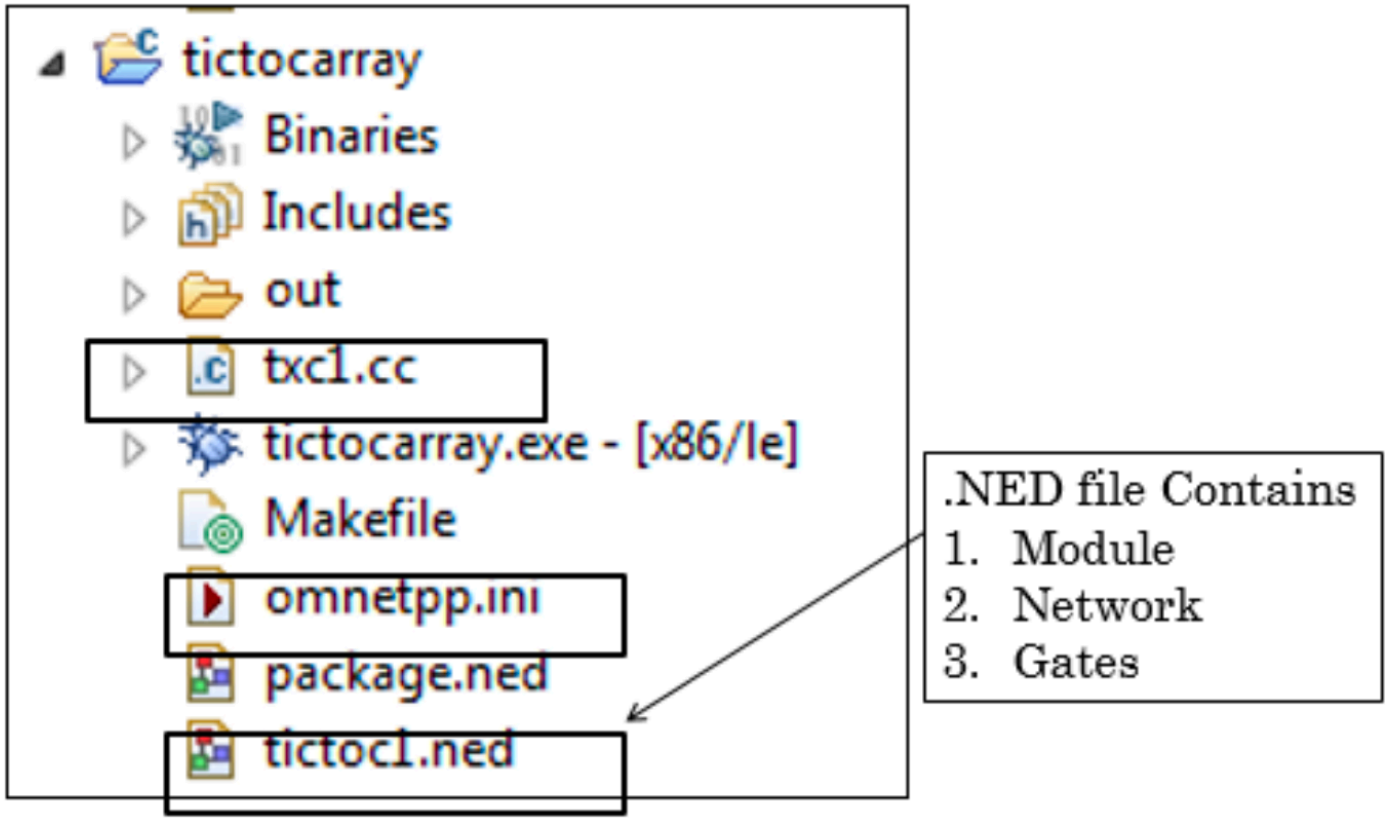
Classes that are part of simulation class library

The following classes are the part of simulation class library

- Module, gates, parameter, channel
- Message, packet
- Container class (e.g. queue and array)
- Data collection classes
- Statistics and distribution estimated classes
- Transition and result accuracy detection classes.

OMNeT++ Consists of

- NED language topology description(s**)(.ned files)**
- Message definitions **(.msg files)**
- Simple module sources. They are C++ files, with .h/.cc suffix.



How OMNeT++ Works?

- When Program started
  - **Read all NED file**s containing model topology
  - Then it **reads a configuration file**(usually called omnetpp.ini)
- **Output** is written in **result file**
- **Graph is generated from result file** using Matlab, Phython etc

NED Features

NED has several features which makes it **scale well** to large project

- **Hierarchical**
- **Component**-based
- **Interfaces**
- **Inheritance**
- **Packages**
- Metadata **annotation**

The Network

- Network consists of
  - Nodes
  - Gates and
  - Connections

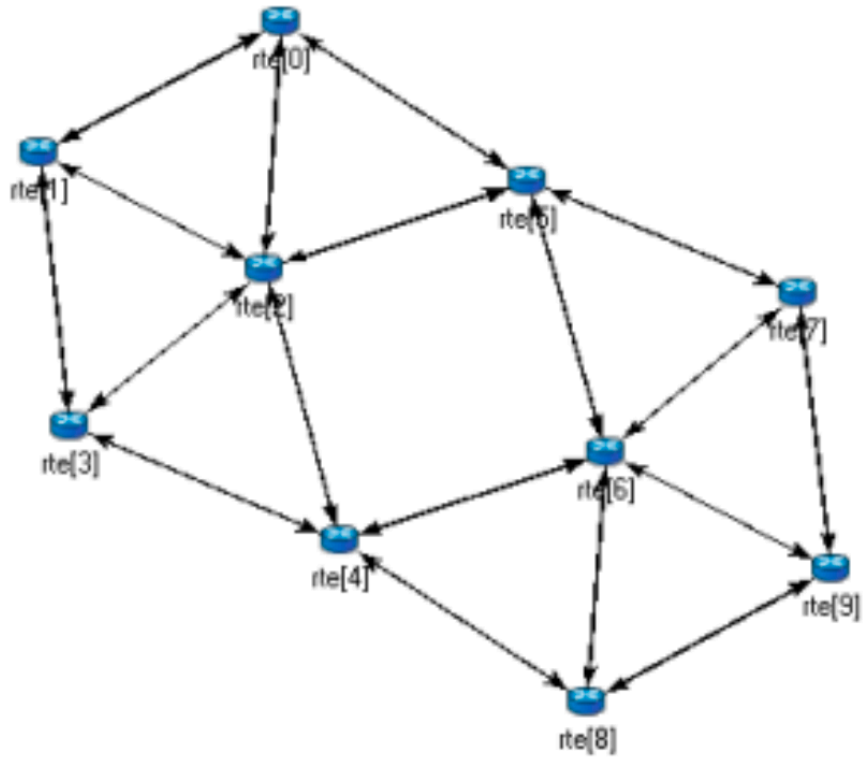


Fig: The network

SIMULATION TOOLS

Existing Simulators

JavaSim:

- Pros
  - Very modular
  - Easy to use
- Cons
  - Geared for wired inter-networks
  - No wireless support, not efficient due to overhead

GlomoSim:

- Specific for mobile wireless networks.
- Built as a set of libraries. The libraries are built in Parsec(a C-based discrete event simulation language).
- Layered architecture with easy plug-in capability.

SSFNet and Glomosim are not better than NS-2 in terms of design and extensibility.

SIMULATORS

NS-2: De facto standard for network simulations

- Does support wireless simulations
- A primitive energy model is present.
- Object oriented design and Lots of documentation.
- Uses Tcl to specify the Components, and Otcl to glue them together.

Cons:

- Difficult to use and learn
- Interdependency among modules pose difficult to implement new protocols.
- Originally built for wired networks, later extended for wireless.
- Supposedly, does not work well for large topologies.

SIMULATORS: NS-2

NS-2 is well known general-purpose network simulator developed in 2002.

It is an object-oriented discrete event simulator based on a combination of C++ and Otcl.

Many extensions are implemented in NS-2 for WSN research, like radio energy model, S-MAC stack, etc.

NS-2 Webpage: http://www.isi.edu/nsnam/ns/

SIMULATORS: NS-2 CONT.

- Advantage:

  NS-2 supports several different types of networks, since it is a general-purpose network simulator. Also, NS-2 has a large number of users, and the technical support is the best among all simulators.

- Disadvantage:

  NS-2 is not specifically designed for wireless sensor network. Its performance on WSN is not better than other simulators.

SIMULATORS: SENSE

Battery Model:

- Linear Battery
- Discharge Rate Dependent and/or Relaxation Battery

Application Layer:

- Random Neighbor
- Constant Bit Rate

Network Layer:

- Simple Flooding
- A simplified version of AODV/ADOV (Ad hoc On Demand Distance Vector) without route repairing
- A simplified version of DSR (dynamic source routing) without route repairing
- Self Selective Routing (SSR)
- Self Healing Routing (SHR)

SENSE

MAC Layer:

- NullMAC
- IEEE 802.11 with DCF

Physical Layer: Duplex Transceiver

Wireless Channel:

- Free Space
- Adjacency Matrix

Simulation Engine: CostSimEng (sequential)

https://www.nrl.navy.mil/itd/ncs/products/sensorsim

SENSORSIM

Extension to NS - 2.

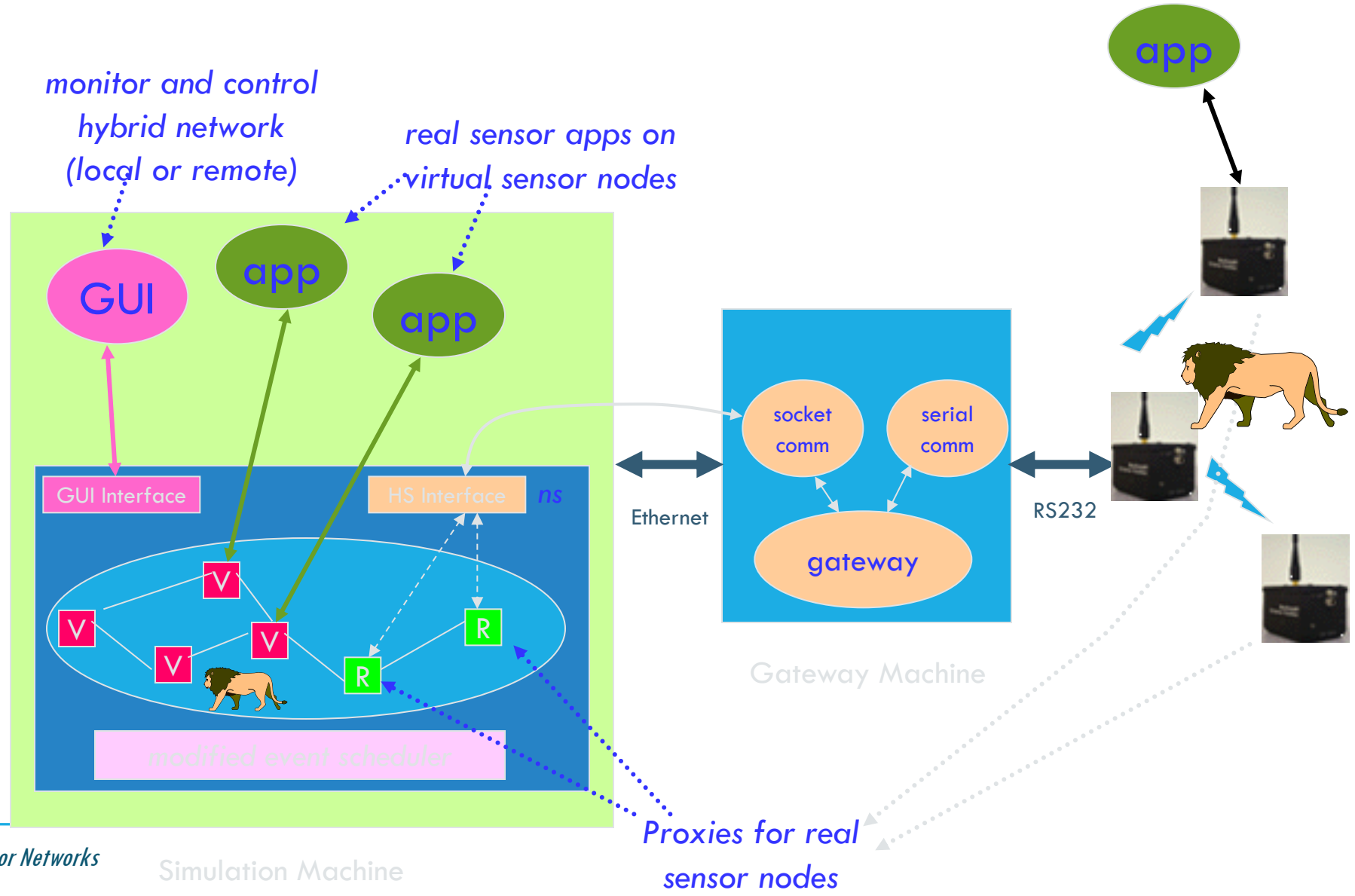
- Provides battery models, radio propagation models and sensor channel models.
- Provides a lightweight protocol stack.
- Has support for hybrid simulation.
- Must be integrated with NS - 2.

SensorSim: A Simulation Framework for Networks
Sensor

Sung Park, Andreas Savvides, and Mani B. Srivastava
Electrical Engineering Department, University of California in Los Angeles
7702-B, Boelter Hall, Box 951594, Los Angeles, CA 90095-1594
Emaih ~spark, asavvide, mbs}@ee.ucla.edu

http://compilers.cs.ucla.edu/emsoft05/ParkSavvidesSrivastava00.pdf

SENSORSIM ARCHITECTURE



SENSORSIM ARCHITECTURE OVERVIEW

Sensor NW has three types of nodes:

- ***Sensor nodes: monitor immediate environment, with many***

transducers

- ***Target nodes: generate various stimuli for sensor nodes***
- ***User nodes: client and administration of sensor network***

Separate channels:

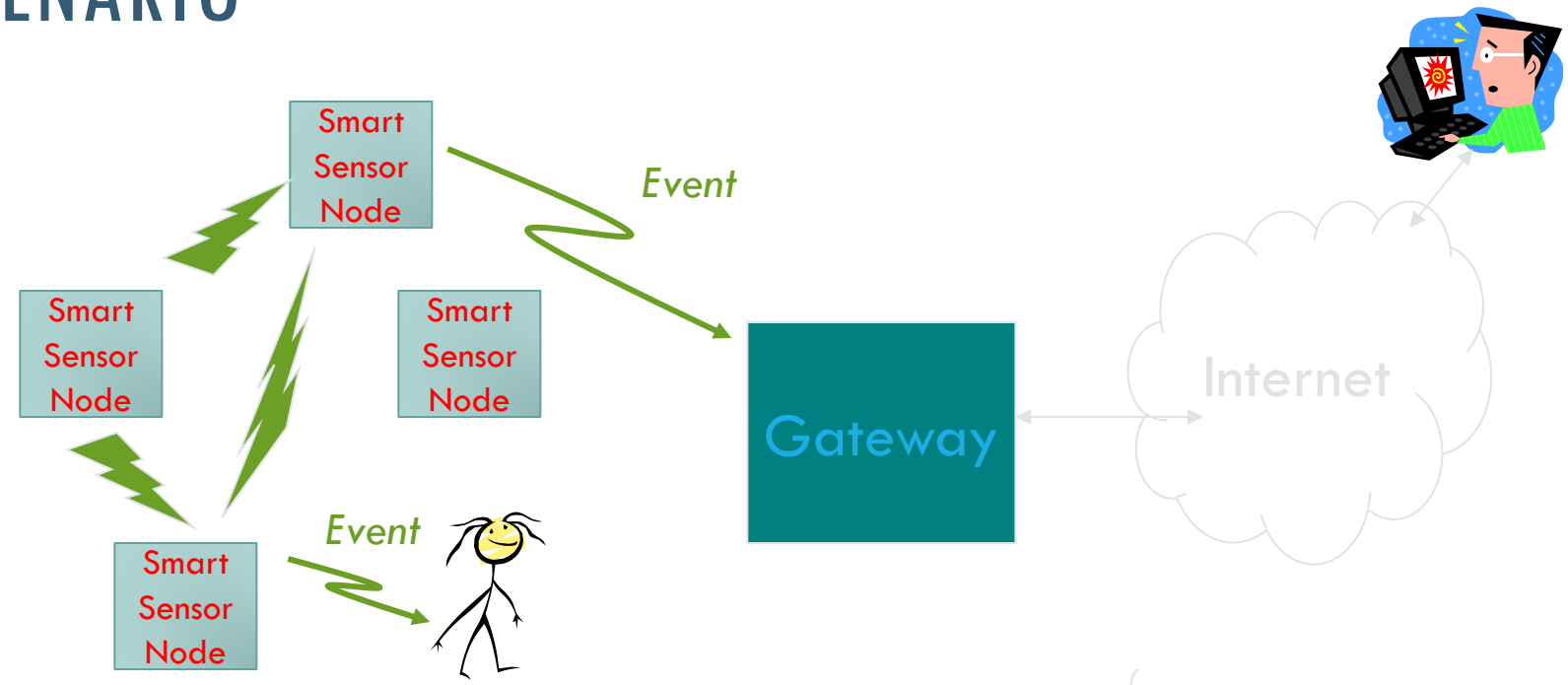
- ***Sensor channels: communication among sensor nodes and target***
- ***Network channels: to user node or gateways and onward***

transmission to other network.

- Concurrent transmission possible
- Easier to model complex behavior of sensor nodes, reaction to

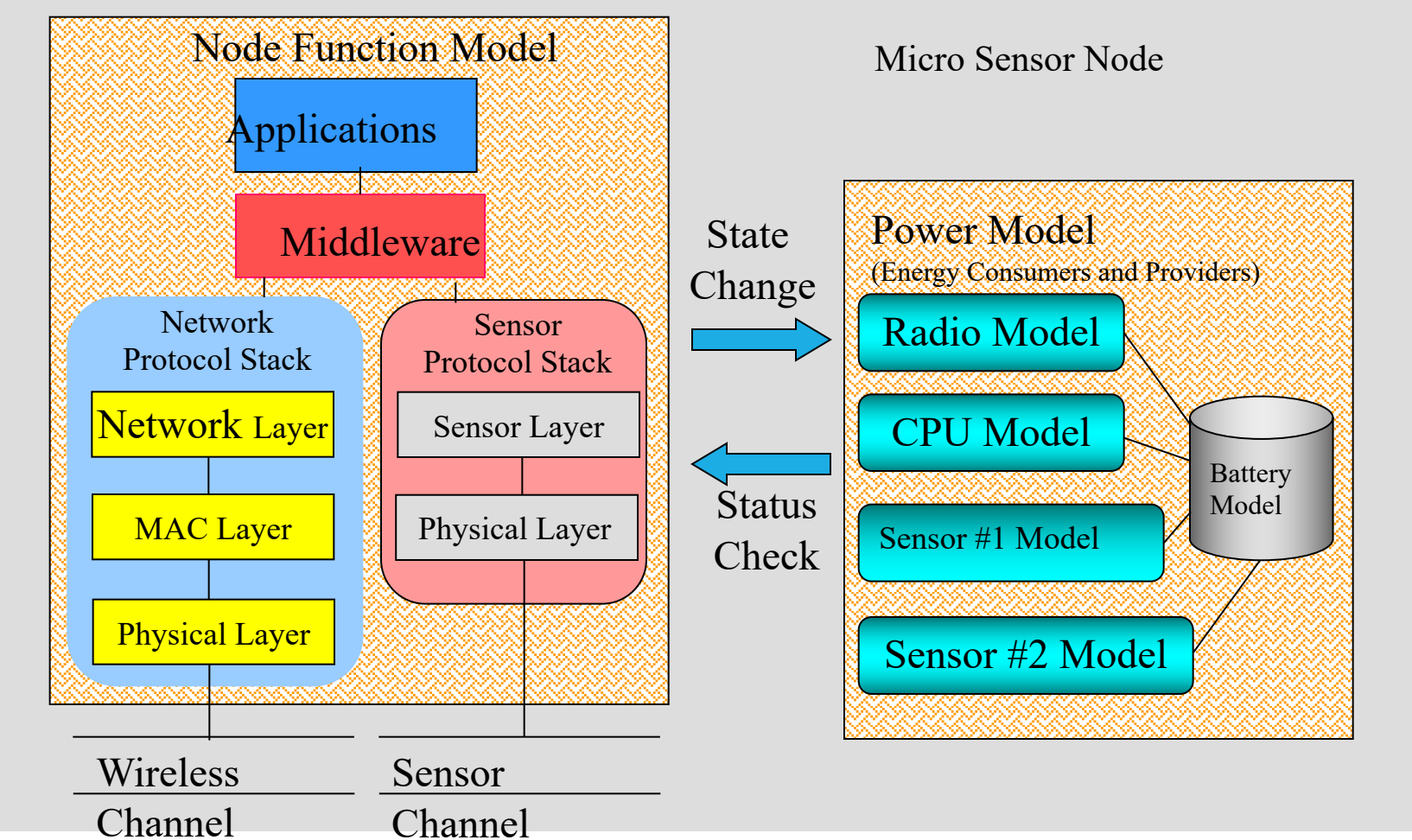
multiple sensor signals.

TYPICAL SCENARIO



- Nodes coordinator: local processing among neighbors to combine their results
    - lower network traffic, higher-level sensory tasks
- Application: large scale, dynamically changing, robust sensor colonies

SENSOR NODE MODEL IN SENSORSIM

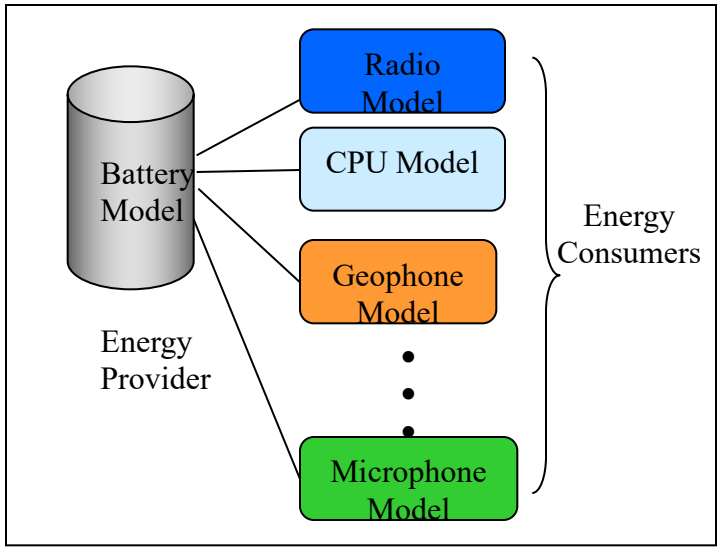


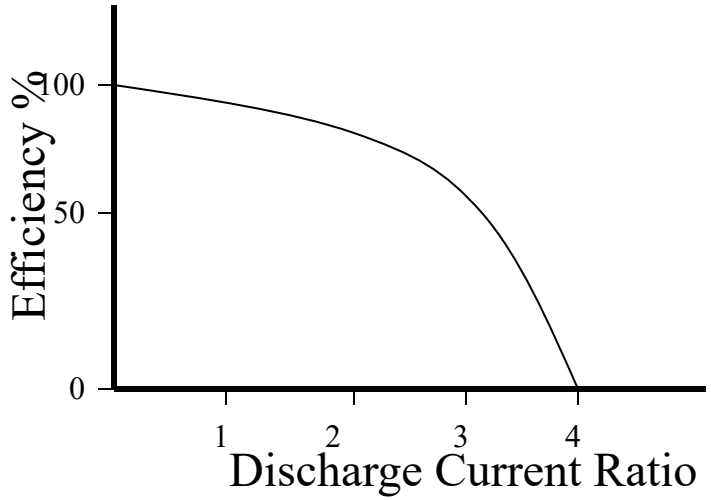
BATTERY MODEL

Ideally battery capacity is decided by the amount of active material stored in a cell

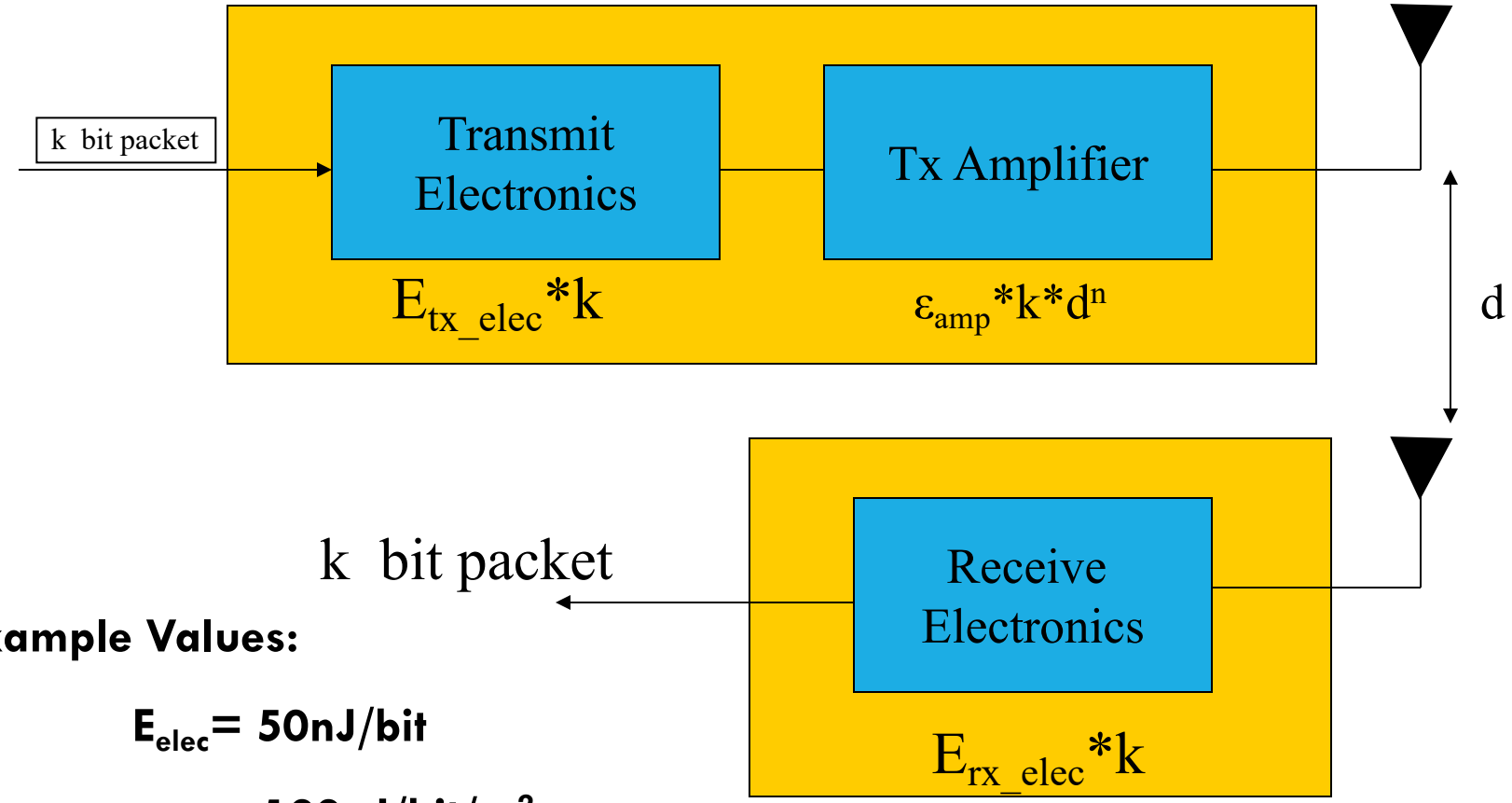
In reality, this depends on how the battery is discharged

- **discharge rate**
- **discharge profile**
- **operating voltage and power drained**





RADIO MODEL



Example Values:

E_{elec}= 50nJ/bit

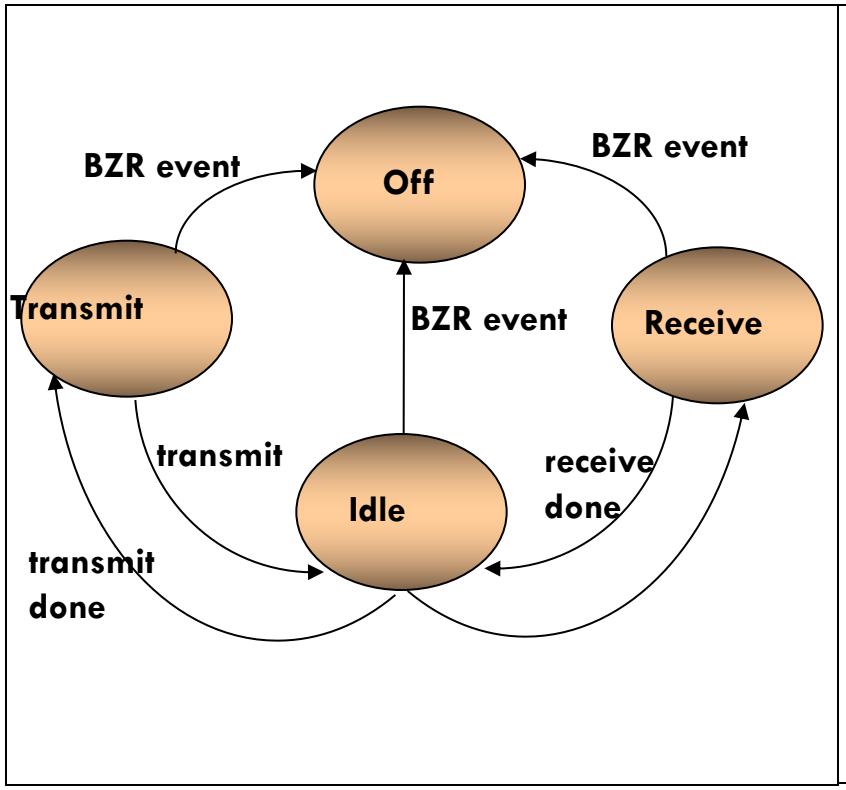
ε_{amp}= 100pJ/bit/m^2

POWER MANAGEMENT SIMULATION

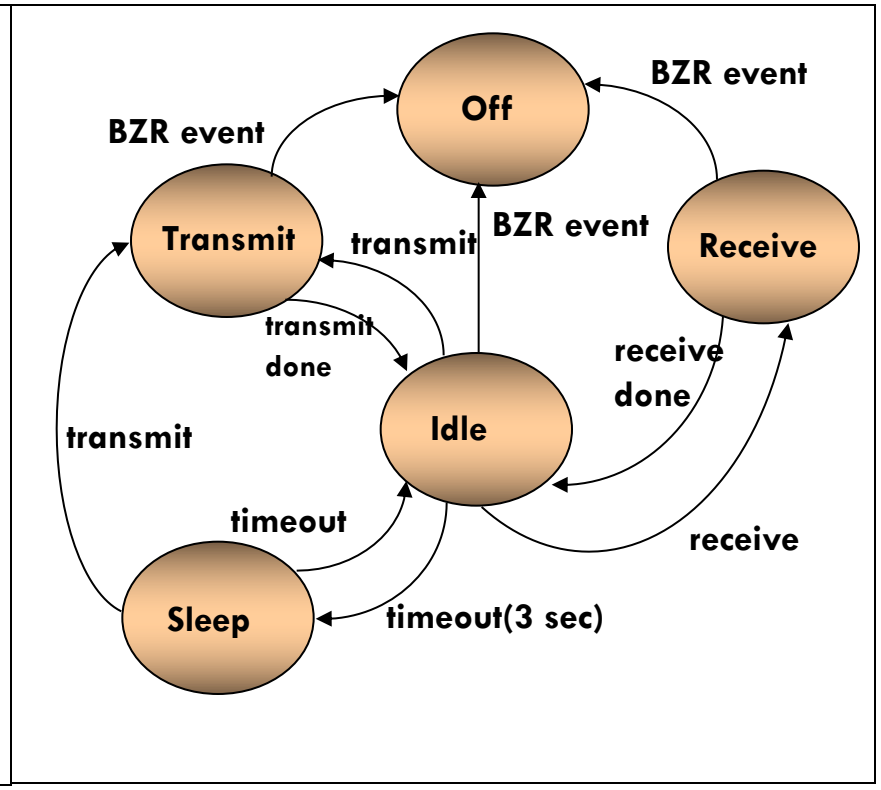
- Battery Zero Reached(BZR) Event:
  - The BZR event is used to indicate when the energy stored in the battery is completely drained.
  - The BZR event is created by the Battery Model every time the battery model handles the BCP event.
  - Battery Threshold Reached (BTR) Event: The BTR event is similar to BZR event. The difference comes from the fact that BTR event is inserted when the energy level of the battery reaches a certain threshold.

  - Battery Change Power(BCP) Event: The BCP Event is created by the energy consumers to change their rate of power draw.

Power Management Model

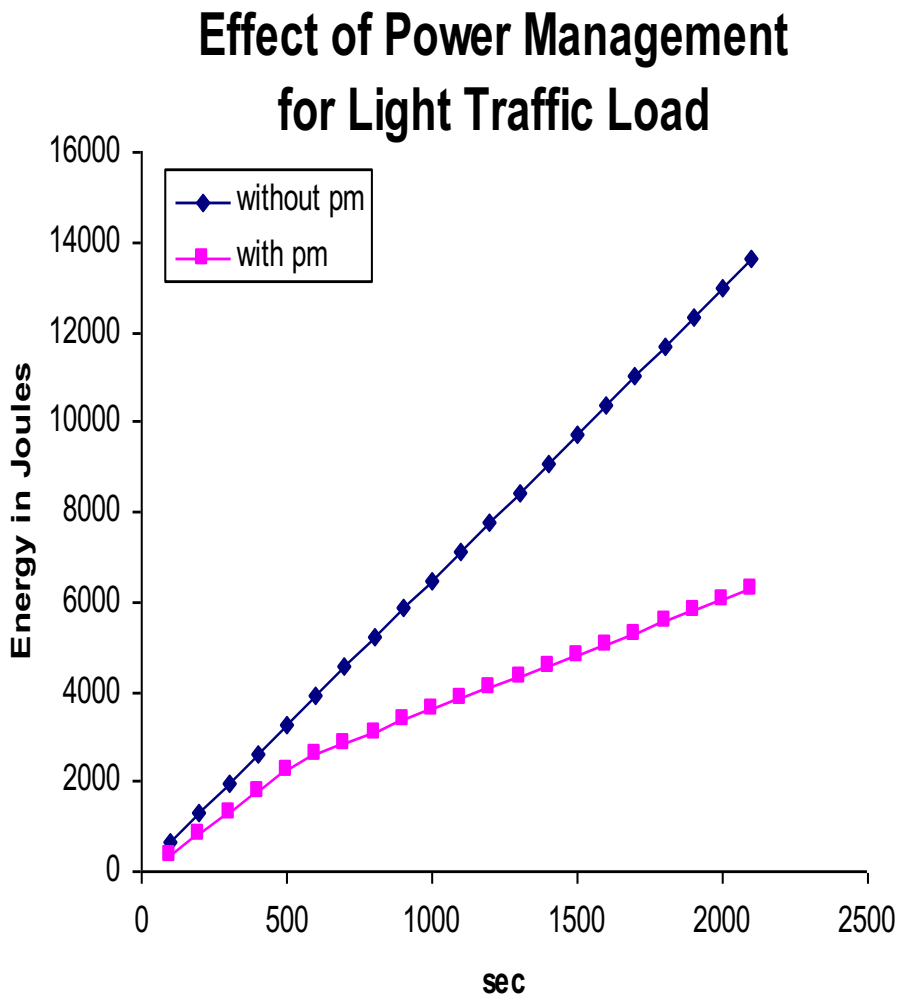


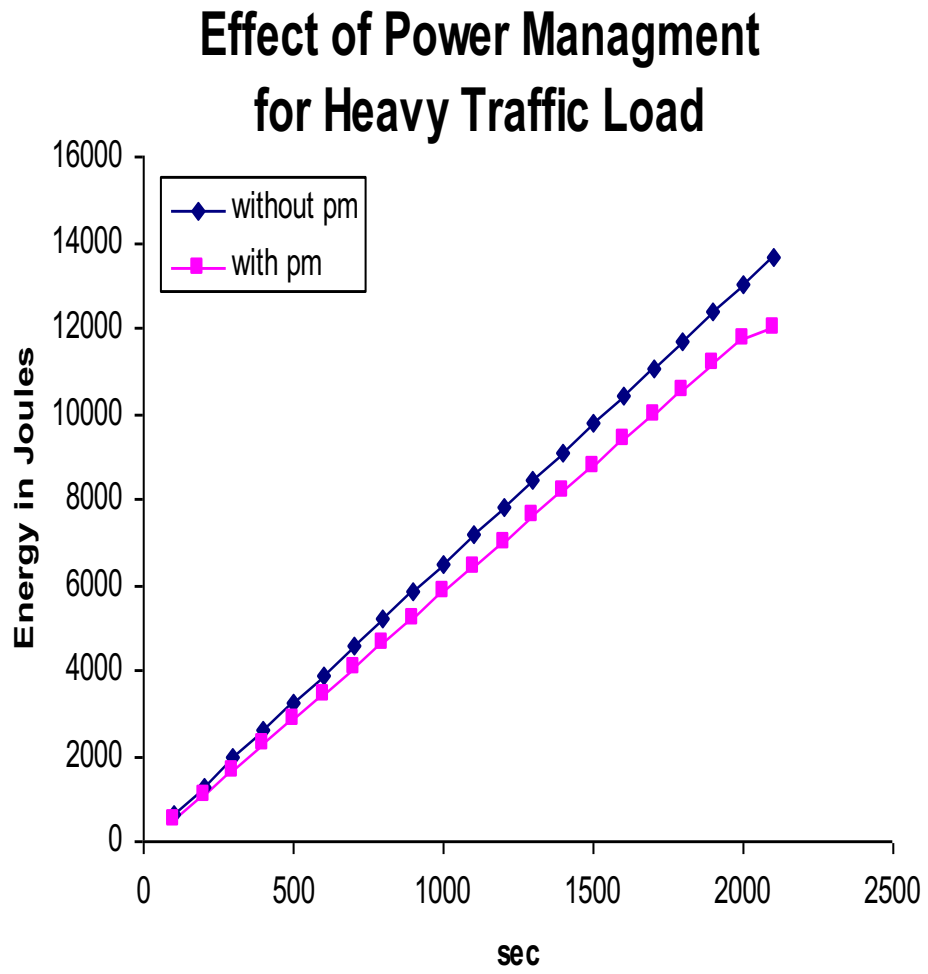
Without Power Management



With Power Management

POWER MODEL SIMULATION RESULTS





SIMULATORS: J-SIM

J-Sim is an open-source, component-based compositional network simulation environment fully developd in Java by UIUC and Ohio State University.

J-Sim is a dual-language simulation environment in which classes are written in Java and glued together using Tcl/Java.

J-Sim has specific wireless sensor network package to support a typical wireless sensor network environment.

J-Sim Webpage: http://www.j-sim.org/

SIMULATOR: J-SIM CONT.

- Advantage:

  J-Sim is totally developed in JAVA, and then can be easily installed on different platforms. Also, J-Sim has already included specific wireless sensor network packages with both battery and power model.

- Disadvantage:

  the wireless sensor network environment is fixed in J-Sim, and it is the only topology supported in J-Sim. Also, only IEEE 802.11 is supported in J-Sim as the MAC protocol for WSN. Actually, the most common used MAC protocol is S-MAC.

SIMULATOR: SENS

SENS is the component-based simulator for wireless sensor network developed by UIUC in 2004.

SENS supports to link the application source codes with a library which translates sensor node API calls to SENS Application APIs.

A full set of simple propagation models are implemented in SENS to allow develops to test a wide variety of settings in different environment.

SENS Webpage: http://osl.cs.uiuc.edu/sens/

SIMULATOR: SENS CONT.

- Advantage:

  SENS provides a very useful feature to link the application codes with an API translation library, which makes a SENS possible target for TinyOS.

- Disadvantage:

  SENS is less flexible than any other network simulators. Users can not change the MAC protocol and any other low level network protocols.

EMULATOR: TOSSIM

TOSSIM is designed specifically for TinyOS by UC Berkeley.

TOSSIM runs the same TinyOS application codes which can be executed on MICA series motes.

A few low level components should be replaced in order to translate real hardware interrupts to discrete simulator events driven by the simulator event queue.

TOSSIM Webpage: http://www.di.unipi.it/~ste/MaD-WiSe/download-11.htm

SIMULATOR: TOSSIM CONT.

Advantage:

TOSSIM simulates the realistic wireless sensor network hardware. It will greatly help researchers to debug their TinyOS application codes and evaluate network performance without doing actual experiments on real hardware.

Disadvantage:

TOSSIM requests each node to run the same TinyOS application codes, which are nearly impossible in the realistic network scenarios. TOSSIM's compilation step loses the fine-grained timing and interrupts properties of the code.

EMULATOR: ATEMU

ATEMU is developed by Maryland University, and it is the first instruction-level emulator .

ATEMU 30 times slower than TOSSIM.

ATEMU offers an accurate emulation model in which each MICA mote can run different application code.

ATEMU Webpage: http://sourceforge.net/projects/atemu/

SIMULATOR: ATEMU CONT.

- Advantage:

  ATEMU is the most accurate instruction-level emulator for wireless sensor network research. Also, it provides the step-by-step execution and breakpoints feature to debug TinyOS application codes.

- Disadvantage:

  Compared to TOSSIM, ATEMU scarifies the simulation speed to achieve the accuracy. The result of 30 times slower than TOSSIM is sometimes not acceptable if we simulate the large-scale wireless sensor network.

EMULATOR: AVRORA

AVRORA is developed by UCLA in JAVA, and it is another instruction-level emulator.

AVRORA balances the speed and accuracy between TOSSIM and ATEMU through the approach of event queue.

AVRORA provides many useful features to support the research on the wireless sensor network, like control flow graph generation, energy analysis.

AVRORA Webpage: http://compilers.cs.ucla.edu/avrora/index.html

SIMULATOR: AVRORA CONT.

Advantage:

AVRORA achieves both the speed and accuracy of simulation. Also, it is the most flexible emulators compared to both TOSSIM and ATEMU.

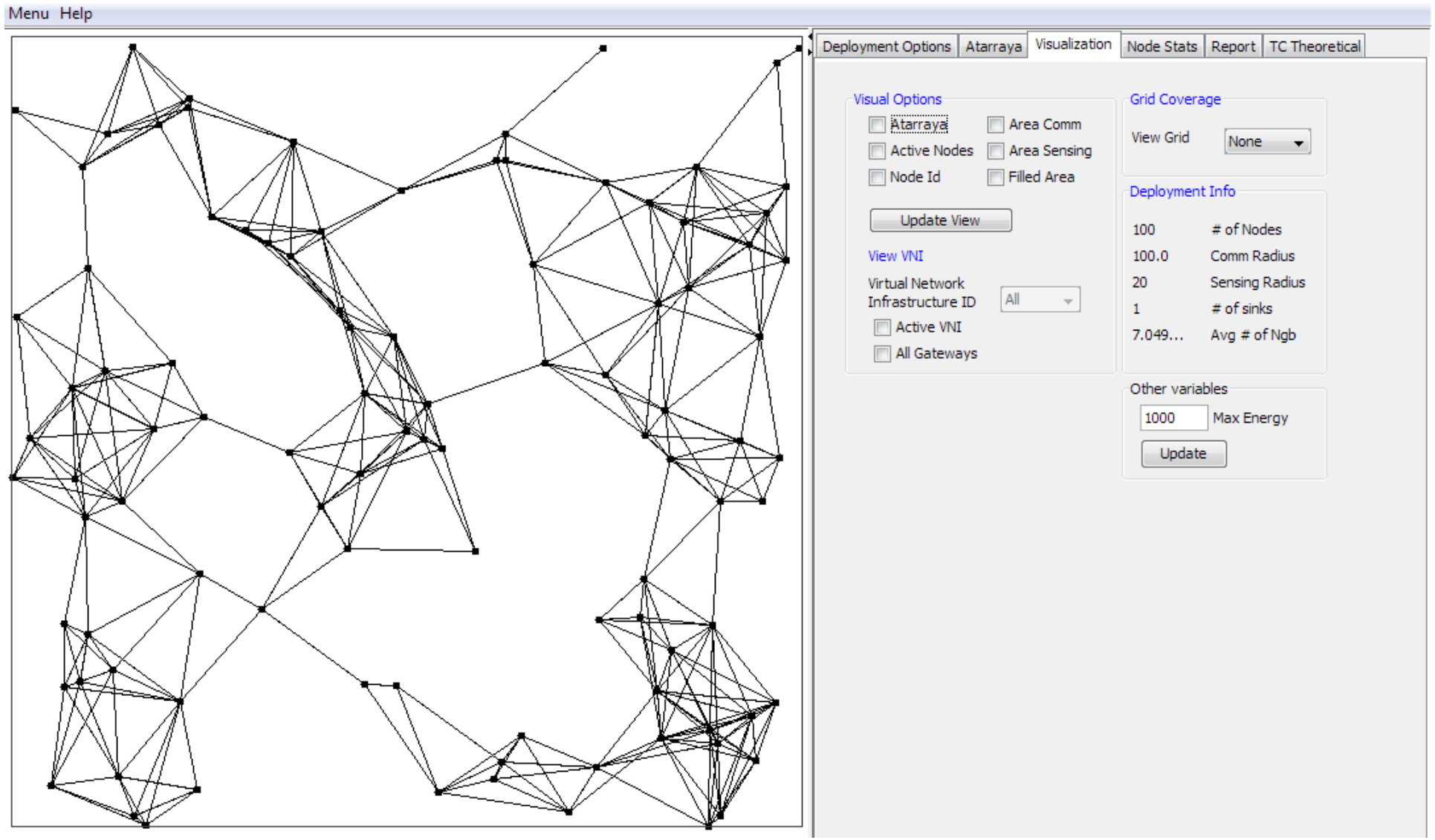
Disadvantage:

The network configuration approach is not as good as ATEMU through XML. Also, the performance of AVRORA still need to be evaluated, since it is a new developed one.

ATARRAYA

Atarraya is a simulation tool for teaching and researching topology control algorithms and protocols for wireless sensor networks.

- The main idea of Atarraya is to provide a simple tool for creating and testing old and new topology control protocols.
- Atarraya includes both topology construction and topology maintenance protocols and algorithms.
- The inclusion of topology maintenance is a unique aspect of Atarraya.
- Atarraya's modularity makes it very easy to create new protocols.


Menu Help
Deployment Options
Atarraya
Visualization
Node Stats
Report
TC Theoretical
Visual Options
Atarraya
Area Comm
Active Nodes
Area Sensing
Node Id
Filled Area
Update View
View VNI
Virtual Network Infrastructure ID
All
Active VNI
All Gateways
Grid Coverage
View Grid
None
Deployment Info
100
# of Nodes
100.0
Comm Radius
20
Sensing Radius
1
# of sinks
7.049...
Avg # of Ngb
Other variables
1000
Max Energy
Update

Prowler

.

Prowler is a probabilistic wireless network simulator capable of simulating wireless distributed systems, from the application to the physical communication layer.

Prowler, running under MATLAB, provides an easy way of application prototyping with nice visualization capabilities.

Although Prowler provides a generic simulation environment, its current target platform is the Berkeley MICA mote running TinyOS

Prowler version 1.21

File Edit View Insert Tools Window Help

Application

Application name:

spantree

Info Parameters...

Simulation

Radio definition:

radio_channel

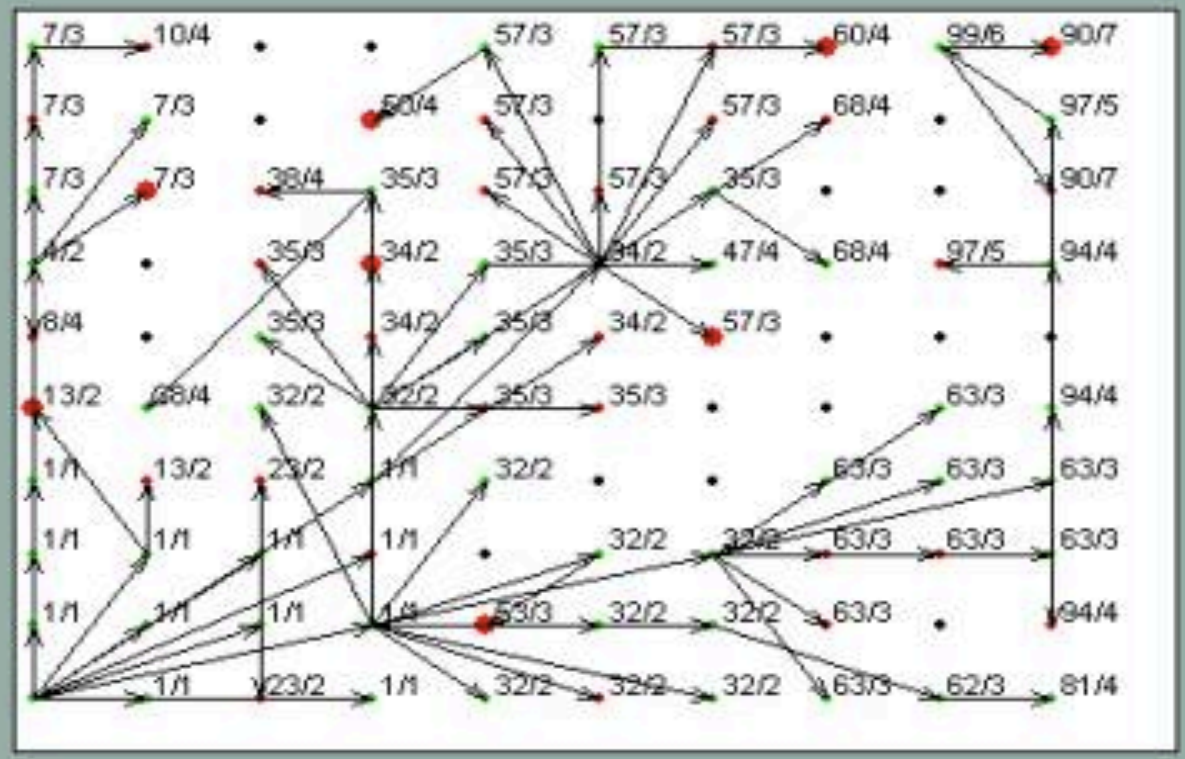
Info Parameters...

Display

Fast animation

- [x] Show events
- [ ] Show LED's
- [ ] Show grid on plot
- [ ] External display

Start Stop Continue



BitTime	Sec	Event	ID
10107	0.25	Channel_Idle_Check	51
10107	0.25	Packet_Sent	84
10108	0.25	Packet_Receive_End	74
10108	0.25	Packet_Receive_End	75
10108	0.25	Packet_Receive_End	83
10108	0.25	Packet_Receive_End	85
10108	0.25	Packet_Receive_End	94
10108	0.25	Packet_Receive_End	95
10108	0.25	Packet_Receive_End	96
10110	0.25	Channel_Idle_Check	72
10113	0.25	Channel_Idle_Check	55

Prowler - Simulation Parameters

Radio Transmission

Ideal signal power:

P_rec_id=P_trans * f(x), where x is the distance.

f(x)= 1./(1+x.^2)

Fading effect:

P_rec=P_rec_id*(1+alpha(x))*(1+beta(t)),

where alpha and beta are random variables with normal distribution N(0,s):

s_alpha= 0.45 s_beta= 0.02

Transmission error:

p_error= 0.05

Reception conditions:

Signal can be received if P_rec > 0.1

Signal can be received if SINR > 4

Receiver's noise variance (RNV) = 0.025

Idle limit = RNV * 4

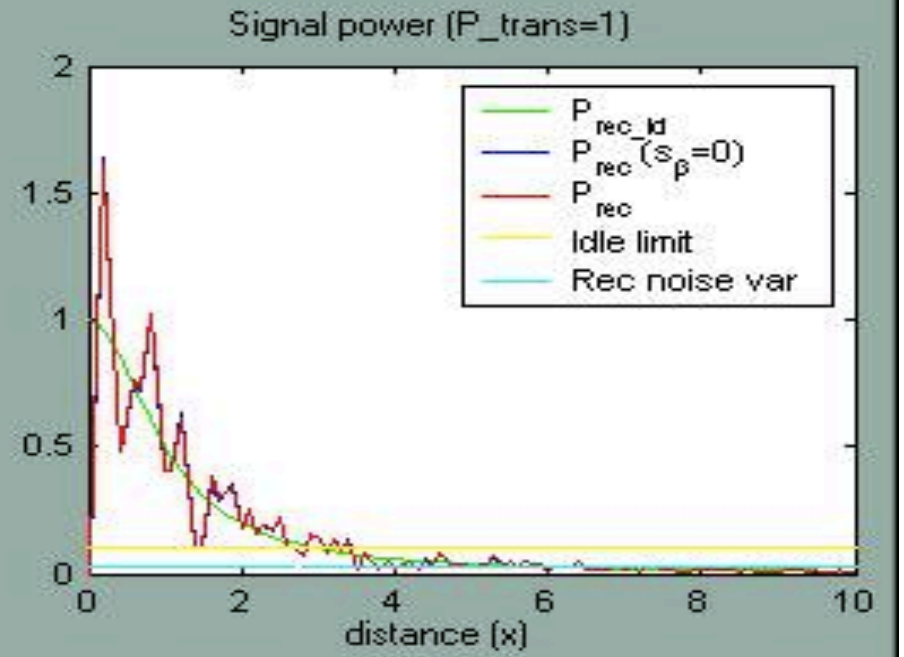
MAC-layer

Waiting time: 200 + 128 * rand

Backoff time: 100 + 30 * rand

Packet length: 960 Unit: bit-time

(rand has uniform distribution between 0 and 1)



lin Plot range: 10

Note: One random representation is plotted.

Simulation

Simulation stops at 100 sec

Help

Cancel

Save settings

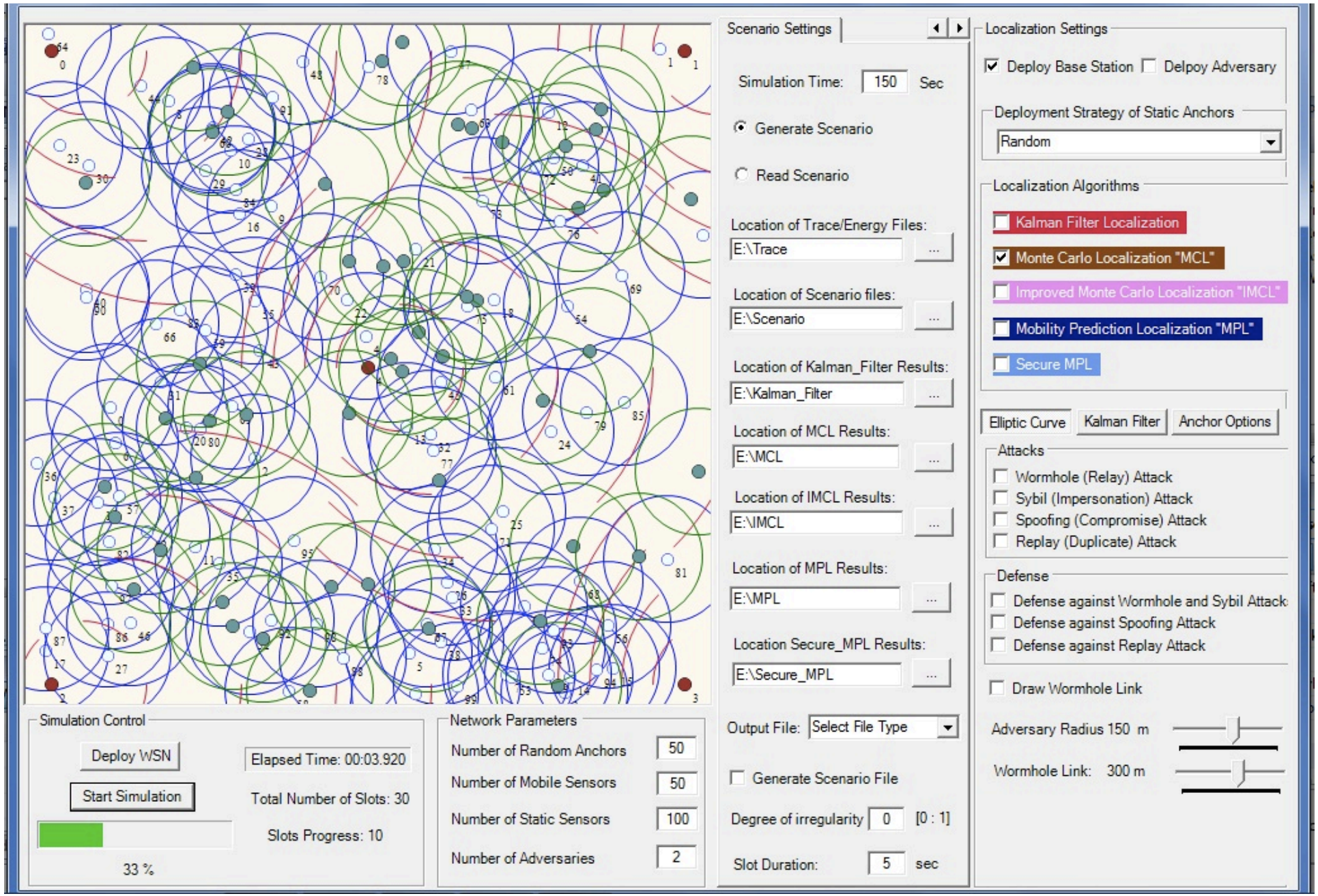
Apply

Load settings Load default

Close

WIRELESS SENSOR NETWORK LOCALIZATION SIMULATOR

- A determination of the location of sensor nodes is a simulation task that Wireless Sensor
- The program comes with eight localization algorithms, while other required procedures may be implemented.
- Numerous parameters that define network topology include: network size, locators deployment strategy and antenna type, as well as the path loss and node mobility.
- The graphical overview of the network is presented in the GUI below


Scenario Settings
Simulation Time: 150 Sec
Generate Scenario
Read Scenario
Location of Trace/Energy Files:
E:\Trace
Location of Scenario files:
E:\Scenario
Location of Kalman_Filter Results:
E:\Kalman_Filter
Location of MCL Results:
E:\MCL
Location of IMCL Results:
E:\IMCL
Location of MPL Results:
E:\MPL
Location Secure_MPL Results:
E:\Secure_MPL
Output File: Select File Type
Generate Scenario File
Degree of irregularity 0 [0 : 1]
Slot Duration: 5 sec
Localization Settings
Deploy Base Station
Delpoy Adversary
Deployment Strategy of Static Anchors
Random
Localization Algorithms
Kalman Filter Localization
Monte Carlo Localization "MCL"
Improved Monte Carlo Localization "IMCL"
Mobility Prediction Localization "MPL"
Secure MPL
Elliptic Curve
Kalman Filter
Anchor Options
Attacks
Wormhole (Relay) Attack
Sybil (Impersonation) Attack
Spoofing (Compromise) Attack
Replay (Duplicate) Attack
Defense
Defense against Wormhole and Sybil Attack
Defense against Spoofing Attack
Defense against Replay Attack
Draw Wormhole Link
Adversary Radius 150 m
Wormhole Link: 300 m
Simulation Control
Deploy WSN
Start Simulation
Elapsed Time: 00:03.920
Total Number of Slots: 30
Slots Progress: 10
33 %
Network Parameters
Number of Random Anchors 50
Number of Mobile Sensors 50
Number of Static Sensors 100
Number of Adversaries 2

HYBRID SIMULATION

Real application support

- The same applications that run on the real nodes can also run on the simulated nodes

Interaction with real nodes

- Real sensor nodes can participate in the simulation

Advantages

- Enables the use of real traffic from the sensor channel that is currently not well understood and the models are not yet mature
- Validate protocols and applications running on the real nodes by being able to test these applications in large networks
- Study the behavior of sensor network protocols and applications at scale

HYBRID SIMULATION CHALLENGES

Virtual Time Synchronization between ns and external entities

- done between ns and external processes

Real and Virtual Time Synchronization

Hybrid Modeling of Wireless Channel

- collision between packets from real and virtual nodes
- Open Problem !