SIMULATION OF SENSOR NETWORKS USING TOSSIM

Pravin Wararkar¹, Dr. H K Dahule² · Shyam Patidar³ · Dhananjay Joshi⁴ · Sreyashrao Surapreddi⁵ & Gyandeep Bhumarkar⁶

¹Department of Electronics & Telecommunication Engineering, SVKM’s Narsee Monjee Institute of Management Studies, (Deemed-to-be-University), Mukesh Patel School of Technology Management & Engineering Shirpur, India, Email: pwararkar@gmail.com
²Department of Physics, Shri Shivaji Education Society Amravati’s, (Nagpur University), Shivaji Science College, , Congress Nagar, Nagpur, India, Email: hk dahule@yahoo.co.in
³Department of Mechanical Engineering, SVKM’s Narsee Monjee Institute of Management Studies (Deemed-to-be-University) Mukesh Patel School of Technology Management & Engineering, Shirpur, India Email: er. spatidar@gmail.com
⁴Department of Computer Engineering, SVKM’s Narsee Monjee Institute of Management Studies (Deemed-to-be-University) Mukesh Patel School of Technology Management & Engineering, Shirpur, India Email: Dj4query@gmail.com
⁵Department of Civil Engineering, SVKM’s Narsee Monjee Institute of Management Studies, (Deemed-to-be-University) Mukesh Patel School of Technology Management & Engineering, Shiripur, India Email: sreyash05@gmail.com
⁶Department of Civil Engineering, Government Polytechnic College (Department of Technical Education & Skill Development) Government Polytechnic College, Raisen, Raisen, India, Email: gyandeepbhumarkar@gmail.com

ABSTRACT
Recent advancement in wireless communications and electronics has enabled the development of low-cost sensor networks. The sensor networks can be used for various application areas (e.g., health, military, home). For different application areas, there are different technical issues that researchers are currently resolving. Development of the right simulation tools has been a key step in systems research progress for several areas. In general, simulation can provide a way to study system design alternatives in a controlled environment, explore system configurations that are difficult to physically construct, and observe interactions that are difficult to capture in a live system.

Keywords: Simulation, Sensor Networks

INTRODUCTION

SENSOR NETWORKS
A sensor network is composed of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it. The position of sensor nodes need not be engineered or predetermined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this also means that sensor network protocols and algorithms must possess self-organizing capabilities.

Smart disposable microsensors can be deployed on the ground, in the air, under water, on bodies, in vehicles, and inside buildings. A system of networked sensors can detect and track...
threats (e.g., winged and wheeled vehicles, personnel, chemical and biological agents) and be used for weapon targeting and area denial. Each sensor node will have embedded processing capability, and will potentially have multiple onboard sensors, operating in the acoustic, seismic, infrared (IR), and magnetic modes, as well as imagers and micro radars. Also onboard will be storage, wireless links to neighboring nodes, and location and positioning knowledge through the global positioning system (GPS) or local positioning algorithms.

These sensor network consists of multifunctional sensor nodes that are small in size and communicate untethered in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks. Sensor networks represent a significant improvement over traditional sensors.

Another unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an onboard processor. Instead of sending the raw data to the nodes responsible for the fusion, they use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data.

**APPLICATIONS OF SENSOR NETWORKS**

The above described features ensure a wide range of applications for sensor networks. Some of the application areas are health, military, and home.

In military, for example, the rapid deployment, self-organization, and fault tolerance characteristics of sensor networks make them a very promising sensing technique for military command, control, communications, computing, intelligence, surveillance, reconnaissance, and targeting systems.

In health, sensor nodes can also be deployed to monitor patients and assist disabled patients. Some other commercial applications include managing inventory, monitoring product quality, and monitoring disaster areas.

**COMMUNICATION ARCHITECTURE**

The sensor nodes are usually scattered in a sensor field as shown in following Fig.1.

![Sensor nodes scattered in a sensor field](image)

Each of these scattered sensor nodes has the capabilities to collect data and route data back to the sink.

Data are routed back to the sink by a multihop infrastructureless architecture through the sink as shown in Fig. 1. The sink may communicate with the *task* manager node via Internet or
satellite. The design of the sensor network as described by Fig. 1 is influenced by many factors, including fault tolerance, scalability, costs, operating environment, sensor network topology, hardware constraints, transmission media, and power consumption, production costs, operating environment, sensor network topology, hardware constraints, transmission media, and power consumption.

**HARDWARE CONSTRAINTS**

A sensor node is made up of four basic components, as shown in Fig. 2:
- a sensing unit,
- a processing unit,
- a transceiver unit, and
- a power unit.

They may also have additional application-dependent components such as a location finding system, power generator, and mobilizer.

![Figure 2: The components of a sensor node.](image)

Sensing units are usually composed of two subunits: sensors and analog-to-digital converters (ADCs). The analog signals produced by the sensors based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the processing unit.

The processing unit, which is generally associated with a small storage unit, manages the procedures that make the sensor node collaborate with the other nodes to carry out the assigned sensing tasks.

A transceiver unit connects the node to the network. One of the most important components of a sensor node is the power unit. Power units may be supported by power scavenging units such as solar cells. There are also other subunits that are application-dependent.

Most of the sensor network routing techniques and sensing tasks require knowledge of location with high accuracy. Thus, it is common that a sensor node has a location finding system.

A mobilizer may sometimes be needed to move sensor nodes when it is required to carry out the assigned tasks.
All of these subunits may need to fit into a matchbox-sized module[1]. The required size may be smaller than even a cubic centimeter[2], which is light enough to remain suspended in the air. Apart from size, there are some other stringent constraints for sensor nodes. These nodes must[3] consume extremely low power, operate in high volumetric densities, have low production cost, be dispensable and autonomous, operate unattended, and be adaptive to the environment.

NETWORK SIMULATION

In communication and computer network research, network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/routers, data links, packets, etc) using mathematical formulas, or actually capturing and playing back observations from a production network. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions. When a simulation program is used in conjunction with live applications and services in order to observe end-to-end performance to the user desktop, this technique is also referred to as network emulation.

SIMULATORS

The simulator (or network simulator) is the program in charge of calculating how the network would behave. Such software may be distributed in source form (software) or packaged in the form of a dedicated hardware appliance. Users can then customize the simulator to fulfill their specific analysis needs. Simulators typically come with support for the most popular protocols in use today, such as IPv4, IPv6, UDP, and TCP.

Free/open source network simulators used in research include ns, OMNeT++, and GloMoSim (the latter no longer under active development), which run on Linux, FreeBSD, SunOS, Solaris, Microsoft Windows and other operating systems. Ns and GloMoSim are widely used in universities.

SIMULATIONS

Most of the commercial simulators are GUI driven, while some network simulators require input scripts or commands (network parameters). An important output of simulations are the trace files. The network parameters describe the state of the network (node placement, existing links) and the events (data transmissions, link failures, etc). Trace files can document every event that occurred in the simulation and are used for analysis. Network simulators can also capture this type of data directly from a functioning production environment. This data capture may be done at various times of the day, week, month, in order to reflect average, worst-case, and best-case conditions. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

SIMULATION TECHNIQUES

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events -- such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.
Some network simulation problems, notably those relying on queueing theory, are well suited to Markov chain simulations, in no list of future events is maintained and the simulation consists of transiting between different system "states" in a memoryless fashion. Markov chain simulation is typically faster but less accurate and flexible than detailed discrete event simulation. Some simulations are cyclic based simulations and these are faster as compared to event based simulations.

Simulation of networks can be a difficult task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be fantastically large. Specialized techniques such as "control variates" and "importance sampling" have been developed to speed simulation.

SIMULATORS FOR SENSOR NETWORKS

More prominent simulators for sensor networks available today are:

1. **NS-2 [5]**: The mother of all network simulators has facilities for carrying out both wireless and wired simulations. It is written in C++ and oTCL. Since it is object-oriented, it is easier to add new modules. It provides for support for energy models. Some example applications are included as a part of the package. It has the advantage of extensive documentation.

2. **GloMoSim [4]**: GLobal MOobile Information systems SIMulator is a scalable simulation environment for wireless and wired network systems. It is written both in C and Parsec. It is capable of parallel discrete-event simulation. GloMoSim currently supports protocols for a purely wireless network. A basic level of Parsec knowledge and thorough C knowledge is sufficient to carry out simulations.

3. **SensorSim [http://nesl.ee.ucla.edu/projects/sensorsim/]**: is a simulation framework for sensor networks. It is an extension to the NS simulator. It provides the following: Sensing channel and sensor models, Battery models, Lightweight protocol stacks for wireless micro sensors, Scenario generation and Hybrid simulation. It is geared very specifically towards sensor networks and is still in the pre-release stage. It does not have proper documentation.

4. **TOSSIM**: TOSSIM captures the behavior and interactions of networks of thousands of TinyOS motes at network bit granularity. It is further described in the following sections.

**TinyOS**

TinyOS is an operating system specifically designed for sensor networks. It has a component-based programming model, provided by the nesC language, a dialect of C. TinyOS is not an OS in the traditional sense. It is a programming framework for embedded systems and set of components that enable building an application specific OS into each application. A TinyOS program is a graph of components, each of which is an independent computational entity. Each TinyOS component has a frame, a structure of private variables that can only be referenced by that component.
The event-driven nature of sensor networks means that testing an individual mote is insufficient. Programs must be tested at scale and in complex and rich conditions to capture a wide range of interactions. Deploying hundreds of motes is a daunting task; the focus of work shifts from research to maintenance, which is time-consuming due to the failure rate of individual motes. A simulator can deal with these difficulties, by providing controlled, reproducible environments, by enabling access to tools such as debuggers, and by postponing deployment until code is well tested and algorithms are understood.

**Introduction To Tossim**

Accurate and scalable simulation has historically been a key enabling factor for systems research. TOSSIM is a simulator for TinyOS wireless sensor networks. By exploiting the sensor network domain and TinyOS’s design, TOSSIM can capture network behavior at a high fidelity while scaling to thousands of nodes.

By using a probabilistic bit error model for the network, TOSSIM remains simple and efficient, but expressive enough to capture a wide range of network interactions. Using TOSSIM, several bugs in TinyOS have been discovered, ranging from network bi-level MAC interactions to queue overflows in an ad-hoc routing protocol. Through these and other evaluations, it can be shown that detailed, scalable sensor network simulation is possible.

**TOSSIM ARCHITECTURE**

Figure 3 shows a graphical overview of TOSSIM. The TOSSIM architecture is composed of five parts: support for compiling TinyOS component graphs into the simulation infrastructure, a discrete event queue, a small number of re-implemented TinyOS hardware abstraction components, mechanisms for extensible radio and ADC models, and communication services for external programs to interact with a simulation.

**WORKING OF TOSSIM**

TOSSIM captures the behavior and interactions of networks of thousands of TinyOS motes at network bit granularity.

TOSSIM takes advantage of TinyOS’s structure and whole system compilation to generate discrete-event simulations directly from TinyOS component graphs. It runs the same code that runs on sensor network hardware. By replacing a few low-level components (e.g., those shaded in Figure 6), TOSSIM translates hardware interrupts into discrete simulator events; the simulator event queue delivers the interrupts that drive the execution of a TinyOS application. The remainder of TinyOS code runs unchanged. TOSSIM uses a very simple but surprisingly powerful abstraction for its wireless network.

The network is a directed graph, in which each vertex is a node, and each edge has a bit error probability. Each node has a private piece of state representing what it hears on the radio channel. This abstraction allows testing under perfect transmission conditions (bit error rate is zero), can capture the hidden terminal problem (for nodes a,b,c, there are edges (a, b) and (b, c) but no edge (a, c)), and can capture many of the different problems that can occur in packet transmission (start symbol detection failure, data corruption, etc.).
By seamlessly supporting the PC-based TinyOS tool-chain, TOSSIM allows developers to easily transition between running an application on motes and in simulation. To improve its usefulness to TinyOS developers, TOSSIM has mechanisms that allow GUIs to provide detailed visualization and actuation of a running simulation.

**TinyViz**

TinyViz, the TOSSIM visualization tool, illustrates the capabilities of TOSSIM’s communication services. TinyViz is a Java-based graphical user interface for TOSSIM, allowing simulations to be visualized, controlled, and analyzed.

TinyViz provides visual feedback on the simulation state and mechanisms for controlling the running simulation, e.g., modifying ADC readings and radio loss probabilities. TinyViz provides a plugin interface allowing developers to implement their own application-specific visualization and control code within the TinyViz engine. Figure 4 shows a screenshot.

![Figure 3: TOSSIM Architecture: Frames, Events, Models, Components, and Services](image)

![Figure 4: TinyViz connected to TOSSIM running an object tracking application.](image)
The right panel shows sent radio packets, the left panel exhibits radio connectivity for mote 15 and network traffic. The arrows represent link quality and packet transmissions.

TinyViz has a set of default plugins that provide basic debugging and analysis capabilities. Two of these, the network and control plugins, were described above. Further examples include a plugin that displays in list format all debug messages and another that graphically displays the data in radio and UART packets. A sensor plugin that displays mote sensor values in the GUI allows the user to set individual mote sensor values during simulation; this plugin can be extended to more complicated sensor models.

A radio model plugin changes radio connectivity based on distances between motes in the GUI and graphically displays link probabilities, providing basic mechanisms for experimenting with how networks behave under change.

We will also demonstrate the simulation of few built in programs in TOSSIM

CONCLUSION

As per simulation carried out, TOSSIM will be most useful for further advancement of sensor Networks

REFERENCES


