ON MULTI-AGENT STRATEGIES FOR EFFICIENTLY DESIGNING COMPETENT INTEGRATED HARDWARE-GENERIC SIMULATION SYSTEMS OF WIRELESS SENSOR NETWORKS

Alexander Filippou
University of Bolton, UK,
alexfilippoy@yahoo.gr

Dimitrios A. Karras
Sterea Hellas Institute of Technology, Greece, Automation Dept., Psachna, Evia, Hellas (Greece) P.C. 34400,
dakarras@ieee.org, dimitrios.karras@gmail.com,
dakarras@teiste.gr

KEYWORDS
Wireless Sensor Networks, WSN, Multi-Agent Systems, Simulation, MCU Emulation, OpenCL, GPGPU

ABSTRACT

This research deals with several key issues concerning WSN design process. Due to the nature of these networks, debugging after deployment is unlikely, thus a well-organized testing methodology is required. WSN simulators could achieve such a task, but still code implementing mote sensing and RF behaviour consists of layered and/or interacting protocols that for the sake of designing accuracy are tested working as a whole, running on specific hardware. Simulators that provide cross layer simulation and hardware cross platform emulation options may be regarded as a significant milestone for improved WSN design process. The herein proposed multi-agent simulation architecture aims at designing a novel generic WSN simulation system independent of specific hardware platforms but taking into account all hardware entities and sensor network events for testing and analysing the behaviour of a pragmatic WSN system. Details of the implementation are provided as well as a preliminary validation of the simulator in C# involving motes based on two different hardware platforms, randomly deployed over WSN motes.

1 INTRODUCTION

A WSN is a distributed system. It consists of a usually large number of autonomous devices that form a network. The diversity of missions and environments deployed in, introduces issues and parameters of paramount importance during design process. Success of this process is considered delivering **specific code running on specific hardware**, both meeting mission and production cost requirements.

In general, a mote is a device that consists of a medium access hardware interface, a processing module and a sensor array. In case of WSN, the medium is the RF channel, and the hardware interface is a RF transceiver. In case of submerged SNs the medium is water (acoustic signals) and the hardware interface is a microphone and a loudspeaker. The trivial case scenario is a Wireless Sensor Network, running on batteries, with limited computational ability and memory, operating in a harsh and hostile environment. By using simulation tools we gain pre-deployment knowledge estimating the network's behavior. In most cases, the design and implementation of application and protocol stack code, running on specific hardware setup, are viewed through energy consumption, security and production cost prisms.

The first task of a WSN after deployment is to configure itself. Every mote uses its transceiver to establish connections with its neighbors, in order to construct a topology. The mote acquires its location which is unknown at the beginning, through collaboration with other motes, starting from a few motes with known locations. The Localization protocol responsible for the above task uses physical quantities such as RSS AoA, ToA, to calculate the mote's location. After identifying its neighbors, and being identified, the mote is part of the network, able to produce sensor data, propagate data to sink, collaborate with neighboring motes to perform a computational or sensing task, create cluster, and so on. The total activity of a mote extends to multiple levels – or layers – each having its own procedure and parameters to calculate QoS. The overall performance is derived from the combination and cross layer code collaboration (Masri W.,Mammeri Z.,2008), and as stated in (Bernd-Ludwig Wenning et al., 2008) does not necessarily means optimal performance in every layer.

In the next part of this paper, we highlight topology, simulation and hardware design issues that back up our choices in design of our optimal multi-agent based simulator. We spot the parameters taken in consideration in order to calculate the simulation metrics in each case. Our goal is to design a sensor simulator able to perform cross layer code, communication medium and environmental simulation, while keeping an inside view in every aspect of this process, giving every detail needed in order to extract conclusions about network behavior, in mote, local (an area containing a number of motes) or global (entire network) level.

2 A CRITICAL OVERVIEW OF WSN RESEARCH

From our point of view, WSN coding may be categorized in two main categories. **Application Code** and **Network / Maintenance Code**. Application code is responsible for the collection and interpretation / processing of sensor data, or for actions in case of collaborating actuators. Application code operates on the platform provided by Network / maintenance code, which implements the WSN backbone. The backbone includes network protocol stack, topology, security, and configuring protocols, and provides services to the

Proceedings 29th European Conference on Modelling and Simulation ©ECMS Valeri M. Mladenov, Petia Georgieva, Grisha Spasov, Galidiya Petrova (Editors)
ISBN: 978-0-9932440-0-1 / ISBN: 978-0-9932440-1-8 (CD)

application code clear of such issues. All above share in most cases limited resources, eg memory, and processing time. Simulating a protocol spans from a single protocol testing to testing the entire MCU code. The simulator runs protocol models or emulates the processing unit of the mote running the actual code to be uploaded. Modelling protocol is prone to errors of inaccuracy, for there is a gap between model and final code, in terms of behaviour and interactions with the rest of the code in the MCU memory. Moreover, any mote after deployment completes tasks related to network configuration, starting from authentication of neighbours and proceeding to synchronization and localization.

A. Authentication

After deployment, a mote searches for neighbours. This is done by transmitting its presence at maximum power, waiting for replies. Any protocol that uses the network leaves traces in the mote's transmissions. Keeping this in mind and in the case of authentication, in order to gain access, an adversary's mote presents itself as a trusted one, becoming part of the network, by being included in neighbouring mote's list of motes that have faulty authenticated it. Authentication protocols have thein own front of attack, thus security issues, starting from their transmission traces. This is an attack on topology, varying from communication disturbing, gaining access to cryptographic keys and routing tables in order to launch fake messages to sink (S. Dziembowski et al., 2009), (Karlof, C. et al., 2003)

B. Synchronization

Due to the nature of oscillators, every mote's clock function differs from the ideal (c(t)=t) and is modelled (Serpedin E. et al. , 2009) as :

$$C_i(t) = \phi_i t + \omega_i t + e$$

The parameters φi and ωi, are called the clock offset (phase difference) and clock skew (frequency difference), in relation to the reference clock respectively, and e stands for random noise. The goal of synchronization is to provide to each mote the reference clock, by calculating these parameters for every mote i. (Giridhar A. et al., 2011): Other tasks depend on the accuracy of the synchronization protocol :

1) Data Fusion:The interpretation of collected data in most cases requires that the data is time stamped. For example in target tracking, if the target if k covered, k event packets will be created for the same target. In network processing by neighboring motes according to event time, will determine the report packet sent to sink.

***2) Power Management*:** Using redundant motes organized in groups, a network of sensor motes creates a reduced topology by switching off groups in turns. Accurate synchronization is required for the sleep-wake-up circle of topology maintenance (Slijepcevic, et al. , 2001).

***3) Slotted transmission schemes*:** Slotted ALOHA, TDMA use time synchronization in order to determine the slot time boundaries. Accurate synchronization avoids collisions, reduces retransmissions thus saves energy.

4) Protocols: Localization protocols, routing protocols (eg LEACH) require synchronization or time stamped messages. It is clear that evaluation of a synchronization protocol is related to cross layer simulation, due to dependency among protocols. The LEACH protocol uses TDMA scheme for in-cluster communication, thus its performance is proportional to synchronization protocol accuracy. Also, simulation of the synchronization protocol without assumptions requires a **hardware aware simulator able to emulate e.g. clock drift**.

C. Localization

After identifying neighbors, every mote estimates its position by activating the **Localization Protocol.** The Localization Protocol uses hardware input parameters related to physical quantities such as received signal strength (RSS), time of arrival (ToA, TDoA), angle of arrival (AoA) or sound (Timo Ojala et al, 2011), to calculate the mote's location. This process begins from motes with known locations (anchors) and gradually is spread throughout the entire network. Some localization schemes rely on additional hardware to operate, or their simulation is somehow hardware related:

1) Time Difference of Arrival, is used in two ways: difference between two signals of different nature e.g. RF and ultrasonic, and difference between RF signals of different motes of known location. The former scheme requires additional hardware, increasing complexity and mote cost. In addition, ultrasonic pulse reception suffers from severe multipath effects. The distance between the motes is derived obviously **without the need of synchronization**. But due to multipath effects, this method provides fairly accurate distance estimation.

2) Roundtrip propagation time measurements measure the difference between the time when a signal is sent by a sensor and the time when the signal returned by a second sensor comes back to the original sensor. Since the same local clock is used to compute the roundtrip propagation time, there is no synchronization problem (Guoqiang Mao, 2009). The response of the second mote must include the processing time of the reply, which is sent to the first sensor to be subtracted. It is clear that processing time in the second mote depends on its clock, and on the number of machine cycles needed to execute code that handles and transmits the reply. The above properties depend on the chosen MCU or MCUs in case of an heterogeneous WSN. In addition, in some energy saving strategies, one solution is adjusting the clock to lower frequencies through software during runtime. This spots the need for simulation tools that **can handle different MCUs, clocks and clock adjustments** and able to **calculate processing time of a subroutine**.

3) Beacons using directional antennas, transmit their location and angle of transmission, or motes use their directional antenna to calculate transmissions of anchors. In general, protocols using AoA require a channel simulator that can handle sector type transmissions e.g. Opnet (www.opnet.com).

***4) Underwater*:** In (He, T., et al. 2006) a localization protocol is proposed for underwater sensor networks using the DNR (dive end rise) of anchors equipped with GPS, in order to update their location, and then using sound waves to transmit their locations. Besides that a

pressure sensor in every mote helps in estimating the depth underwater. The whole scenario takes place not on a 2d plane but in 3d space. The localization is constant, considering the fact that the simulated UWSN may operate in a area with strong currents.

In every case, simulation of localization protocols require **(a)**. Hardware aware simulator, able to emulate behavior of components such as antennas, MCUs for accurate time measurement **(b)**. An accurate medium simulator (RF spectrum, sound) able to represent the nature of signal propagation (multipath fading, reflection, diffraction, attenuation). **(c)**. An environmental simulator, in case that environmental properties or phenomena (obstacles, sea currents, water pressure, forest fires (Bernd-Ludwig Wenning et al., 2008)) should be taken in consideration.

Authentication. Synchronization and Localization protocols are activated during **topology construction phase** and during **every topology maintenance phase** (M.A Labrador,,et al., 2009). The topology construction phase provides the initial (full) topology, where motes have authenticated all their neighbors, and are synchronized. The topology maintenance phase updates neighbor motes tables by deleting energy depleted, deactivated or destroyed motes, or by inserting new ones, in case of a redeployment over a blind hole. Topology protocols such as (Slijepcevic, et al., 2001) provide the reduced (logical) topology, the platform on which routing operates. However, in (Watteyne T. et al, 2008) authors introduce a rooting protocol under the concept of virtual coordinates, which are randomly initialized in each mote, and updated each time the mote relays a packet. This updating process consists of the sending mote retrieving the virtual coordinates of its neighbors and updating its virtual coordinates by using a centroid transformation. In this case, localization and routing are processes that run concurrently.

Concluding, all the above three types of protocols have their own trace in motes transmissions, and share processing time. The fact that sensing ability looses MCU focus is taken in consideration during design process.

D. Routing

In (Khan M, et al., 2008) the routing protocol Directed Diffusion (Inatanagonwiwat, C.,, et al., 2000) is examined, using a **diagnostic tool**, plugged in Tossim (Levis, P., et al. 2003). The case study is the fact that despite good communication, the protocol fails to deliver packets of interest to the sink. The diagnostic simulator is motivated by the idea that, in a distributed computing environment, nodes have to interact with each other in some manner defined by their distributed protocols in order to perform tasks correctly. These interaction patterns are the concatenation of distributed sequences of events on multiple nodes. In a correctly functioning system, these sequences of events follow a path that the protocol is designed to handle. The challenge is to identify special sequences of events which are responsible for the failure among hundreds of other common sequences that are logged during the execution but are unrelated to failures (Khan M,, et al., 2008). In this fine work the need for **fine grained debugging** is pointed out, providing also a solution as a plug in tool. However, Tossim **lacks** ability of simulating different code in every mote and using a different hardware platform eg a PIC or a Freescale MCU based mote. In (Bernd-Ludwig Wenning et al., 2008), The EMA routing protocol is introduced, in which main concept is the environmental awareness of the protocol. The low consumption orientation of routing protocols such as LEACH (W.B. Heinzelman, et al., 2002), PEGASIS (S. Lindsey, et al., 2002) TEEN (A. Manjeshwar, et al., 2001), is not the optimal solution when a mote is facing destruction threat from e.g. forest fire. The protocol estimates also the probability of that threat in order to use resources of motes that are about to fail.

The Greedy-Face-Greedy (GFG) and Greedy Perimeter Stateless Routing (GPSR) protocols are the most widely adopted geographic routing protocols for WSN (H. Frey, et al., 2006). Every node applies Gabriel Graph Transformation to the connectivity graph, for the protocol to perform over its planar version the left hand rule. In this way, protocol recovers from geographical dead ends caused by blind holes. The assumption that communication regions are perfectly circular disks does not hold in real-world propagation conditions, causing the protocols to fail, in case (Watteyne T. et al, 2008) of obstacles. In addition Gabriel Graph Transformation does not function with limited positioning accuracy (T Watteyne, et al., 2007).

Concluding, protocol relies on topology to operate. Topology protocols or topology related protocols provide the platform whose accuracy routing protocol QoS depends on. In (Bernd-Ludwig Wenning et al., 2008) it is shown that even efficient protocols do not operate well in all situations. The goal is the overall performance of protocol code and this is tested by cross layer simulation, provided the ability to monitor protocol interactions.

E. Sensing

The sensor array of a mote is consisted of at least one sensor or sensor device. They are mostly analog, and connected to MCU through an A/D converter. In most cases, the MCU-ADC channel is shared among all sensors of the mote, using a time slot mechanism. Sensors function according to their physical properties, such as range, energy consumption, response delay, drifting over time etc. Their performance characteristics introduce problems that span from coding to security and life expectancy of the sensor itself, and as a consequence, life expectancy of the network (Jon S. Wilson ed, 2005):

Transfer Function: Is the function that shows the relationship between physical stimulus and the electrical signal. The signal is processed (eg amplified, converted to digital) and transferred to the MCU for interpretation by the application level. **The problems that arise** in order select the suitable sensor is the **computational load** of the interpretation, and the **energy consumption** of the in-circuit processing of the signal. The above spot the need of modeling the sensor itself, and for simulator tools that have the ability to switch between sensor devices and able to model the hardware specs of these devices, including energy consumption of the circuitry that supports the sensor.

Sensitivity: Is the factor between variation of stimulus and consequent variation of electrical signal. A sensor is sensitive, if a small change of stimulus results in a large change of the electrical signal. Is sensitivity of the selected sensor devices adequate for the desired detect ability of the entire network? Is it safe to assume detect ability modeled by an deterministic (binary) sensing model? What computational load would contribute a process that detects events in a continuously streaming sensor data in case of a stochastic sensing model? (e.g. correlating sound) (Patrick Kuckertz et al., 2007).The answer lies on the ability of the simulator to model real world phenomena or scenarios. A crack on the concrete construction of a bridge lasts for milliseconds and has to be identified among sounds and noises of various amplitudes and frequencies e.g. car tires, engines, contraction – expansion of metal skeleton etc. Simulating WSN for sniper detection through identifying sound source position of multiple sensor readings requires modeling of the environment e.g buildings, echoes, sound attenuation etc.

Dynamic Range or Span: The range of physical signal that can be converted to electrical according to specs, is the dynamic range of the sensor device. The response of the sensor outside of this range is faulty. In real life, it is impossible to detect inaccurate reading coming from the sensor array, unless we use a variety of sensor types to measure the same physical quantity. During design process it is essential to know the probability of an inaccurate reading coming from sensor array, and the impact on the network. To cope with this task, we need simulation tools that **(a)** model the sensor, the environmental phenomena of which we collect readings, and **(b)** provide the ability to implement metrics over the accuracy of these readings.

Noise and Resolution: The output signal of a sensor, is contaminated with noise, which is usually distributed across the frequency spectrum. In case that noise is similar to minimum detectable signal fluctuation (**resolution**), the performance of the system is limited, thus during the design process, the selections in hardware are examined on how they contribute to noise.

Sensing Range: A sensor's sensing area is a disc or sector, sphere or cone. The sensitivity of a sensor may depend on the distance of the stimulus. The binary sensing model is not always a wise choice without taking in consideration the sensor specifications related to stimulus distance from the device. When referring to network lifetime, the dominant issue is energy consumption. Main resource is considered to be the battery, and main consumer the transceiver. Active sensors e.g. accelerometers, require energy to provide output electrical signal. Thus the volume of measurements affect network lifetime, and along with communication management and MCU sleep-wake up cycle, sensor activation management needs to be included in the total energy consumption policy. Chemical sensors, sensors used in contamination spread scenarios, detection of explosives, toxics etc. employ substances, e.g. electrolytes, which quantity decrease in every measurement. Draining a sensor of that resource makes the mote useless in terms of coverage. In addition, the sensor is exposed to unlimited numbers of chemical combinations (Fraden J., 2004), able to alter the sensor's behavior.

In conclusion, sensors as electronic devices have properties that may have a major impact on sensor network behavior. Modeling of sensors, along with the environmental properties or phenomena that provide data as sensor measurements, contributes to the fact that it is feasible to obtain a fine grained detail of the environment – WSN interaction, revealing possible failures or bug prone design choices during simulation.

Coverage and Connectivity : Every sensor has a sensing range. Range varies from type to type, spanning from a single point e.g. temperature sensor to an area usually of circular shape. A point of an area or space is covered if it is in range of at least one sensor. Coverage is the QoS that quantifies the ability of a deployed sensor network to report an event or a reading in the deployment area. Sensor readings and their processing produce data to be routed to sink. The Sink is aware of the events spotted or data produced by motes if routing protocol succeeds in delivering data. Thus apart from being in range of at least one sensor, a point in the area of interest is covered, if there is a routing path available to deliver consequent data to sink. If there is no available path, then from the sink's view point, there is no response from the motes of the disconnected area, and even though an event is tracked by a sensor, this area is regarded as a blind hole.

Every mote has a sensing range (coverage) for each type of sensor it carries and a communication range (connectivity). Assuming that a WSN is static and isotropic [Habib M., et al., 2009], and modeling a network as such, reduces our ability to test the protocol performance (of any layer) over a transmission power – energy management or mobility scheme (Muhammad Imran et al., 2011). Finding neighbors by transmitting In full power or in permitted power according to battery power level, is one of choices in reacting to a connectivity loss scenario. In general a mote has the ability to adjust its transmission power during runtime, if the total resource management strategy contains the ability. That includes the **hardware choice of including a battery level indicator on the mote**.

In (Balister P., et al., 2009), the model of Trap Coverage is introduced. From our point of view this model is a realistic one, for it allows lack of coverage over bounded areas, in other words, the WSN continues to operate when in real life blind holes appear due to node energy depletion. The same model generalizes full coverage, when the diameter of the permitted blind hole is near 0.

Apart from managing transmission power, mobility (Muhammad Imran et al., 2011) is among choices, where mobile motes are moving to a position that restores connectivity. From obviously expensive motes, we move to motes that we are able to use in large quantities: Deploying redundant motes, organizing them to operate in turns in order to prolong network lifetime. In Cover Sets (Slijepcevic, et al. , 2001), (Cardei, M., et al., 2005),(Cheng,

M.X., et al., 2005) motes are organized in sets that cover the same area, and scheduled in a way that one set is active at the time. In many schemes, like (Slijepcevic, et al. , 2001), (Cardei, M., et al., 2005), (Cheng, M.X., et al., 2005) a cross layer simulation is needed, for deriving in this case the appropriate MAC layer, considering the flood in transmissions.

Event detection in most cases requires multiple readings from different sensors. Processing of sensor data in mote clusters or at the sink, produces the report message. Even though simulated models of mote collaboration, successful routing at most circumstances and hard conditions, data fusion at the sink may conclude to success in event tracking thus coverage, after deployment, hardware details may prove the protocols inadequate. The assumption that sensors of a mote can sense at the same time is not valid in case of a single MCU ADC channel. Sensors are triggered in turns, and in case of a **time bounded** event (a crack of concrete) some sensors of the sensor array of the mote will miss the event. Dealing with the problem in (Saukh O.,, et al., 2008) authors introduce the concept of **logical sensor** , in which sensors are distributed over different motes, kept active and the cluster of motes that provide all types of sensors is considered as a logical sensor. In the cluster the fusion of data occurs, creating the necessary report to be routed to sink. The problem of **sensor cost** is dealt also. In the same concept, the problem of minimizing expensive sensors while maintaining coverage is dealt with logical sensors. In the past, one of the assumptions made, was the disk model of a sensor, usually binary. Technology has to offer a variety of products, in this case sensors, with properties to be taken into consideration in the design process of a WSN. There are directional sensors (e.g. ultrasonic, cameras, etc), thus simulating a WSN equipped with these sensors requires simulator with the ability to model in every extension the nature of sensors used. In (Liang C.K., et al., 2008) authors propose **Maximum Coverage with Rotatable Angles (MCRA)**, in which the goal of the protocol is to increase coverage by turning the orientation of directional sensors, while minimizing the angles of rotation. Minimizing angle of rotation concludes to low energy consumption, minimum movement detection of the deployed WSN, and positioning of motes in adequate manner to cover a given area (surveillance applications)

Concluding, coverage is not only a matter of sensing radius and routing backbone. Details that affect coverage are spread across layers, and cross layer simulation is needed to obtain precise evaluation of code – protocol performance, in terms of reducing mote cost, increasing network lifetime, and of course coverage itself.

3 THE PROPOSED MULTIAGENT SYSTEM SIMULATOR ARCHITECTURE

A. An Overview of the Design issues of the Multi-Agent System Architecture

In the previous part, we spotted details taken into consideration while designing a protocol or the entire protocol collection running on mote's MCU. Hardware specific simulators like Tossim and Atemu lack flexibility concerning hardware for they are bounded with a specific platform. Avrora emulates every mote in its own thread, and thus performs and scales according to thread ability of the computer it runs on. None of the above takes advantage of GPGPU abilities (CUDA, OpenCL)

We propose a multiagent system architecture, able of scaling, providing fine grain detail of the simulation, and configuring all the parameters of RF channel, Environment and Hardware. Our goal is to provide a multi agent simulation tool, to serve among others as a WSN Simulator. Simulation is an integral part of WSN design process. Due to complex and distributed nature of WSNs, tests and evaluations are carried out safely by simulation. A well equipped simulator provides accurate results in as many aspects of the WSN as the designer needs. In other words, flexibility in applying metrics, in modeling devices and environment, and speed are desirable features.

A cross layer simulation reveals hidden interactions. In any case, a protocol success is restricted by the success of its collaboration with the rest of code in the MCU memory. Like Avrora, Tossim and Atemu, we choose emulation as base for our simulator. There are some advantages over protocol code modeling:

1. **MCU emulation is by default cross layer**. Designers are freed from code modeling, and synchronizing and linking of protocols. The model-implementation gap is eliminated, for modeling phase is skipped. The code written for simulation, will eventually be uploaded in the target MCU.
2. **Monitoring of code execution**: Compiling C into machine code, makes code monitoring easier. There is no complexity at machine code level, and every instruction is executed consuming precise **amount of time and energy.**

In Atemu, XML files are used to describe connections between devices on a mote. We take a little further the use of the XML standard, using XML to describe the MCU itself, making our simulator platform independent.

In http://code.google.com/p/jmc68k/, http://ref.x86asm.net/ XML files are used to model the instruction set of Motorola 68000 and Intel 8x86 architectures respectively. These files contain the amount of information need in case of an Assembler – Disassembler. In http://code.google.com/p/jmc68k/, for every instruction, JAVA code completes the emulator. In other words, in http://code.google.com/p/jmc68k/ the XML file **does not** describe the operations over memory contents of the 68000 processor.

MCU Modeling: In our proposal, the emulator is consisted of two components. The first is an array of bytes. During runtime it contains MCU's memory contents. Registers, flags, program memory, ports, data memory, etc are considered as memory and occupy a number of bytes in this array. The other component performs operations done by instructions upon bytes of memory. For example, the 8 bit AVR instruction set consists of 142 instructions. Each instruction is described by the opcode length usually constant, clock cycles of execution, and operations upon memory. The mnemonic

ANDI is described in Atmel 8-bit AVR Instruction Set. Rev 0856l-AVR-07/10 :

1. Syntax : ANDI Rd,K
2. Operation : Rd←Rd AND K
3. Opcode 0111 KKKK dddd KKKK LENGTH 1 WORD (2 BYTES)
4. Affection on flags (SREG) and Program Counter (PC)
5. Clock Cycles 1

In the XML file, 2-5 are described for every instruction. Op code length and clock cycles may vary from instruction to instruction. Clock cycles are related to MCU oscillator, which may change by software during runtime, Timers and ADCs, interrupts and any other built in device of the MCU is modeled separately.

An XML file describes an MCU or FPGA including the instruction set, timers, ADCs. It also **contains energy consumption of the clock cycle according to oscillator**. Using multiple XML files we setup a network with many MCU types, each running on its own clock. Every emulated MCU receives a signal from the global clock, to move one machine cycle forward. The global clock is a variable increased by a basic time step, the **maximum common divider** of all MCU clocks. Synchronization of the above process is achieved by (A. Filippou, et al., 2010):

```
For (I=0;;I++)
{
   If  (I mod a=0) then     Component (a)
   If (I mod b=0) then      Component (b)    ……    }
```

Every emulated MCU after moving one machine cycle forward (which is in many cases executing one instruction) waits until it receives the next signal from the global clock. Global clock signals the components to be activated, and waits for every component to complete before increasing the basic time step. In this way we describe functionality in a single file without coding, and details needed to produce the XML file of an MCU are mostly in the MCU reference [Atmel 8-bit AVR Instruction Set. Rev 0856l-AVR-07/10]. Designer is able to support its own simulation with the XML description of his choice MCU/FPGA.

In the first part of this paper we discussed several aspects of WSN protocol coding. The implementation of the protocols in C results - after compiling for the target MCU – in bytes, which ends up to emulator's virtual program memory. Being able to monitor MCU memory during runtime gives us several advantages:

1. Data Memory monitoring: MCU virtual memory contents such as sensor data, variables used in code, routing tables, authenticated neighbors etc, in the form of bytes. Memory monitoring provides the ability to keep track of variables and data. Interpretation of these bytes reveals information that code stores during runtime. Instances of memory may be stored in a database during runtime, in order to become subject to an extensive **data mining**.

2. Register / Port memory monitoring: From emulator's point of view, special function registers, general purpose registers and ports are regarded as memory. Their virtual space is implemented by assigning their functionality to the appropriate number of bytes of physical memory, the first component of the emulator. For example, in PIC 16F887, the program counter (PC) is a 13 bit register implemented by 2 registers PCLATH and PCL. PCLATH is located &A0h of all 4 memory banks, and PCL is located &02h of all banks also. The MSB of PC includes bits 0-4 of PCLATH, and LSB includes bits 0-7 of PCL. Program Counter Monitoring allows monitoring of program execution. For example if the IF THEN ELSE block:

IF A>0 THEN <*block 1*> **ELSE** <*block 2*> **END IF**

is compiled, the compiler places variable A in memory and the first instruction of each block (1 and 2) occupies a specific byte of program memory. By monitoring PC we obtain information on program flow, time consumption, diagnostic simulation (Khan M,, et al., 2008), debugging, and in general ability to apply metrics.

Emulator and every component of the simulator has a monitor. Monitor applies metrics described by script, and/or provides interface to external application e.g. MySQL database. Regarding security issues, monitor provides vision to protocol behavior in terms of security, including execution and data monitoring. It emulates tampering by making MCU memory contents visible to an external component or application, which virtualizes a hacking attempt, or simulates an attack model.

Connecting Components: Every component of the simulator e.g. MCU emulator, transceiver emulator/simulator, sensor emulator/simulator, has a plug port and a signal port. Plug port is the gateway through which component interchanges data. Signal port is the port from which is activated or activates another component. For example, emulating the PIC 16F887, plug port is the MCU ports. Ports are connected to MCU memory, e.g. PORTA is placed in &05h of 1st bank and controlled by TRISA register in &85h of 2nd bank. Through signal ports the emulator is activated by its TIMER component which it is also activated by the simulator's global clock. The connection backbone of the simulator is the implementation of these ports. It is implemented with byte arrays, and any plug uses as many bytes it needs to represent interchanged data. In this way user may add his own functionality, as long as they provide signal and plug ports. Every component's interface is signal and plug ports, and communicates with the rest of the simulated world through them. These ports are connected to the connection backbone (array of bytes). There is no mote representation in the simulation, there are just components waiting global clock to signal their next step forward. Mote setup is derived **logically** by its component connections. The overall design is presented in the following figure 1a.

Channel Modeling: MCU sends data to be transmitted to transceiver emulator. Transceiver emulator communicates with RF channel model through a communications backbone, which is also an array of bytes. The size of plugs in bytes is according to channel modeling. Transceiver and RF channel model can exchange any amount of data (size of plug) during a time step, **as long as it is translated in bytes**. This amount of data may be a bit, a spectrum slice of a quantum of time

(a vector containing amplitudes on every frequency), or a hole packet.

When a transceiver transmits data in a time step, it sends to RF channel model the coordinates (X,Y,Z), power and of course data itself. RF channel model stores data and coordinates as long as the signal is above SNR, according to its transmission power. When a transceiver listens to RF channel, then the model calculates what would arrive to antenna, using valid stored signals. Till a transceiver activates its antenna, RF **channel model is inactive**. In case of a CSMA/CD scheme, a transceiver transmits and listens to RF Channel at the same time.

Sensing: Sensors in general are modeled as functions. They communicate with the environment model through the physical quantities backbone which is also implemented in bytes. They are connected to the ADC of the MCU, through the connections backbone. Environmental model produces data to be forwarded to sensors through the physical quantities backbone. To do so, it uses functions of physical phenomena, executes scenarios (target tracking) or simply streams data from a data set.

B. Implementation Details of the Multi-Agent System Architecture

Based on the above detailed analysis our proposal comprises four basic architectural implementation modules: (a) Agents (b) The Agent Controller (c) Interfaces (d) Services.

Agents: There are two types of agents. (a) Built in agents. They are commonly used components such as plotters, visualizers, or emulators. They are at the disposal of the user and not necessarily part of every simulation. (b) User defined agents. User writes code to define the agent behaviour. These are implemented by an interpreter, whose functionality is described later.

The Agent Controller: The controller activates or deactivates agents according to simulation clock and/or event handling instructions written in user defined code. In case of a simulation clock, the controller uses a basic time step which is the greatest common divisor GCD of all time steps of the clocks of time dependant agents. In case of time independent agents, their code is activated at every pace of the controller.

Interfaces: The interfaces of agents are implemented with a byte array. Two agents communicate through a set of bytes eg from index a to index b in this array, using b-a bytes. These two integers (a and b), describe the interface. All interchanged data (numerical quantities, text, fluctuation of a quantity for a period of time eg a waveform from t=k to t=k+10, signals or combinations) is described in bytes. Every interface may be used by two or more agents putting interchanged data in wide scope. An event handling mechanism notifies every agent using the interface (eg form a to b) that contents are changed. An agent may use as many interfaces as user requires. By using interfaces the user may project data from inside the agent to defined scope. Below is an example of a mote built up using four agents and the interface array. The parts of the interface array that are used by this mote are coloured with grey.

In the example below the PIC emulator uses 3 interfaces, of different width. Similarly, we could consider in the mote model the AVR processor or any other processor. The pace of the emulator is one machine cycle, (usually for MCUs execution of one instruction). In the lab we may measure the exact energy consumption of one cycle. This emulator uses as input the compiled program to be uploaded to the real device. The battery agent subtracts from a starting quantity energy consumption of MCU and radio, sending a shutdown signal to MCU emulator in case of energy depletion deactivating the virtual mote. Sensor array agent sends analog data to MCU emulator. The interface is as wide as needed to describe the possible value range of data. Due to ADC conversion latency, the ADC may be modelled separately using a fifth agent.

Services: There are two types of services: (a) built in library. They are functions and procedures widely used in fields of engineering or science. (b) User defined: user writes code to describe the functionality of their procedure or function. Services are sets of procedures and/or functions in scope of any agent. They are activated when called by agents and their data is valid even when they are inactive. An implementation example is the RF medium. When a mote transmits, sends to RF medium service its coordinates, time, and transmission power along with data. RF medium service stores these values in a table. When a mote listens, the service calculates the signal at his location, according to all active signals above a given SNR. In case of an CSMA-CD MAC layer, both services (transmitting, listening) are active at the same time.

C. The proposed Agent based Interpreter

The interpreter consists of three main parts: (a) A 4XN integer array (b) The actual code that executes user programs (c) A byte array where data is stored.

1) 4XN Integer Array The simple program is:

1. If (A>0) 2. B← B+1 3. else 4.C←C+5 5. Endif 6. D←B+C

The Interpreter ends up translated in tokens and stored in the 4XN integer array. There are 4 basic types of token supported:

(a) decision (b) operator (c) assignment (d) variable or value. At the first column contains the token code. Columns 2 and 3 contain pointers pointing at the next token to execute in the token (4XN) array. Column 4 contains pointer pointing to data memory. In detail :

Decision: if A>0 is true then the next token to execute is the assignment token in line 2, else executes assignment token in line 4. Assuming that decision token code is number 1, and x and y are the indexes that tokens in lines 2 and 4 are stored respectively, then the line in which the decision token is stored in the 4XN token array would be like:

1	x	Y	

There is no pointer to data memory for there is no value involved. The expression A>0 is stored directly below the decision token row.

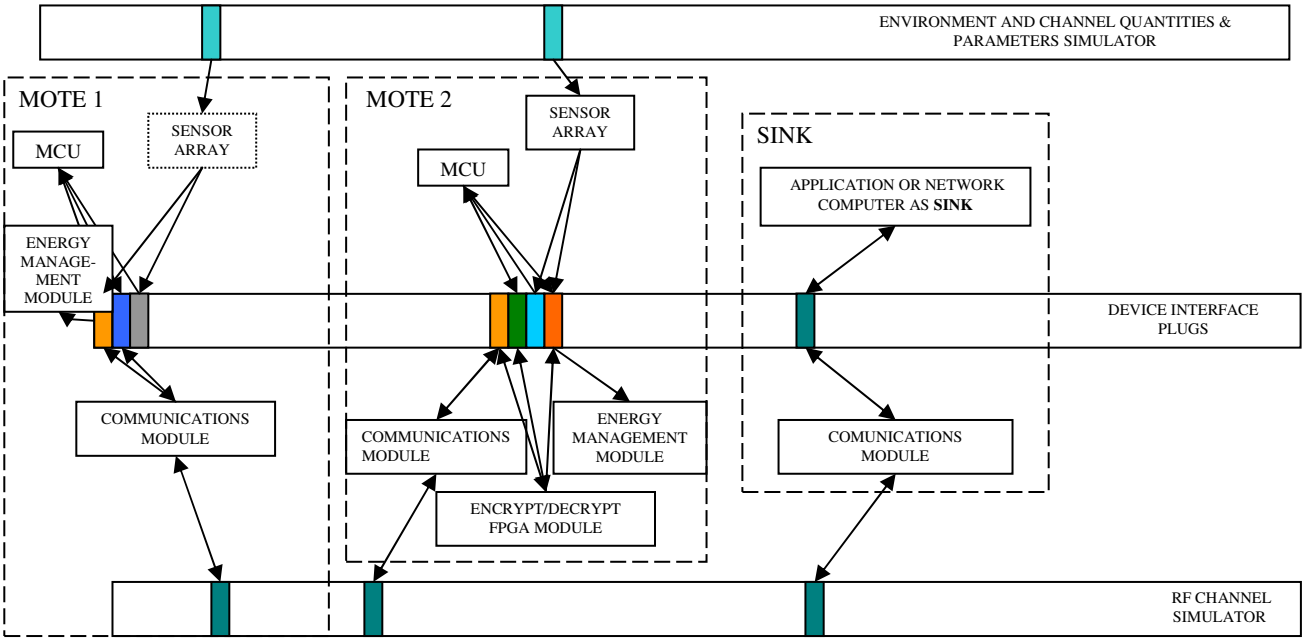


Figure 1a. The Multi-Agent System Architecture of the proposed WSN Simulator

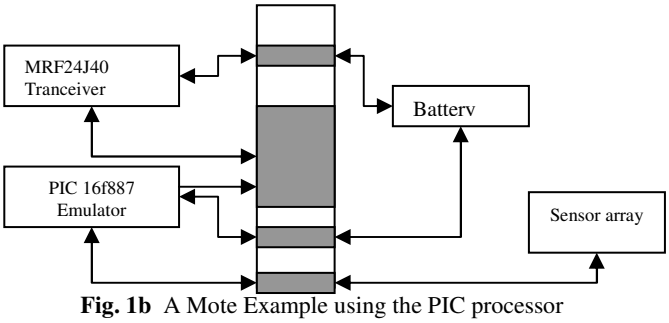


Fig. 1b A Mote Example using the PIC processor

When the interpreter reaches a decision token, evaluates the expression below and according to its truth selects next token (x or y). With the decision token we may also implement for, while, do until statements using the pointers x,y accordingly.

Operator: Operator > compares variable A with value 0. Assuming that operator > token code is 2, and variable-value token code is 3 then the expression is stored:

Indx	1	2	3	4
10	1	x	y	
11	2	12	13	
12	3	0	0	m
13	3	0	0	n

Memory
<A>
0

Fig 2. 4XN Token Array and Memory Array

The algorithm that evaluates expressions is :

```
Evaluate (x) { y=Token(x,2); z=Token(x,3)
IF z=0 and y=0 THEN return mem(Token(x,4));
ELSEIF z=0 THEN return op(Token(x,1),evaluate(y))
ELSEIF y=0 THEN Return op(Token(x,1),evaluate(z))
ELSE
Return op(Token(x,1),evaluate(y),evaluate(z))
ENDIF },
```

where x,y,z are indexes in the 4XN array of tokens, and m,n are indexes in the data memory (array of type byte).

Assignment: The assignment token evaluates the expression on its right, stores the result in memory, and proceeds to next token execution. The memory pointer in column 4 in an assignment row, points the location to store the results. The arrays needed are as follows:

Memory	
indx	
m	<A>
n	0
o	<B>
p	1
q	<C>
r	5
s	<D>

Index	1	2	3	4
10	1	14	18	0
11	2	12	13	0
12	3	0	0	m
13	3	0	0	n
14	4	22		o
15	5	16	17	0
16	3	0	0	o
17	3	0	0	p
18	4	22		q
19	5	20	21	0
20	3	0	0	q
21	3	0	0	r
22	4			s
23	5	24	25	0
24	3	0	0	o
25	3	0	0	q

Fig 3. The representation of the program in the two arrays as above described.

Data Types: The interpreter engine uses its own set of basic data types: Char, string, int, real, bool, byte, binary and their combinations (structs). Binary is used for binary numbers bigger than 255. From the user's point of view, data memory and interfaces are transparent. From inside the agent, interface is seen and used as a simple variable or a struct. The interpreter's engine is responsible for any transformation needed, relieving user from the task. The basic advantage of this interpreter implementation is that is simple and can be used also:

MCUs: The interpreter is uploaded in the MCU EEPROM. Application and protocol coding now becomes data. Sink transmits code along with data altering WSN mission at runtime.

In the 4XN array we may store 1 or more programs. Each program starts at a specific index of this array. The interpreter using a priority mechanism may execute these programs concurrently, by executing a number of tokens of each, in every pace. This feature makes the interpreter able to serve as a base of a WSN OS.

GPGPU: The function evaluate(x) is the part of the interpreter that calculates expressions. The other part is a simple switch (x) statement where x is the token code.

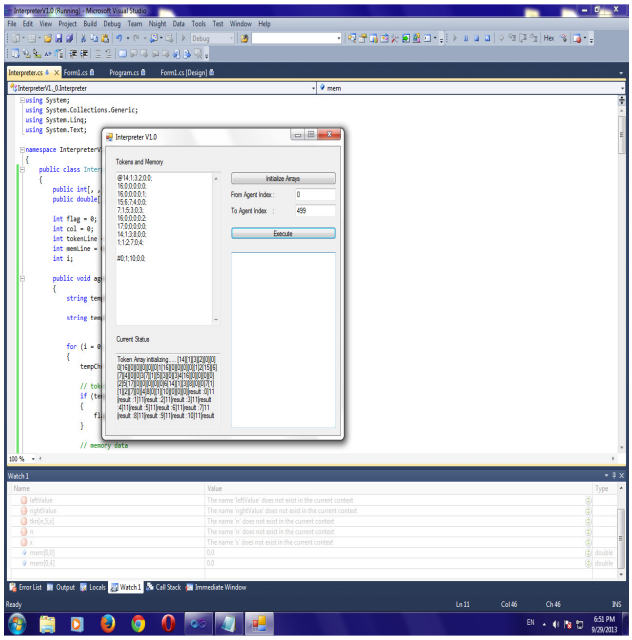
Using a non recursive version of evaluate(x) we eliminate external dependencies and calls. One would expect difficulty in handling the sum of data involved due to variety of data types. In our case all data and code are contained in two arrays. Code and data can fit in a CUDA thread. In other words, one agent in every thread. The amount of data is constant (arrays). Both are copied to GPU memory using:

#define X 100; #define Y 4; #define Z 1000 ; Int Tkn [X] [Y]; Byte mem [Z]; cudaMemcpy(dTkn,Tkn,X*Y*sizeof(int), cudaMemcpyHostToDevice); cudaMemcpy(devicemem, mem,z,cudaMemcpyHostToDevice);

4 PRELIMINARY VALIDATION OF THE NEW DESIGN FOR VARYING MOTE PROCESSORS COMPARED TO OTHER STATE OF THE ART APPROACHES

The proposed multi-agent simulation system architecture has been implemented in C#, and has been based in the emulation of motes using either AVR or PIC processors in a random selection of which mote holds the AVR or the PIC processor. In the following screen shot we present a sample simulation run of a WSN comprised of 500 motes, while our runs ranged from 100 to 5000 motes. Each mote runs eternally a simple programme like the above presented ones, emulating either the AVR or the PIC processor executing a subset of basic AVR or PIC commands (depending on the processor building up the specific random mote). The programme and all AVR or PIC specifications are loaded through parsing an associated XML File. All motes, for simplicity reasons, but without loss of generality run the same exactly AVR or PIC programmes. Each mote is designed and runs as an agent, independently of the others. However, the control agent synchronizes all these 100 to 5000 agents very efficiently. There are no delays for simulations of such a scale, and the overhead of controlling all agents, with regards to computing time consumption, is negligible when the number of motes ranges from 100 to 5000. These delays vary from 3*E^(-7) seconds for 100 motes to 5*E^(-5) seconds for 5000 motes, running in a Desktop PC with INTEL i3 processor, 4Gbytes RAM and Windows 8.1 operating system.

The proposed architecture is somewhat similar with the architecture of SENSE (Chen, Gilbert, et al., 2005). SENSE is built on top of COST. A simulation is dealt as a composition of components. Each component is implemented in C++, and communicates with other components via inports and outports. Inports and outports are used to reduce interdependencies between components and allow code reuse. The universal interface mechanism in our architecture provides the ability of interchanging any type of data and of any size and organization (structs). An interface may serve as a private channel used by two agents, or as a group channel, only by the use of two indexes that specify the channel width and location on the byte array. Avrora, AvroraZ, tossim and Atemu, are AVR emulators, with AvroraZ (de Paz Alberola, et al., 2008) giving the ability of emulating the cc2420 chip. All are limited to AVR MCU's implementations, and they do no support network-communication level simulation (Yi, Sangho, et al., 2008). SensorMaker (Yi, Sangho, et al., 2008) is written in C, provides network information in fine grain, (packet collisions, packet path, cluster layout) along with mote information mainly energy level. However, the modelling–coding gap still remains for the designer to fill in



5 CONCLUSIONS AND PROSPECTS

Based on literature reports, we conclude the set of desired features an efficient WSN simulating tool should deliver: (1) Easy to use (Yi, Sangho, et al., 2008) (2) code reuse (Chen, Gilbert, et al., 2005) (3) Hardware emulation (Levis, P., et al. 2003), (de Paz Alberola, et al., 2008) (4) Visualization and interpretation of data (Yi, Sangho, et al., 2008) (5) Network toolbox (Yi, Sangho, et al., 2008) (6) precise timing (Levis, P., et al. 2003), (de Paz Alberola, et al., 2008).

Our herein developed proposal for a generic multi agent WSN simulator has been carefully considered as able to fulfil the above characteristics. Using agents and services the user may implement hardware emulation, independently of the processors involved in mote architecture and design, channel – medium (RF or sound) modelling, environmental phenomena modelling, event monitors and handlers, data visualisation interpretation and storage. All components of the simulator are connected directly, or via the multi type interface structure. The simulation controller activates each agent (a) according to a time interval (time dependant agents), (b) at every pace (continually), (c) or according to an event (a change in an interface, a memory location, a variation of an environmental parameter etc). Agent or service behaviour is implemented in code which is executed by the interpreter or the simulator. The interpreter's back end is en array of tokens in which the user's code is translated. The interpreter's front end is a C# like language, but in the future others could be supported to (Matlab, Delphi, Basic etc.). Controller may

control channel (RF or sound) simulation, Environmental (temperature variations over an area) simulation and device simulation/emulation (model/real code). This agent based proposal is evaluated in preliminary experiments, concerning mainly the overhead delays imposed in the simulations, when the number of WSN motes ranges from 100-5000, and when the motes involve randomly either AVR or PIC processors running the same interpreted programs. The prospects of this proposal consider further implementations and evaluations in real life projects handling complex events and not simple similar codes as in the current stage.

REFERENCES

Masri W.,Mammeri Z. "On QoS Mapping in TDMA Based Wireless Sensor Networks" Wireless and Mobile Networking IFIP Joint Conference on Mobile Wireless Communications Networks MWCN 2008.

Bernd-Ludwig Wenning et al. "Environmental Monitoring Aware Routing in Wireless Sensor Networks" Wireless and Mobile Networking IFIP Joint Conference on Mobile Wireless Communications Networks, MWCN 2008

S. Dziembowski, A. Mei, A Panconesi "On Active Attacks on Sensor Network Key Distribution Schemes" Algorithmic Aspects of Wireless Sensor Networks 5th Int. Workshop ALGOSENSORS 2009 Rhodes Greece July 10,11 2009

Karlof, C., Wagner, D.: "Secure routing in wireless sensor networks: attacks and countermeasures." Ad Hoc Networks 1(2-3), 293–315 2003.

Serpedin E. Chaudhari Q. "Synchronization in Wireless Sensor Networks Parameter Estimation Peformance Benchmarks and Protocols" Cambridge University Press 2009

Giridhar A., Kumar P.R. "The Spatial Smoothing Method of Clock Synchronization in Wireless Networks" Theoretical Aspects of Distributed Computing in Sensor Networks Springer 2011

Slijepcevic, S., Potkonjak, M.:" Power efficient organization of wireless sensor networks." Proceedings of IEEE International Conference on Communications, vol. 2, pp. 472–447 2001

Guoqiang Mao, Barış Fidan "Localization Algorithms and Strategies for Wireless Sensor Networks" Premier Reference Source – information Science Reference 2009

He, T., Stoleru, R., Stankovic John, A." Range free localization. Technical report" , University of Virginia (2006)

Timo Ojala et al "UBI-AMI: Real-Time Metering of Energy Consumption at Homes Using Multi-Hop IP-based Wireless Sensor Networks" Advances In Grid and Pervasive Computing GPC Oulu Finland 2011 pp 274-284

M.A Labrador, P.M. Wightman "Topology Control in Wireless Sensor Networks" Springer 2009

Watteyne T. et al "Centroid Virtual Coordinates–A novel Near – Shortest Path Routing Paradigm", Comp. Networks, 2008

Levis, P., Lee, N., Welsh, M., Culler, D.: "Tossim: Accurate and scalable simulation of entire tinyos applications." First International Conference on Embedded Networked Sensor Systems (SenSys 2003) (November 2003)

Khan M, Abdelzaher T.,Gupta K.K "Towards Diagnostic Simulation in Sensor Networks" Distributed Computing in Sensor Systems 4th IEEE International Conference DCOSS 2008 Santorini Island Greece June 11-14 2008

W.B. Heinzelman, A.P. Chandrakasan, H. Balakrishnan," An application-specific protocol architecture for wireless microsensor networks", IEEE Transactions on Wireless Communications , 1(4): 660-670, October 2002.

S. Lindsey, C.S. Raghavendra, "PEGASIS: power efficient athering in sensor information systems", in: Proc. IEEE Aerospace Conference, Big Sky, Montana, March 2002.

A. Manjeshwar and D.P. Agrawal," TEEN: A Protocol For Enhanced Eficiency in Wireless Sensor Networks", in Proceedings of the 1International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, CA April 2001.

H. Frey and I. Stojmenovic "On delivery guarantees of face and combined greedy-face routing algorithms in ad hoc and sensor networks" in twelfth ACM Annual International Conference on Mobile Computing and Networking (MOBICOM) Los Angeles CA USA ACM September 23-29, 2006, pp 390-401

T Watteyne, et al. "On using virtual coordinates for routing in the context of wireless sensor networks" in 18th Annual Int. Symposium on Personal Indoor and Mobile Radio Communications (PIMRC) Athens Greece IEEE, 3-7/9/2007.

Inatanagonwiwat, C., Govindan, R., Estrin, D.: "Directed di□usion: A scalable and robust communication paradigm for sensor networks." Mobicom 2000, Boston, MA, USA (2000)

Jon S. Wilson ed. "Sensor Technology Handbook" Elsevier 2005.

Patrick Kuckertz et al. "Sniper fire localization using Wireless sensor networks and genetic algorithm based data fusion" Military Communications Conference 2007 MILCOM 2007 IEEE (2007)

Muhammad Imran et al., "Application-Centric Connectivity Restoration Algorithm for Wireless Sensor and Actor Networks" Advances in Grip and Pervasice Computing GPC 2011 pp 243-253

Habib M., Ammari "Challenges and Opportunities of Connected k Covered Wireless Sensor Networks - From Sensor Deployment to Data Gathering " Springer, 2009

Fraden J. "Handbook of Modern Sensors – Physics Designs and Applications" Springer Verlag 2004.

Balister P., Kumar S., Zheng Z., Sinsha P., "Trap Coverage : Allowing Coverage Holes of Bounded Diameter in Wireless Sensor Networks" Infocom/IEEE 2009

Cardei, M., Thai, M.T., Li, Y., Wu, W.: "Energy-efficient target coverage in WSN". In: IEEE INFOCOM (2005)

Cheng, M.X., Ruan, L., Wu, W.: "Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks." IEEE INFOCOM (2005)

Saukh O., Sauter R, Marron P.J. "Time-Bounded and Space-Bounded Sensing in WSN" Distributed Computing in Sensor Systems DCOSS 2008, LNCS 5067, pp. 357–371

Liang C.K., Chen Y.T " The Target Coverage Problem in Directional Sensor Networks with Rotatable Angles". Distributed Computing in Sensor Systems DCOSS 2008, LNCS 5067 pp. 264–273

Chen, Gilbert, et al. "SENSE: a wireless sensor network simulator." Advances in Pervasive Computing and Networking (2005): 249-267

de Paz Alberola, Rodolfo, and Dirk Pesch. "AvroraZ: extending Avrora with an IEEE 802.15. 4 compliant radio chip model." Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks. ACM, 2008

Yi, Sangho, et al. "SensorMaker: A wireless sensor network simulator for scalable and fine-grained instrumentation." Computational Science and Its Applications–ICCSA 2008 (2008): 800-810.

A. Filippou, D.A. Karras, P.M. Papazoglou and R.C. Papademetriou "On a Novel Simulation Framework and Scheduling Model Integrating Coverage Mechanisms for Sensor Networks and Handling Concurrency" GDC/CA 2010, CCIS 121, pp. 277–286