Atarraya: A Simulation Tool to Teach and Research Topology Control Algorithms for Wireless Sensor Networks

Pedro M. Wightman[1]
University of South Florida
4202 E. Fowler Ave.
Tampa, FL 33620
pedrow@cse.usf.edu

Miguel A. Labrador
University of South Florida
4202 E. Fowler Ave.
Tampa, FL 33620
labrador@cse.usf.edu

ABSTRACT

Topology Control is a well-known technique for saving energy in wireless sensor networks. Despite the fact that topology control algorithms and protocols have been extensively studied, they are currently unavailable in most, if not all, simulation tools. In this work we introduce Atarraya, a discrete-event simulation tool specifically designed for testing and implementing topology control protocols for wireless sensor networks. The simulation tool includes structures for designing both topology construction and topology maintenance protocols. In addition, Atarraya includes several key algorithms and applications that along with its graphical user interface can be used to support teaching activities. The correctness of the tool is validated using well-known results available in the literature.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]; I.6.8 [**Types of Simulation**]: Discrete event; C.2 [**Computer-Communication Networks**]; C.2.2 [**Network Protocols**]: Protocol Verification.

Keywords

Topology construction; topology maintenance.

1. INTRODUCTION

In the last years, the topic of Topology Control (TC) has received a lot of attention by the wireless ad hoc and sensor networks research community, as it is a very well-known technique for saving energy in Wireless Sensor Networks (WSNs). TC's main objective is to build a topology with a reduced number of active nodes and links that also preserves some important network characteristics, such as network connectivity and area coverage.

In the literature of topology control, many protocols have been presented to create this reduced topology on a wireless network. Most of these protocols were evaluated using simulation tools in order to deal with cost, experimental, and scalability problems derived from an appropriate evaluation, which involves a large number of nodes given that the effect of TC can be better appreciated in large and dense networks.

Nonetheless, despite the fact that topology control algorithms and protocols have been extensively studied, using both mathematical and proprietary simulation tools, they are currently unavailable in most, if not all, well-known and accepted simulation tools currently used by the networking research community.

The unavailability of such a simulation tool has important implications. From the research point of view, there is no trusted available tool where to implement new topology control algorithms and test them under the same assumptions and conditions. Therefore, fairness in results comparing different schemes is always an issue. Also, having no common tool consumes a lot of precious time implementing well-known algorithms for comparison. Finally, and more importantly, more and new research in the area is not encouraged.

Atarraya supports the new definition of topology control that considers topology control as two independent processes that work in an iterative manner. First, there is the process that builds the initial (reduced) topology, called Topology Construction. Then, there is the process that maintains (restores, rotates, or recreates) the reduced topology during the lifetime of the network, called Topology Maintenance. As of today, this is the first simulator that works based on this paradigm. This characteristic allows the seamless combination and integration of both protocols, which is very useful for determining the optimal pair that extends the lifetime of the network, which is the main goal of topology control protocols and many other protocols and algorithms for wireless sensor networks. From the teaching viewpoint, it is well-known that the visualization capabilities of simulation tools are a powerful method to explain complex concepts in an easy manner. Here, also, the unavailability of such a simulation tool does not allow professors, not researchers working on topology control, to teach and encourage experimentation in topology control further.

In this work we introduce Atarraya, a new simulation tool

[1]Professor on leave from Universidad del Norte, Barranquilla, Colombia. http://www.uninorte.edu.co

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIMUTools 2009, Rome, Italy
Copyright 2009 ICST, ISBN 978-963-9799-45-5.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

for teaching and researching topology control algorithms and protocols for wireless sensor networks. Atarraya is a Java-based, event-driven modular simulator for designing and comparing protocols for topology construction, topology maintenance, sensor-data management, and routing in wireless sensor networks.

In its current state, Atarraya is an excellent tool not only for research to develop and test new topology control algorithms, but also for teaching. Atarraya's graphical user interface shows how topology control protocols work, showing how the protocols shape the network topology during their execution. In addition, Atarraya includes necessary mechanisms to experiment with classical theoretical results related to topology control and wireless sensor networks, such as the giant component experiment and the critical transmission range (CTR) [3], the calculation of the Minimum Spanning Tree of a graph, and others, which are excellent examples for teaching fundamental concepts. Moreover, it can be used to illustrate common procedures found in most communication protocols, like a hello-reply protocol, or flooding schemes, and evaluate them in the context of the metrics that the simulator offers, like energy consumption and message overhead.

The rest of the paper is organized as follows. The next section discusses the related work. Section 3 discusses the scope of the simulator. In Section 4 the Atarraya simulation tool is described in detail, including its internal structure, the design of protocols, its features, etc. Section 5 describes the most important options included in the tool as well as the main assumptions and models utilized. Section 6 presents examples that illustrate some of the experiments that can be run. Finally, Section 7 concludes the paper.

2. RELATED WORK

Although there are many commercial and open source simulation tools available for networking research, the availability of reusable topology control simulation models is very limited. First, commercial tools, like Qualnet or Opnet, and open source tools, like ns-2, Omnet++, JiST/SWANS, and several others, are full blown network simulators, which include all major communication protocols in the OSI and TCP/IP communication protocol stacks at the Physical, Data Link, Network, and Transport, and Application layers. However, given that topology control is not clearly defined as a layer in any of the classical stacks, it is usually not included in the pool of available protocols. Second, and reinforcing the last aspect, topology control, contrary to MAC, routing, and transport layer protocols, is not a main stream area of research and the protocols do not necessarily need to be used in many other simulation scenarios.

The other important reason that explains the lack of related work on simulation and topology control is that these commercial and open source simulators are very complex tools, which require a steep learning curve in order to modify existing code or include new code to implement new algorithms. More importantly, this learning curve is associated with the fact that these tools offer many protocols and many features, that are, in most cases, not needed to evaluate the performance of topology control algorithms and protocols. To the best of our knowledge, only the work of [2], which describes the implementation of a very simple topology control protocol in Opnet, and the work of [9], which presents the implementation of a centralized topology control algorithm in ns-2, are publicly available. Although other tools have been implemented in Matlab, C, or Java, they are proprietary, and were developed to test a particular algorithm of interest.

In this work we present Atarraya, a discrete, event-driven simulation tool designed for researching and teaching topology control in wireless sensor networks. It offers a simple platform that allow researchers to develop and test topology control algorithms, in a very simplified communication stack, in order to offer a simpler and faster implementation of their algorithm. Atarraya comes with several well-known algorithms for rapid comparison analysis. In addition, it offers a complete tool for topology generation, statistics with relevant metrics for topology control, a visualization module, and some tools that can be used for reproduce or validate analytical work on topology control. These last aspects are of important teaching value.

3. TOPOLOGY CONTROL AND VIRTUAL NETWORK INFRASTRUCTURES - VNI

Atarraya includes new concepts and features not available thus far. For example, the literature on topology control usually refers to those algorithms in charge of reducing the initial topology, and ignores the fact that once the reduced topology has worked for some time, it has to be changed again; otherwise having topology control will be worse than not having topology control at all. Although some proposed topology control algorithms have also includes some sort of procedure to change the reduced topology, a formal definition about this important aspect in topology control has been missing.

In our recent work [5, 6], we have extended the scope and proposed a new definition of topology control, including a complete taxonomy. Under this new definition, topology control now consists of two processes. The first process consists of those algorithms in charge of reducing the initial topology, a process that we now call *Topology Construction*. The second process consists of those algorithms that will change the reduced topology for a new one when the current topology can no longer offer the service it was called to provide. We call this process *Topology Maintenance*. Therefore, topology control is now a continuous process that iterates from topology construction to topology maintenance until the network has no more resources to provide the service it was designed for. Atarraya includes complete support of topology control having separate modules for topology construction and topology maintenance algorithms, so any topology maintenance mechanism could actually make use of any topology construction algorithm and vice versa.

One of the most common assumptions in topology control algorithms is that they generate only one reduced topology. In our proposed taxonomy, these algorithms are classified as dynamic techniques, as they generate one reduced topology on the fly. However, we have found that another plausible way to perform topology control is to have several reduced topologies and rotate them as needed. These algorithms were classified as static, as all the reduced topologies were calculated at the beginning and stored in memory. This is similar to the lights of a Christmas tree, in which one subset of lights among various subsets (topologies) is active at any given point in time, and rotated over time. This rotation distributes the energy consumption between the subsets, while

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

keeping the network covered and connected. Each one of this subsets is what we call a *Virtual Network Infrastructure (VNI)*. Atarraya, also includes support for multiple trees to implement static topology maintenance techniques. In order to implement this feature, the simulator includes data structures to support multiple VNIs in every node, plus the possibility to differentiate events that belong to different VNIs in the queue of events.

4. ATARRAYA'S INTERNAL STRUCTURE

This section describes the internal structure of Atarraya. First, its main functional components. Then, the structure of the protocols is described in more detail, including how they communicate with the main class and with other protocols, how to initialize the nodes, and how to handle protocol events.

4.1 Functional Components

This section describes Atarraya's main functional components and how they interact with each other. The functional components offer the "big picture" necessary to understand the critical components of Atarraya. Figure 1 presents a global view of the internal structure of the simulator, which consists of the main simulator thread, the node handler, the batch executor, and the display manager. These elements are described next.

4.1.1 The main simulation thread

This is the core of the system. The simulator thread, defined in the class *the_sim*, is in charge of fetching the next event from the simulation event queue, and sending the event to the node handler for execution. An instance of this class is created by the method StartSimulation() whenever a simulation is executed. This class contains the event queue, the simulation clock, the display manager, the database with the data about the nodes, and the simulation agent, which is in charge of storing the simulation results for the reports in the respective logs.

When an instance of the *the_sim* class is created, it is necessary to add the initial events to the queue before the thread is started. The first thing the thread will do once started is to check if there are any events in the Event Queue. If the thread is started without any events, it will consider that an error has occurred, and the simulation will be suspended.

Once the first events have been loaded into the queue, the simulator thread can get started. The thread starts a loop that will execute until one of the three termination conditions is true: there are no events in the queue, all the nodes have reached the final state in the topology control protocol, or the protocols have called for the end of the simulation (for example, the TM protocol has found that the sink has no more neighbors, so the network is dead). If the first condition occurs and the simulator has not been notified that the protocols finished execution, it means that there was an error during the simulation, and it will be notified on the simulation report.

In the loop, the first thing the thread does is to verify if the event is valid. If so, the event will be registered (if this option was selected by the user), the simulation clock will be updated, and the event will be sent to the *NodeHandler*. There, the event will be delivered to the appropriate *EventHandler* according to the protocol. Once the event is executed, the simulator will go back to the loop and start again the process. The simulator updates the clock with the execution time of the events based on the fact that all the events in the queue are sorted by their projected execution time, so there is no such thing like a trip to the past.

Once the simulator breaks the loop by any of the finalization conditions mentioned above, the thread goes to the report construction section, saving all the events and statistics, as selected by the user. This section also takes into account whether or not the simulation is part of a batch execution, in which case all the data from all previous executions is kept until the last one finishes. All this information is stored in data structures that are stored in the report files after the simulation is finished. Reading this section of the code will provide the user with information about all the options for each configuration of report in both single and batch simulation cases. Once the simulation and the report building section finish, the thread ends too.

In the current version of the simulator, just one simulator thread can run at a time because there is only one data structure to store the topology, which is localized in the *atarraya_frame* class. Individual instances of the data structure running several simulations in parallel will consume all the resources of the Java virtual machine, especially if the network topologies are big.

4.1.2 The node handler

This class is in charge of defining the protocols to be used in the simulation and routing the event to the appropriate protocol once received from the simulation thread. The *NodeHandler* class defines four possible protocols that a node can have running during a simulation: Topology control, topology maintenance, sensor-data management, and communication-routing protocols. Given that there are different algorithms for each type of protocol, the main purpose of this class is to make that selection transparent to the rest of the simulator, so that no details about the selection are required in order to execute the simulation. When a simulation is started, this class creates the instances of the selected protocols in each of the four different categories. The simulation thread sends the next event from the event queue to the *NodeHandler* class. Once the event is received, it is routed to the appropriate protocol based on the protocol identifier included in the event.

4.1.3 The batch executor

The main purpose of the BatchExecutor thread is to perform operations that require multiple executions, such as creating a set of topologies, performing a large number of simulations, and the Giant Component test. Since these operations are run on a thread independent from the main one, the graphical user interface does not freeze while these operations are being executed, which allows for the interaction between the user and the simulator even while some of these operations are running in the background. This class is instantiated whenever one of the mentioned operations is started.

4.1.4 The display manager

The display manager, or *newpanel* class, is the one in charge of the graphical representation of the topologies. The heart of this class is the override of the Paint method of this class that extends a Panel class. All the painting options for the topology are defined in this method. The other methods

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

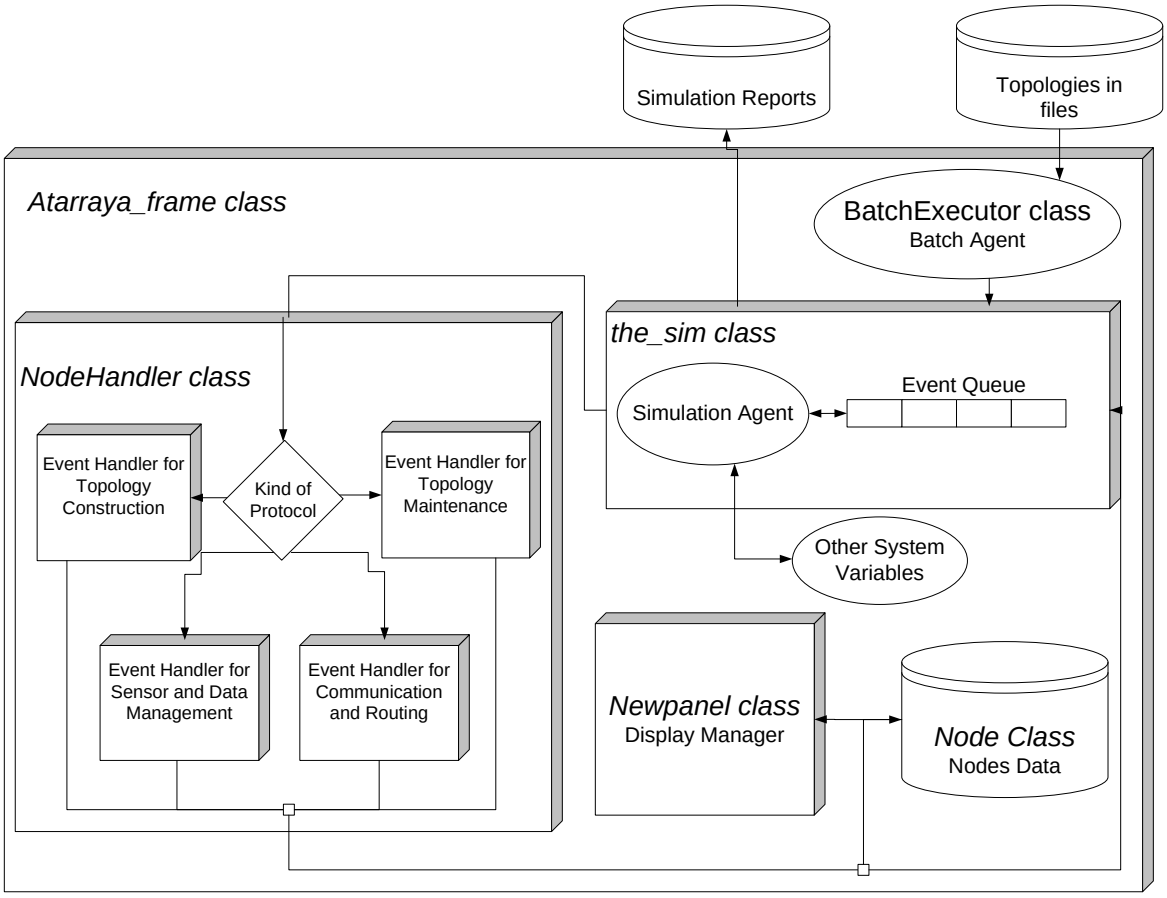

Figure 1: Atarraya's functional components.

perform minor but necessary actions like obtaining information about the options, providing coordinates from the deployment area, etc. This class was defined as a private class of the *atarraya_frame* class so it can have direct access to the topology data structure.

Atarraya provides several options for topology visualization, which can be seen in more detail in the visualization options in the main window of the simulator. The most relevant visualization options are the following:

- MaxPower Topology: This is the original view of the topology with all nodes transmitting at full power, and all the links that their unit disks provide.
- Single selected network configuration or tree: In this view the user defines which of the Virtual Network Infrastructures he or she wants to see. More on VNI later in Section 5.4. The default configuration is Black in most protocols.
- All network configurations: If several configurations are defined in a certain topology, this view allows the user to see all of them and appreciate the differences between them.
- Active network configuration: Each node is assumed to be able to maintain several VNIs, but use only one at a time. This view allows the user to see in real-time in which network configurations the nodes of the topology are.

4.2 Protocol Structure and Design

This section introduces the design and structure of the protocols in Atarraya. The next subsections describe the types of events that a protocol in Atarraya can model, how states are labeled, how each protocol communicates with the *atarraya_frame* class, how protocols interact with each other, how nodes are initialized, and how the simulator handles events.

4.2.1 Simulation Events and the EventHandler

Given that Atarraya is an event-driven simulator, everything that happens during a simulation is an event, so protocols must be defined in terms of cause-effect when certain event occurs. Each of these types of events triggers some internal actions in the node that might modify its status, data structures, etc., and could also cause the generation of new events in the future. The *EventHandler* class is the one that model the structure of a protocol in Atarraya and handles all the events. Common examples of events in Atarraya are sending and receiving messages. Also, Atarraya includes provisions to program an event in the future and invalidate a programmed event.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

The *HandleEvent* method is the core of the protocol, as it defines the actions taken by the protocol when an event occurs. The unique parameter that this method receives is the event taken from the event queue.

The events are classified based on an event label. Each protocol defines a set of labels for all the events that it uses. These labels are defined in the *constants* interface. The first action taken by the *HandleEvent* method is to recover all the fields from the event and store them in temporary variables. Depending on the nature of the protocol, the classification of the events can be done in different ways: Label-then-State or State-then-Label. In the first case, the most important information is the label of the event, which becomes the main classification factor. Once the label is found, the code inside determines if the state of the node is important or not for the execution of the actions associated with the event. In the second case, the most important information is the state of the node. This methodology is useful when there are not many types of events but each type is interpreted differently based on the state of the node.

4.2.2 Node state labels

In general, a good number of topology construction protocols use node states to represent the evolution of the protocol. In Atarraya, nodes can be in any of the following four states: Initial, Active, Inactive, and Sleeping states. These states are assigned to the protocols on the *Node_Handler* class. The values defined as the parameters are usually defined in the *constants* interface, and they are all positive integer values. For example: *tc_protocol_handler.setLabels(S_INITIAL_STATE, S_ACTIVE, S_SLEEP, S_SLEEP);*

Given that most topology control protocols implemented in Atarraya are completely distributed, the sink cannot call the end of the protocol because it has no information about the state of all the nodes. That is the reason why Atarraya knows that a protocol has finished when all the nodes have reached the final state. Each topology control protocol can define which states are selected as the final states. This is done in the method *CheckIfDesiredFinalState(int s)* that is defined in every *EventHadler*, which is invoked in the *atarraya_frame* class when the simulation agent is trying to verify if the topology control algorithm is finished. In the following example, the protocol is selecting the active and the inactive states as the final states of the nodes.

```
public boolean CheckIfDesiredFinalState(int s){
        if(s==active || s==inactive)
            return true;
        return false;    }
```

Atarraya stops whenever the nodes of the topology are in any of the selected states, no matter if there are still events in the queue.

4.2.3 Communication with the atarraya_frame class

Each protocol receives a reference to the instance of Atarraya's main frame, as defined on the *NodeHandler*. This reference allows the protocol to have access to variables from the main class. In order to access the variables from the simulator, the protocol needs to use the method *father.getVariable(int code)*, where the parameter *code* is a defined label for the sets of variables that can be accessed. This list can be found in the *constants* interface, and in the *atarraya_frame* class where the *getVariable()* method is defined. For example, if we need to know how many nodes are in the topology (including the sink nodes), the following line returns this value: *tam = father.getVariable(NUMPOINTS).*

When the protocol needs to get information about a node or modify it, the method to use is *getNode(int id)*, where *id* is the unique id of the node. In order to set node i in the initial state of the protocol in the VNI _vniID, the following line can be used: *getNode(i).setState(initial,vniID).*

4.2.4 Interaction with other protocols

There will always be some level of communication between protocols. For example, inter-protocol communication is needed to avoid situations like one node wanting to send a data message without having a route to the sink. Given that in Atarraya every event in the simulation goes to the same queue, it is necessary to determine to which protocol it must send the event to. Each type of protocol has its own identifier label, which is included in the event definition. This allows the *Node_Handler* to send the event to the appropriate protocol.

One of the premises of Atarraya is to create modular protocols that can be used in as many combinations as possible with the other protocols. Accordingly, protocols in Atarraya can only generate events in other protocols. For example, once a node reaches the final state of its topology construction algorithm, it can notify the topology maintenance protocol to start the maintenance procedure. Most of the times these inter-protocol events are meant to initiate or stop certain activity, so the protocol and how it works internally are completely independent, but the other protocols can decide the starting points. The following example illustrates a topology construction protocol when it invokes the topology maintenance protocol: *pushEvent(new event_sim(temp_clock+DELTA_TIME, temp_clock, receiver, receiver, INIT_EVENT, "", temp_tree,TM_PROTOCOL)).*

4.2.5 Initialization of nodes and the initial events

The *init_nodes(int vni)* method is used to set the nodes ready to start the execution of the simulation. Nodes are set to their initial states, and any previously defined events regarding other protocols and all necessary variables are set to their default values. This method is invoked in the *StartSimulation* method in the *atarraya_frame* class, for all nodes, including the sink. The following code is an example of a *init_nodes* routine, in which every node is set to its initial state, every state label is defined, any existent programmed event in the queue is cancelled, and the execution of the topology maintenance and sensor and data management protocols are reset.

The *initial_event(int _id, int _vniID)* method is used to define the first events to be included in the queue, before the simulation agent is started. Remember that if the simulation agent finds an empty queue it considers that the simulation has finished in an incorrect way. This method needs two parameters: the ID of the node that will perform the first event, and the VNI ID. This method is also invoked in the *StartSimulation()* method in the *atarraya_frame* class, but only for the sink nodes. If no sink nodes are defined in your topology, make sure that events are included in the queue using the *init_nodes(_vniID)* method.

5. HOW TO USE ATARRAYA

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

Atarraya offers a variety of options for designing and experimenting with topology control algorithms. This section provides an brief user guide on how to use the simulation tool and all its available options.

The first step is to have a clear understanding of the simulation scenarios to be run. Here, the user needs to know in advance which protocols he or she wants to use; whether the experiment is just a preliminary test or an exhaustive performance evaluation, what is the nature of the topologies that she is planning to use, what type of statistics are needed, and so forth. In this section, these questions are answered in order that a user may create and run successful simulations with Atarraya.

5.1 Selection of the protocols

Atarraya includes four types of protocols: Topology construction, topology maintenance, sensor-data, and communication protocols. Atarraya can be set to work in either of the following two modes related to topology control: Topology construction only, or All protocols. The first mode is designed to test a specific topology construction algorithm and measure the initial reduced topology that the algorithm produces. In order to select this mode, the user only needs to select a topology construction protocol.

The second mode is intented to test not only the reduced topology but the lifetime of the network, based on the combination of all the protocols, i.e. topology construction and topology maintenance. In order to select the second mode the user needs to select a protocol in each of the protocol categories, i.e. select a topology construction and a topology maintenance protocol, otherwise Atarraya will not allow the user to run the experiment.

The topology construction protocols included in Atarraya are based on algorithms presented in published papers. Currently, Atarraya includes the A3 [4], EECDS [10], and CDS-Rule-K [7, 8] algorithms. Although all types of topology construction protocols might be implemented, such as those based on changing the transmission range of the nodes, hierarchical protocols, cluster-based protocols, etc., the current version of Atarraya includes only those based on the Connected Dominating Set concept. In addition, and for the sake of comparison with a wireless sensor network without topology control, Atarraya offers a protocol that does not reduce the topology at all, called JustTree. The only service that this protocol provides is the creation of a forwarding tree to implement the constant gateway forwarding protocol.

Atarraya also includes all the topology maintenance techniques included in [5], i.e., static, dynamic and hybrid topology maintenance techniques. They are generic algorithms that work with any of the topology construction protocols. The simplest topology maintenance protocol included in Atarraya is the No Topology Maintenance protocol, which does nothing to maintain the initial topology, but monitors it until it dies. The protocol is in charge of informing Atarraya when this event happens. In Atarraya, the termination policy is defined as the moment at which the sink stops receiving information from the nodes.

The **sensor-data management protocol** models the behavior of the sensors, regarding variables like data transmission frequency, data aggregation policies, etc. Atarraya provides a simple protocol for sending and receiving messages without data aggregation. Nodes transmit data packets at predefined times, and forward every received data packet based on the forwarding policy explained in the following paragraph.

Although communication or routing protocols are not the focus of this simulator, some kind of routing procedure is necessary so packets can reach the sink. Given that the topology control protocols implemented in Atarraya are designed to produce a tree-like reduced topology, the tool provides a very simple forwarding algorithm that allows packets to reach the sink node: The **constant gateway forwarding protocol**. In this protocol, packets are always forwarded to a default gateway unless the destination of the packet is a direct neighbor, in which case it will be sent directly to that node. In a tree-like topology, the gateway of a node is simply its parent node. In this fashion, the packet will finally reach the sink node, since it is the root of the tree.

Atarraya does not include any routing protocol, but defines a data structure that models a routing table. This data structure allows users to develop more advanced routing protocols than the one currently implemented. In addition, the routing table can store a limited amount of packet sequence numbers (events have a field for this purpose) that allow the implementation of other forwarding algorithms like flooding-based protocols, or save the last versions of the routing protocol information packets, like routing tables on a Distance Vector protocol, or the last neighborhood information in a Link-State protocol.

In Atarraya's graphical user interface, the panel named *Atarraya* presents the available protocols. In addition, this panel contains the controls for the simulation agent, the report configuration options, simulation events and statistics panels, and the batch simulations controls.

5.1.1 Other protocols

Atarraya has been used to validate analytically derived equations and show special effects or behaviors of topology control algorithms. This section describes five tools included in Atarraya that are very well suited for educational purposes, as they allow users to:

- Calculate the Critical Transmission Range (CTR) based on the formulas of Penrose-Santi and Santi-Blough [3].
- Reproduce the experiment to obtain the Giant Component figures: Greatest Connected Component and Ratio of Connected Topologies.
- Reproduce the experiment that proves connectivity of the CTR formula of Santi-Blough for 2 dimensional deployments.
- Calculate the Minimal Spanning Tree on a given graph and provide the sum of the selected edges.
- Save the neighborhoods of a graph in a file.

The Minimal Spanning Tree of a graph is a classical tool for graph analysis - Prim's algorithm was implemented. Regarding the last tool, the information provided by it can be used to define and solve linear programming optimization problems on graphs, like finding a minimal set cover of the graph.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

5.2 Energy and communications model

In the design of Atarraya simplicity and focus on reaching a better understanding of the behavior of the topology control algorithms was embraced. As such, several assumptions were made to make the simulator simpler while still good enough for achieving its main objective. These assumptions are related to the energy and the communication models utilized in the tool.

The energy model included in Atarraya is based on the following formulas, taken from [1]:

$$E_{TXbit} = E_{elect} + \left(E_{amp} \cdot \left(\pi r^2\right)\right)$$

$$E_{RXbit} = E_{elect}$$

In addition, Atarraya also makes the following assumptions:

- During the idle time, a node does not spend energy. Even though this assumption has been proven untrue because being idle might be as costly as receiving data, this is still an assumption that can be done in some experiments, for example, the delay time for a topology construction protocol is very short compared to the total amount of time the network is going to be active, so the energy spent on idle listening is negligible. This function is being implemented for the next version of the simulator.
- The nodes are assumed to have one radio for general messages and a second radio for control messages: The main radio is used in all operations when the node is in active mode, and the second one (low power cheaper radio) is used to send and receive control packets to "wake up" the main radio. Only the main radio can be turned off, which means no messages will be received and no energy will be used. The secondary radio is assumed to use half the energy of the main radio.
- The sink node has a infinite source of energy. In general, the sink node is assumed to be powered from an external source of energy.

The communication model used in Atarraya is based on the following assumptions:

- The communication range of the nodes is a perfect symmetric unit disk. If $d_{x,y} \leq r_x \rightarrow x$ and y can see each other.
- A constant bit error rate (BER) is defined for the whole network. This is a simple implementation of an error model. Whenever a packet is going to be sent, a random number is generated and compared to the message error rate (that depends on the size of the message). If the random number is greater, the message is received, otherwise it is lost. The default value for the BER is 0, which means there is no packet loss. No sum of partially received packets will build a complete packet.
- Atarraya assumes that there exists a Data Link Layer that deals with packet losses and retransmissions, but it does not model this. In order to model some of the consequences of the operations of the MAC layer, the packets are delayed a random amount of time in order to model delays occurring due to retransmissions, contention, etc. The variable that defines the maximum delay value can be found in the *constants* interface, by the name of *MAX_TX_DELAY_RANDOM*. The default value for this variable is 0.2 time units.

5.3 Type of experiments

Atarraya offers two types of experiments: single topology based, that uses the visual representation of the topology, and the batch execution mode that simulates a large set of topologies. The single topology based type is good for protocol design and debugging. The batch type is made for full scale evaluation and analysis.

During the protocol design phase, it is very important to have the capability to run a small number of controlled topologies one at a time to be able to compare the results of several runs on a known scenario. This is a debugging phase where many changes are introduced in the protocol until it behaves as intended. During this process, a visual representation of the topology and the performance of the topology control algorithm is very helpful.

Once the protocol has reached a stable version that has worked well in several single topologies, the protocol needs a more exhaustive test with a larger number of topologies in order to analyze its average behavior. The batch execution mode allows the researcher to run simulations with hundreds of random topologies. In this case, the visual representation of the topologies is not necessary; actually, it would slow down the simulation process.

5.4 Structure of a topology

A topology in Atarraya is composed of four basic elements: Deployment area, regular nodes, sink nodes, and virtual network infrastructures or VNI. The deployment area is an abstract concept, which is useful for visualization purposes. It is a rectangle in which the user deploys the nodes of the network. In order to define the deployment area, the user needs to define its width and height.

The set of regular nodes is usually the biggest set of elements in the topology. They are in charge of monitoring the environmental variable or variable of interest, sending this information to the sink and routing packets. As with any wireless sensor devices, regular nodes are very limited in terms of resources.

The sink nodes are special nodes that, in most scenarios, are included to receive the information from all active nodes in the network. They serve as bridges between the wireless sensor network and any type of external network used to transport the sensed data to its final destination somewhere else in the Internet. In some cases they are also in charge of initiating, executing, and/or controlling the topology construction and maintenance protocols, routing protocols, etc.

Despite the fact that in real life the hardware configuration of the sink nodes is different compared to a regular wireless sensor device, Atarraya uses the same data structure to model both nodes, with the main difference that sink nodes are considered to have an unlimited amount of energy.

Atarraya also support several topologies at the same time through the VNI feature. It identifies the different VNIs with numbers from 0 to 6, and visually with colors: Black, Red, Blue, Green, Orange, Pink, and Yellow. Each node is assumed to have a separated data structure for each VNI, so the information of each one is independent from each

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

Table 1: Parameters for deployment creation

Parameter	Description
Number of Nodes	Integer value
Communication Radius	r_{comm} is an integer value
Sensing Radius	r_{sens} is an integer value
Position Distribution	- Uniform in a square area of size h and w, centered in (x, y) - Normal with $x' \in N(\mu_x, \sigma)$ and $y' \in N(\mu_y, \sigma)$ - Grid H-V: Distribute nodes in the deployment area with a distance of r_{comm} between nodes, so nodes are adjacent with their vertical and horizontal neighbors - Grid H-V-D: Distribute nodes in the deployment area with a distance of $r_{comm} \cdot \sqrt{2}$ between nodes, so nodes are adjacent with their vertical, horizontal and diagonal neighbors - Constant position: (x, y) for all nodes in the creation word - Center: (x, y) is the center of the deployment area - Manual: Use the mouse to select the position in the deployment area
Energy Distribution	- Constant position: $e \in \Re$ for all nodes in the creation word - Uniform with maximum energy e_{max} - Normal with $e' \in N(\mu_e, \sigma)$ - Poisson with λ_e
Sink?	This parameter determines if the nodes in the creation word are sink nodes
VNI Selection	This parameter associates sink nodes to the selected VNI
Inter-query time	Frequency of querying the sensor for readings. Only used by sensor-data protocols
Inter-reset time	Time period between the execution of time-triggered topology maintenance protocols
Energy threshold	Energy percentage differential used to invoke energy-triggered topology maintenance protocols. Every time the energy of a node changes this energy threshold value, since the last invocation of the topology maintenance protocol, the node will invoke the protocol again. For example, if the node has all its energy and the value is 0.10, the node will invoke the protocol at 90%, 80%, 70%, etc.

other. Regular nodes have no affiliation with a particular VNI, while each sink node is associated with a VNI to which it serves as a sink node. All sink nodes are assumed to be regular nodes by the VNI, that are not associated with them, keeping their characteristic of unlimited energy.

The following list contains all the available options to define the network topology:

- Sink or No Sink: Even though most wireless sensor networks contain one or more sink nodes, Atarraya allows the user to define a wireless sensor network without sinks.
- One or multiple sinks in a single VNI: Atarraya allows to define a single network with a single sink or multiple sinks.
- One VNI or multiple VNIs: Having multiple VNIs is the way in which the static global topology maintenance techniques were implemented, where there are several subsets of topologies and one sink.

5.5 Structure of the nodes

In terms of the nodes, Atarraya can manage homogeneous networks, in which all nodes have the same characteristics, and heterogeneous networks, where there is at least one node that is different from the others.

When creating a topology, the simulator works based on subsets of homogeneous nodes. Each set defines a family of nodes that share the same characteristics. For example, the user can define a set of "weak" nodes with low transmission range and low energy, and a set of "powerful" nodes with high transmission range and higher energy. Atarraya also allows for different random distributions for the location and energy of each group of nodes, that will allow the user to create denser zones in the topology, or zones with nodes that have less energy.

The data structure that models these families of nodes is called *Creation Words*. These creation words are created based on the values defined on the *Deployment Options* panel in the simulator. The user can create as many creation words as desired: these families can model from a single node to the complete set of regular nodes. In the panel there are two list boxes where the creation words are stored until the topology is created: The regular nodes list (top) and the sink creation words (bottom). There are three buttons for adding a new creation word, removing a selected creation word, and clearing both list boxes.

Table 1 summarizes the parameters that can be defined for a homogeneous family of nodes, as well as some other parameters that are related to the execution of the topology maintenance and the sensor-data protocols. All these options can be found in the *Deployment Options* tab, under the tabs named *Main Parameters* and *Other Parameters*.

5.6 Topology control performance metrics

Topology control protocols are compared based on a specific set of metrics that most of the available tools for simulating wireless networks do not include. The following list enumerates the most important metrics to measure the performance of the topology control algorithms.

For the evaluation of topology construction algorithms, Atarraya includes as performance metrics the number of active nodes per VNI, the overhead in terms of number of packets, and the area of coverage.

For the evaluation of topology maintenance algorithms, the metrics include the number of active nodes reachable from the sink during execution, the lifetime of the network, the number of data messages received by the sink, and the

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

network coverage during execution.

5.7 Simulation results

Atarraya offers three types of result logs that can be obtained from a set of experiments: General statistics, network lifetime, and the simulation event logs. The **general statistics log** can register the state of several variables in a periodical manner, or provide just one snapshot of the network at the end of the simulation. The most usually consulted variables are the simulation clock, number of nodes, number of sink nodes, number of active nodes, number of dead nodes, average node degree, average level of nodes (if level is used by the protocol), number of messages sent and received, number of data messages received by the sink, energy on the tree, energy spent, and area of communication coverage.

These values are stored in a text file where each individual row contains a snapshot of the variables of the network in the moment at which data was collected. If a experiment includes more than one topology, the user can decide if the data is going be stored individually per execution, or if all the results are going to be summarized in a single file. The usual format in which data is presented is in comma separated values (.csv), which is readable by most data analysis programs, like Excel, Matlab, etc. However, if the user does not want them to be saved in files, the *Report* panel has a text area that holds the statistics generated by the experiments in csv format. If just one topology is simulated, the user can use the *Stats* text area in the *Atarraya* panel, which presents the results in plain text format.

The **network lifetime log** registers the status of the active topology. This log stores information every time the topology maintenance protocol is invoked, or when a node dies. This specific log cannot be stored in individual files per execution; it is stored in a summarized format in csv format.

The information stored in the network lifetime log by a single topology is represented in four rows. The first row registers the moment in which the information of the network was calculated. The second row represents the number of active nodes that can still reach the sink node - those that still can provide information to the sink. This value is important because if the sink gets isolated, no matter how many active nodes remain, all of them are useless because the information they produce gets lost. In the case of having more that one VNIs, the program works with the active VNI in the nodes. The third row shows the ratio between the number of active nodes that can reach the sink (value found on the second row) and the maximum number of active nodes. This value is the percentage of active nodes that are still alive and connected to the sink. Finally, the fourth row contains the percentage of covered area by the active nodes in the second row. This information can be very useful in order to compare efficiency between protocols, in terms of number of active nodes and real covered area.

The **simulation event log** registers all the events generated by the simulator during the execution of a single topology. The complete set of events allows the user to debug the protocol by seeing the sequence of operations executed. This information is only available in individual files per topology.

Atarraya not only offers statistics from the complete simulation point of view, but from the individual node perspective. It is always very useful to know the status of the nodes at any given point in time to check if the algorithms are working as desired. This information can be found in the *Node Stats* panel. In order to get the information of a node, the user can do it in two ways: dragging the mouse pointer to the desired node and clicking over it, or typing the id number of the node in the text field and press the button Generate Stats. Once the node is selected, the point in the topology turns orange and increases in size.

Atarraya also allows for the visual representation of the reduced topology. The main window of Atarraya is divided in two environments: The deployment visualization area and the control area. One of the panels in the control area contains the visualization options: changing view from MaxPower graph to the reduced topology (Atarraya mode); showing the active nodes (Parent mode), communication and sensing coverage areas, and node id's; drawing a grid over the deployment the area, etc.

6. SIMULATION EXAMPLES

In this section we show some experiments run with Atarraya. The first example shows the classic experiment of the Giant Component [3], the second experiment compares several topology constructions algorithms, and the final one shows a sensitivity analysis using one topology maintenance protocol.

The giant component is a classical experiment to show the behavior of connectivity when the communication radius of the nodes is increased. Figure 2 shows the size of greatest connected component (GC), in percentage against the total size of the network, and the percentage of connected topologies versus the communication radius. The experiment was performed with 3 different network sizes: 10, 100 and 1000 nodes, over 1000 randomly generated topologies where the nodes wer uniformly distributed in a square area of 100x100 meters. Figure 2 shows the results produced by Atarraya, which are similar to the ones presented by Santi in his book [3].

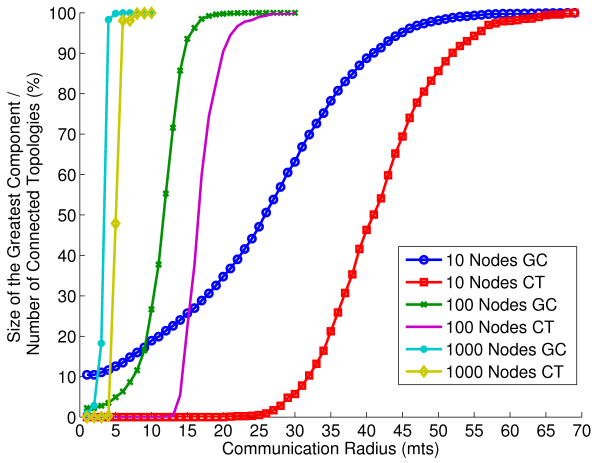

Figure 2: The giant component experiment.

The following example compares the performance of three topology construction algorithms, A3, EECDS, and CDS-Rule-K [4]. The performance metric is the average number of active nodes produced by the topology construction protocol on the initial topology. Figure 3 shows these results. In these tests other metrics can be used, like the number of sent

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565

messages, energy spent, area or coverage, and essentially all the variables found in the experiment's report.

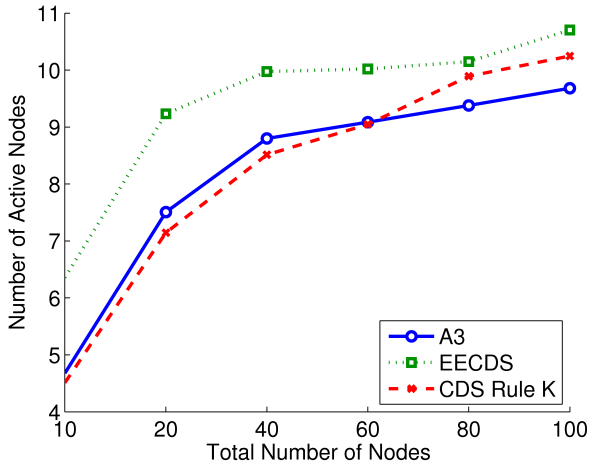

Figure 3: Active nodes vs. network density.

The final example shows a sensitivity analysis performed on a topology maintenance scheme, which tests the influence of the energy threshold in the lifetime of the network. Figure 4 shows the number of active nodes over time of the topology maintenance protocol called Dynamic Global Energy Topology Recreation (DGETRec), using 4 different energy thresholds, when the thresholds are set to 5%, 10%, 25%, and 50% of nodes' remaining capacity. Using this topology maintenance scheme, a new topology construction takes place using the A3 algorithm every time a node reaches the energy threshold set, i.e., when it has consumed 95%, 90%, 75% and 50% of its total energy. This kind of experiments are very useful when the designer is trying to determine the best maintenance policy for a given topology construction protocol.

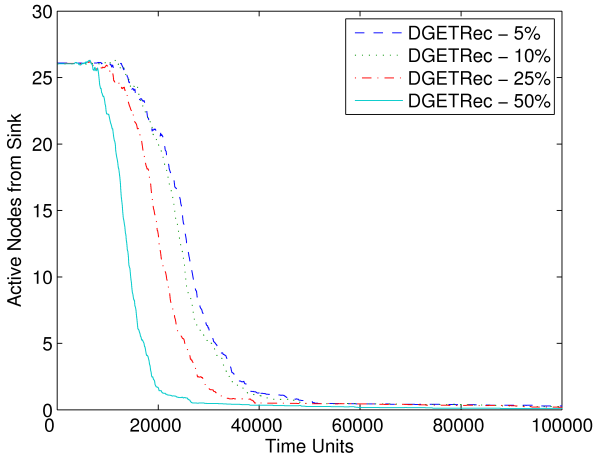

Figure 4: Example of a test of sensitivity of topology maintenance protocols in Atarraya

7. CONCLUSIONS

In this work we present a simulation tool for teaching and researching topology control algorithms and protocols for wireless sensor networks. The tool, called Atarraya, is described from the point of view of the developer and the user. First, the internals of the tool are described. Then, a brief user guide with the main options and features available is included. Finally, some examples are presented, which illustrate the potential of the tool for both teaching and research. Teaching experiments for educational purposes that can be used in the classroom include the replication of the giant component, or the calculation of the critical transmission range. Examples of research experiments are also included showing the performance of three diffrent topology construction algorithms and one topology maintenance scheme.

This simulation tool is still far from its full potential; for example, work is needed in important areas like the energy and communication models, in the implementation of more protocols, and in mobility models, extending its capabilities to wireless mobile ad hoc networks. It is hoped that by making the tool publicly available, it will trigger more research in the topic of topology control and the tool will get expanded. The application and the code of Atarraya is available at: http://www.csee.usf.edu/~labrador/Atarraya.

8. REFERENCES

[1] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1:660–670, 2002.

[2] N. Krishnamurthi, S. J. Yang, and M. Seidman. Modular topology control and energy model for wirless ad hoc sensor networks. *Online Proceedings of OPNETWORK Š04*, 0, 2004.

[3] P. Santi. *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons, September 2005.

[4] P. M. Wightman and M. A. Labrador. A3: a topology construction algorithm for wireless sensor networks. *Proc. of IEEE Globecom 2008*, 2008.

[5] P. M. Wightman and M. A. Labrador. *Topology Control in Wireless Sensor Networks*. Springer, 2009.

[6] P. M. Wightman and M. A. Labrador. Topology maintenance: Extending the lifetime of wireless sensor networks. *Proc. of WWASN 2009 (submitted)*, 2009.

[7] J. Wu and F. Dai. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(10):908–920, 2004.

[8] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd ACM international Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 7–14, 1999.

[9] L. Xu, H. Bo, L. Haixia, Y. Mingqiang, S. Mei, and G. Wei. Research and analysis of topology control in ns-2 for ad-hoc wireless network. *Complex, Intelligent and Software Intensive Systems, International Conference*, 0:461–465, 2008.

[10] Z. Yuanyuan, X. Jia, and H. Yanxiang. Energy efficient distributed connected dominating sets construction in wireless sensor networks. In *Proceeding of the 2006 ACM International Conference on Communications and Mobile Computing*, pages 797–802, 2006.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5565
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5565