Distributed Data Management for Large-Scale Wireless Sensor Networks Simulations

Stephen Wylie*
Dept. of EECS
Northwestern University
Evanston, Il

James Heide*
Dept. of EECS
Northwestern University
Evanston, Il

Besim Avci†
Dept. of EECS
Northwestern University
Evanston, Il

stevo,jamesh,besim@eecs.northwestern.edu

Dennis D. Vaccaro*
Grinnell College
Dept. of CS
Grinnel, IA

Oliviu Ghica†
Dept. of EECS
Northwestern University
Evanston, Il

Goce Trajcevski†
Dept. of EECS
Northwestern University
Evanston, Il

vaccarod@grinnel.edu

oliver,goce@eecs.northwestern.edu

ABSTRACT

We tackle two important problems that arise in simulation-based studies of various data-related properties in the context of Wireless Sensor Networks (WSNs): (1) reducing the turnaround time for completing the simulations in a large-scale parameter space; (2) providing database functionalities for a more detailed insight into the simulation's evolution. Towards these goals, we have developed the DiS-SIDnet (Distributed System for Simulation and Integrated Development for Wireless Sensor Networks). Leveraging upon our earlier works on the SIDnet-SWANS tool [3], DiS-SIDnet not only provides the ability of a synchronized execution of the simulations in a distributed environment, but also maintains the simulation data over the parameter-space in a relational database. In addition to post-simulation queries that can be posed to the database, we also provide the feature of specifying triggers that can generate notifications upon detecting certain events of interest during the simulation process.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Measurement

Keywords

Simulation, Sensor Networks, Databases

1. INTRODUCTION

Simulation is a methodology of studying the behavior of real-life systems via mimicking their dynamics and parameters in a suitable implementation. Over the years, there has been a plethora of methodologies proposed for simulating various phenomena [12].

When it comes to investigating the properties of Wireless Sensor Networks (WSN) and analyzing their behavior, actual testing in real-life settings before the deployment may be cost-prohibitive and sometimes even impossible. As the popularity of the WSN and their application domains have increased, researchers have developed several simulators and emulators that have been used for gathering data pertaining to, e.g., energy efficiency and lifetime, routing delays, etc... [5, 7, 13]. More recently, actual testbeds were made available to the scientific community for testing their finding on real motes, e.g., Kansei [2] with over 200 motes; TWIST [6] with over 100 motes. Projects like MoteLab [11] actually provide opportunities for a web-based login into a system, scheduling a particular task, and then querying the results of the execution that are stored in a corresponding instance of mySQL database.

However, when it comes to investigating various properties of interest for WSNs in scenarios with thousands of nodes, one still needs to rely on a simulation. Firstly, as we mentioned, it is the shear cost-factor of the actual deployment. Secondly, it is not a straightforward task to create all the different environmental conditions that one may need to investigate during the testing phase. For example, one may be interested in the impacts of simultaneous temperature

*Research supported by NSF-CNS: REU-1041567
†Research supported by NSF-CNS: 0910952

EDBT 2012, March 26–30, 2012, Berlin, Germany.

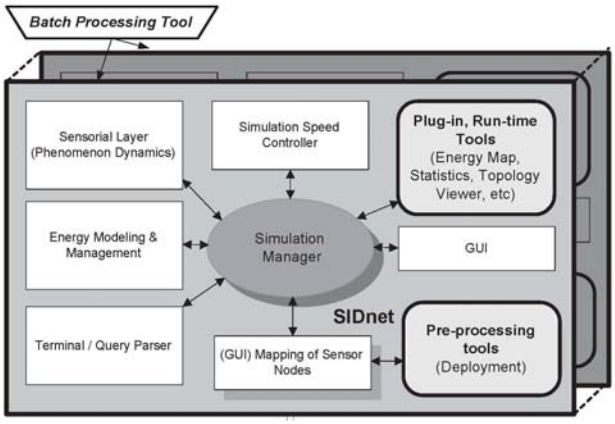

Figure 1: SIDnet-SWANS Features

increase in different regions of interest for the application domain; or the impact of an instant death of a large collection of nodes on the Quality of Service, etc...

Motivated by these issues, we have originally developed the SIDnet-SWANS tool [3] which provides a variety of options for the users to program, in terms of:
(1) Particular methodology (e.g., a routing protocol)
(1) fluctuations of the phenomena under consideration;
(2) properties of the WSN to be monitored (e.g., energy consumption or latency of a given protocol);
(3) batch re-execution of the simulator over a collection of different (values for) parameters' sets.
Figure 1 illustrates the relationship of the main components of the SIDnet-SWANS simulator that we have demonstrated in our prior work [3]. At the heart of the motivation for the current work are the observations that:

- Repeating tens of thousands of executions to cover the simulation over the entire desired parameters' space would sometimes take days to complete.
- Investigating the evolution of (the simulated behavior of) the WSN is not a common feature, although users may be interested in queries like, for example: "Retrieve the locations of all the nodes which have lasted over 2 hours during the high temperature fluctuations in more than 40% of the region of interest".
- Even though for some particular instance(s) of the WSN's evolution the user may wish to receive a notification regarding a property of the collective simulations/observations data, one could only rely on simulating the in-network types of events a'la TinyDB [8]. In other words, one can not receive notifications about some "meta-properties" that could steer the course of the (simulated) behavior like, for instance: "Notify me when more than 60% of the nodes have dissipated over 85% of their initial battery charge, in scenarios in which less than 30% of the initial nodes are dead and the fluctuations of the environmental conditions have not been frequent during 12 hours"

To provide these type of capabilities of investigating the evolution of the overall simulation process, we have developed the DiS-SIDnet extension of the SIDnet-SWANS simulator, which is the subject of this demonstration.

2. OVERVIEW OF DIS-SIDNET

In order to speed up the completion of a large collection of simulation runs, it is desirable that different instances of the execution are executed in parallel. Towards that, our main desideratum was to provide a flexible environment where the users/programmers can:

- Partition the executions along different (semantic) dimensions.
- Select the available resources to run the partitions.

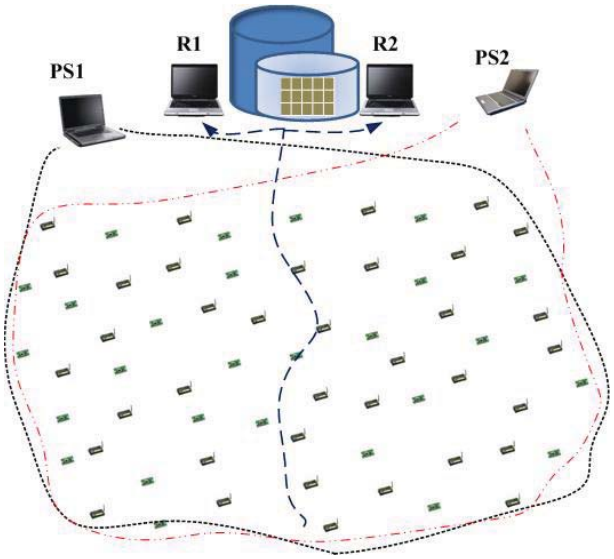

Figure 2: Combining simulation runs

An illustration of these features of DiS-SIDnet is provided in Figure 2. Instances of SIDnet are executing on four different computers PS_1, PS_2, R_1 and R_2 where:

1. PS_1 and PS_2 are executing concurrent tasks pertaining to different (disjoint) subsets of the *Parameters Space*.
2. R_1 and R_2 are executing concurrent tasks pertaining to different *Geographic Regions* of the WSN.

In the current implementation, one of the machines is designated as a "server", while the rest of them are acting like its "clients". Each client maintains a snapshot of the simulation execution pertaining to its own subset of the values across different dimensions, and the data is copied to the designated server either periodically or upon the completion of the entire subset of the simulation-runs at a given client.

If a trigger is specified for the purpose of raising a notification (e.g., based on the values of the monitored phenomena, or to the network status like energy-level per node being below certain threshold) and a particular client can detect the desired events, that client will send the notification to the server. Otherwise, upon subsequent downloads of partial

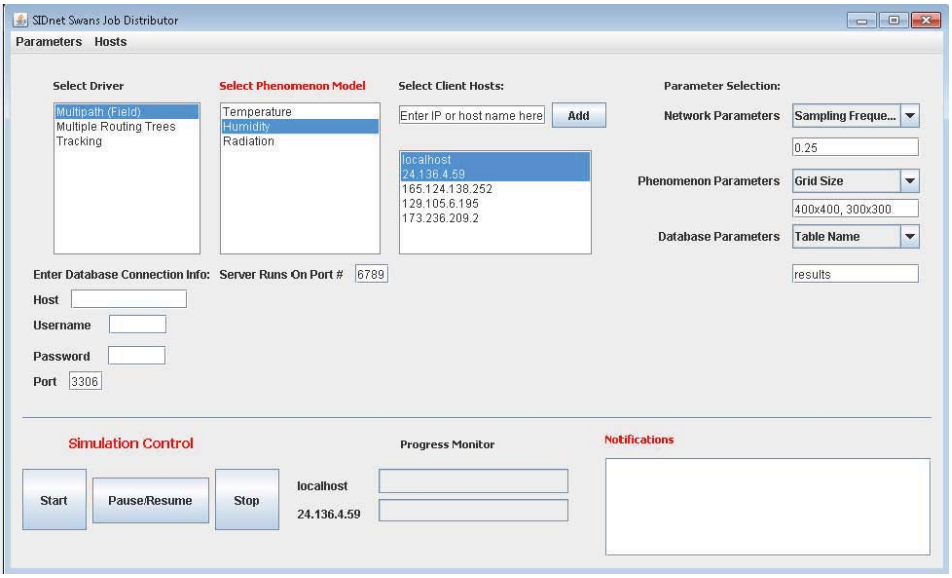

Figure 3: DiS-SIDnet User Interface

executions, the server will detect the event of interest (and fire the trigger) after combining the data from the different clients.

Figure 3 shows a screenshot of the main interface of the DiS-SIDnet server. As illustrated, the user has the opportunity to select IP-address(es) of the client machines that will be used to run concurrent instances of the simulation.

Upon determining the participating client, the selection of the driver determines which kind of a simulation will be executing – e.g., testing a particular routing protocol, testing a particular tracking protocol, etc. Subsequently, the user selects:

- Source file for modelling the fluctuations of a particular physical phenomenon. Similarly to the original SIDnet-SWANS implementation [3], we partition the geographical space into a grid of equal-sized cells, and we specify the functions describing the fluctuations of the phenomena for each cell. For example, if the monitored/sampled phenomena is temperature, we can limit the range for "cold" cells to be between -5°C and +3°C, growing linearly within that range during the simulation time.
- Parameters of interest like, for example, the number of nodes in a region of certain dimensions; the sampling frequency of a given node; the type of motion (if tracking scenarios are simulated); etc. For each parameter, the user is asked to specify the range of values, along with the "step" if needed.
- How a subset of the values from a given range of the corresponding parameters is to be distributed among the participating clients.
- The database[1] and the corresponding tables where the data from different simulation runs will be stored.
- Lastly, the user is given the option of specifying SQL queries and triggers which, upon insertion of new tuples from a given simulation-run, can raise notifications upon detecting certain events (provided the specified condition also holds). For example, the user may wish to be notified when more than 25% of the nodes within a rectangular region have dropped their battery reserves below 10% from the initial value, provided that the rate of change of the monitored phenomenon has not exceeded 20% of the total range per minute in the past 30 minutes.

[1]mySQL is used in the current implementation.

3. DEMO SPECIFICATION

We now present in detail the steps that will be taken in the individual demonstrations of the DiS-SIDnet, each of which is expected to run for 15-20 minutes. Every demonstration will execute the batch of the selected simulation-runs on three laptops (client plus two servers).

(**S1**) First, we will illustrate the use of the option that provides selection of different *drivers*. We plan on offering two such choices:

- Alternating among multiple routing trees used for data aggregation (cf. [9]).
- Minimizing the number of tracking principal nodes during the process of tracking mobile objects in WSN (cf. [4]).

As part of this step of the demo, we will explain the role of the driver both in the context of the individual instances of SIDnet-SWANS, as well as the Dis-SIDnet. As an example, Figure 3 shows a selection of the driver program for multipath routing using field-based paradigm [10].

(**S2**) Subsequently, we will demonstrate the selection of the *phenomenon*:

In this part of the demonstration, we will illustrate the coupling between the (simulation of the) particular phenomenon and the corresponding driver.

- For the data aggregation with multiple routing trees part, we will focus on "natural" values, such as temperature and humidity (cf. Figure 3).
- For the tracking scenarios, we will focus on models of motions (Random Way-Point, Gauss-Markov Mobility Model [1]).

(**S3**) Next, we will present the selection of the participating *clients IP addresses* from among the available ones.

(**S4**) The fourth step of the demonstration will illustrate the detailed selection of the *parameters* that describe the properties of the simulations from several different categories:

- *Network parameters*, e.g.,: initial battery charge, density of coverage, geographic dimensions/coordinates, sensing range, sampling frequency, epoch duration, etc. For example, in Figure 3 we show a sampling frequency of 0.25Hz, which is, sampling every four seconds.
- *Phenomenon parameters*, e.g.,: size/granularity of the grid covering the geographic region, source file containing the specifications of the phenomenon fluctuations per grid-cell, etc.
- *Database parameters*, e.g.,: table(s) where the tuples from individual runs will be stored and SQL statements for the queries of interest and/or triggers of interest.

We note that (**S4**) needs to be repeated for each client whose IP-address has been selected in (**S1**).

(**S5**) After the setup-steps, the fifth step of the demo will actually start the execution of the simulation-runs on the server and the selected clients. Throughout this step, the audience will be able to see the effect of the separate instances of SIDnet-SWANS executing concurrently.

(**S6**) The next-to-last step of the demo will illustrate the *run-time interaction features* at two levels:

- Globally: In addition to monitoring the progress (in terms of portion of completed runs in each host), the server has the ability to *Pause* a particular run (and the *Resume* it), as well as completely *Stop* it.
- Locally: Using the existing features of the SIDnet-SWANS (cf. [3]), the users can monitor the energy map of the network, or isolate a particular node and monitor its activities during the course of a simulation (e.g., packets sent/received).

(**S7**) The last portion of the demo will present the *database features* of the DiS-SIDnet:

- Raising notifications: we will use the interactive tools of the SIDnet-SWANS to instantly "kill" a large portion of the nodes in a given area, and show how the pre-set trigger (cf. (**S4**) above) fires, notifying the user that the energy level has dropped below certain threshold for a large count of nodes.
- Displaying answers to pre-defined SQL queries (cf. (**S4**) above).

4. REFERENCES

[1] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5), 2002.

[2] Emre Ertin, Anish Arora, Rajiv Ramnath, Mikhail Nesterenko, Vinayak Naik, Sandip Bapat, Vinod Kulathumani, Mukundan Sridharan, Hongwei Zhang, and Hui Cao. Kansei: A testbed for sensing at scale. In *SenSys*, 2006. http://ceti.cse.ohio-state.edu/kansei/.

[3] Oliviu Ghica, Goce Trajcevski, Peter Scheuermann, Zachary Bischoff, and Nikolay Valtchanov. Sidnet-swans: A simulator and integrated development platform for sensor networks applications. In *SenSys*, pages 385–386, 2008.

[4] Oliviu Ghica, Goce Trajcevski, Fan Zhou, Roberto Tamassia, and Peter Scheuermann. Selecting tracking principals with epoch-awareness. In *Proceedings of the 18th ACM SIGSPATIAL GIS Conference*, pages 222–231, 2010.

[5] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: A software environment for developing and deploying wireless sensor networks. In *USENIX Annual Technical Conference*, 2004.

[6] Vlado Handziski, Andreas Kopke, Andreas Willig, and Adam Wolisz. Twist: A scalable and reconfigurable wireless sensor network testbed for indoor deployments. Technical Report TKN-05-008, Technical University Berlin, 2005.

[7] Philip Levis, Nelson Lee, Matt Welsh, and David E. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys*, 2003.

[8] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM TODS*, 30(1), 2005.

[9] Goce Trajcevski, Oliviu Ghica, Peter Scheuermann, Roberto Tamassia, and Isabel F. Cruz. Alternating multiple tributaries + deltas. In *DMSN*, 2008.

[10] Goce Trajcevski, Oliviu Ghica, Peter Scheuermann, Marco Zuniga, Rene Schubotz, and Manfred Hauswirth. Improving the energy balance of field-based routing in wireless sensor networks. In *GLOBECOM*, pages 1–5, 2010.

[11] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. Motelab: A wireless sensor network testbed. In *IPSN*, 2005.

[12] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation (2nd Edition)*. Academic Press, 2000.

[13] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Workshop on Parallel and Distributed Simulation*, 1998.