SIMULATION-DRIVEN EMULATION OF COLLABORATIVE ALGORITHMS TO ASSESS THEIR REQUIREMENTS FOR A LARGE-SCALE WSN IMPLEMENTATION

Dimitris V. Manatakis, Michael G. Nennes, Ioannis G. Bakas, Elias S. Manolakos

Department of Informatics and Telecommunications, University of Athens, Greece
{dmanatak, eliasm}@di.uoa.gr

ABSTRACT

Assessing how the performance of a decentralized wireless sensor network (WSN) algorithm's implementation scales, in terms of communication and energy costs, as the network size increases is an essential requirement before its field deployment. Simulations are commonly used for this purpose, especially for large-scale environmental monitoring applications. However, it is difficult to evaluate energy consumption, processing and memory requirements before the algorithm is really ported to a real WSN platform. We propose a method for emulating the operation of collaborative algorithms in large-scale WSNs by re-using a small number of available real sensor nodes. We demonstrate the potential of the proposed simulation-driven WSN emulation approach by using it to estimate how communication and energy costs scale with the network's size when implementing a collaborative algorithm we developed in [12] for tracking the spatiotemporal evolution of a progressing environmental hazard.

Index Terms— WSN implementation, continuous object tracking, energy consumption, environmental hazard.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are becoming a mature technology that is used increasingly for tracking single or multiple moving targets (e.g. vehicles, humans, animals, objects etc.) [1-3]. The rapidly dropping cost of WSNs has spurred a lot of interest towards large-scale WSN solutions for detecting and tracking diffusive hazardous phenomena, such as wildfires, noxious gases, oil slicks, etc. [4-11]. Generating accurate and timely predictions of a hazard's spatio-temporal evolution characteristics (e.g. its direction and speed etc.) is of great importance to the authorities trying to confront the situation and manage its potentially catastrophic consequences.

Tracking a spreading hazard is, however, a fundamentally different problem from tracking individual target(s), since a diffusive phenomenon, also referred to as "continuous object", may cover a large geographical region and its size and shape may change continuously and in complex ways. The key idea of the methods proposed in the literature for continuous object tracking [4-11] is to try to delineate its boundaries, by identifying, at each time step, the sensor nodes that are located closest to it. Hence these methods suffer, by construction, from the severe limitation that they require unrealistic sensor node densities (i.e. thousands of deployed sensor nodes per km^2) to achieve reasonable accuracy, a fact that renders them impractical even for medium-scale environmental monitoring applications. Another disadvantage of these methods is that they do not provide information of predictive value regarding the spatiotemporal evolution of the hazard.

To address these severe limitations we have introduced in [12,13] a distributed approach which is based on probabilistic modeling and can continuously estimate, with high accuracy, the space- and time-varying evolution characteristics (direction and speed) of a progressing hazard using low density WSNs. As for all WSN schemes, computer simulations can be used to assess the expected estimation accuracy as a function of the network's density. However simulations fail to provide: a) accurate energy consumption estimates and how they scale with the size of the network, and b) information about the processing and memory requirements of the distributed algorithm's implementation. Since having such estimates is very important before attempting to deploy a large-scale WSN for environmental monitoring application the real question becomes, how can we meet this requirement without having to deploy a large-scale WSN?

To address this question we introduce a method which allows us to emulate the operation of a large-scale WSN deployment for environmental applications by reutilizing only a small number of real sensor nodes. The key idea of the proposed method is to re-allocate (virtually reposition) the available sensor nodes so that they implement WSN nodes located close to the hazard's front line as it evolves. Sensor nodes re-allocation has been used before in [14] to evaluate the performance of geographical routing protocols in large scale WSN. The scheme used in [14] has been

Acknowledgments: This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

specifically designed for evaluating routing protocols before network deployment and cannot be applied for the evaluation of energy aspects of collaborative WSN algorithms in general. To the best of our knowledge our proposed emulation scheme is the first attempt to use a small number of sensor nodes to realistically estimate the energy consumption of a collaborative algorithm in a large-scale WSN implementation. We demonstrate its capabilities using the distributed algorithm we introduced in [12,13] for estimating the spatiotemporal evolution parameters of diffusing environmental hazards. WSN emulation provides convincing evidence that the collaborative algorithm is suitable for large-scale WSN deployment since it respects the memory, processing and energy constraints of commodity sensor nodes used in WSN implementations.

The rest of the paper is organized as follows: Section 2 presents the basics of the distributed algorithm in [12] to the extent needed to follow the rest of the paper. Section 3 presents the WSN implementation of the algorithm. WSN emulation results are presented and discussed in section 4. In section 5 we conclude by summarizing our findings.

2. IN-NETWORK PROCESSING ALGORITHM FOR HAZARD EVOLUTION ESTIMATION

The main idea of the distributed processing algorithm presented in [12,13] is as follows: Depending on its communication range every sensor node has a defined *neighborhood* (see Figure 1a). As the hazard's front enters a neighborhood and gets detected, an ad-hoc *cluster* of three nodes is formed dynamically. It consists of the *Master* node, which is responsible for forming the ad-hoc cluster, and two additional *Helper* nodes, used to assist the Master in performing a hazard's local *model update*. Each sensor node is initialized with a *prior model* which represents the evolution characteristics (speed, direction, orientation) of a *local front* line segment of length equal to the diameter of the node's ideal circular *communication region* (see Figure 1b). Every time the model is updated, the *posterior model* parameters are propagated to the rest of the neighbors.

In the algorithm, a sensor node may at any time assume one of the following statuses:

Quiescent: Default and initial status.

Master: Responsible for cluster formation and for updating the local front's model parameters.

Slave: Responsible for monitoring the phenomenon upon receiving a Master's request. A Helper is a special Slave.

The algorithm can be separated into three phases. To facilitate their presentation we will use a simple example.

Cluster formation: When sensor node S_0 detects (say at time t_0) the approaching hazard's front (red dashed curve in Figure 1b), it checks to see if it satisfies some necessary conditions (presented in [12,13]) to become a Master node. If the conditions are met, S_0 initiates a *local timer* and broadcasts a message to notify its neighbors about its newly

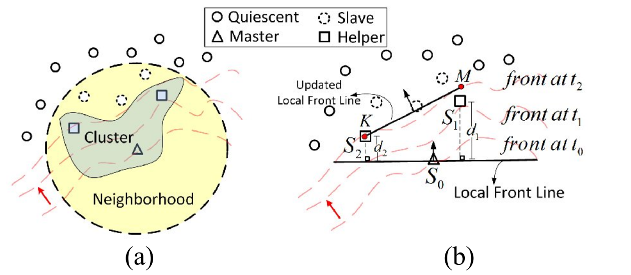

Figure 1: (a) Sensor node neighborhood. An ad-hoc cluster is formed as soon as the hazard is detected. (b) Local front model updating procedure. See text for details.

assumed Master status. When the neighbors receive this message they become its Slaves. In the sequel, Master S_0 waits until it receives two hazardous event *detection messages* from two Slaves (those to be used as its Helpers) before initiating the prior model's updating procedure.

Model updating: Let's assume that at local times t_1 and t_2 (relatively to t_0) Master node S_0 receives the two detection messages it has been waiting for from Slaves S_1 and S_2 respectively w.l.o.g. (see Figure 1b). Using times t_1,t_2 and the Euclidian distances d_1,d_2 of the Helpers' locations from S_0*'s* local front line (see Figure 1b), Master S_0 can estimates the coordinates of two new points *K, M,* (shown as red dots in Figure 1b) and use them to determine the orientation of the updated (moved) local front line. Finally, using the analytical solutions (algebraic expressions) of a KL-divergence minimization problem formulated and solved in [12], S_0 can estimate the parameters of its posterior belief model, to complete the updating of its model.

Model propagation: After the model's update, Master node S_0 broadcasts a *propagation message* containing its new model parameters. When the enslaved neighbors receive this message, they replace their prior model with the new one. In addition, Master S_0 sends to the Helper that detected most recently the hazard (w.l.o.g. it is assumed to be S_2) a unicast *become-the-new Master* message. If S_2 *satisfies* the necessary conditions to assume the Master's role it accepts the request and initiates a new cluster formation procedure. If S_2 *does not* satisfy the necessary conditions, it rejects S_0's offer. When S_0 receives a *rejection message* from S_2, it attempts the same negotiation with its other Helper, S_1. If S_1 also declines the offer, Master S_0 gives up with its two Helpers, broadcasts in its neighborhood a *propagate-my-model message* (containing its posterior model parameters) and returns to Quiescent status. When its neighbors receive this message they also broadcast a similar message in order to further push S_0's posterior model out to *their* neighbors.

3. WSN IMPLEMENTATION

The WSN implementation of the distributed algorithm was based on the affordable Atmel Raven evaluation kit [15] consisting of AVRRAVEN boards (sensor nodes) and

RZUSBSTICK boards (sink node). The AVRRAVEN board has three main modules [16]: The ATmega3290 8-bit MCU which has 32 KB ISP flash memory, 1 KB EEPROM, 2KB SRAM and is responsible for handling the on board sensors. The ATmega1284P MCU, which has 128 KB ISP flash, 4KB EEPROM, 15KB SRAM, and is responsible for handling the 2.4GHz AT86RF230 radio transceiver designed for low-cost IEEE 802.15.4 applications. Its transmission power can be adjusted in the range [-17dBm, 3dBm] and its reception sensitivity was fixed to -101dBm.

The distributed algorithm was coded in C on the IPv6 ready RTOS Contiki [17]. The RTOS and the C code were loaded on the AVRRAVEN boards using the AVR Dragon programmer [18] and occupied 71KB on the ATmega1284P, i.e. 55.4% of its total ISP flash memory. We also designed a Java application, called RavenObserver, running on the host PC, to monitor the WSN and collect data from the sensor nodes during the conducted experiments.

Deploying a large-scale WSN to validate an in-network algorithm is unrealistic. To overcome this fundamental limitation we developed a simulation-driven emulation procedure which allows us to mimic the behavior of a large-scale WSN, during the evolution of a diffusive hazard, using only a small number of real sensor nodes. The basic idea of the proposed scheme is to cleverly re-use sensor nodes which are no longer able to participate in the distributed algorithm. Specifically we developed a technique which allows us to virtually re-position these nodes forward, in the direction of the hazard's front movement. For its virtual repositioning to be possible, a node should satisfy the following conditions: a) it must be at Quiescent state, and b) it must have detected the front of the phenomenon.

Using the Matlab-based WSN simulator presented in [12,13] we were able to create simulation scenarios with different sensor node densities, deployment strategies, and progressing hazard front evolution characteristics. For our evaluation we modified the simulator so that it can also generate an ASCII file containing the following setup record for each sensor node: {node ID, location coordinates, time of hazard's detection, IDs of its neighbors}. Using this file as input, the RavenObserver coordinates the re-use and virtual repositioning of the available sensor nodes (6 in our case), in order to emulate the behavior of the large-scale WSN as prescribed by the Matlab simulation.

Assuming that *N* real sensor nodes are available when a WSN emulation experiment is initiated, RavenObserver checks the Matlab generated file and extracts, for the *N* nodes that have detected the phenomenon first, the aforementioned setup records. Then, the WSN sink node takes over and sends this information to the available *N* real sensor nodes. Upon reception, the RavenObserver starts an internal timer and checks in the Matlab generated file the hazard's expected detection times for the aforementioned sensor nodes. When the timer reaches the detection time of a sensor node, RavenObserver asks the sink to send a special message to it in order for that node to start emulating the detection of the hazard. Upon reception of this message, the node changes its status and acts as prescribed by the algorithm. Finally, when a node is no longer able to participate in the algorithm, it sends a special message to the sink, which in turn informs RavenObserver that this sensor is available for re-allocation. Then RavenObserver finds the record for the node that is expected to detect the phenomenon next (according to the hazard's evolution simulation) and asks the sink to forward this record to the freed sensor node. This emulation method works w.l.o.g with any number *N* of available sensor nodes *(N>3),* albeit the emulation time depends on that number.

4. EVALUATION RESULTS

We now present experimental results, obtained using the available ATMEL Raven sensor nodes that helped us assess the processing, communication and energy efficiency of the collaborative algorithm and how it is expected to scale with the WSN's size in practice. Since the main contribution of the specific algorithm is its ability to estimate accurately the hazard's evolution parameters using low density WSNs, in our experiments we used node densities that are considered low for environmental applications. Specifically we used 5 x 10^{-5}, 7.5 x 10^{-5}, 10^{-4} sensors/m^2, which correspond to 50, 75 and 100 sensor nodes deployed within an area of 1 *km*^{2}. In order to establish that we have a connected network we use the transmission (Tx) powers shown in Figure 3. A Matlab program was used to generate random sensor node deployments. With *N* AVR Raven nodes available, we can emulate deployments in which every sensor node has at most *N-1* neighbors, and *N* is larger than 3.

To simulate realistically the behavior of a diffusive hazard we used a wildfire simulation software called FLogA (Fire Logic Animator) developed in our group [19]. FLogA is a web-based interactive software tool which allows us to draw a forest area on Google Earth anywhere in Europe, insert ignition points, simulate realistically and geo-animate the behavior of the evolving fire line under different prevailing wind conditions. Using FlogA we have generated five different fire scenarios affecting the same square forest area of 1 *km*^{2} in Hymettus mountain, Attica, Greece. For each scenario, the fire ignition points were placed at different locations, giving rise to very different wildfire front evolution patterns. The duration of each wildfire experiment was set to *180min* in order to guarantee that most of the forest area would be affected by the wildfire.

During the emulation of the network's operation RavenObserver collects the following information from the available Raven sensor nodes: a) the number of the received/transmitted (Rx/Tx) messages b) the number of the Rx/Tx Bytes c) the energy consumed by Rx/Tx operations and d) the computation time required for each model update. This data is collected by the sink node. At the end of the

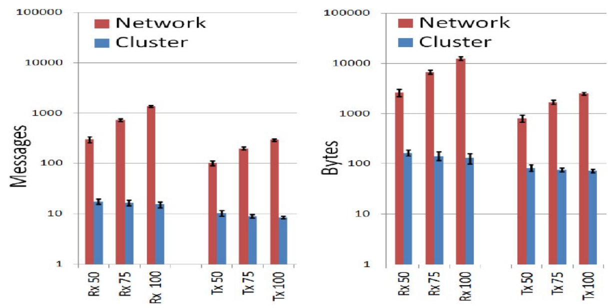

Figure 2: Mean and stdev of received/transmitted Messages/Bytes, at the Cluster and Network level for different WSN densities.

emulation, it is analyzed at two levels: *(a) Network-level:* Considers the data of all sensor nodes participating in the simulation, *(b) Cluster-level:* Analyzes the data of the nodes participating in the local front's model updating procedure (Master and its two Helpers).

Figure 2 provides, for each analysis level, the mean of the Rx/Tx Messages and Bytes and their stdev, for three density scenarios. The statistics for each scenario were computed considering all the corresponding Rx and Tx Messages/Bytes for the 50 different simulation runs (10 random deployments * 5 wildfires). We observe that the bar plots for the Messages (Bytes) follow similar trends as the number of deployed sensors increases. This is as expected due to the direct relation between Messages and Bytes for a given density. From the cluster-level analysis, we observe that as the number of deployed sensors increases the number of the Rx and Tx Messages and Bytes decreases a little. This behavior is justified if we consider that: a smaller number of sensor nodes in a given area implies fewer neighbors which in turn implies that it is more difficult for a Master node to "inherit" its Master status to one of its Helpers. As mentioned in section 2 (Model propagation), when a Master cannot find a new qualified Master, it is forced to broadcast a message to its Helpers, so that they can propagate its updated model to their neighbors. These extra "negotiations" are responsible for the small increase of Messages (Bytes) as the number of the deployed sensors decreases. However, at the network level an increase in the number of deployed sensor nodes leads to an increase in the total number of model updates, and therefore the total number of the Rx and Tx Messages and Bytes increases accordingly.

Figure 3 provides, for the 100 sensor nodes density scenario, the mean and stdev of energies consumed for Rx/Tx. Similar trends are followed for the other densities and the corresponding plots have been skipped due to lack of space. For both levels of analysis, the energy plots show that the required Rx energy is on average larger than the Tx energy. At a first glance this may seem counterintuitive, but it can be explained if we consider that most of the messages are of broadcast type, which means that a single Tx corresponds to many Rxs (by nodes in the same neighborhood). Furthermore, a more detailed analysis at the cluster level shows that the Master consumes for Tx (blue

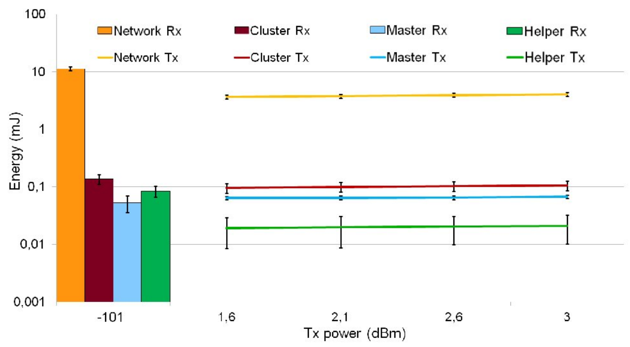

Figure 3: Mean and stdev of energy consumed for Rx and for Tx operations (as a function of the node's Tx power). Network and Cluster-level analysis for the 100 nodes scenario.

line) about 5 times more energy than both of its Helpers combined (green line). This happens because the Master transmits many more messages than its Helpers during the model's forward propagation (see section 2). The large stdev observed for the Helpers Tx energy is due to the negotiation which takes place between the Master and either one, or both, of its Helpers (in sequence) during the model's propagation phase.

Measuring the energy consumed by the nodes for processing and sensing tasks during emulation was not feasible since the AVRRAVEN nodes do not provide such functions. However, since it is well known [20,21] that the radio communication is the prominent energy consumer in a WSN, we can safely conclude that the provided Rx/Tx Network level energy results provide a good estimate of the lower bound of the total WSN energy consumed by the 100 nodes during the *180min* of operation captured by the wildfire simulation. This conclusion is further supported by the fact that the sensor nodes activate (exit the Quiescent status) only for the time period where the fire front line is close to their vicinity (emulated period). Furthermore, the mean computation time for a model update, required by the 8-bit ATmega1284P MPU of the Raven sensor node clocked at 8MHz, is about 510msec. (with a stdev of 5msec).

5. CONCLUSIONS

We introduce a simulation-driven WSN emulation procedure which allows us to estimate, even by using a small number of sensor nodes, the energy consumption and scalability of collaborative algorithms as the WSN's size increases. We demonstrate the validity of the approach by evaluating an algorithm developed in [12,13] for tracking the spatiotemporal evolution of spreading hazards. The results clearly indicate that this algorithm is suitable for a large-scale WSN deployment, since it respects WSNs' communication, processing, memory and energy constraints. The same emulation approach can be followed to assess the practicality of large-scale WSN deployment of other in-network algorithms of similar nature for environmental monitoring applications.

6. REFERENCES

[1] J. Liu, J.E. Reich and F. Zhao, "Collaborative in-network processing for target tracking," *EURASIP, J. Appl. Signal Processing,* vol.2003, pp. 378-391, Mar. 2003.

[2] L. Chen, M. Cetin, and A.S. Willsky, "Distributed Data association for multi-target tracking in sensor networks," *in Proc. Int. Conf. Information Fusion* Philadelphia, PA, July 2005.

[3] J. Liu, M. Chu, J.E. Reich, "Multitarget Tracking in Distributed Sensor Network," *Signal Processing Magazine*, vol.24, issue 3, pp. 36-46, May 2007.

[4] X. Ji, H. Zha, J. J. Metzner, and G. Kesidis, "Dynamic Cluster

Structure for Object Detection and Tracking in Wireless Ad-Hoc Sensor Networks," *In Proc. of the IEEE Int. Conf. on Communications*, 2004, pp. 3807-3811.

[5] J. Kim, K. Kim, S. Chauhdary, W. Yang, M. Park, "DEMOCO: Energy-Efficient Detection and Monitoring for Continuous Objects in WSN", *IEICE Trans.on Comm*, vol.E91-B, pp. 3648-3656, Nov.2008.

[6] W. Chang, H. Lin, Z. Cheng: "CODA: A Continuous Object Detection and Tracking Algorithm for Wireless Ad Hoc Sensor Networks". *Consumer Communications and Networking Conf*, pp. 168-174, 2008.

[7] C. Zhong, M. Worboys, "Energy Efficient Continuous Boundary Monitoring in Sensor Networks" Technical Report, 2007. Available: http://ilab1.korea.ac.kr/papers/ref2.pdf.

[8] M. S. Jin, F. Yu, S. Park, E. Lee, S.-H. Kim, "Localized Mechanism for Continuous Object Tracking and Monitoring in Wireless Sensor Networks," in *Proc. of the IEEE Conf Autonomous Decentralized Systems*, pp. 1-8 2009.

[9] W. Lee; Y. Yim; S. Park; J. Lee; H. Park; Sang-Ha Kim, "A Cluster-Based Continuous Object Tracking Scheme in Wireless Sensor Networks," *Vehicular Technology Conference (VTC Fall), 2011 IEEE* , vol., no., pp.1,5, 5-8 Sept. 2011.

[10] S. C. Tu, G.Y. Chang, J.P. Sheu, W. Li, and K.Y. Hsieh, "Scalable continuous object detection and tracking in sensor networks **"** *J. Parallel Distrib. Comput. 70(3):212-224* (*2010*).

[11] C. Yang; Q. Li; J. Liu, "A multisink-based Continuous Object Tracking in wireless sensor networks by GIS," *Advanced Communication Technology (ICACT), 2012 14th International Conference on* , vol., no., pp.7,11, 19-22 Feb. 2012.

[12] D. V. Manatakis, E. S. Manolakos, "Collaborative Sensor Network algorithm for predicting the spatiotemporal evolution of hazardous phenomena," *In Proc. SMC 2011 (special session on Collaborative Wireless Sensor Networks)*, Anchorage-Alaska, October 2011, pp. 3439-3445.

[13] D. V. Manatakis, E. S. Manolakos, "Predictive Modeling of the Spatiotemporal Evolution of an Environmental Hazard and its Sensor Network Implementation" *In Proc. ICASSP 2011*, May 2011, Prague-Czech pp. 2056-2059.

[14] B. Pavkovic , J. Radak, N. Mitton, F. Rousseau, I. Stojmenovic, "From real neighbors to imaginary destination: emulation of large scale wireless sensor networks "*ADHOC-NOW'12 Proc. of the 11th int. conf. on Ad-hoc, Mobile, and Wireless Networks,* pp. 459-471, isbn: 978-3-642-31637-1.

[15] AVR Raven – Atmel Corporation http://www.atmel.com/tools/AVRRAVEN.aspx. Last accessed (20/10/2013).

[16] AVR2016: RZRAVEN Hardware User's Guide http://www.jm.pl/karty/ATAVRRZRAVEN.pdf Last accessed (20/10/2013).

[17] A. Dunkels, B. Gronvall, and T. Voigt. "Contiki: A lightweight and flexible operating system for tiny networked sensors". *In Proc. of the 29th Ann. IEEE Int. Conf. on Local Computer Networks, LCN '04*, pp. 455–462, Wash., DC, 2004.

[18] AVR Dragon – Atmel Corporation http://www.atmel.com/tools/avrdragon.aspx. Last accessed (20/10/2013).

[19] N. Bogdos, E. S. Manolakos, "A tool for simulation and geo-animation of wildfires with fuel editing and hotspot monitoring capabilities,'" *Elsevier Journal of Environmental Modeling and Software*, Vol.46, August 2013, pp. 182-195.

[20] E Shih, S Cho, N Ickes, R Min, A Sinha, A Wang and A Chandrakasan "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks", *In Proc. of ACM MobiCom'01,* pp. 272-287 Rome, Italy.

[21] Raghunathan, V.; Schurgers, C.; Sung Park; Srivastava, M.B., "Energy-aware wireless microsensor networks," *Signal Processing Magazine, IEEE* , vol.19, no.2, pp.40,50, Mar 2002.