Case Study on the Reliability of Unattended Outdoor Wireless Sensor Systems

Agnelo R. Silva
EE - Electrophysics Department
University of Southern California
Los Angeles, CA
Email: agnelosi@usc.edu

Mahta Moghaddam
EE - Electrophysics Department
University of Southern California
Los Angeles, CA
Email: mahta@usc.edu

Mingyan Liu
EECS Department
University of Michigan
Ann Arbor, MI
Email: mingyan@eecs.umich.edu

***Abstract*—Designed to support in-situ soil moisture measurements for the National Aeronautics and Space Administration (NASA) SMAP Mission, the SoilSCAPE project started more than 4 years ago and currently it has more than 141 sensor nodes and 500 soil probes spread in 7 sites, each one with a Wireless Sensor Network (WSN) connected to a central data server by means of cellular text (SMS) and 3G links. These unattended WSNs are among the largest existing outdoor networks (in terms of node sparsity) and they are operating continuously for more than 2 years. All regular sensor nodes employ non-rechargeable batteries and the failure rate due to software and electronic issues is steadily zero during the years. A very high reliability of the system is paramount in order to reduce the total cost of ownership (TCO) of WSN solutions deployed in unattended and harsh scenarios because in many cases the maintenance costs may be multiple times the value of the device to be fixed. Therefore, we designed a Watchdog Timer (WDT) solution which is supported by a special software engineering methodology involving software and hardware modules that are *WDT-aware*. The main focus of this paper is to describe this novel WDT approach for embedded systems and to analyze its effectiveness for unattended systems, in particular outdoor WSNs.**

I. Introduction

Wireless Sensor Networks (WSNs) were introduced almost 15 years ago as a potential energy-efficient solution for spatially distributed sensors with very constrained energy resources. WSNs have been successfully applied in distinct scenarios, such as building automation, industrial process monitoring, environmental monitoring, security, etc. [1]. A strong evidence of the importance of WSNs is the crescent adoption of the IEEE 802.15.4 and ZigBee standards [2], [3]. Matching with the mentioned energy constraint, WSN nodes typically have few hardware resources in terms of processor and memory capabilities. Due to such restrictions, in general it has been widely accepted that the resilience of WSNs must be guaranteed by means of extensive network collaboration and a significant number of nodes since these devices, individually, may not be very reliable.

The typical trade-off in WSNs involves network performance (i.e., data throughput and latency) and energy consumption [1]. This aspect is the main difference between WSNs and any other wireless technology, even including modern mobile devices. Therefore, the low-level networking layers of WSNs are designed to operate at relative low data-rate (e.g., typically around 100 to 250 kbps, or less) in order to keep the radio module of the devices inactive for the majority of the time. Certain WSN scenarios are even much more complex because the energy profiles of the sensor nodes are heterogeneous. For instance, a WSN node can a) be directly plugged to mains, or b) employ energy-harvesting techniques, or c) have a nominal lifetime dictated by the usage of non-rechargeable batteries [4]. Moreover, in other cases, some of the WSN nodes are mobile ones and ad-hoc schemes must be in place. In order to deal with any of these scenarios, off-the-shelf WSNs typically employ solutions that strongly depend on the collaboration among the nodes. Accordingly, it has been observed that the majority of the simulations in the WSN literature involve a significant number of nodes (e.g., $>$100) with relative small inter-node distances (e.g., 10 to 100m) [5].

The SoilSCAPE project started more than 4 years ago and currently it is still evolving with more than 141 sensor nodes and 500 soil probes spread in 7 sites, each one with a Wireless Sensor Network (WSN) connected to a central data server by means of cellular text (SMS) and 3G links. The main goal of this project is to provide real-time in-situ soil moisture data for certain sites in order to validate/calibrate the algorithms employed by the SMAP Mission which is expected to provide estimations for soil moisture by means of Radar and Radiometer Remote Sensing [6].

The unattended WSNs used by the SoilSCAPE project are among the largest existing outdoor networks (in terms of node sparsity) and they are operating continuously for more than 2 years. Very impressive is the fact that almost all nodes employ non-rechargeable batteries, that are typically reported in the WSN literature as lasting *in practice* only few weeks or months. For the SoilSCAPE case, the minimum battery lifetime was conservatively designed for 15 months of operation while the majority of the nodes already surpassed this goal. The only exception in terms of energy storage is the Local Coordinator (LC) node which employs a regular energy system (with solar panel) typically used in weather stations. This is the case due to the high energy costs for the LC-Data Server link (SMS or 3G, in our case).

In this work, the main focus goes to the development of WDT techniques that result in a very reliable WSN solution for the SoilSCAPE project. The examples and associated discussions are provided in a way to facilitate the adoption of some of the concepts and engineering guidelines in generic scenarios involving unattended and critical devices, not necessarily WSN nodes. The novelty of our proposed WDT-

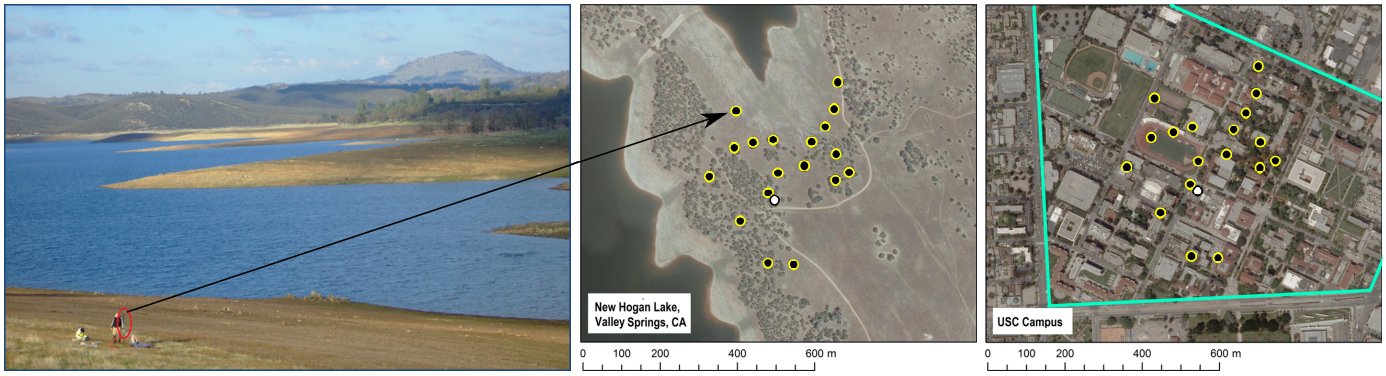

Fig. 1. One of the SoilSCAPE sites: the high degree of network sparsity is clearly visible when virtually transposing the same node topology to the University of Southern California campus. These nodes use off-the-shelf 2.4 GHz radio modules (10 dBm transmitting power level) and non-rechargeable batteries. No collaboration among nodes is in place and each node directly communicates with the data collector node (LC, marked with a different color).

based solution is related to two aspects. First, the hardware WDT technique adds a functionality that is time-proved to be much more effective than the regular *reset* action. Second, in terms of software engineering, we introduce the concept of a *WDT-aware* system where the choice of restarting/rebooting a device is not only associated with a *frozen* state, but with a broader definition for erratic states. We believe that this latter systemic aspect is the main contribution of this work, with particular higher impact on unattended outdoor WSNs.

This work is organized as follows: in Section II, the motivations regarding the SoilSCAPE project is given in order to provide some field and laboratory lessons regarding the deployment of outdoors WSNs. In Section III, multiple WDT techniques are discussed and the main goal of the section is to show missing aspects at the current approaches involving WDTs. The existing work with Virtual Finite State Machines is introduced in Section IV in order to show how to identify erratic states and how to associate this action with the WDT techniques. Finally, in Section V, the impact of using such WDT guidelines at the SoilSCAPE project is discussed and final remarks are provided.

II. Motivation

The SoilSCAPE project is potentially one of the most energy-efficient solutions so far developed for large and sparse outdoor WSNs in particular due to these characteristics:

- The application itself has very low data-rate: data collection (i.e., soil moisture and temperature) every 5 to 60min. (by default, every 20min.).
- No collaboration among nodes occurs: a regular node only communicates with the LC node.
- The network overhead is smaller than 1%.
- With the exception of the data collector (LC), no infrastructure node is in place.
- The energy consumption is highly homogeneous among all regular nodes.
- Because there are very big gaps between communication transactions (i.e., 2 to 4 seconds), multipath and similar effects are strongly minimized and it is possible to achieve inter-node distances very close to the theoretical maximum. This feature is particularly important for low-power and sparse outdoors networks.
- The network architecture we have developed, called Ripple-2, is open to any communication transceiver/module that can provide at least peer-to-peer wireless communication. The point-to-point data transfer reliability is ultimately enforced at the application-level, no matter if the lower networking layers provide this support or not.

In Fig. 1, it is illustrated one the remarkable characteristics of the Ripple-2 architecture used at the SoilSCAPE project: a high degree of network sparsity. To achieve such solution with all the previously mentioned features, we had to engage in a preliminary systemic investigation of the problem. The main aspects of this investigation are listed below. The details regarding each of these items, with the exception of the last one, are given at the provided references. In this work, the main focus goes to the last aspect regarding WDT techniques:

(a) Battery performance under extreme weather conditions [4].
(b) Techniques to achieve non-rechargeable battery lifetime close to the expected value considering its nominal energy capacity [7].
(c) Impact of enclosure (water and temperature damage) and solar panel characteristics for outdoor WSN nodes [5], [6], [7].
(d) Network overhead of typical WSN solutions [5].
(e) Effects of transients due to voltage regulators and systems involving rechargeable batteries and solar panels [4].
(f) Effective energy consumption of a node in sleep state [4], [7].
(g) Impact of having a star-like network topology and a non-collaborative approach in low data-rate WSN applications [4], [7].
(h) Effects of having an application-layer overlay activating and deactivating off-the-shelf radio transceivers and WSN modules [7], [8].
(i) Hardware and software Watch-Dog Timer (WDT) techniques to provide a robust solution for unattended nodes in outdoors.

At the beginning of the SoilSCAPE project, the approach was to use off-the-shelf WSN solutions. On August 2010,

two experimental sites were selected (Ann Arbor, MI and Canton, OK, USA) and ZigBee nodes [9] with a traditional energy scheme (solar panel + rechargeable batteries) were employed in a preliminary prototype called Ripple-1 [6]. After few months, we realized that Ripple-1 and off-the-shelf WSNs in general do not scale well when a high node sparsity is also important. We investigated the largest outdoor WSN deployments so far reported and a specific metrics was developed to better understand the sparsity issue [5]. That study shows that high physical coverage area in WSNs typically imply the use of a very high number of nodes.

In fact, the majority of the WSNs simulations usually employ more than 100 nodes to prove certain aspect related to scalability. Nonetheless, if we assume that only 30 sensor nodes are available for a certain WSN project, it can be more important to cover a 1000x1000m^2 area rather than a 200x200m^2 one. This is particularly true if the measurements have high spatial correlation as in the case of the SoilSCAPE project. In other words, if the environmental measurements are highly correlated at least at some physical spots, it is typically more strategic to install the available sensing resources in larger areas than to concentrate all sensor nodes in one or few spots. Besides a higher network sparsity, it is also important to drastically reduce the energy cost due to the intrinsic collaboration among WSN nodes. This aspect is not significant for high data-rate WSNs, but it is very critical for low duty-cycled applications, as investigated in [8]. Therefore, we started the design of the Ripple-2 architecture [7], [8].

This new architecture has a cross-layer networking approach which comprises application-layer software modules and a hardware module for power management and software-rule enforcement. It is an open architecture designed to support any kind of radio module that allows at least point-to-point communication (off-the-shelf WSN modules are also supported). Nothing is assumed in relation to the availability of communication error control at these radio modules. To support these design foundations, a complex system was necessary. The initial code of the Ripple-2 sensor node had more than $3,000$ lines (currently, $1,757$). Also, the code of the data collector node (with SMS/3G link) has almost $8,400$ lines. Moreover, the complexity of the underlying hardware module is also significant. For instance, our current regular sensor node employs 5 digital switches, 2 watchdog timers, and 1 energy sub-system to charge supercapacitors (the radio module is the load). As shown in Fig. 2, while an off-the-shelf WSN node would be basically comprised of the radio module (big circle), our Ripple-2 node has many more hardware components in order to achieve an energy efficiency at least 3 folds higher than typical WSN devices. Although this goal was ultimately achieved [8], the main trade-off is the risk of having a non-reliable solution. That is, such level of system complexity may potentially introduce hidden issues and decrease the system's reliability. This case study paper highlights our strategy to achieve a very high reliable solution for the SoilSCAPE project. Accordingly, one very effective way to deal with scenarios where WSN nodes are trapped in erratic conditions is to simply restart them (we must assume that these actions are rarely necessary) and this is the topic of the next section.

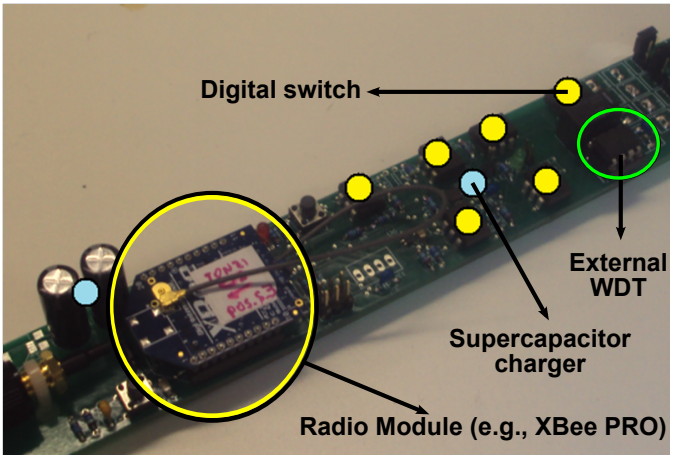

Fig. 2. Ripple-2 node: the additional hardware and software modules aiming energy-efficiency can potentially decrease the reliability of the solution. Accordingly, a WDT-aware design was adopted to allow the node to autonomously deal with problems without human intervention.

III. WDT: First Step Toward Reliable Systems

The adoption of component redundancy and the watchdog timer (WDT) technique are well-known topics at the embedded systems literature [10], [11], [12], [13] and they are typically regarded as ultimated solutions to achieve high-availability in computer-based systems. The WDT technique is basically a way to automatically reset the main processor after the device has been trapped in an erratic state for a certain amount of time. Clearly, WDT is a drastic solution that mimics the power-off/on action performed by a person as a last effort to solve a problem. A significant number of modern microcontrollers have an internal WDT and such level of implementation is fairly documented and does not require hardware modifications. Nonetheless, the mentioned solutions may not provide the expected results for embedded systems, in particular unattended WSNs deployed in harsh environments for two main reasons. First, a *reset* action is not always sufficient and second, if after a reboot, the WSN node can still be trapped in an erratic state that would require a human intervention.

The Ripple-2 architecture evolved from the adoption of a simple and traditional WDT scheme to a mission-critical oriented design and this process was based on progressive and long-term *laboratory* experiments and now validated by long-term field results. Our current Ripple-2 solution for both regular and LC nodes addressed the previously mentioned two potential issues regarding the use of WDT. Typical questions that we faced during this process and our conclusions are listed next:

1) Is the component redundancy the proper solution?
A generic redundancy of components, such as processor or radio module, may potentially decrease rather than increase the reliability of the solution. The new switching mechanisms that are eventually necessary when redundant modules are used may be potential sources of new and hidden issues. A more strategic and cost-effective approach is to perform careful tests of the various sub-modules in order to verify if any of them is particularly critical. For instance, we monitored what happened with the energy system of a node employing solar panel and rechargeable batteries at freezing temperatures.

In such critical energy-related scenario, we concluded that drastic oscillations of the voltage regulator may stop the processor even employing a regular WDT protection. For this specific scenario, the redundancy of components would not make the node more reliable.

Conclusions: The use of redundant components is only recommended for modules that have high probability of failure and, in this case, we are assuming that it is not possible to simply adopt a better solution. This is typically the case when long-distance communication links are involved. For this example, a higher cost (or, alternatively, with a lower bandwidth) connectivity solution can be used only as a backup for the cheaper one (or with a higher bandwidth). Another example, is the usage of multiple energy reservoirs, such as backups batteries [14]. It is important to highlight that these guidelines (and the remaining ones in this section) are not tied to WSN scenarios. For instance, remote and unattended SCADA (supervisory control and data acquisition) devices is another example of systems that can benefit with the proposed guidelines.

2) How to characterize an *erratic* state condition?
The typical approach used in WDTs is to add a code procedure that resets the WDT (this action is called *WDT refresh*) and this procedure is expected to be regularly called. If the main processor *freezes*, the WDT refresh does not occur and the WDT device eventually resets the main processor. Note that the processor in this case was in an erratic condition, but there are other scenarios where the *system* may also be in an erratic condition. This is a *key* observation in our studies regarding reliability: the concept of *state* must always be considered inside a certain context. Therefore, without a proper formalization of the system, an erratic state may have different interpretations and a system may be trapped in one of such erratic scenarios where the WDT technique is ineffective because the WDT refresh is occurring normally.

For instance, consider a scenario where a WSN node is in a loop, moving from state A to B and then to A, always waiting for a message reply from the network. Can we consider this an erratic scenario? The answer potentially depends on additional aspects such as, how much time the node is waiting, if there is another high priority task that the node is not performing due to this logical loop, etc. Note that not the necessarily the code programmer knows the answer, but the *system designer* must provide an answer. Our conclusion is that an *erratic* condition of a WSN node is characterized by the non conclusion of a *transaction* for a certain amount of time. The term transaction is associated with an expected sequence of events and the overall idea is that all processes are controlled by strict timing settings. In Section IV, software engineering guidelines that were used at SoilSCAPE project will be presented.

Conclusions: The interpretation of an erratic state is seldom associated with a) a frozen state (non-responsive processor) or b) code loops. For both cases, it seems that it is sufficient to add a software routine that regularly refreshes the WDT thus avoiding the reboot of the machine. This solution is very well known and effective for the majority of the real-world cases. However, it is also important to consider cases where the main processor is perfectly operating but the device is an erratic condition from a system point of view. In this case, WDT techniques alone are not sufficient and additional approaches, such as the one to be presented in SectionIV, are still necessary. Remember that if an unattended device presents such problem only once in a year, the human intervention still be necessary to manually restart this device.

3) How the power management sub-system is related to the reliability of the node?
Oscillations at the voltage rail used by the main processor is extremely critical because it can lead to irreversible damage of components, such as the real-time clock (RTC) chip, the EEPROM, and the system file of an SD card. Voltage regulators alone may not be the solution because the specifications of these devices may not hold for certain variations of the input voltage. In this context, we observed that a higher reliability is achieved when we separate the power line of the main processor and its satellite low-energy devices (e.g., RTC chip) from the rail lines used for *power-hungry* sub-systems, such as the radio module.

Conclusions: Because modern microcontrollers support low voltage levels (e.g., 1.8V), it is important to protect this component from issues regarding energy depletion. This is particularly important for devices that mainly rely on energy-harvesting. If the main processor is always protected (i.e., the its voltage rail never goes below 1.8V), it is always possible for this processor to log information regarding energy aspects. For instance, when the affected WSN node finally has enough energy to communicate, it is possible to inform the detailed energy-related reason why the node had not communicated before. In this sense, the reliability of the solution can eventually increase when a new version of the device has its design based on such kind of information.

4) How reliable is the WDT solution?
Many modern processors have an internal WDT and it may be used as the first level of the WDT solution. However, it is also possible to use an external WDT device, such as one RTC chip connected to the reset line of the main processor. This solution is more effective than the internal WDT and it can be employed as a second level WDT with very small energy consumption depending on the choice of the RTC chip. Nonetheless, we also observed that the most efficient WDT solution is an external microcontroller (MCU) completely dedicated to the WDT function. Such intelligent device allows a higher level of reliability control because it is an independent device that does not require any kind of cyclical setup which is the case when an external RTC is employed. Therefore, this WDT processor may serve as the third level WDT. When more than one of these 3 WDT layers are in place, the reliability of the solution can be drastically increased.

Conclusions: In short, when more than one WDT device is in place, the solution can be very reliable. Three forms of WDT mechanisms are possible: internal, external RTC-based, and external MCU-based. Typically, the energy cost of the WDT solution also follows this order, while the reliability level follow the reverse order. A strategic approach is to use the internal or the external RTC-based for conventional WDT actions, such as to restart the processor when it is

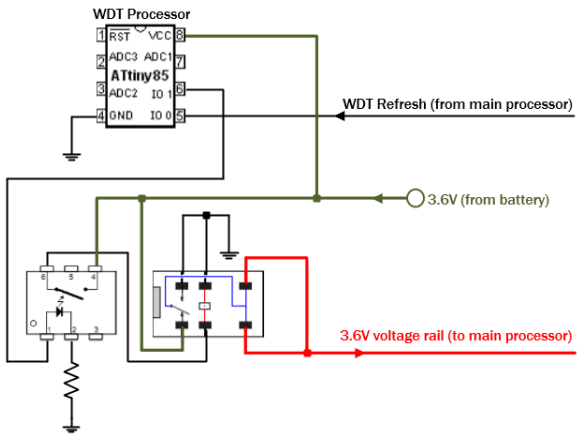

Fig. 3. External WDT scheme used to increase the reliability of WSN nodes.

trapped by a frozen or a loop caused by programming issues. For these cases, the WDT can potentially solve the problem after a short time (e.g., 1 min.). On the other hand, the external MCU-based WDT can be dedicated for long-term transactions, such the case when a node cannot communicate with other node after a certain amount of time (e.g., 30 min.). The same external WDT can still be used to enforce rebooting commands that came from higher levels (main processor, coordinator node, Data Server, etc.).

5) How effective is the reset action?
Unfortunately, a reset action on the main processor may not effectively solve a problem with a device in an erratic state. Moreover, due to the increasing availability of commercial sub-modules that have their own MCUs (e.g., RTC chips, radio and data-storage modules, digital sensor probes, etc.), potential *system* reliability issues can increase. For instance, we observed cases where radio modules from diverse manufacturers stopped communicating with the main processor of our Ripple-2 node. The issue was not solved with a reset and, even much more critical, it was not solved with a power-off/on action at the main processor: the issue was solved when the radio module was powered-off/on.

A similar issue was also observed for the main processors. Unfortunately, even a power-off/on effect may not be effective all times. This is the case because modern microcontrollers support low voltage levels and under very low-current drain circumstances, it is possible that the existing capacitance at the microcontroller voltage rail maintains the processor active and trapped in a freezing/loop state even after the power-off/on action. To make the scenario even more complex, when active devices are connected to the I/O lines, the processor also may not be properly restarted because it can still be *powered* via these I/O lines. In order to address the mentioned issues, we designed a very effective solution that properly operated in all scenarios we have investigated. The circuitry is shown in Fig. 3 and it comprises of a mechanical relay activated by an opto-coupler. The mechanical relay has the role of switching the voltage rail of the main processor. The novelty in this solution is the fact that during activation (WDT reset action), the voltage rail (load side only) is short-circuited to ground and it is maintained in this configuration for a certain amount of time. Therefore, any residual voltage level at the lines of the processor eventually vanishes. If peripheral devices are attached to the processor, such lines must be switched-off or, alternatively, all peripherals must also be powered-off. Finally, observe in Fig. 3 that even if the WDT device fails, the system continues to operate normally.

Conclusions: Any form of WDT action involving solely the *reset* line of a processor/device is highly prone to reliability issues. In particular for unattended remote devices, it is recommended that a reset action to be properly *translated* in hardware by a power-off/on action that effectively affects all components of the system, not only the main processor. We highlight that the circuitry in Fig. 3 was only truly effective when the power-off time was set to 3 seconds, which is a particular value for our Ripple-2 devices and must be configured according to each case.

The main focus of this section is the proper implementation of WDT techniques. Nonetheless, the eventual WDT action is the reboot of the equipment, a form of drastic intervention. A way to reduce the need of such interventions is the formal modeling of the sates of the system. Moreover, by analyzing the evolution of the machine states, the main processor can identify scenarios where the system is in an erratic/non-expected/unknown state. Once this erratic state is identified, the main processor can issue a command to the external WDT to restarting all the system. We call such system a *WDT-aware* one and its fully realization requires the formalization of the states of the system, as discussed in the next section.

IV. Virtual Finite State Machines + WDT

The WDT solution can be an effective solution but data loss my be involved in a WDT action. In this section, guidelines will be provided in order to design the system with a proper correctness level where the WDT restarting actions are rarely necessary. One way to achieve a balanced solution regarding software design robustness and quick development is the formal modeling of the system states. Unfortunately, even simple cases of real-world machines quickly become very complex due to very high number of state machines (SMs), an issue known as *explosion of states*. If the purpose of these SMs is to facilitate the analysis of the system by a human being, then the mentioned issue makes the formalization of states almost useless. This fact potentially explains the lack of SMs analysis at the current software engineering techniques applied to embedded systems.

A potential way to conciliate the adoption of SMs and the potential issue of explosion of states is the approach called virtual FSMs (VFSMs) introduced in [15], where a set of possible distinct state machines (SMs) are virtually grouped together. The intuition behind the concept of a virtual set of SMs grouped at state A is the existence of intermediary states A_1, A_2, .., and A_n that are strongly related with each other and cannot be broken into smaller parts out of the context of the state A. Note that such concept is relatively similar to the concept of a database (DB) *transaction*. The fundamental difference is that typically DB transactions are associated with well controlled scenarios and exceptions are relatively straightforward to be managed. On the other hand,

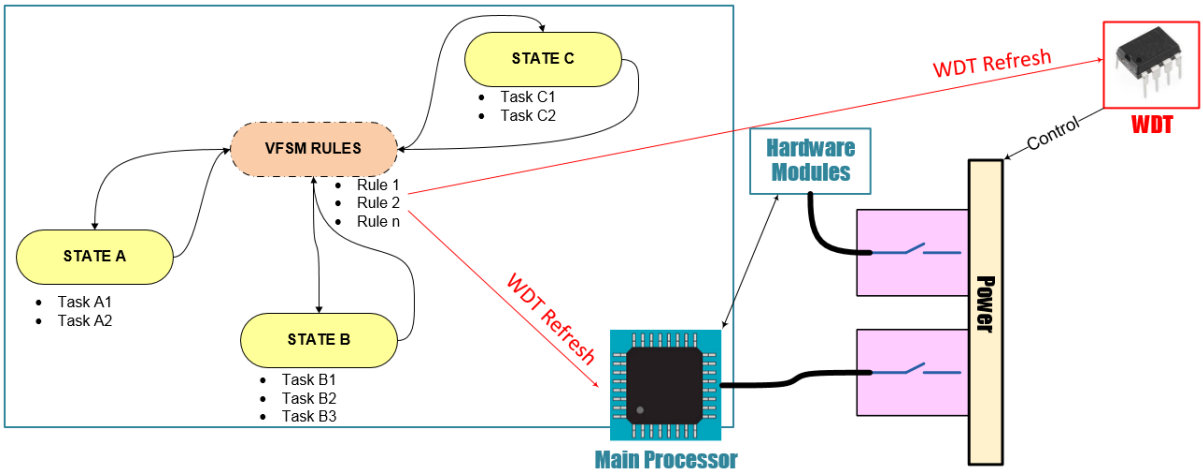

Fig. 4. The virtual finite state machine (VFSM) model represents the system behavior with its exceptions treatment associated with WDT actions.

embedded systems, such as the ones involving WSN nodes, are normally exposed to many non-controlled scenarios (e.g., energy depletion issues, communication failures, etc.).

The following simple example illustrate how we have used VFSMs at the Ripple-2 design. Assume that a WSN node is currently at the virtual state A comprised of SMs:

- A_1: node received message PROT1
- A_2: node identified in the message PROT1 what is the requested measurement type to be performed

Depending of the A_2 outcome, the node goes to distinct SMs (e.g., B, C, etc.). What makes A_1 and A_2 a single virtual SM is the fact that the node cannot change its SM while A_2 is not completed. But what would happen if a message PROT2 is received while the node makes its transition between A_1 and A_2? Similarly, what would happen if an attached intelligent probe interrupts the WSN node by sending its measurements related to a critical event? To make the scenario even more complex, assume that the two mentioned exceptions also occurred at the same time when an additional external RTC chip interrupts the main processor regarding a regular scheduled task that must be performed, such as saving all the logs at the memory into a SD card? Observe that if the system designer does not have answers for these questions, the system may be not correctly programmed and the node can potentially go to a non-function status while operating without any hardware failure. This is a kind of scenario that the WDT techniques discussed at the previous section can help to mitigate, but it also means that the system is poorly designed and effectively not very reliable since WDT actions can be associated with loss of data and events.

For the mentioned example, if the processor moves from A_1 to any state other than A_2, the VFSM model is clearly violated because a grouped set of SMs cannot be never broken. If however, it is very important to handle each of the mentioned priority cases just after the execution of A_1, then a new VFSM model must be provided where A_2 is actually another regular machine state, not a sub-state. However, if we allow too many exceptions, the complexity of the VFSM model quickly increases and its usefulness decreases. Another important aspect involving SMs in VSFM is the strict adoption of timeouts: the machine cannot be in a certain state/sub-state for more time than expected. As expected, these timing values are strongly dependent on the user-defined constraints and hardware/network capabilities. In many cases, they are based on empirical observations whenever maximum accuracy is necessary.

Our balanced solution for a similar scenario came in the form of an exception treatment procedure which is always called after the complete execution of each virtual SM transition (A, B, ...), such as *VFSM RULES* in Fig. 4. The last command at the state A is the change of the node status to the new SM (e.g., from A to C). This new state is then formally validated according to the state of the software variables as verified by the exception treatment *VFSM RULES*. Some of these variables may be modified during the partial treatment of an external event while the node was executing in state A (in this case, such variables are not critical for the running SM A). The role of the exception treatment is to confirm the state C or to move the machine to a new state X according to predefined rules. When a specific scenario is not addressed by a specific rule, the node can follow two approaches: a) return to the initial START VM where all software variables are reseted (equivalent to a hardware reset) or b) ignore the exception. For instance, this latter approach can be used for the mentioned case of the reception of message PROT2 while the machine transitions between A_1 and A_2. When the VFSM model, such as shown in Fig. 4 is evaluated and all expected virtual SM and transitions between them are properly defined, the solution is potentially robust. Again, any non-expected scenario (no existing rule) can be properly handled by the default treatment (i.e., software or hardware reset).

V. Results

As a result of the careful design of the Ripple-2 architecture for the SoilSCAPE project, we have achieved exceptional levels for both energy-efficiency and robustness [8]. No infrastructure node, such as routers or repeaters, is necessary besides the data collector (Local Coordinator, LC) for very sparse deployments. The regular nodes and even the LC node can hibernate with power consumption smaller than 20μW. This is particularly impressive considering the fact that typical infrastructure nodes cannot even sleep. Moreover,

the effective network overhead is typically smaller than 1% and the energy-efficiency of the nodes is multiple folds higher than the state-of-the-art in WSNs. Even more important, the energy depletion of the network is highly homogeneous. For instance, in a recent battery replacement trip to one of the sites that was operating for more than 2 years, we observed that the majority of the nodes stopped working at the same 3-month window. As expected, the TCO of the solution is drastically reduced when the maintenance visit can be properly planned and all nodes are involved. Therefore, we have achieved the functional goals of the SoilSCAPE project.

Regarding the reliability of the solution, the problems we faced were uniquely related to external agents, such as animals destroying the soil probe cables, water damage at the soil moisture probes, and expected temporary SMS/3G connection issues due to the high distance between the sites and cellular towers. The failure rate due to software and electronic issues is steadily zero during the years. We observed that from all LC nodes, 5 never used the WDT technique and 2 used WDT once during the long period of 1-3 years. These results indicate that the overall system is very robust, nonetheless for some unknown causes, maybe related to hardware issues, the nodes that required the WDT action could properly operate after the problem without human intervention.

ACKNOWLEDGMENTS

This work was performed at the University of Southern California and at the University of Michigan under support from the National Aeronautics and Space Administration, Earth Science Technology Office, Advanced Information Systems Technology Program. Agnelo R. Silva is also supported by the Brazilian National Research Council (CNPq) scholarship under the Brazilian Programme *Science without borders*.

REFERENCES

[1] I. F. Akyildiz and et.al., "Wireless sensor networks: a survey," *Computer Networks Journal (Elsevier)*, vol. 38, pp. 393–422, March 2002.

[2] IEEE 802.15.4 Standard, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE, NJ, USA, 2006.

[3] ZigBee Alliance, "ZigBee Specifications," ZigBee Standard Organization, San Ramon, CA, USA, 2008.

[4] A. Silva and et. al., "Power-management techniques for wireless sensor networks and similar low-power communication devices based on nonrechargeable batteries," *J. Comp. Netw. and Comm.*, pp. 1–10, 2012.

[5] A. Silva, M. Moghaddam, and M. Liu, *Design of Low Data-Rate Environmental Monitoring Applications - The Art of Wireless Sensor Networks - H. M. Ammari (ed.)*, ch. 3, pp. 51–94. Springer-Verlag Berlin Heidelberg, 2014.

[6] M. Moghaddam and et. al., "A wireless soil moisture smart sensor web using physics-based optimal control: Concept and initial demonstrations," *IEEE J. Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 3, pp. 522–535, December 2010.

[7] A. Silva and et. al., "An Adaptive Energy-Management Framework for Sensor Nodes with Constrained Energy Scavenging Profiles," *Intl J. of Distributed Sensor Networks*, no. 272849, pp. pp. 1–33, 2013.

[8] A. Silva, M. Liu, and M. Moghaddam, "Ripple-2: a non-collaborative, asynchronous, and open architecture for highly-scalable and low duty-cycle WSNs," *ACM Mobile Computing and Communications Review*, vol. 17, pp. 55–60, January 2013.

[9] "XBee DigiMesh 2.4 - XBee-PRO - Wireless connectivity using the DigiMesh protocol." http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-digimesh-2-4.

[10] F. Stajano and R. Anderson, "The grenade timer: Fortifying the watchdog timer against malicious mobile code," in *7th Intl Workshop on Mobile Multimedia Communications (MoMuC' 00)*, (Tokyo, Japan), 2000.

[11] P. Dutta and et al., "Trio enabling sustainable and scalable outdoor wireless sensor network deployments," in *In Proc. IEEE IPSN' 06*, pp. 407–415, 2006.

[12] M. Demirbas and et. al., "INSIGHT: Internet-sensor integration for habitat monitoring," in *Intl. Symp. on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, pp. 558–563, 2006.

[13] S. Brown and C. Sreenan, *Software Update Recovery for Wireless Sensor Networks - Sensor Applications, Experimentation, and Logistics*, vol. 29. Springer Berlin Heidelberg, 2010.

[14] A. Silva, M. Liu, and M. Moghaddam, "WSN-SA: Design Foundations for Situational Awareness Systems Based on Sensor Networks," in *IEEE Global Humanitarian Tech. Conf. (GHTC' 13)*, (San Jose, CA), 2013.

[15] F. Wagner, "VFSM executable specification," in *Proceedings of the IEEE International Conference on Computer System and Software Engineering*, (The Hague, The Netherlands), pp. 226–231, 1992.