Resilient Sensor Networks Utilizing Dynamic Resource Reuse

A novel paradigm towards fault-tolerant wireless sensing

Sharief M. A. Oteafy and Hossam S. Hassanein
School of Computing
Queen's University
Kingston, Canada
{ oteafy | hossam }@cs.queensu.ca

***Abstract*— An important mandate for the operation of Wireless Sensor Networks (WSNs) is ensuring resilient operation. While current solutions, such as over-deployment, ruggedized packaging and re-mobilization of sensing nodes, offer viable remedies, the question of resilience lies beyond sheer functionality with fault tolerance. That is, a trade-off has commonly grown to be a myth, we could either have a resilient network, or an efficient one. In this paper we present a model that adopts a resilient paradigm of resource reuse (RR), named RR-WSNs, whereby the individual resources of sensing systems would be re-utilized and assigned to different applications as failures occur. We elaborate on our definition of functional coverage in WSNs, and how application requirements are met by assigning functional tasks to viable resources in the region of interest. We present an optimal mapping via a binary integer programming formulation. We present our results from Matlab implementations, demonstrating functional longevity.**

Keywords*— *Resilient operation; Sensor Networks; Novel paradigm; Resource Reuse; Functional Coverage;

I. INTRODUCTION

The spectrum of applications of Wireless Sensor Networks (WSNs) is ever increasing. Once a limited technology with dedicated utility, now WSNs are expanding to horizons unforeseen a decade ago. At their core premise, WSNs delivered sensing at various locations that are tethered via wireless transceivers. Thus, presenting a major leap from the established wired sensing paradigms. Today, we envision their utility in sensor and actuator networks, and their integration with prominent directions such as cloud-centric sensing and the Internet of Things (IoT).

An important design dimension of WSNs is ensuring autonomous operation and network longevity. Many definitions were presented to address both properties, either in singularity or in aggregated metrics. For the former, the notion of minimizing human interference to mitigate network faults/failures is considered the common definition. For the latter, definitions varied significantly as to when we deem network lifetime to be over. Obsolete definitions depending on first failure or last failure are seldom used in current literature. This is mainly due to the lack of context in which such first/last failures occurred, and how they in fact affect network operation (i.e. coverage, connectivity, etc). While other definitions that attempted to integrate failure-context in the metric, such as when the first partition in the network occurs, or when the first coverage hole is created, are more relevant. A more empowering definition – although not recent – was presented by Dessler and Dietrich in [1]. They defined network lifetime as a measure of remaining functional nodes in a given region, whereby faults and varying nodal density would dictate new functional zones.

Our prime concern is ensuring resilient operation for WSNs. Resilience in a broad definition reflects the ability of the network to handle faults/failures beyond a small number of nodes. That is, maintain network operability under multiple failures. Different models are adopted to ensure resilient operation, however network longevity often suffers. A typical example is implementing energy-draining protocols that cross-validate sensor readings to ensure conformity and correctness of data sent as sensor reports [2].

In this paper we present a novel paradigm in WSNs, which capitalizes on granular resource reuse in sensing systems. At its core, the presented paradigm (RR-WSN) enables a flexible view of available sensing/communication resources in the WSN and wireless devices in its vicinity to actively potentiate more applications and higher resilience levels. Our contribution lies in presenting a new model for maintaining and prolonging functional longevity, and ensuring functional zones are reconfigured based on application demand. We also present an elaborate classification of fault-tolerance procedures, highlighting the stages of fault detection and resolution.

The remainder of this paper is organized as follows. In Section II we present an elaborate discussion of the types of failures faced in WSNs, and the stages covered to detect, classify and resolve such failures. We present a brief overview of the RR-WSN model, in light of earlier work by Oteafy and Hassanein [3][4]. Section III follows with an elaborate discussion on the resilient RR-WSN paradigm, and the granular objective of maximizing functional energy impact (FEI), a metric presented in this paper. We present an optimization problem for finding the functional sets of resources to meet a given set of application requirements, and formulate it as a binary integer programming problem. Our results and performance evaluation are presented in Section IV, and conclusions are drawn in Section V.

978-1-4799-2903-0/14/$31.00 ©2014 Crown

II. Background and Motivation

A. Failure types and conditions

Failures in sensing systems, and wireless networks in general, have been tackled in the literature extensively [5]. When merged together, a new class of (inherently complex) problems has been aggregated to form the field of resilience in WSNs. The problem has been tackled in two main directions. The first direction observes the required output, compared to an expected one (from history based logging or a benchmark) to decide if the results imply a faulty system. The other direction observes the actual hardware (via sensors) or software governing the protocols (via selective debugging) to monitor its operation for faulty behavior [3]. Both result in detection of failures; and demonstrate varying accuracy based on the adopted approach and its applicability to the specific components. Having this information forms the basis of self-healing WSNs; a design consideration prominent in harsh environment.

From the perception of WSNs, fault resilience has been at the core of design considerations [6]. The aforementioned detection paradigms are applied to detecting both communication and operation faults [7]. It is important to note that they seldom occur in isolation, and more often than not a cascading effect in errors/faults takes place. For example, failure of a node which formed a pivot in the topology, would result in network partitioning; hence an operational error causes a communication problem. On another front, errors in reports transmitted or control messages, could result in significant overhead of coordination and re-aggregation of information in a certain region in the network, thereby depleting the respective nodes' power and hindering their performance.

B. Phase-based fault-tolerance

The process of identifying faults, and all the steps taken to solve them and maintaining network operation, is non-trivial. The scope and accuracy by which this is approached depends highly on the available network resources, application and the required accuracy. The following 5 steps outline both the processes that need to be carried out to establish fault resilience, and the order in which they need to be done. Jeopardizing any of these stages would often impact both the efficiency by which faults are resolved, and the accuracy of the system that would result after fault-resolution. These five steps are detailed as follows:

1) Detection

A reliable mechanism for fault detection needs to be operational during network lifetime, identifying the erroneous behavior and the threshold after which a fault is declared. Such mechanisms vary in whether this threshold is static or dynamic, set by the node or externally (remotely by user or another entity in the network at a higher level in the hierarchy, such as the cluster head of sink) [2]. Moreover, this threshold needs to be sensitive to neighboring nodes and their readings. As such, a value detected by only one node which significantly surpasses the threshold, often gains more significance if it occurs in neighboring nodes, since many physical factors impact a region not only a node.

2) Classification

The type of error needs to be identified. The first indication is understanding its pattern of occurrence. Is it a fault that happened in isolation, with no previous similar records? If not, how often does the fault occur? Here two important factors come in play, is the error intermittent with a certain frequency? Is the frequency variable or static? If not, then what are the triggers for such intermittent faults? Are they environment

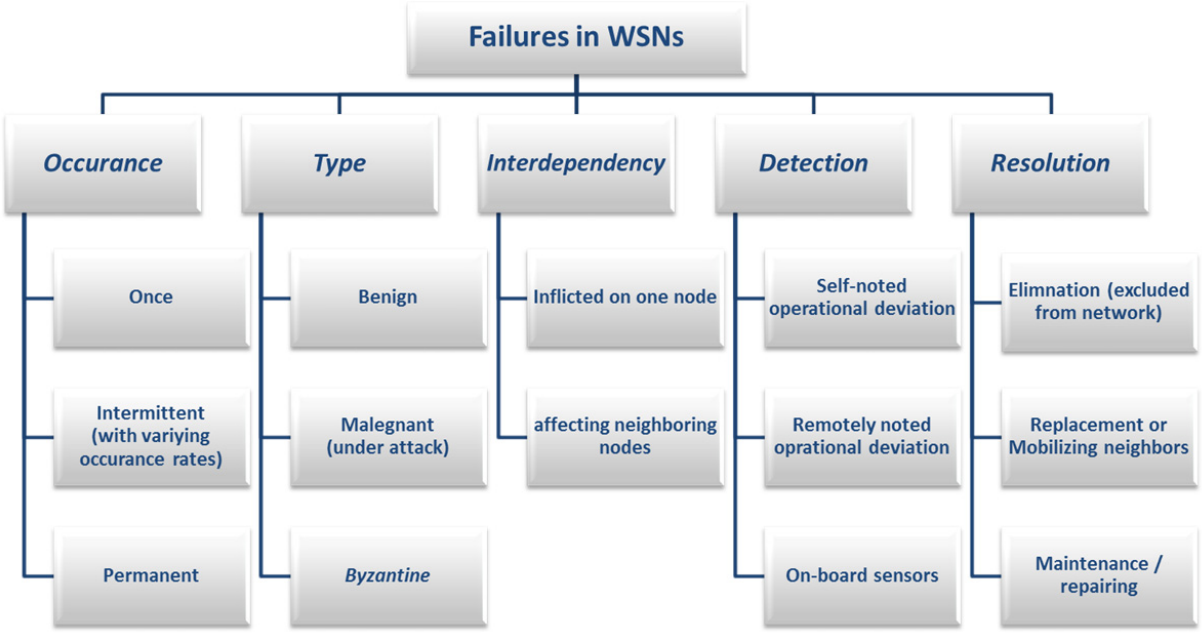

Fig. 1. Failures in WSNs and pertaining factors.

based, hardware induced or involve tampering? Finally, whether intermittent or occurs in singularity, is the error based on *byzantine behavior* [8], or is it a benign fault? In byzantine faults, nodes behave incorrectly, deceptively or inconsistently. Byzantine fault tolerance encompasses a significant research track that was triggered in 1980.

The easiest error to classify is the permanent case. That is, if a node failed, then investigations into frequency, triggers and other factors could prove superfluous, accordingly dealing with it just entails maintenance (step 5 here).

3) Location and Zoning

Another important aspect of fault resilience is zoning; i.e. identifying the affected region in the network. Accordingly, the fault detection mechanism should be able to identify the position of the error(s) with adequate accuracy. Adequacy, although qualitative here, reflects the importance of accuracy yet without jeopardizing network operation and exhausting its resources to achieve it.

If the topology/protocols are geographically based, then this would translate to an actual coordinate in the region. On the other hand, if the network is not based as such, and follows a "hop-based" approach, then the location would be tagged "relative" to the sink. If it is a cluster based topology, then the affected cluster(s) should be identified, and so on.

Moreover, this should also encompass the number and regions of the affected nodes. If it's a single node, then zoning is as trivial as locating it. On the contrary, if multiple nodes take part in the identified error, then the problem of location and identification becomes much harder to track and report. Assessing the fault and locating it to only a subset of the affected/causing regions, would result in sparse resolution in the next two steps. This would result in flawed network operation regardless of the efficiency of isolation and maintenance.

4) Isolation

Resolving a fault, once identified and classified, entails carrying out physical and/or software changes to the affected nodes. That is, for example, removing malicious nodes from a deployed WSN or replacing a faulty sensor board on a group of nodes. This could often result in network dissection; when the affected nodes act as pivots on the topological view of the network (as a graph). For the duration of maintenance, the step to be elaborated upon below, all nodes undergoing change should be isolated from network activity until fixed/replaced. Simply put, their flawed contribution to readings, communication and control (co-ordination) would hinder network performance, and their operation should be capped until cured.

5) Maintenance

Different types of failures dictate varying methods of maintenance. Before rushing into network maintenance, it is crucial to investigate the previous 4 steps to be able to efficiently and optimally address the failure(s). That is, without accurate detection of failures, followed by correct classification, locating them and isolating their effect from the network, trying to fix such failures would merely resolve to "trained" trial and error; at best.

Simply detailing what would be done to resolve every type of failure is both superfluous and merely incomprehensive. This is evident from the multitude of protocols and architectures for WSNs developed for harsh environments, and magnified by the diversity of applications for which they are tweaked. In efforts of abstraction, it would suffice to elaborate upon the methodologies by which network maintenance is carried out.

Maintenance schemes typically fall under three categories. The most common is eliminating the affected/failed nodes from the network, simply by letting it deplete its power, or removing it from network topology; i.e. disconnect it. Another direction is attempting to replace the damaged nodes, which requires either manual placement in field (persons, robots, etc), or remote dispersion (by plane, robots, etc) if that was the original deployment scheme. A more demanding approach is maintaining or repairing them, which requires remote intervention, if possible, for software faults via Dynamic reprogramming [9], or actual visits to the site of deployment for trained personnel to repair the faults components. The latter is usually adopted for higher-end large nodes deployed in accessible regions.

C. Resource Reuse paradigm

Sensor nodes are typically equipped with a multitude of resources. They mostly include, communication (transceivers), memory (mostly flash), and processing capabilities (Micro controller units – MCUs); in addition to sensing platforms that are mostly application dependent. The notion of common resources between co-existing WSNs is at the core of our RR-WSN paradigm. We capitalize on expanding the utility of resources in sensing nodes (**R**) beyond the network in which it was deployed. An underutilized resource in a network could be much needed in a neighboring one, aiding it in a functional zone in which a failure/hole has occurred.

We thus adopt RR-WSN, where resources are aggregated and represented in a pool (resource pool **ReP**) according to their attributes. We categorize and utilize sensor resources under six categories: their 1) functional capabilities, e.g. take pictures, videos, etc, 2) levels of operation, e.g. resolution of picture, range of transceiver, etc, 3) power consumption at each level, 4) region of fidelity, e.g. coverage, 5) duty cycling scheme, and finally 6) location. An elaborate discussion of resource abstraction, and conformity measures, are detailed in [3].

Similarly, applications to run on the ReP are dissected into functional requirements (**F**) that match the aforementioned representation of resources attributes. Thus, a mapping of functional requirements **F** to available resources **R** is reduced to an optimal assignment problem for all applications over all available resources. This is also elaborated upon in [3] and [4].

III. RR-WSN PARADIGM FOR WSN RESILIENCE

The goal of RR-WSN is to find the optimal assignment of resources to functional requirements in a given sensing network. In this Section we present a formulation for the

assignment problem of functional requirements to resources over a multitude of WSNs, to run concurrent applications and mitigate failures by resource reuse to maintain functional zones per need. This formulation spans heterogeneous nodes with varying resources, residual energies and anchoring sinks (network owners).

We adopt rounds to cater for changing requirements or failures, both in nodes and network topology. That is, after deciding on the resources/functionalities, and obtaining an optimal assignment, the network operates under this assignment until a change occurs. We denote this duration as a round $\boldsymbol{\tau}$.

At each round $\boldsymbol{\tau_k} \in \mathbf{T}$ the sets **F** and **R** are obtained and populated. Hence, using the resource pool (**ReP**) the aggregation of applications dictates the mapping of $\boldsymbol{F \rightarrow R}$ denoted as $\boldsymbol{\Psi_k}$ for each round $\boldsymbol{t_k}$. Intrinsically, there could be many matchings for which **F** could be mapped on **R**, i.e., solutions for $\boldsymbol{\Psi_k}$. Thus, we denote $\boldsymbol{\Psi_k}^*$ as an optimal mapping of $\boldsymbol{F \rightarrow R}$ minimizing functional energy impact (**FEI**) defined as

Definition: Functional Energy Impact (**FEI**) is an indicator of the percent of energy consumed by node $\mathbf{n_i}$ to perform a functional request $\mathbf{f_{j,m}}$ of type $\boldsymbol{\theta_m} \in \boldsymbol{\Theta}$ relative to its energy reservoir dedicated for resource class $\boldsymbol{\theta_m}$.

That is, each node slices its energy reservoir into portions, to match the resource classes it holds. All instances of that resource that are included in the global network ReP would be capped by that portion of energy. For example, if a node has a temperature sensor that manifests into 3 usable instances, and 20% of its available reservoir is dedicated to temperature sensing, then the FEI of using one of its temperature sensors is computed relative to 20% of the node's energy. The FEI measure is taken to ensure that a node, with minimal energy consumption, will not be exploited to carry multiple functional requests being penalized by its relatively larger energy reservoir.

This argument strikes a more important definition that would affect any argument on energy efficiency, which is network lifetime. Many definitions exist in the literature on the point in time in which network lifetime ends, most notably considering the first node to die, first coverage hole created (by death of nodes) or the first partitioning in the network. The latter would occur when a pivotal node (i.e. in a graph representation) would fail, therefore rendering the network communicating in two (or more) isolated sub-networks.

In our RR-WSN model we assume network lifetime to be the point in time in when a given functional zone fails. Zone sizes vary by application. Lifetime definitions that aim for connectivity or coverage metrics assume pre-set functional operation by its nodes. However, in our design we assume that coverage is a functionality that could be re-assigned to neighboring nodes if available, and long-range connectivity could be re-established by probing the transmission resources of neighboring nodes. Thus, lifetime becomes a function of the resources that are available in given vicinities on which such tasks could be offloaded to. Hence, we ultimately depend on the abundance of resources which are provided by wireless nodes.

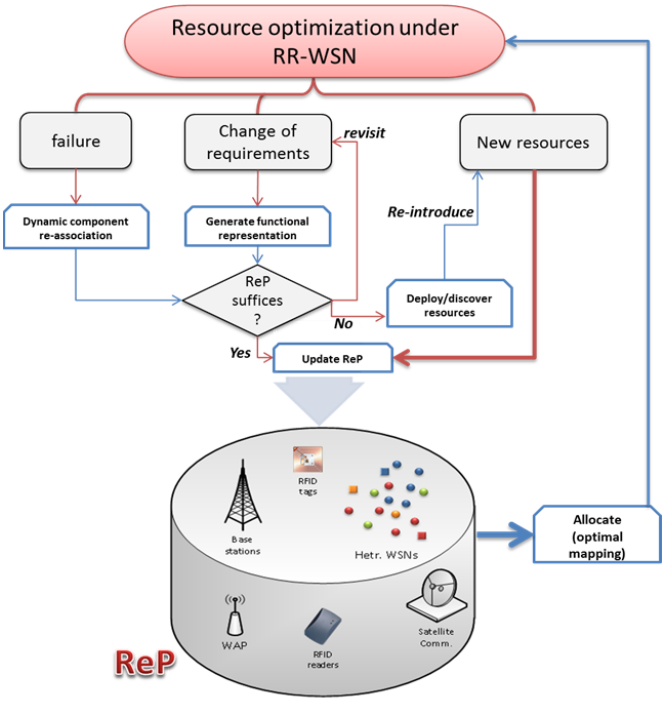

Fig. 2. overview of RR-WSN optimal mapping model

Thus, we hereby identify the problem of finding $\boldsymbol{\Psi_k}^*$ as an optimization problem with two sets of constraints, namely: (1) functional constraints, whereby each functional request $\boldsymbol{f_{j,m}}$ must be met by a given resource instance $\boldsymbol{r^k_{i,n}}$ if available (2) node constraints that ensures each resource utilized in a given node abides by the energy cap dedicated to that resource class. Since our model runs in rounds, this mapping tolerates changes as resources change, and nodes have the flexibility to increase or decrease the energy portions assigned to its different classes.

A. Assumptions

Optimal mapping of applications' functional requirements to available resources ensures that the network operates under pre-set fair constraints. Moreover applications are offloaded efficiently over multiple networks without resource starvation or exhaustion. We adopt a linear programming (LP) formulation to solve the optimal mapping problem, i.e., finding $\boldsymbol{\Psi_k}^*$. System overview is depicted in Fig. 2 to highlight the utility of resource instances and functional requirements and their mapping.

We assume that WSNs and their SNs, municipal, industrial and institutional wirelessly accessible static nodes form a pool of resources for the cross-platform utilization of our paradigm. No assumptions are made on access network types, as research on vertical handoffs and multi-homed devices already established leverage to multi-access schemes. Moreover, for static WSN deployments, the sink acts as the median of communication and thus renders access-network matching a trivial issue.

We also assume multiple applications, for varying domains, requesting functionalities from this pool of resources. As such, a single resource could be probed for its functionalities by different applications. We assume that RR-WSN resources are already deployed and reachable. Active nodes holding resources are assumed to have a measurable reservoir of energy usable by the attached resources with predetermined consumption indicators.

This does not however imply that both resource availability and energy reservoirs cannot change over time, as this is already catered for at the beginning of each round. A new round is triggered by changes in application requirements, failures in the network such that a functional requirement is no longer met, or new nodes introduced into the network(s). Fig. 2 depicts the factors that dictate the initiation of a new round of RR-WSN, and the functional reaction of the system.

B. Optimal mapping

Prior to finding a potential $\boldsymbol{\Psi_k}$ a set of preprocessing steps are required. First, we need to aggregate the resources available and the functional requests made by the set of applications. We note the total resources in the network as $\mathbf{R}$ defined by

$$\boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{\alpha}}^{\boldsymbol{\beta}} = \begin{cases} 1 & \text{if resource instance } \boldsymbol{\beta} \text{ of class } \boldsymbol{\alpha} \text{ node } \boldsymbol{n_i} \text{ used} \\ 0 & \text{otherwise} \end{cases}$$

where $|\boldsymbol{N}|$ is the total number of nodes in the underlying networks, $\boldsymbol{\Theta_i}$ are the resource classes encompassed by node $\boldsymbol{n_i} \in \mathbf{N}$, and $\boldsymbol{k}$ is the number of instances available from each $\boldsymbol{\Theta_i}$ denoting how many functional requests could be served by this resource class in $\boldsymbol{n_i}$.

Then, for each functional request, the set of potential resources that could fulfill it needs to be constructed, denoted as $\boldsymbol{\varphi_j}$. Thus, we aggregate a vector $\boldsymbol{\Phi}$ of size $|\boldsymbol{F}|$ over all functional requests. Formally,

C. BIP formulation

In WSNs there are typically many parameters which designers seek to optimize, of most important impact on network longevity is power consumption. However, the notion of least energy consumption is not necessarily the most optimal use of energy reservoirs over the network; it might not lead to the longest network lifetime. For example, nodes which have potentially higher end resources, could perform a given task with lower power consumption than neighboring nodes. In a greedy algorithm seeking to utilize the resource with least power consumption, these nodes could be exploited to their depletion, rendering their other resources depleted as well

To strike a balance, we define the following indicators to aid a fine tuned assessment of FEI. First, for each node $\boldsymbol{n_i} \in \mathbf{N}$, we denote the power consumption for each $\boldsymbol{r_{i,\alpha}} \in \boldsymbol{\Theta_i}$ as $\boldsymbol{\theta_{i,\alpha}}$ mW, where $\boldsymbol{\alpha} \in \{1, |\boldsymbol{\Theta_i}|\}$. This is already an attribute of each resource that is pre-defined (according to its manufacturer specifications). Each node $\boldsymbol{n_i}$ allocates a percentage of its residual energy to each resource class, to dictate a dynamic assignment of its local battery to its available functionalities. This portion per class is denoted as $\boldsymbol{\delta_{i,\alpha}}$ for each $\boldsymbol{r_{i,\alpha}}$. The total residual energy in node $\boldsymbol{n_i}$ is denoted as $\boldsymbol{\varepsilon_i}$ which dissipates as its resources are assigned to functional tasks.

In RR-WSN, the goal is to determine if a given resource instance $\boldsymbol{r_{i,\alpha}^{\beta}}$ will be serving a given functional requirement $\boldsymbol{f_{j,m}}$. Thus, we define the binary utilization variable for each resource instance $\boldsymbol{r_{i,\alpha}^{\beta}}$ as:

$$\mathbf{R} = \bigcup_{i=1}^{|N|} \bigcup_{\alpha \in \Theta_i} \bigcup_{\beta=1}^{k} \boldsymbol{r_{i,\alpha}^{\beta}} \tag{1}$$

To calibrate the functional energy impact (FEI) of allocating a given resource $\boldsymbol{r_{i,\alpha}^{\beta}}$ to a $\boldsymbol{f_{j,m}}$ we denote it as $\boldsymbol{\gamma_{i,\alpha}^{\beta}}$ defined by:

$$\boldsymbol{\gamma_{i,\alpha}^{\beta}} = \frac{\boldsymbol{\theta_{i,\alpha}}}{\boldsymbol{\delta_{i,\alpha}} * \boldsymbol{\varepsilon_i}} \tag{2}$$

Thus, we define our Binary Integer Programming (BIP) problem as minimizing the functional energy impact (FEI) of carrying all the functional requests in $\boldsymbol{F}$ over their Fidelity sets $\boldsymbol{\Phi}$, while maintaining nodal energy and functional boundaries.

$$\textbf{min } \boldsymbol{FEI} = \sum_{i=1}^{|N|} \sum_{\alpha \in \Theta_i} \sum_{\beta=1}^{k} \boldsymbol{\gamma_{i,\alpha}^{\beta}} * \boldsymbol{\mu_{i,\alpha}^{\beta}} \tag{3}$$

subject to the following constraints

$$\forall\, \boldsymbol{\varphi_j} \in \boldsymbol{\Phi} \quad \sum_{m=1}^{|\varphi_j|} \boldsymbol{\mu_m} = \mathbf{1} \tag{4}$$

$$\forall\, \boldsymbol{n_i} \in \mathbf{N} \quad \sum_{\alpha \in \Theta_i} \sum_{\beta=1}^{k} \boldsymbol{\mu_{i,\alpha}^{\beta}} \leq \boldsymbol{\phi_i} \tag{5}$$

$$\forall\, \boldsymbol{n_i} \in \mathbf{N} \left(\sum_{\alpha \in \Theta_i} \left(\sum_{\beta=1}^{k} \boldsymbol{\mu_{i,\alpha}^{\beta}} * \boldsymbol{\theta_{i,\alpha}} \right) \right) * \boldsymbol{\tau} \leq \boldsymbol{\varepsilon_i} \tag{6}$$

Thus, our aim is to minimize the total FEI over all resources used, ensuring in constraint (4) that each functional requirement is met by one and only one resource. Also, we denote the nodal capacity of node $\boldsymbol{n_i} \in \mathbf{N}$ by $\boldsymbol{\phi_i}$, enforcing that any given node cannot exceed a predetermined limit of resources used, shared across all its resource classes $\boldsymbol{\theta_i}$. This is enforced in constraint (5). Finally, in constraint (6) we ensure that each node only offer resources within its energy capacity $\boldsymbol{\varepsilon_i}$ over all of its used resources.

IV. RESULTS

We ran our simulations on MATLAB. The experiment setup is adaptive, allowing for independent runs under different network distributions and random locations for both resources and functional requests.

We note that at each run the set of constraints for the BILP formulation is constructed according to the fidelity regions of resources in the current setting. We utilized the bintprog solver developed by Mathworks® in solving the BILP formulation.

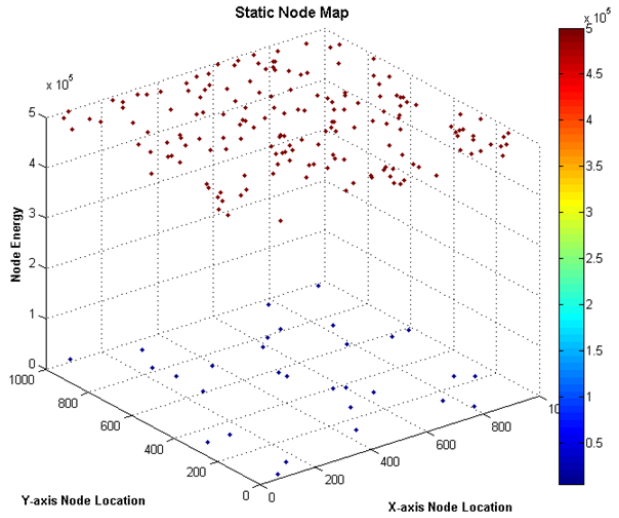

Fig. 3. A typical distribution of nodes over a 1000 x 1000 region, differentiated by their residual energy

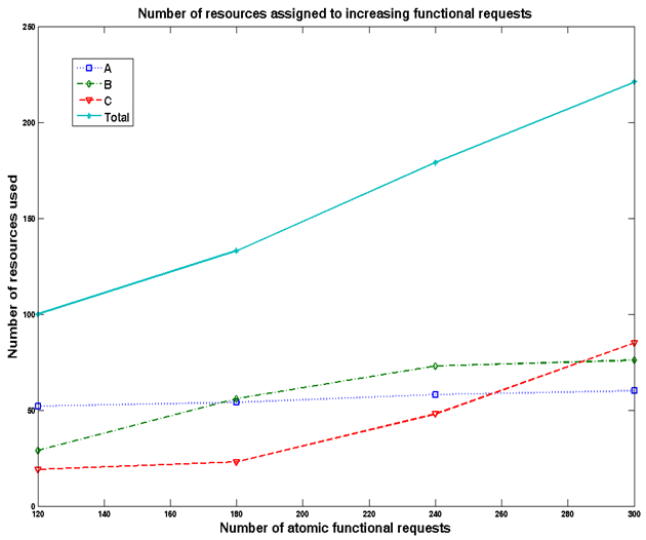

Fig. 4. Resources assigned to functional requests over three networks running RR-WSN as failures occur.

The experiment is setup in a region of size 500 x 500 meters, and nodes having an initial set of resources chosen from {'Temperature sensor' ; 'Micro controller' ; 'Memory'; 'Transceiver' ; 'Camera' ; 'Radar' }. The simulations are run with controlled and varying variables, which are highlighted in each scenario. Our simulations are run for 10 times to achieve representative means. For a sample run in a region of 1000 x 1000 meters, a sample distribution of the nodes with varying battery reservoirs is demonstrated in Fig. 3.

It is important to note the impact of utilizing non-network resources for meeting a set of application requirements with failures and changing requirements. We carry out experiments to demonstrate the tendency of WSNs, under RR-WSN, to utilize non-local resources to satisfy functional requests should failures occur. Thus, we setup a scenario with three networks, A, B and C. Each network encompasses a set of unique and heterogeneous resources, even though functionalities could match. We experiment with a varying number of application requirements, and measure the yielding assignment of resources to these functional requests. We initialize the networks with 50 nodes each, dispersed randomly over a 100 x 100 m region. The trend is shown in Fig. 4.

In the initial phase of operation, when the less-energy demanding resources of network A could suffice the functional requests, there is higher dependency on its resources. However, as the number of functional requirements increase due to failures, beyond an energy-efficient loading on network A, the more energy-demanding resources of networks B and C get assigned functional requests to restore functioning zones.

I. Conclusions

Managing failures in WSNs is quite challenging. The harsh environments in which sensors are deployed, and the stringent operational mandates, often deem resilience a challenging design dimension. In this work we presented the novel approach of Resource Reuse to the failure mitigation problem.

Having WSN resilience achieved by proactively enlisting the utility of wirelessly abundant resources, we leverage operation in regions where faults occur, and maintain functional coverage by re-assigning tasks to available resources. Thus, we present a model that alleviates the famous tradeoff of resilience to energy conservation in WSNs.

Acknowledgment

This research is funded by a grant from the Ontario Ministry of Economic Development and Innovation under the Ontario Research Fund-Research Excellence (ORF-RE) program.

References

[1] F. Dressler and I. Dietrich, "Lifetime Analysis in Heterogeneous Sensor Networks," 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools, pp. 606-616, 2006

[2] Brade, T., Zug, S., & Kaiser, J. "Validity-Based Failure Algebra for Distributed Sensor Systems", In *IEEE 32nd International Symposium on Reliable Distributed Systems (SRDS),*pp. 143-152, 2013.

[3] S. Oteafy and H. Hassanein, "Re-usable Resources in Wireless Sensor Networks: A Linear Optimization for a Novel Application Overlay Paradigm over Multiple Networks", IEEE Global communications (Globecom), pp.1-5, Dec. 2011.

[4] S. Oteafy and H. Hassanein, "Utilizing transient resources in dynamic wireless sensor networks", IEEE Wireless Comm. and Networking Conf.: Mobile and Wireless Networks,WCNC, pp.2124-2128, 2012.

[5] K. Marzullo, "Tolerating failures of continuous-valued sensors," *ACM Transactions on Computer Systems (TOCS),* vol. 8, pp. 284-304, 1990.

[6] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance.* Springer, 2006.

[7] J. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta. "In-network outlier detection in wireless sensor networks." *Knowledge and information systems,* Vol. 34, No. 1, pp.23-54, 2013

[8] A. Vempaty, L. Tong, and P. Varshney, "Distributed inference with byzantine data: State-of-the-art review on data falsification attacks." *IEEE Signal Processing Magazine, Vol.*30, No, pp.65-75, 2013.

[9] N. Shafi, K. Ali, and H. Hassanein, "No-reboot and zero-flash over-the-air programming for Wireless Sensor Networks" *IEEE SECON,* pp. 371-379, 2012.