Toward Highly-Available WSNs for Assisted Living

Safwan Al-Omari and Weisong Shi

Department of Computer Science
Wayne State University
{somari, weisong}@wayne.edu

ABSTRACT

In response to the consistent increase of elder people living in their apartments, and the need for innovative non-obtrusive tools to connect elders to their caregivers, we started an initiative with the Institute of Gerontology at Wayne State University to explore the application of wireless sensor networks (WSNs) for the monitoring of elder people and the communication of potential emergency conditions to their remote caregivers. Motivated by the fact that sensor nodes are resource-constrained and error-prone on one hand, and mission urgency on the other hand, we argue that high availability is a vital requirement that viable WSNs for assisted-living have to acquire. We propose the use of classical reliability theory techniques to tackle this issue in a systematic way. We develop analytical models of the WSN availability in terms of the availability of the underlying sensor nodes. These models help in planning for the required number of nodes and the way these nodes are scheduled `ON` and `OFF`. Our preliminary results show that using node scheduling almost doubles the expected WSN total uptime.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences—*Health*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Reliability and life testing*

General Terms

Reliability, Algorithms

Keywords

Wireless Sensor Networks, Node Scheduling, Modeling

1. INTRODUCTION

It is expected that adults age 65 years and older will account for more than 18% of the U.S population by the year 2025 [7]. In 1991, DHHS created the Healthy People 2000 project, which is the first national effort targeted at reducing disability and promoting physical health in older adults. Despite this initiative, the numbers of elders with one or more physical disabilities is increasing. In collaboration with the Institute of Gerontology at Wayne State University, we propose to apply the technology of WSNs as a non-obtrusive tool to better monitor the activities of elders living in their apartments, providing an innovative approach to connect seniors to their caregivers that facilitates the communication of any emergency conditions. A prevalent emergency condition to which elders are susceptible is falling. According to the Centers for Disease Control and Prevention, falls are the leading cause of injury deaths and the most common cause of nonfatal injuries and hospital admissions for trauma. More than one third of adults age 65 and older fall each year, and more than 60% of the people who die from falls are 75 years of age and older. Thus, detecting falls and responding quickly is critical. If falls are detected and responded to immediately, more lives will be saved.

WSNs can be deeply deployed in the physical world in a non-obtrusive fashion and provide remote and continuous monitoring of the environment for potential target events. A typical WSN consists of a set of sensor nodes. These sensor nodes are similar to low-power minicomputers with central processors and random-access memory, and thus the capability of computation and storage. Upon deployment, the sensor nodes self-organize into a connected network and continuously monitor the environment. Once an emergency event is detected by a set of sensor nodes, an alert is relayed through a special sensor node (i.e., gateway) over the Internet to a remote nursing station, who's personal responds appropriately. We coin this type of network as SAILNet, which stands for **S**ensor **A**ssisted **I**ndependent **L**iving **N**etworks.

Despite their evident potential, successful WSNs deployments in such mission-critical applications is hindered by the resource constraints of the underlying sensor nodes including power, computation, and communication quality. These limitations render the sensor nodes highly unreliable and susceptible to frequent failures. Given the unreliability and the urgency of the WSN mission, *we recognize high WSN availability as a major concern that needs to be addressed systematically along our journey toward implementing viable WSNs for assisted-living.* By systematic we mean the development of analytical tools that capture sensor node failure behavior and allow for modeling WSN availability in terms of the availability of the underlying sensor nodes. These models not only help in protocol design for SAILNet, but also allow us to answer more fundamental questions con-

HealthNet'07, June 11, 2007, San Juan, Puerto Rico, USA.

cerning deployment parameters such as predicting the number of nodes needed to empower the system (i.e., WSN) to withstand failures and maintain availability. This promotes autonomy of SAILNet and reduces human intervention.

Developing such models requires a clear understanding of sensor node failure models. Sensor node failures may be the result of a software or a hardware failure. The former can be handled through restarting and/or re-programming faulty nodes over the air. On the contrary, hardware failures have no easy remedy. In this work, we target the fail-stop hardware failures. From the perspective of hardware failures, we can identify two application categories: in-door [6] and out-door [11]. The latter tends to have a hostile deployment environment, which results in more frequent and unexpected node failures due to harsh environmental conditions [13]. Whereas, the former tends to provide more controlled deployment environment, which results in fewer and more expected sensor node failures. Therefore, adopting the usage-based failure model, in which the probability of failure depends on the time a node spends in the `ON` mode, is more reasonable. Despite the controlled deployment environment in in-door WSNs, sensor nodes are still failure-prone due to their low cost design. Therefore, WSN deployments are envisioned to involve high degree of node redundancy [4, 13] to overcome these limitations.

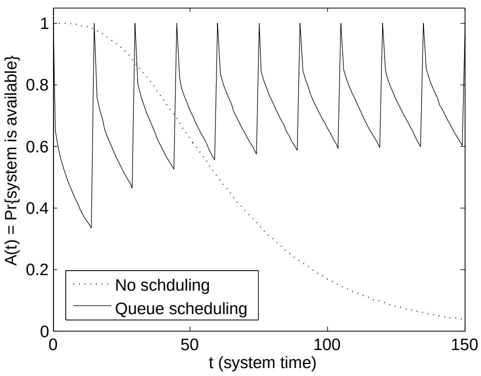

Figure 1: Effect of node scheduling on the availability of WSN.

Note that turning a node `OFF` in the usage-based failure model prevents node failures. Hence, node scheduling does indeed affect WSN availability behavior. In Fig. 1, we show an interesting result on how node scheduling can affect WSN availability. The x-axis represents time, and the y-axis represents system availability. In the *queue scheduling* scheme, the m sensor nodes are divided in groups of κ' nodes each, these groups are turned `ON` sequentially in a queue like schedule one group at a time. As shown in the figure, *queue scheduling* exhibits better average availability than that of *no scheduling*, in which all the nodes are turned `ON` since the beginning.

Node redundancy and scheduling, which has been referred to as topology management, has been studied extensively in the literature [1, 4, 5, 6, 12]. In previous efforts, node redundancy and scheduling is used to work around the sensor node power constraint and to extend the WSN lifetime beyond the lifetime of a single sensor node. Nodes are redundantly deployed so that multiple sensors are able to perform the same function. Some nodes are turned `OFF` to save energy, while others stay `ON` and perform the function. Two functions were considered, connectivity and coverage. Our work complements their work by emphasizing availability as a new requirement on the node redundancy and scheduling algorithms. As topology (either connectivity or coverage) does not pose a big challenge in SAILNet (Section 2), we prefer to use redundancy management instead of topology management in this paper.

Inspired by the natural analogy between WSNs for SAILNet and classical systems, we use techniques from classical reliability theory to model the WSN as a κ-out-of-m system. In this setting, the system (i.e., WSN) is said to be available (i.e., functioning) as long as there are at least κ out of the m deployed sensor nodes are available (i.e., non faulty). Based on our preliminary analytical model results, we find that the WSN exhibits almost double the expected total uptime under *queue scheduling* scheme in comparison to using the *no scheduling* scheme.

Our contribution in this paper is three-fold. First, we model the WSN availability using sound techniques from the reliability theory [9]. Second, we use the model to solve the redundancy management problems: *deciding the number of redundant nodes needed to meet desired availability behavior, and the way these nodes should be scheduled to improve availability.* Third, we show that scheduling nodes in the usage-based failure model does indeed improve availability.

The rest of the paper is organized as follows. Section 2 provides a deployment strategy of the sensor nodes in SAILNet; Section 3 presents formal definitions and the assumed sensor node failure model; In Section 4, we move on to present our node scheduling schemes and their availability modeling, and finally, formalize and solve the redundancy management problems; Section 5 presents our preliminary evaluation results; Finally, we present our current status and future work in Section 6.

2. STRUCTURING WSN FOR SAILNET

2.1 A two-tier sensor node deployment

In in-door WSN deployment, we believe that a two-tier structuring of the sensor nodes such as the lately proposed Tenet architecture makes a perfect match [8]. Each room in the target apartment is equipped with several sensor nodes, which cluster together and form a single-hop network with a predetermined and more powerful cluster head (i.e., first tier). In the second tier, the cluster heads organize into another single-hop network with a gateway connected to the Internet. To avoid interference and increase network capacity, cluster heads may use different wireless channel to communicate with the gateway. Unlike the prevalent large-scale environmental deployments, coverage and connectivity do not pose a big challenge. However, since the target events are life-threatening, the WSN need to be highly available and resilient to node failures.

Therefore, each room is redundantly equipped with m sensor nodes out of which κ sensor nodes are needed available in order to have a functioning cluster. κ sensor nodes are needed instead of only one node to rule out sensor reading errors due to faulty sensors and noisy environment [10]. Deciding the value of κ is out the scope of this work and is considered as input to our model. However, deciding the value of m to rule out node failures is our topic in this paper. Due to low cost of sensor nodes, using node redundancy to overcome resource limitations is an inevitable solution in the WSN research community [4, 13]. Therefore, redundantly deploying m sensor nodes when only κ are needed to work around sensor node unreliability is inevitable.

2.2 SAILNet as a classical system

In classical reliability theory, a system is viewed as a set of independent components that are connected serially, in parallel, or as compromise (i.e., κ-out-of-m). In a serial system, the system is functioning if and only if all of its components are functioning. In a parallel system, the system is functioning as long as at least one component is functioning. In a κ-out-of-m system, at least κ components are required functioning for the system to be considered functioning. We can see that a κ-out-of-m system can be easily mapped into sensor node cluster, and the set of clusters into a serial system. The natural analogy between WSNs and classical systems is the fundamental motivation behind our approach of using classical reliability theory techniques in our work. In addition to this analogy, we have three more important reasons for using classical reliability theory. First, WSN are deeply embedded in the physical world, which makes them susceptible to the same environmental conditions and suffer similar tear and wear effects to those components of classical systems. Second, Using component redundancy is a well-accepted and well-developed approach in boosting overall system reliability in classical systems. Likewise, sensor node redundancy is a well-accepted and promising solution to overcome resource limitations including sensor node unreliability. Third, the failure of one sensor node does not cause the failure of other nodes in WSNs (i.e., independent sensor node failures). Nonetheless, all sensor nodes are expected to follow a similar failure model as they suffer similar environmental conditions and once a node fails, it never becomes available again (i.e., fail-stop).

3. PROBLEM STATEMENT AND METRICS

The purpose of our work is to model the availability of the sensor cluster (denoted as $A(t)$) in terms of the availability of the underlying components (denoted as $S_i(t)$ for sensor i) under two scheduling schemes: *no scheduling* and *queue scheduling*, formalize the redundancy management problem in each scheduling scheme, and finally to compare their performance in terms of average availability and the expected total uptime. We use techniques from reliability theory [9] to develop these analytical models and to formalize and solve the redundancy management problems. Basically, we model the availability of the system as a κ-out-of-m system. We assume that all the sensor nodes have similar initial power and perform similar workload, which enable them to function for an identical maximum period of time T_{max}. In the rest of the paper, we use the terms system and cluster interchangeably.

Average availability (denoted as $\mathrm{avg}_{A(t)}$) is defined as $\frac{\sum_{t=0}^{T_{max}} A(t)}{T_{max}+1}$. The total uptime time and expected total uptime are denoted as U and $\mathrm{E}[U]$ respectively and defined as the total time and expected total time, in which the system is available. In Subsection 4.1 and Subsection 4.2, we present formal definitions of U and $\mathrm{E}[U]$ and show that $\mathrm{E}[U] = \mathrm{avg}_{A_{no}(t)} \cdot (\mathrm{T}_{max} + 1)$ for both the *no scheduling* and *queue scheduling* schemes.

The redundancy management problem has slightly different settings in each scheduling scheme. In the *no scheduling* scheme, all the nodes are made `ON` since the beginning and so, we are only left with finding the required number of nodes (i.e., m) to meet some availability requirements (i.e., $\mathrm{avg}_{A(t)}$ or $\mathrm{E}[U]$). Whereas in the *queue scheduling* scheme, the m nodes are divided into groups of κ' nodes each, and made `ON` in a queue-like manner, therefore, we have two problems to solve. First, find m and corresponding κ' to meet some availability requirements. Second, given m, find κ' that optimizes availability in terms of either $\mathrm{avg}_{A(t)}$, or $\mathrm{E}[U]$. Note that *no scheduling* scheme is indeed a special case of the *queue scheduling* scheme (i.e., make $\kappa' = m$), however, we prefer to model the *no scheduling* scheme separately for two reasons. First, modeling the *no scheduling* scheme is easy as a classical κ-out-of-m system. Second, this scheme needs *no scheduling* management at all, which makes its implementation different than that of *queue scheduling*.

The lifetime of usage-based components is typically divided into three periods, each with a different failure rate. First, an early period with decreasing failure rate, these failures are due to design and manufacturing faults. Second, a stable period with a very low and stable failure rate. Third, a wear-out period at the end of the component lifetime with increasing failure rate, these failures are due to normal wear and tear. A widely accepted approach to model this behavior is to use a bathtub-shaped *failure rate* function, denoted as $(\lambda_i(t))$. $\lambda_i(t)$ represents the conditional probability intensity that node i will fail in the next moment, given that it has survived until time t (i.e., $\lambda_i(t) = Pr\{X_i \in [t+\mathrm{d}t] | X_i > t\} = \frac{-S_i(t)\prime}{S_i(t)}$), where $X \in [\,0, T_{max}\,]$ represents node i lifetime and $S_i(t)$ is known as the survival function and represents the unconditional probability that node i has no failures by time t and so the node is available at time t. It is known that $S_i(t) = exp\{-\int_0^t \lambda(\tau)\mathrm{d}\tau\}$ [9]. In this paper, we use a failure rate function proposed lately in [3]. $\lambda_i(t)$ and the corresponding $S_i(t)$ are defined as follows:

$$\lambda_i(t) = a\, b(a\, t)^{b-1} \,+\, (\frac{a}{b})(a\, t)^{\frac{1}{b}-1} \,+\, h_\circ \tag{1}$$

$$S_i(t) = \exp\{-(a\, t)^b \,-\, (a\, t)^{\frac{1}{b}} \,-\, h_\circ t\} \tag{2}$$

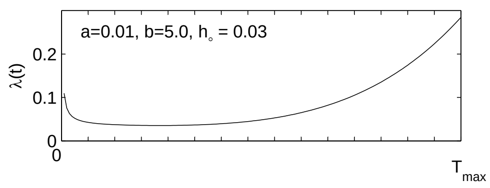

Figure 2: Sensor node failure-rate function.

Fig. 2 depicts $\lambda(t)$ with assumed values for a, b, and $h_\circ$. Based on our assumption that once a node dies it never becomes available again (i.e., fail-stop), we may think of $S_i(t)$ as the availability of node i at time t, which equals to $Pr\{node\ i\ is\ available\ at\ time\ t\}$. Also, note that since all the nodes follow the same failure model, we can simply use $\lambda(t)$ and $S(t)$ in the rest of the paper.

4. AVAILABILITY MODELING

In the following two subsections, we present availability modeling of two scheduling schemes, *no scheduling* and *queue scheduling*, including formalizing and solving the redundancy management problems in the context of these scheduling schemes.

4.1 The no scheduling scheme

We simply model the availability of the cluster as a classical κ-out-of-m system. We say that the sensor cluster is available at time t with probability $A_{no}(t)$ if and only if there exits at least κ nodes available at time t, put formally as follows:

$$A_{no}(t) = \sum_{i=\kappa}^{m} \binom{m}{i} S(t)^i \cdot (1 - S(t))^{(m-i)} \tag{3}$$

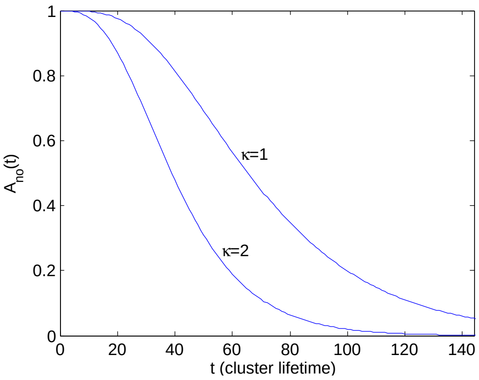

Figure 3: No scheduling availability.

Fig. 3 shows an $A_{no}(t)$ with $m = 12$ and $\kappa = 1$ and 2, and a $T_{max} = 144$. To find $\mathrm{E}[U]$, note that the system as a whole exhibits a fail-stop behavior following its fail-stop components (i.e., sensor nodes). In other words, once there are less than κ sensor nodes available, the system fails and never becomes available again. Therefore, U is equivalent to a random variable representing the time until first failure (denoted as τ) with $Pr\{\tau > t\} = A_{no}(t)$. Note that the random variable $\tau \geq 0$, and so the expected time until first failure and hence $\mathrm{E}[U]$ can be calculated as follows:

$$\mathrm{E}[U] = \sum_{t=0}^{T_{max}} A_{no}(t) \tag{4}$$

As there is *no scheduling* in this scheme (i.e., all nodes are `ON` all the time, we are left with one question that hope the model to answer:

Problem 1: Given a value of κ, find the lowest m needed to meet either $\mathrm{avg}_{A_{no}(t)}$ or $\mathrm{E}[U]$.

Problem 1 is a simple optimization problem that can be solved iteratively over m starting with $m = \kappa$, incrementing m by one each time, and checking whether the current value of m meets the requirement (i.e., $\mathrm{avg}_{A(t)}$ or $\mathrm{E}[U]$). To find $\mathrm{avg}_{A(t)}$, we simply follow the definition to calculate it for a given m value. From the definition of $\mathrm{avg}_{A(t)}$ in Section 3, we get $(\mathrm{T}_{max} + 1) \cdot \mathrm{avg}_{A_{no}(t)} = \sum_{t=0}^{\mathrm{T}_{max}} A_{no}(t)$, by substituting in Equation (4), we get:

$$\mathrm{E}[U] = \mathrm{avg}_{A_{no}(t)} \cdot (\mathrm{T}_{max} + 1) \tag{5}$$

Thus, the value of m needed to meet $\mathrm{avg}_{A(t)}$ requirement, is the same value of m that is needed to meet $\mathrm{E}[U]$. In other words, the solution of Problem 1 for $\mathrm{avg}_{A(t)}$ and $\mathrm{E}[U]$ is the same.

4.2 Queue scheduling scheme

Queue scheduling divides the m nodes into $\eta = \frac{m}{\kappa'}$ groups (denoted as g_i, where $i = 1, \ldots, \eta$). Each group, g_i, consists of κ' nodes, where $(\kappa \leq \kappa' \leq m)$. Given these η groups, the time is divided into η epoches with equal periods denoted as $\Delta = \frac{T_{max}}{\eta}$. Each group g_i is turned `ON`, in a queue-like scheduling, at the beginning of its corresponding epoch (denoted as ϵ_i).

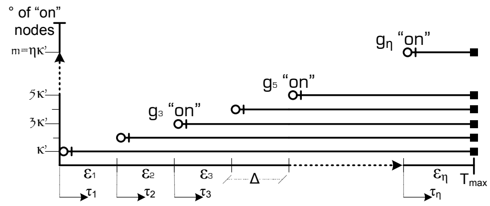

Figure 4: Time line of queue scheduling.

Fig. 4 depicts the *queue scheduling* time line. The x-axis represents time, while the y-axis represents the total number of `ON` nodes shown as discrete values multiple of κ'. Unlike in the *no scheduling* scheme, the total uptime (i.e., U) in *queue scheduling* is different than the time until the first failure since the system may fail and become available again when a new group is turned `ON`. However, in a single epoch, the system exhibits a fail stop behavior. Therefore, we can define the total uptime of the system as the summation of random variables representing the time until first system failure in each time epoch (shown as τ_i in Fig. 4). Formally, $U = \sum_{i=1}^{\eta} \tau_i$, where, $0 \leq \tau_i \leq \Delta$. Hence:

$$\mathrm{E}[U] = \sum_{i=1}^{\eta} E[\tau_i] \tag{6}$$

Now, we turn our attention to find the system availability ($A_Q(t)$), which also represents the probability distributions of the random variable τ_i. Perhaps the first thing that comes to mind when trying to model the availability of *queue scheduling* is the renewal process model. Unfortunately, the fact that renewals (i.e., bringing node groups (g_i) `ON`) are asynchronous to failures causes two major incompliances with the classical renewal process model assumptions. First, *lack of instantaneous repairs.* In other words, should the system fail during epoch i (i.e., ϵ_i), it will not be available until the beginning of the next time epoch (i.e., ϵ_{i+1}). Recall that in our usage-based failure model assumption in Section 3, nodes have to be in `OFF` mode to avoid failures, which makes them un-responsive to external events and therefore can not be asked to become active on the spot in case a failure is detected. On the other hand, in unattended and remotely administered WSNs, human intervention is infeasible and violates the key non-obtrusive application requirement. Second, *non-homogeneity of the availability probability distribution during different time epoches.* In other words, τ_i in Fig. 4 are not identically distributed.

In light of the above *queue scheduling* algorithm complications, we use a recursive numerical function to model the system availability at an arbitrary time instance (i.e., $A_Q(t)$). Let $Q(t, e, i, p)$ be the probability that there are exactly i

```
1. function result=Q(t, e, i, p)
2. t' = t - (e - 1) * Δ
3. if (e == 1){ // base case
4.    if (i > k'){
5.       return 0 // no enough nodes to choose from
6.    }
7.    return (k' i) * S(t')^i * (1 - S(t'))^(k'-i) * p
8. }else{
9.      minJ = max(0, i - (e - 1) * k')
10.     if (min > k'){
11.        return 0 // no enough nodes to choose from
12.     }
13.     maxJ = min(i, k')
14.     tmpP = 0
15.     for (j = minJ; j <= maxJ; j++){
16.        current_p = p * (k' j) * S(t')^j * (1 - S(t'))^(k'-j)
17.        tmpP = tmpP + Q(t, e - 1, i - j, current_p)
18.     }
19.     return tmpP
20. }}
```

Figure 5: Q: finds recursively the probability of having exactly i nodes available.

nodes available at time t, then:

$$A_Q(t) = \sum_{i=\kappa}^{\kappa' \cdot e} Q(t, e, i, 1.0) \text{ , where} \tag{7}$$

$$e = \lfloor \frac{t}{\Delta} \rfloor + 1$$

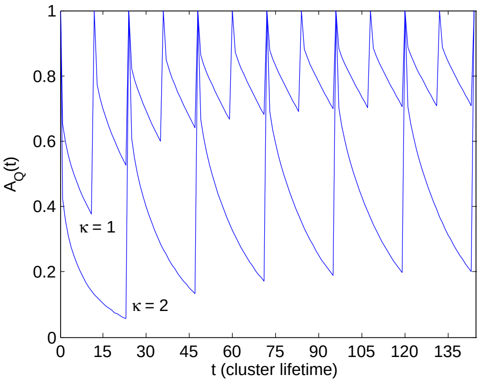

Figure 6: Queue scheduling availability.

The function **Q** finds the probability of having exactly i nodes available out of $e \cdot \kappa'$ nodes that are already turned `ON` by the time t. Q considers all the possible i node combinations by looping recursively over e, which represents the current group. Fig. 5 lists the pseudocode of function Q. In line 2 of Fig. 5, Q finds the total `ON` time (t') of the current group (i.e., e) as a shift of the global time t. Hence, the availability of the current node group becomes $S(t')$. Lines 3 through 6 in Fig. 5 represent the base case scenario (i.e., $e = 1$), in which only one group of nodes exists. Therefore, Q returns the probability of having i nodes available out of κ' `ON` nodes. Lines 15 through 17, loops recursively over all the possible combinations. Fig. 6 shows $A_Q(t)$ for $m = 12$, $\kappa = 1$, and $\kappa = 2$, and $T_{max} = 144$. κ' is set equal to κ for simplicity.

Now, we shift our gear to formalize and solve the redundancy management problems for the *queue scheduling* scheme. As in *no scheduling* scheme, first redundancy management problem is concerned with finding the needed number of nodes (i.e., m) to meet the desired availability requirement put formally as follows:

Problem 2: Given κ, find the lowest m and corresponding κ' needed to meet either $\text{avg}_{A_Q(t)}$ or E[U].

Again, we may solve this simple optimization problem iteratively over m and κ' starting from $m = \kappa$, incrementing m by one, and checking against the requirement. For each value of m, κ' is changed from κ up to m. To find E[U], we may re-write $avg_{A_Q(t)}$ as a piece-wise summation with Δ time intervals as follows:

$$\text{avg}_{A_Q(t)} = \frac{\sum_{t=0}^{\Delta-1} A_Q(t) + \cdots + \sum_{t=(\eta-1)\cdot\Delta}^{\eta\cdot\Delta-1} A_Q(t)}{T_{max} + 1} \tag{8}$$

Note that the i^{th} summation term in Equation (8) equals to $E[\tau_i]$, hence, from Equation (6), $E[U]$ equals to $avg_{A_Q(t)} \cdot T_{max} + 1$. Therefore, optimizing for $avg_{A_Q(t)}$ is the same as optimizing for E[U].

We can re-formulate Problem 2 to find the optimal κ', given a budget of m nodes from which at least κ nodes should be available. Putting the problem this way is useful in two scenarios. First, if there is a limited budget on the allowed number of nodes m in the planning phase. Second, if κ' needs to be adapted during the operational phase by considering the actual new m nodes available in the WSN after some node failures. The optimization problem is put formally as follows:

Problem 3: Given m and κ, find κ' that maximizes $\text{avg}_{A_Q(t)}$.

Problem 3 can be solved iteratively over κ' starting from $\kappa' = \kappa$ and incrementing κ' one by one until $\kappa' = m$, and finding the optimal κ'. Fig. 7 shows solutions of Problem 3 for $\text{avg}_{A_Q(t)}$. We can observe that κ' does not follow a particular pattern and could take any value between κ and m. For example, For $m = 8$ and $\kappa = 2$, Optimal κ' is 4, whereas, changing m to 9, decreases κ' to 3.

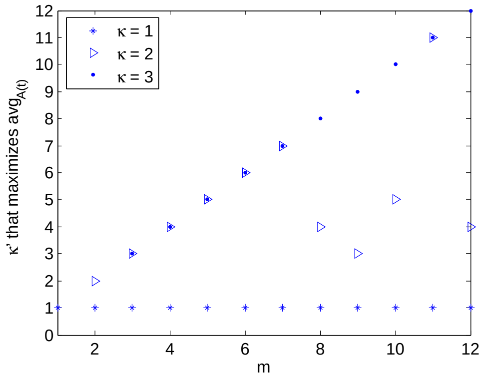

Figure 7: The solutions of problem 3 for $\text{avg}_{A(t)}$.

5. EVALUATION RESULTS

In this section, we use our models to show that node scheduling has the ability to improve the system availability. We compare the performance of *no scheduling* and *queue* *scheduling* schemes in terms of $avg_{A(t)}$ and $\mathrm{E}[U]$ as the number of nodes, m, increases. We use values of 12 and 1 for m, and κ respectively. In *queue scheduling*, κ' is chosen to optimize for $\mathrm{avg}_{A_Q(t)}$. More details of evaluation can be found in our technical report version [2].

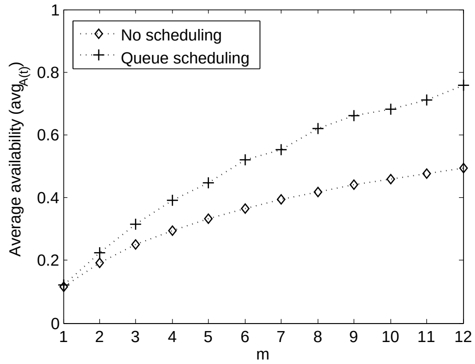

Figure 8: Comparing $avg_{A(t)}$ under no scheduling and queue scheduling.

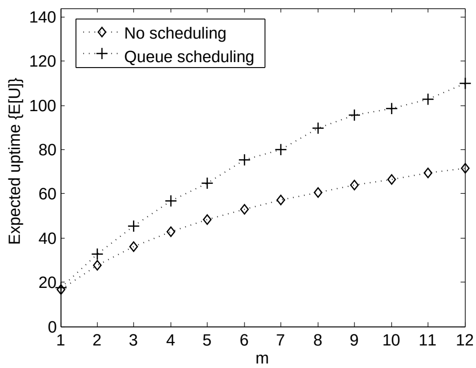

Figure 9: Comparing E$[U]$ under no scheduling and queue scheduling.

In Fig. 8, we can see that *queue scheduling* exhibits consistent and larger increase in the average availability compared to *no scheduling*. For example, for $m = 12$ and $\kappa = 1$, *queue scheduling* almost doubles the average system availability compared to *no scheduling*. For $m = 12$ in Fig. 8, the average availability goes from 0.5 up to 0.75. Also, note that as m and κ get closer (i.e., less redundancy), the performance of *queue scheduling* and *no scheduling* becomes closer, which is simply because *queue scheduling* converges to *no scheduling*. In other words, if m and κ are the same, the only way to schedule the nodes is to make all of them `ON` since the beginning.

In Fig. 9, we compare *no scheduling* and *queue scheduling* in terms of the expected total uptime as m increases. Again, we observe that *queue scheduling* outperforms *no scheduling* by increasing the total time in which the system is available. For example, for $m = 12$, the system total uptime is almost 105 hours when *queue scheduling* is used compared to less than 60 hours when *no scheduling* is used. The improvement is almost double.

In summary, we conclude that *queue scheduling* outperforms *no scheduling* significantly.

6. CURRENT STATUS AND FUTURE WORK

High availability is a vital aspect of SAILNet, we demonstrate how reliability theory can be used systematically to address it. Our analytical models help in predicting the required number of nodes needed to meet availability requirement. Furthermore, based on our preliminary results, we show that in usage-based sensor node failure model, node scheduling can boost the WSN availability. We are actively collaborating with the colleagues at Institute of Gerontology and local senior apartments and developing a SAILNet prototype that monitors their living environment. More information about the project can be found at `http://sail.cs.wayne.edu`.

In our future work, we plan to perform more extensive simulations to validate our analytical modeling. We also, plan to devise more performance metrics to quantify the availability of the WSN and study the effects of node scheduling.

7. REFERENCES

[1] S. Al-Omari and W. Shi. Redundancy-aware topology control in wireless sensor networks. In *Proc. of CollaborateCom'06*, 2006.

[2] S. Al-Omari and W. Shi. Availability modeling and analysis of autonomous in-door wsns. Technical Report MIST-TR-2007-001, Wayne State University, January 2007.

[3] M. Bebbington, C. Lai, and R. Zitikis. Useful periods for lifetime distributions with bathtub shaped hazard rate functions. *IEEE Tran. Reliability*, 55(2):245–251, 2006.

[4] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. *IEEE Tran. on Mobile Computing*, 3(3):272–285, 2004.

[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: An energy-efficient coordination algorithm for topology maintenance in ad-hoc wireless networks. In *Proc. of MobiCom'01*, 2001.

[6] W. Conner et al. Experimental evaluation of synchronization and topology control for in-building sensor network applications. In *Proc. of WSNA'03*, 2003.

[7] E. Dishman. Inventing wellness systems for aging in place. *Computer*, 37(5):34–41, 2004.

[8] O. Gnawali et al. The tenet architecture for tiered sensor networks. In *Proc. of SenSys '06*, 2006.

[9] B. Gnedenko, Y. Belyayev, and A. Solovyev. *Mathematical Methods of Reliability Theory*. Acadamic Press, 1969.

[10] X. Luo, M. Dong, and Y. Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Trans. Computers*, 55(1):58–70, 2006.

[11] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. of SenSys'04*, 2004.

[12] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proc. of MobiCom'01*, 2001.

[13] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. of ICDCS '03*, 2003.