Statistical Trust Establishment in Wireless Sensor Networks

Matthew J. Probst and Sneha Kumar Kasera
School of Computing, University of Utah
Email: {mprobst, kasera}@cs.utah.edu

Abstract— **We present a new distributed approach that establishes reputation-based trust among sensor nodes in order to identify malfunctioning and malicious sensor nodes and minimize their impact on applications. Our method adapts well to the special characteristics of wireless sensor networks, the most important being their resource limitations. Our methodology computes statistical trust and a confidence interval around the trust based on direct and indirect experiences of sensor node behavior. By considering the trust confidence interval, we are able to study the tradeoff between the tightness of the trust confidence interval with the resources used in collecting experiences. Furthermore, our approach allows dynamic scaling of redundancy levels based on the trust relationship between the nodes of a wireless sensor network. Using extensive simulations we demonstrate the benefits of our approach over an approach that uses static redundancy levels in terms of reduced energy consumption and longer life of the network. We also find that high confidence trust can be computed on each node with a relatively small memory overhead and used to determine the level of redundancy operations among nodes in the system.**

I. Introduction

Due to the criticality of many WSN applications including monitoring and early warning systems, it is crucial that the information obtained from these networks be trustworthy. Decisions based on the sensor network data can have serious economic and social impact. Therefore, nodes in a sensor network must perform their functions reliably. However, due to their limited capabilities for economic viability, deployment in "unfriendly" physical environments, and risk of physical attacks, not all sensor nodes can be expected to behave reliably at all times [1]. It then becomes necessary to identify malfunctioning and malicious or compromised nodes, and isolate them. Detecting such misbehaving nodes from a location external to the network is difficult. This is because sensor nodes perform in-network data processing and aggregation. Wireless sensor networks can be secured most effectively against misbehaving nodes if the nodes closest to the source of the problem themselves can detect such misbehavior and react accordingly.

Currently, to deal with node misbehavior, critical sensor network deployments require sufficient redundancy to meet the needs of the particular application. However, complete redundancy typically requires a minimum of triple the amount of hardware (and energy expenditures) to ensure that a 2/3 Byzantine consensus can be achieved when a sensing or aggregation discrepancy is encountered. Such full redundancy has traditionally required a constant level of energy expenditure irrespective of network threat and misbehavior levels.

Wireless sensor networks must protect themselves from a variety of threats. In WSNs, typically, a large number of sensors are deployed in some area of interest. These sensor nodes are expected to work unattended even in naturally harsh physical conditions. They are also often deployed in accessible areas where they could be physically attacked. The harsh physical conditions, or physical attacks, could result in malfunctioning of the sensor devices. Sensor nodes could also be compromised by tampering, and replicated. Additionally, malicious sensor nodes could be dropped into the area of deployment. These malicious sensors could eavesdrop on sensor communications, pose as legitimate nodes, disrupt the functioning of the sensor networks by imposing themselves as "nodes-in-the-middle", and disrupting service in a variety of ways. We have loosely classified the different types of misbehaviors in a WSN below. This classification is not intended to be comprehensive. Our goal here is to identify the type of misbehavior our research has targeted.

- *Misbehavior 1:* Sensor nodes malfunction but are not malicious.
- *Misbehavior 2:* Malicious attacker nodes (possibly dropped in the sensor field) eavesdrop on communications between genuine nodes, impersonate genuine nodes, and generate denial-of-service traffic or signals. However, in this threat model, there are no compromised nodes.
- *Misbehavior 3:* Compromised nodes, although appearing to be genuine, malfunction maliciously. They are also likely to cause the second type of misbehavior.

Our research focuses on misbehavior 1. We have established a trust system in sensor networks where nodes could malfunction but are not malicious. However, our trust system can also be a useful component for any solution that addresses misbehavior 3. This is because a compromised node might be able to authenticate itself correctly and still malfunction maliciously. Our trust system detects malfunctioning, whether malicious or not. Our research however, does not address misbehavior 2, which requires suitable authentication and privacy mechanisms. In addressing misbehavior 1, we have focused on the following three basic functions of WSNs - accurate data collection, data routing, and data processing and aggregation. In this paper we present a new distributed approach that establishes reputation-based trust among sensor nodes that allows the system to dynamically adapt its redundancy based on the confidence that nodes have between each other to behave correctly (trust). We show that a significant amount of energy can be conserved and the sensor network life extended when redundancy is varied according to the changing levels of trust between nodes.

The remainder of this paper is structured as follows. Section II summarizes existing work on building trust. Section III describes our trust system, its various components, and our trust computation methodology. In Section IV, we evaluate our trust model and in Section V, we present conclusions and potential future areas of research.

II. Related Work

Trust has been studied in a variety of networks and applications. A large number of trust models have been proposed in social networking. In this section, we review only those existing works that are somewhat related to our research. Golbeck and Hendler [2] extend the concept of the semantic web to include reputation ratings. The algorithm they present is based on voting to derive either a complete trust or complete lack of trust in an entity. No partial trust is derived with their algorithm. They do not account for history of interactions and they assume perfect connectivity. There is no discussion of how to cache reputation ratings. Cahill *et al* have outlined the importance of considering both risk as well as trust when making decisions [3]. If risk is low, then the action threshold for trust can be low. The trust model component of our research is based on the hypothesis of Carbone *et al*, which introduces a model that takes uncertainty into account. Caching trust is discussed in [4] but only to the extent of caching ciphers. Cache eviction based on content is not discussed. Gray *et al* introduce the importance of calculating trust based on the "Small Worlds" approach [5]. They recommend a cache eviction algorithm also based on a "Small Worlds" approach. Reputation-based trust has also been proposed for peer-to-peer systems. Ganeriwal *et al* present a Bayesian based approach for building WSN trust [6]. Bayesian methods, though memory efficient, are not suitable for delay-tolerant networks where significantly stale information may arrive at the same time as fresh information. Chen and Yeager have constructed a decentralized trust system for the Sun JXTA platform [7]. They also take a Bayesian approach and use discrete trust ratings which cannot provide the same level of accuracy as continuous trust ratings. The *confidence* levels that they propose are not based on statistical confidence intervals. Although the above existing research addresses the problem of trust, none looks at building trust specific to *resource-limited* and *delay-tolerant* wireless sensor networks. Theodorakopoulos and Baras in [8] present an algorithm for forming trust in Ad Hoc networks based on seminirings. This approach however lacks the ability to easily decay the usefulness of previous experiences based on the risk sensitivity (aversion level) of each node independent of other nodes. Our method for trust calculation allows each node in a system to independently evaluate and weigh the experiences of other nodes without reliance on summarized recommendations of other nodes. Avinash *et al* in [9] present a reputation based system that precludes the ability for nodes to perform their own assessment of original experience evidence. Their system is also delay in-tolerant. Yu *et al* in [10] present an information theoretic framework for trust evaluation. Their framework along with the work of Kraniewski *et al* in TIBFIT [11] do not leverage statistical confidence intervals nor do they address energy consumption optimizations. There is also a growing amount of research on security in WSNs (e.g., [1], [12]–[16]. This research mainly addresses misbehavior 2, and to some extent misbehavior 3 as described in Section I. Unlike this existing research, our focus is on building trust in the presence of malfunctioning nodes while reducing energy consumption. Trust is not a replacement for security nor is security a replacement for trust. Trust and security rather complement each other. Within a system building trust may require the use of secure protocols and maintaining security may be aided by trust establishment and maintenance.

III. Trust System

Social networks serve as an example by which we created a trust model for wireless sensor networks. Social trust is built in two phases. Before we directly interact with an individual, we might postulate a preconceived level of trust in that individual. Preconceived trust is formed from evidence we are given from other individuals in our social network. We automatically discount the accuracy of the obtained information based on our trust in the individuals who are generating and passing the information. We tend to trust information received via our direct social peers more than information received from the second layer of our social peers. The second phase of building trust is to interact directly with the individual or observe the direct interaction of others with the individual and start to establish a history of trust with that individual. In the case of WSNs, these observations to the sensing, routing and aggregation behavior of other nodes may be made by overhearing the radio communications of these nodes.

A. Context-Specific Trust

Social trust in relationships may be built over days, months, years or even decades. Each individual might have a different valuation of trust built over time. *Earning trust* may take a different length of time depending on the circumstances. Again, the trust we form with other individuals is limited to specific contexts based on the interaction we have had with them. Typically we do not trust our preferred automobile mechanic with legal questions nor do we trust our preferred lawyer with questions regarding fuel injection systems.

Following the social network model, we have postulated that, given a network of context-specific and directional-trust relationships connecting two entities, any entity can place a confidence rating on any piece of data/fact/statement generated by another entity that falls within the given context. In the case of WSNs the data produced by other nodes can be sensor readings, routed data, or aggregated data. We have further presumed that a confidence interval about the trust rating may be established to allow the entities to make accurate decisions. In the case of WSNs, this confidence interval will assist nodes in making decisions for routing, sensing, and data aggregation. Data should not be routed through nodes that can not be trusted. Likewise data collected from a misbehaving node or routed or aggregated through should not be propagated through the network. Nodes may need to expend more power sensing to take over for a neighbor node whose sensors can no longer be trusted. Nodes should not include sensor readings in aggregation processing from nodes that cannot be trusted to provide generally accurate readings.

B. Collection of Experiences

We describe four types of experiences below. For each type of experience, we list the methodology we used to enable a node to turn the experience into a useful piece of information.

Sensor readings: Nodes follow the process of overhearing sensor readings of nearby nodes and then comparing them to their local sensor readings. If the remote sensor readings are correlated closely enough with the local sensor readings (they are within a threshold set by decaying the correlation of values based on the distance between the sensor ranges), then the remote sensor reading is considered to be valid. Overhearing the source node of sensor readings is not the only way to evaluate a sensor reading. Intermediary nodes that have been requested to route raw sensor information (and their respective neighbors that can overhear them) can also evaluate each sensor reading for accuracy albeit with more limited ability given the increased distance from the source nodes sensors. The greater the perceived degradation in sensor accuracy, the less the source sensor node is trusted to accurately sense in the near future.

Experience generation accuracy: This is the evaluation of a neighbor's accuracy in recording direct experiences. To evaluate a neighbor's ability to generate experiences, a node listens to experiences generated and communicated by that neighbor and compares these experiences to its own. The larger the discrepancies in perceptions of experiences, the less trust a node will have in its neighbor's ability to accurately generate experiences. Examples of collection misbehavior include improperly weighing or evaluating an experience.

Observed data propagation accuracy (routing): Neighboring nodes within a certain proximity to a node performing some routing action are able to overhear both the incoming packet and the outgoing packet. These neighboring nodes can compare the outgoing routing destination of each overheard packet to its information in its own routing tables. If the packet apparently advances toward its intended destination, then the routing behavior of the overheard node is considered correct. If the packet does not get routed, gets corrupted or modified, or gets routed along an incorrect path, then the experience is recorded as a misbehavior by the overhearing node.

Observed accuracy of data aggregation: We examined two types of aggregation behavior observance. The first is one where a node is close enough to a neighbor to overhear all aggregation communication (inputs and outputs). In this case a node will simply verify that the aggregation function behaved correctly. The second and more complex case is where the aggregation behavior of a node is to be evaluated for a node that is far enough away not to be able to overhear all its inputs. In this case we relied on nodes that can compare the result of multi-path propagation schemes for data aggregation. Examples of aggregation misbehavior include inaccurately aggregating data due either to processor error or to intentional bias.

C. Trust Computation Methodology

In this section we present our trust computation methodology using experience records as input and providing as an output a trust value and a confidence interval based on those experiences. We present this methodology in the context of one entity, E1, that wishes to form trust in another entity, E2. Although a typical motivation for trust formation between nodes is decision-making, we do not explore different motivations here because the methodology is indifferent to motivation. Before trust is formed, entity E1 observes the behavior of E2 and judges whether the behavior is correct. Each opportunity E1 has of observing and judging the behavior of E2 is recorded in an *experience record*. An experience record contains at a minimum the following pieces of information:

- Identification of the entity (node) being observed. In our example this is the identity of E2. This may be a unique node-id, unique location or some other type of entity identifier.
- Identification of the entity (node) making the observation. In our example this is the identity of E1. This may be a digital signature. The identity of the observer is necessary in the cases where experience records are shared between nodes.
- The context type of the experience/observation. If, for example, E1 judges E2's ability to sense temperature accurately, the context of the experience would be data sensing. In WSNs, data sensing is one important responsibility that nodes fulfill and thus is well served by neighbor observation. Two other important responsibilities are a) data routing, propagation and aggregation, and b) generation of an experience record. Experience record includes E2's ability to observe other nodes accurately, and generate experiences itself.
- A timestamp indicating how long ago the experience took place. This information is important given that experiences become stale over time (nodes may change behavior in the interim).
- The trust value. This is the actual rating of trustworthiness that the observer (E1) assigns the node being observed (E2) for this particular experience. We use x_i to represent the trust value of experience (sample) i
- A weight that the observer (E1) assigned to the experience record indicating the amount of observation that went into generating the experience record. A limited or brief experience would be weighted lower than a longer lasting or more intense experience. We use w_i^c

to represent the context specific weight that an observer assigns to the experience i.

E1 thus observes the behavior of E2 and records these experiences in a local *experience cache*. Over time, these experiences will become stale and E1 may find it necessary to evict an existing record in the trust cache to make room for a newer record. E1 uses this trust cache to store both experiences that it recorded itself as well as experience records it receives from other nodes in the network.

1) Initial Evaluation of Experience Records: When E1 wishes to form a trust confidence interval for E2, it first identifies the context of the desired trust confidence interval (ability to sense data, etc). It queries its experience cache for records that have E2 being evaluated in this context. The goal of E1 is to find the mean of these trust values and to identify a confidence interval about this mean. The typical method for finding a mean ($\overline{x}$) of the sample values is simply to add up all of the values and divide by the number of samples:

$$\overline{x} = \frac{\Sigma x_i}{n}$$

The typical method for finding a confidence interval about this mean is to first estimate the variance of the population σ^2:

$$\sigma^2 = \frac{\Sigma \left(x_i - \overline{x}\right)^2}{n-1}$$

This estimated variance is used to create a confidence interval about the mean [17]:

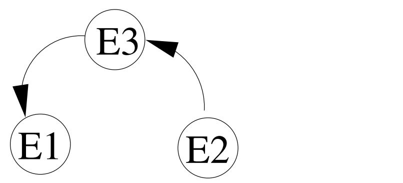
$$\overline{x} \pm t_{n-1,1-\alpha/2}\sqrt{\sigma^2/n} \qquad \text{(eq.1)}$$

where α is 0.10 for a 90% confidence interval, 0.05 for a 95% confidence interval, etc. The t in the above equation represents the $student - t$ distribution. If the confidence interval is sufficiently narrow (as determined by the context), E1 proceeds with its decision-making process. However, if the confidence interval is too wide then additional experiences are collected at the expense of additional resources.

The above method constitutes the basis of our trust computation methodology. Experience records may be received after a significant delay. The significance of an event may be different between observing nodes. A node that creates an experience may be unreliable or malicious. For these reasons, our trust system establishes a confidence interval around a weighted mean [18], [19] to overcome this problems rather than taking a Bayesian approach.

To create the confidence interval around a weighted mean, E1 first calculates a weight W_i for each sample point i. It does this by combining the context specific, level of observation weight w_i^c with a new weight w_i^t that is based on the age of the experience record. The formula behind w_i^t may be chosen at the discretion of E1 but the idea is that the older the sample point (experience record) is, the lower the weight should be. This may be for instance some constant chosen from the interval [0,1] raised to the power of the age. These two weights are combined as follows:

$$W_i = w_i^t w_i^c$$



1) E3 overhears and observes E2's behavior
2) E3 generates experience records
3) E3 passes these experience records on to E1
4) E1 evaluates E3's accuracy in generating experience records and discounts the experience records it receives from E3 based on its trust in E3.

Fig. 1. Building trust via a third party

Using this total weight for each sample point E1 then determines the weighted mean ($\overline{x}_w$) of all of the experiences with E2:

$$\overline{x} = \Sigma\left(\frac{W_i}{\Sigma W_i}x_i\right)$$

From this weighted mean, the unweighted variance (σ^2) is then calculated as usual:

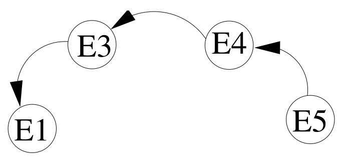
$$\sigma^2 = \frac{\Sigma \left(x_i - \overline{x_w}\right)^2}{n-1}$$

and then turned into a weighted variance (σ_w^2) by E1 via the following manner:

$$\sigma_w^2 = \frac{\sigma^2 \Sigma W_i^2}{\left(\Sigma W_i\right)^2}$$

Armed with the weighted mean $\overline{x}_w$ and the weighted variance σ_w^2, E1 then forms a confidence (eq.1) interval about the weighted mean. To reduce the effect of stale samples and to reduce bias created by correlated samples, the $t_{n-1,1-\alpha/2}$ value is established not by using the usual total number of samples points (n) but instead by using a deflated number of degrees of freedom. This deflated number of degrees of freedom is obtained by simply adding up the sum of all of the total experience weights: ΣW_i. When all total experience weights are in the interval [0,1], the net effect of using this deflated number for degrees of freedom is a widening of the confidence interval. This widening is important due to the reduced confidence we have in correlated and stale values.

2) Incorporating experiences collected by third parties: Although first-hand experiences are the most valuable, it is also valuable for E1 to collect and weigh experiences generated by neighbors of E2. To use experience data produced by other neighbors of E2, E1 must first establish its own trust in the ability of those neighbors to generate experiences. We will use Figure 1 as an example. E3 has generated experiences relating to E2, but until E1 has established its own trust in E3, these experience records cannot be used by E1. E1 starts the process of establishing trust in E3 by comparing experience records it (E1) has created while observing certain sensor network/node behavior to those created by E3 for the same behavior. In this process, E1 is able to collect experiences of E3's ability to generate experiences. It then calculates the confidence



1) E4 overhears and observes E5's behavior
2) E4 generates experience records
3) E4 passes these experience records on to E3
4) E3 passes these experience records on to E1
5) E1 calculates its trust in E4's ability to accuratly generated experience records and E3's ability to accurately route data. E1 then discounts the experience records generated by E4 based on this trust in E4 and E3.

Fig. 2. Building trust in a remote node

interval about the weighted mean of these experience using the equations for weighted mean, weighted variance and a deflated number of degrees of freedom, described above. The resulting confidence interval in the context of E3's *Experience Generation* accuracy is formed by using eq.1. This confidence interval is then transformed by E1 into a fixed point τ_3^{EG} which represents the level of trust E1 places in E3's ability to accurately generate experiences. τ stands for "Trust". The number 3 is the id of the neighbor being evaluated, and EG stands for "Experience Generation". Specifically, the trust level, τ_3^{EG}, is calculated based on the following equation.

$$\tau_3^{EG} = \overline{x}_w - \rho * t_{n-1,1-\alpha/2} * \sqrt{\sigma_w^2/n} \qquad \text{(eq.2)}$$

Here, ρ, ($\epsilon[-1,1]$) is the level of aversion held by E1 (the node doing the evaluation). It identifies the risk E1 is willing to take in E3's experience generation ability. In the worst case, $\rho = 1$, implying that E3 chooses the lowest possible trust value in E3's experience generation ability. In the best case, $\rho = -1$, implying that E1 is willing to accept the highest possible trust value in E3's experience generation capability.

3) Incorporating distant observations: So far we have covered how a node (E1 in our example) can use experience records generated by both itself and immediate neighbors. We will now explain how experience records generated by nodes farther than one hop away (non-directly connected), which can be used in this trust system. This is experience data produced by entities that have potentially never interacted or communicated directly with the node doing the evaluation. For example: E1 wishes to establish a trust confidence interval about E5's sensing accuracy. E1 is not near enough to watch and evaluate E5. An additional entity E4, however, is near enough to evaluate and generate experience data on E5 and it happens to be near enough to E3 for radio communication as illustrated in Figure 2. For E1 to utilize the experience data generated by E4, experience data must be accurately generated and propagated by E4. E1 thus must first evaluate E4's ability to accurately generate experiences in the same manner as described above. However instead of directly being able to evaluate E4, one must instead rely in experience records generated by E3. After receiving experience records relating to E4's ability to accurately generate experiences, E1 combines these into a confidence interval and then in turn into a single trust value, τ_4^{EG}, using eq.2. Likewise, E1 evaluates E3's ability to accurately propagate data. There may exist cases where nodes can accurately generate experiences and sense data, but due to faulty (or malicious) software, fail to route and propagate data accurately. E1 uses all of its available experience records related to E3's ability to propagate data and creates a confidence interval and then a single trust value, τ_3^{DP}, also using eq.2. where τ_3^{DP} in this case represents E1's trust in E3 in the context of "Data Propagation". E1 then uses the trust values it has established in E4 (ability to accurately generate experiences) and E3 (ability to accurately propagate data) to discount the weight of the experiences recorded by E4:

$$W_i = w_i^t w_i^c \tau_4^{EG} \tau_3^{DP}$$

If a certain piece of experience data must be propagated through multiple nodes, then that piece of experience data is discounted by the evaluator's trust in each intermediate node to propagate the data accurately. Hence, the generic equation for assessing the weight of any arbitrary experience record is:

$$W_i = w_i^t w_i^c \tau_{generator-id}^{EG} [\tau_{router1}^{DP} * \tau_{router2}^{DP} * ...]$$

where $\tau_{generator-id}^{EG}$ is the evaluators trust in the node that recorded the experience (in the context of Experience generation) and $\tau_{routerX}^{DP}$ is repeated for each node through which the experience record was propagated. With this method an evaluator establishes trust through a chain of nodes and can use experience data generated by distant nodes.

4) Initial bootstrap of the trust system: Initialization of the system starts by nodes recording their own direct experiences with their physical neighbors. These experiences should include the evaluation of neighbors in at least the two special contexts of Experience-generation and Data-propagation. Each of these direct experiences is used for calculating the trust the evaluator node has in its neighbors. The evaluator calculates the weight (W_i) for each of these experiences as: $w_i^t w_i^c$ Here w_i^t is an age based weight (described in the previous section) and w_i^c is a weight assigned by the evaluator based on the level of contact the experience represents. For each neighbor, the experiences in each of the above contexts are grouped together to form a trust confidence interval in that particular context. For example, a node wishing to form a trust confidence interval in the context of Experience-generation for a particular neighbor will follow this protocol:

1) Observe experience data generated by the neighbor. This experience data would be in some context other than experience generation.
2) Compare that experience data to locally generated experience data and rate the accuracy of the experience data generation. From this comparison a new experience point is generated.
3) Each of these experience points are weighed based on their age and context-weight to form a confidence interval: $\overline{x}_w \pm t_{n-1,1-\alpha/2}\sqrt{\sigma_w^2/n}$.

5) Limited memory for experience data: Sensor nodes usually have limited memory for storing experience data. If a new and apparently useful piece of experience data is acquired and must be added to a completely full experience data store, then an existing piece of experience data must be evicted. The eviction only takes place if the new piece of experience data has a higher "usefulness" than a piece of information already in the cache. We find that the resulting cache replacement pattern is similar to the "small worlds" replacement method as recommended in [5]. In order to gain unfair advantage, certain entities could attempt to flood all receptive nearby entities with messages and requests for interaction in attempt to boost their own trust rating. For this reason, entities would find it beneficial to throttle the rate of new experiences from other entities.

6) Experience correlation: Our statistical methodology for computing confidence intervals expects independent samples. If the experience data are correlated, several samples must be aggregated to generate a single sample [17].

7) Location Awareness: Sensor nodes must have a good sense of their environment in order to evaluate experiences such as sensor values received from other sensor nodes. Location awareness is necessary for extrapolation of sensor data. Typically node location is not known beforehand thus an in-network location awareness system [20] must be used. The location information required for evaluating experiences will be no more than that already required for efficiently routing and aggregating data in a deployed WSN application. Thus, location awareness for the purpose of trust computation should not require any additional resources.

8) Energy Considerations: Our trust system requires sensor nodes to "overhear" messages from neighboring sensor nodes, and also collect trust data from the neighboring sensor nodes. We piggyback trust data on transmission and reception of application messages wherever possible. The system may be combined with other optimizations designed to reduce the overhead of trust formulation. An example of one such optimization is the MIT Leach protocol [21].

IV. Evaluation of the Trust System

Using a custom discrete event simulator, "Trust-Sensim", we have simulated a cluster of nodes with the ability to formulate trust between each other. Here we describe the different aspects of this simulator.

Energy: Each node in the simulation has the energy consumption characteristics for transmitting, receiving, and processing data as well as sleeping of a Telos Rev B WSN node (model TPR2420CA). Each node is powered by a simulated energy capacity of two AA batteries. At the beginning of each simulation, these batteries are given an initial charge normally distributed around 2400 mAh (with a standard deviation of 200mAh).

Positioning: The nodes are positioned in relatively close proximity so as to allow for overlap between their sensing and communication ranges. This allows for all nodes to overhear each other's communications and build trust based on their observations. For this simulation the nodes are statically positioned 5 meters apart in a square grid of NxN nodes that act as a cluster of nodes. We simulate cluster sizes of 2x2, 3x3, 4x4 and 5x5 nodes. The 5x5 cluster having 25 nodes and an edge length of 25 meters. Nodes are aware of each other's identities and positions.

Simulation Events: As in the MIT Leach protocol [21] each node is assigned TDMA time slots for communication with the cluster lead during each round of sensing. For each round of sensing, each node: A) Wakes up on its assigned slot. B) Senses the current temperature and estimated remaining battery charge. C) Transmits the readings to the current cluster lead. D) Receives an ACK message back from the cluster lead which may include system management information such as the identity of the next cluster lead. E) Sleeps until the next round.

At the end of each round the cluster-lead has the responsibility of doing a single high power transmission of the aggregated results to the base station. Any node that disagrees with the aggregated results can do its own high power transmission to the base station. For this simulation the round period was 2 hours. Each cluster lead serves for a total of 25 rounds (50 hours) before a new cluster lead is chosen. During the final round of a cluster lead's tenure, the cluster lead advertises to all of the system nodes the node identify which will serve as the next cluster lead; which is the node with the most available remaining energy. The simulation ends when any node in the system is fully depleted of its energy capacity.

Trust formulation: Each node can enter a mode called "neighborhood watch mode" which causes a node to first wake up at the very beginning of each round of sensing and listen on the radio for transmission of all other nodes. After overhearing the sensing data transmission of other nodes, the node will aggregate the data and transmit the aggregation result to the cluster lead (and any other listening nodes). A node will enter the low-power sleep mode at the very end of the "round" after all other nodes have concluded their aggregated data transmissions.

Neighborhood watch mode allows the system to act in a fully redundant manner. All nodes not only sense and transmit the current cluster lead but they also monitor each other's sensing and aggregation behavior and transmit their own aggregation result to the current cluster lead. Each node also has the ability to enter the "trust formulation mode" which adds the following actions onto the "neighborhood watch mode".

1) Evaluate the sensor readings, data aggregation results and trust experiences overheard from other nodes, based on this evaluation, add new experiences to the local cache of trust experiences.
2) Communicate highly weighted experiences from the local experience cache to neighbors that may be listening via piggy-backing trust experiences on top of sensor reading transmissions to that node acting as the current cluster lead.
3) Probabilistically sleep and skip one or more sensing

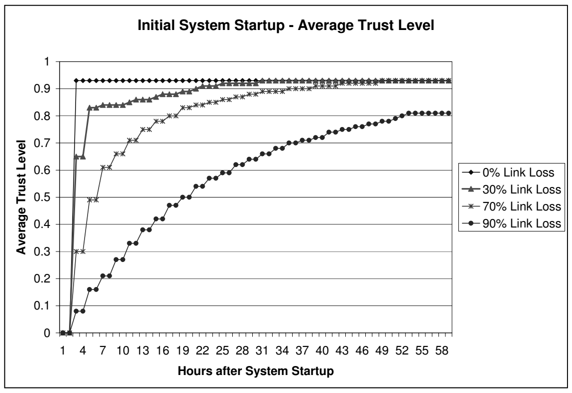


Fig. 3. Effect of Link Loss on Startup Trust Level

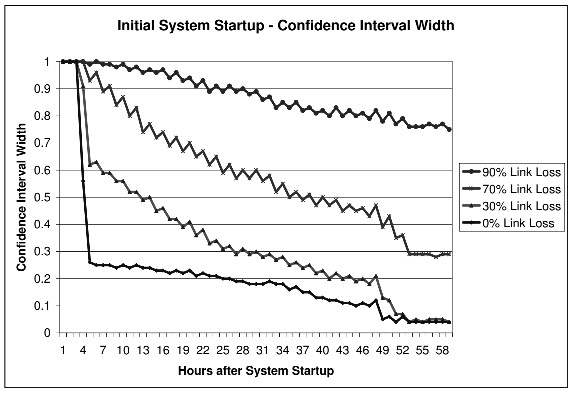


Fig. 4. Effect of Link Loss on Startup Trust Confidence Interval Width

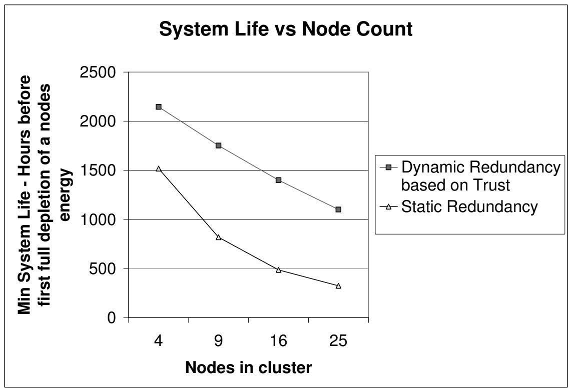


Fig. 6. System Life with Different Node Counts

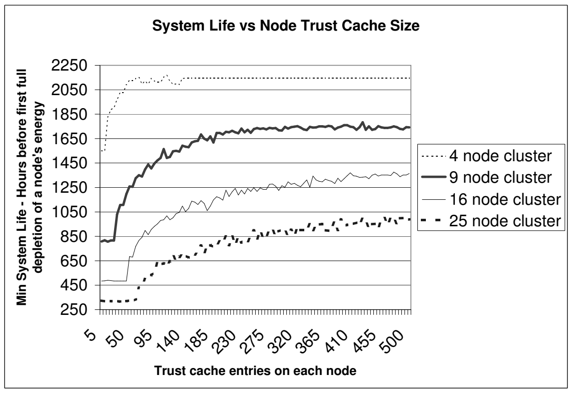


Fig. 7. Cache Size Comparison

rounds based on each node's trust in the current cluster lead and its own neighbors.

On initial system startup no trust exists between nodes. As nodes interact with each other trust is formed. Figures 3 and 4 show the effect of varying levels of link loss on the trust formation process. High levels of link loss prevent narrow confidence intervals from forming and delay, though they do not prevent, the formation of a high trust value.

Nodes that are able to dedicate more memory resources to the formation of trust are able to achieve narrower confidence intervals as shown in Figure 5. There exists a tradeoff to the amount of memory dedicated to the formation of trust and the level of trust achievable (and therefore the life expectancy of a network that uses dynamic redundancy).

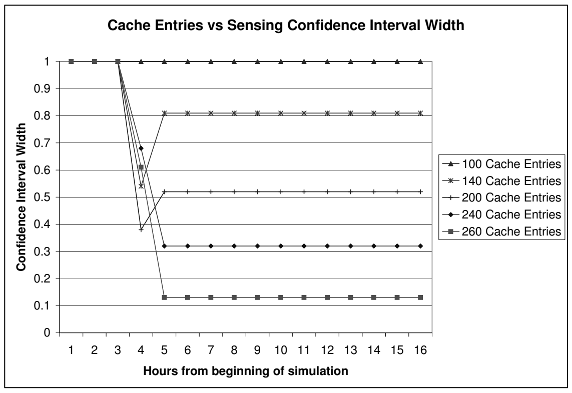


Fig. 5. Cache Size Effect on Startup Confidence Interval Width

Findings

Though building trust requires more memory and computational resources than the neighborhood watch mode alone (due to the trust cache and trust formulation processing), the potential is created for dynamically sleeping through sensing rounds based on current trust in other system nodes. Thus with building trust the potential is created to save energy and extend the life span of the sensor network. We first establish a baseline by comparing three different systems: A) One with no redundancy and neighbor monitoring. In this case, nodes perform only their own duties and never attempt to overhear or monitor the communications of other nodes including the aggregation function performed by the base station. B) A system with full redundancy where all nodes monitor the communications and actions of all other nodes. C) A system with dynamic redundancy based on current levels of trust between nodes. In this simulation we allow each node to have a relatively large trust cache (1000 entries). Figure 6 shows that the achievable life expectancy of a system with trust enabled is well higher than a statically redundant system without trust. We ran the simulator varying the number of nodes in the system as well as the amount of memory dedicated to the trust cache in each system. Figure 7 shows the change in expected

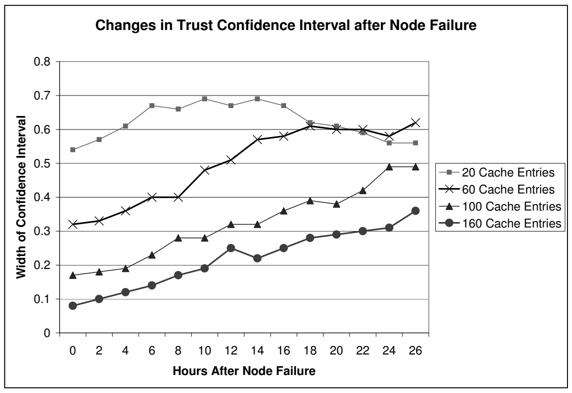


Fig. 8. Simulated Sensor Failure Reaction Time

minimum life of the system (in hours) as the amount of memory available to the trust cache on each node is increased. We find that full redundancy within larger networks requires a considerably higher amount of memory on each node in order to establish trust among nodes. Dedicating a relatively small amount of memory to each node can produce a significant improvement in the life expectancy of the system. As with any typical cache however, there are diminishing returns associated with adding memory to the trust cache on each node. The simulator was also used to test the reaction time for injecting node failures into the system. At approximately 120 hours into the simulation, after trust had been established between system nodes, a sensor failure on one of the nodes was simulated. Figure 8 shows the time required for the system to react (the confidence interval to widen) when a node fails. This graph represents the average trust confidence interval width in the failed node among all other nodes in the system. It is interesting to note that larger trust caches on each node are able to maintain narrower confidence intervals in the event of node failure. This is because with larger caches, a higher number of new experiences are accepted into the cache and usable in calculating trust. Though not shown here, the cluster lead is able to react more quickly than other nodes in the system given that it is awake and thus has one of the first available opportunities to detect node failure. Other nodes in the system that are sleeping at the time of a node failure are unable to detect the node misbehavior until they wake up and are informed by the cluster lead (or their neighbors). It takes more time for the entire system to react with a higher numbers of sleeping nodes. The node with the failed sensor itself may have been sleeping during some of these hours delaying detection by neighbors until it wakes up. A future enhancement to the system might be to assign higher weights to experiences where nodes behavior changes rather than give the same weight to all new experiences. Such a change would assist in drawing the attention to sudden changes in behavior.

V. Conclusions and Future Work

We presented a new distributed approach that establishes reputation-based trust among sensor nodes in order to identify sensor node misbehavior, minimize their impact on applications and maximize energy conservation. We demonstrated the benefits of our approach using extensive simulations. However, we have only tested simple node failures and link loss levels. We plan on investigating the responsiveness of the trust model to malicious misbehavior including both external attackers and existing nodes that have been compromised. We also plan on experimenting with node mobility.

References

[1] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, June 2004.

[2] J. Golbeck and J. Hendler, "Inferring reputation on the semantic web," 2004, http://www.mindswap.org/papers/GolbeckWWW04.pdf.

[3] V. Cahill, E. Gray, J.-M. Seigneur, C. D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. di Marzo Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen, "Using trust for secure collaboration in uncertain environments," *Pervasive Computing*, vol. 2, pp. 52–61, 2003.

[4] M. Sathyanarayanan, "Caching trust rather than content," *Operating Systems Review*, vol. 34, no. 4, pp. 32–33, 2000. [Online]. Available: citeseer.ist.psu.edu/satyanarayanan00caching.html

[5] E. Gray, J. Seigneur, Y. Chen, and C. Jensen, "Trust propagation in small worlds," in *Proceedings of the First International Conference on Trust Management (iTrust2003)*, 2003. [Online]. Available: citeseer.ist.psu.edu/gray03trust.html

[6] S. Ganeriwal and M. Srivastava, "Reputation-based framework for high integrity sensor networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004.

[7] R. Chen and W. Yeager, "Poblano: A distributed trust model for peer-to-peer networks," *Sun Microsystems Technical Paper*, 2000, http://www.jxta.org/docs/trust.pdf.

[8] G. Theodorakopoulos and J. S. Baras, "On trust models and trust evaluation metrics for ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 318–328, February 2006.

[9] A. Srinivasan, J. Teitelbaum, and J. Wu, "Drbts: Distributed reputation-based beacon trust system." in *DASC*, 2006, pp. 277–283.

[10] Y. L. Sun, W. Yu, Z. Han, and K. Lui, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, pp. 305–317, 2006.

[11] M. Krasniewski, P. Varadharajan, B. Rabeler, S. Bagchi, and Y. C. Hu, "Tibfit: Trust index based fault tolerance for arbitrary data faults in sensor networks," *dsn*, vol. 00, pp. 672–681, 2005.

[12] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Proceedings of Sensys 2004*, November 2004.

[13] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *IEEE Computer Society Symposium on Security and Privacy*, May 2005.

[14] J. McCune, E. Shi, A. Perrig, and M. Reiter, "Detection of denial-of-message attacks on sensor network broadcasts," in *IEEE Computer Society Symposium on Security and Privacy*, May 2005.

[15] N. G. Zinaida Benenson and O. Raivio, "Realizing robust user authentication in sensor networks," in *RealWSN*, 2005.

[16] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet (best paper)," in *PerCom*, 2005, pp. 247–256.

[17] A. Law and W. Kelton, "Simulation modeling and analysis," in *McGraw Hill Series in Industrial Engineering and Management Science*, 2000.

[18] D. Krouse and C. Withers, "A visual basic program giving weighted confidence intervals for mean and variance," *Industrial Research Limited Report 1581*, June 2004.

[19] J. M. Bland and S. Kerry, "Weighted comparison of means," *BMJ*, 1998.

[20] L. Lazos and R. Poovendran, "Serloc: Robust localization for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 1, no. 1, pp. 73–100, 2005.

[21] M. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," *Proceedings of IEEE International Conference on Mobile and Wireless Communications Networks, Stockholm, 2002.*, 2002. [Online]. Available: citeseer.ist.psu.edu/handy02low.html