Adaptive Fault-Tolerant QoS Control Algorithms for Maximizing System Lifetime of Query-Based Wireless Sensor Networks

Ing-Ray Chen, *Member*, *IEEE*, Anh Phan Speer, and Mohamed Eltoweissy, *Senior Member*, *IEEE*

Abstract—Data sensing and retrieval in wireless sensor systems have a widespread application in areas such as security and surveillance monitoring, and command and control in battlefields. In query-based wireless sensor systems, a user would issue a query and expect a response to be returned within the deadline. While the use of fault tolerance mechanisms through redundancy improves query reliability in the presence of unreliable wireless communication and sensor faults, it could cause the energy of the system to be quickly depleted. Therefore, there is an inherent trade-off between query reliability versus energy consumption in query-based wireless sensor systems. In this paper, we develop adaptive fault-tolerant quality of service (QoS) control algorithms based on hop-by-hop data delivery utilizing "source" and "path" redundancy, with the goal to satisfy application QoS requirements while prolonging the lifetime of the sensor system. We develop a mathematical model for the lifetime of the sensor system as a function of system parameters including the "source" and "path" redundancy levels utilized. We discover that there exists optimal "source" and "path" redundancy under which the lifetime of the system is maximized while satisfying application QoS requirements. Numerical data are presented and validated through extensive simulation, with physical interpretations given, to demonstrate the feasibility of our algorithm design.

Index Terms—Wireless sensor networks, reliability, timeliness, query processing, redundancy, energy conservation, QoS, mean time to failure.

1 Introduction

Over the last few years, we have seen a rapid increase in the number of applications for wireless sensor networks (WSNs). WSNs can be deployed in battlefield applications, and a variety of vehicle health management and condition-based maintenance applications on industrial, military, and space platforms. For military users, a primary focus has been area monitoring for security and surveillance applications.

A WSN can be either source-driven or query-based depending on the data flow. In source-driven WSNs, sensors initiate data transmission for observed events to interested users, including possibly reporting sensor readings periodically. An important research issue in source-driven WSNs is to satisfy QoS requirements of event-to-sink data transport while conserving energy of WSNs. In query-based WSNs, queries and data are forwarded to interested entities only. In query-based WSNs, a user would issue a query with QoS requirements in terms of reliability and timeliness.

Retrieving sensor data such that QoS requirements are satisfied is a challenging problem and has not been studied until recently [4], [5], [6], [7], [8], [9]. The general approach is to apply redundancy to satisfy the QoS requirement. In this paper, we are also interested in applying redundancy to satisfy application-specified reliability and timeliness requirements for query-based WSNs. Moreover, we aim to determine the optimal redundancy level that could satisfy QoS requirements while prolonging the lifetime of the WSN. Specifically, we develop the notion of "path" and "source" level redundancy. When given QoS requirements of a query, we identify optimal path and source redundancy such that not only QoS requirements are satisfied, but also the lifetime of the system is maximized. We develop adaptive fault-tolerant QoS control (AFTQC) algorithms based on hop-by-hop data delivery to achieve the desired level of redundancy and to eliminate energy expended for maintaining routing paths in the WSN.

The rest of the paper is organized as follows: In Section 2, we survey related work. In Section 3, we discuss the WSN system model and assumptions used in the paper. In Section 4, we develop probability models for computing the lifetime of a query-based WSN as a function of "path" and "source" redundancy being employed, defined as the number of queries that the system is able to execute successfully in terms of QoS satisfaction before failure. We also discuss extensions to the mathematical model developed to deal with software faults, data aggregation, and concurrent query processing which a query-based WSN might experience. In Section 5, we analyze the effect of redundancy on the system lifetime of WSNs, and identify the optimal level of "path" and "source" redundancy that could maximize the system lifetime while satisfying the QoS requirements of queries before failure. Section 6 presents simulation validation. Finally, Section 7 concludes the paper and discusses future work.

- *I.-R. Chen and A.P. Speer are with the Department of Computer Science, Virginia Tech, 7054 Haycock Road, Falls Church, VA 22043. E-mail: {irchen, nphan}@vt.edu.*
- *M. Eltoweissy is with Pacific Northwest National Laboratory, 902 Battelle Boulevard, Richland, WA 99354. E-mail: Mohamed.Eltoweissy@pnl.gov.*

Manuscript received 1 Jan. 2009; accepted 22 Sept. 2009; published online 4 Dec. 2009.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2009-01-0002. Digital Object Identifier no. 10.1109/TDSC.2009.54.

2 Related Work

Existing research efforts related to applying redundancy to satisfy QoS requirements in query-based WSNs fall into three categories: traditional end-to-end QoS, reliability assurance, and application-specific QoS [4]. Traditional end-to-end QoS solutions are based on the concept of end-to-end QoS requirements. The problem is that it may not be feasible to implement end-to-end QoS in WSNs due to the complexity and high cost of the protocols for resource-constrained sensors. An example is Sequential Assignment Routing (SAR) [5] that utilizes path redundancy from a source node to the sink node. Each sensor uses a SAR algorithm for path selection. It takes into account the energy and QoS factors on each path, and the priority level of a packet. For each packet routed through the network, a weighted QoS metric is computed as the product of the additive QoS metric and a weight coefficient associated with the priority level of that packet. The objective of the SAR algorithm is to minimize the average weighted QoS metric throughout the lifetime of the network. The algorithm does not consider the reliability issue.

ESRT [12] has been proposed to address this issue with reliability as the QoS metric. ReInForM has been proposed [6] to address end-to-end reliability issues. ReInForm considers information awareness and adaptability to channel errors along with a differentiated allocation strategy of network resources based on the criticality of data. The protocol sends multiple copies of a packet along multiple paths from the source to the sink such that data is delivered with the desired reliability. It uses the concept of dynamic packet state to control the number of paths required for the desired reliability using local knowledge of the channel error rate and topology. The protocol observes that for uniform unit disk graphs, the number of edge-disjoint paths between nodes is equal to the average node degree with a very high probability. This protocol results in the use of the disjoint paths existing in a thin band between the source and the sink. However, the protocol only concerns QoS in terms of reliability.

In [7], M. Perillo and Heizelman provide application QoS with the goal of maximizing the lifetime of WSNs while satisfying a minimum level of reliability. This maximization is achieved through the joint optimization of scheduling active sensor sets and finding paths for data routing. The lifetime is defined as the sum of the time that all sensor sets are used. The approach uses the strategy of turning off redundant sensors for periods of time to save energy while considering the trade-off between energy consumption and reliability. This approach can extend the lifetime of a network considerably compared with approaches that do not use intelligent scheduling. However, this approach is not scalable and QoS is limited to application reliability only.

Recently, a multipath and multispeed routing protocol called MMSPEED is proposed in [8] which takes into account both timeliness and reliability as QoS requirements. The goal is to provide QoS support that allows packets to choose the most proper combination of service options depending on their timeliness and reliability requirements. For timeliness, multiple QoS levels are supported by providing multiple data delivery speed options. For reliability, multiple reliability requirements are supported by probabilistic multipath forwarding. The protocol provides end-to-end QoS provisioning by employing localized geographic forwarding using immediate neighbor information without end-to-end path discovery and maintenance. It utilizes dynamic compensation which compensates for inaccuracy of local decision as a packet travels toward its destination. The protocol adapts to network dynamics. However, MMEPEED does not consider energy issues. Our work considers energy consumption, in addition to reliability and timeliness requirements as in MMSPEED. Further, we also consider network dynamics due to sensor failures, energy depletion, and sensor connectivity. Utilizing hop-by-hop data delivery, the AFTQC algorithm developed in our work specifically forms m_p redundancy paths for path redundancy and m_s sensors for source redundancy to satisfy the imposed QoS requirements, facilitating the determination of the best (m_p, m_s) that would maximize the lifetime of the WSN.

In [9], QoS is defined as the optimum number of sensors that should be sending information to the sinks at any given time. The protocol utilizes the base station to communicate QoS information to each of the sensors using a broadcast channel. It exploits the mathematical paradigm of the Gur Game to dynamically adjust to the optimum number of sensors. The objective is to maximize the lifetime of the sensor network by having sensors periodically powered down to conserve energy, and at the same time, having enough sensors powered up and sending packets to the sinks to collect enough data. The protocol allows the base station to dynamically adjust the QoS resolution. This solution requires the determination of the amount of sensors that should be powered up a priori to maintain a resolution. QoS metrics for data delivery such as reliability and timelines are not considered.

Clustering SN prolongs the system lifetime of a WSN [1], [2] because clustering reduces contention on wireless channels [13] and supports data aggregation and forwarding at cluster heads (CHs). HEED [1] increases energy efficiency by periodically rotating the role of CHs among SNs with equal probability such that the SN with the highest residual energy and node proximity to its neighbors within a cluster area is selected as a CH. In LEACH [2], the key idea is to reduce the number of nodes communicating directly with the base station by forming a small number of clusters in a self-organizing manner. LEACH uses randomization with equal probability in cluster head selection to achieve energy balance. REED [14] considers the use of redundancy to cope with failures of SNs in hostile environments. We also consider cluster-based WSNs for energy reasons.

Our approach of satisfying application reliability and timeliness requirements while maximizing the system lifetime is to determine the optimal level of redundancy at the “source” and “path” levels. The source-level redundancy refers to the use of multiple sensors to return the requested sensor reading. The path-level redundancy refers to the use of multiple paths to relay the reading to the sink node. Since WSNs are constrained with resources, the AFTQC algorithm developed in this paper utilizes hop-by-hop data delivery and dynamically forms multiple paths for data delivery, without incurring extra overhead to first formulate multiple paths before data delivery. Our contribution is that we identify the best level of redundancy to be used to answer queries to satisfy their QoS requirements while prolonging the lifetime of query-based WSNs.

3 System Model

ACRONYMS

MTTF	Mean time to failure, defined as the mean number of queries that the sensor system is able to execute successfully with QoS satisfaction before failure;
SN	Sensor node;
PC	Processing center;
CH	Cluster head;
WSN	Wireless sensor network.

NOTATION

A	Length of each side of a square sensor area (meter);
n_b	Size of a data packet (bit);
E_{elec}	Energy dissipation to run the transmitter and receiver circuitry (joule/bit);
E_{amp}	Energy used by the transmit amplifier to achieve an acceptable signal to noise ratio (joule/bit/meter^2);
E_o	Initial energy per SN (joule);
$E_{initial}$	Initial energy of the WSN (joule);
$E_{clustering}$	Energy for executing the clustering algorithm (joule);
$E_{threshold}$	Energy threshold below which the WSN depletes its energy (joule);
E_q	Average energy consumption per query (joule);
R_q	Probability that a query reply is delivered successfully within the deadline;
r	Wireless radio communication range (meter);
p	Probability of an SN becoming a CH;
q	SN hardware failure probability;
q_s	SN reading software failure probability;
e_j	Transmission failure probability of an SN with index j;
n	Number of SNs in the WSN;
n_s	Number of SNs per cluster;
N_c	Number of clusters in the WSN;
m_p	Number of paths from a source CH in response to a query;
m_s	Number of SNs per cluster in response to a query;
f	Fraction of neighbor SNs that will forward data;
λ	SN population density (nodes/meter^2);
λ_q	Query arrival rate (times/second);
d_{inter}	Distance between a source CH and the processing center (meter);
d_{intra}	Distance between an SN to its CH (meter);
N^h_{inter}	Average number of hops between a source CH and the processing center;
N^h_{inter}	Average number of hops between an SN to its CH;
i	Index to a path;
j	Index to a node;
k	Index to a neighbor node;
S_{jk}	Progressive transmission speed between two SNs with indexes j and k (meter/second);
$T_{clustering}$	Time interval for executing the clustering algorithm (second);
T_{req}	Query deadline requirement (second);
R_{req}	Query reliability requirement.

A WSN consists of a set of low-power sensor nodes (SNs) typically deployed through air-drop into a geographical area. We make the following assumptions regarding the structure and operation of a query-based WSN:

1. SNs are homogeneous and indistinguishable with the same initial energy level E_o.
2. SNs are deployed into a geographical area of size A^2 with each side of length A. This assumption has been used in the literature [1], [2], [11] to simplify the analysis although the method developed in this paper can deal with other geographical shapes.
3. SNs are distributed according to a homogeneous spatial Poisson process with intensity λ. We assume the domain is relatively free of obstacles and the WSN is dense enough so that the length of a path connecting two SNs can be approximated by the straight line distance. We assume that the WSN deployed is sufficiently dense to satisfy the connectivity condition [10] so that sensors are well connected.
4. The failure behavior of an SN due to environment conditions (i.e., harsh environments causing hardware failure) and software faults is characterized by a hardware failure probability parameter q (where $0 < q < 1$) and a software failure probability q_s (where $0 < q_s < 1$). Both parameters are assumed to be constant.
5. A clustering algorithm (e.g., [1], [2]) that aims to fairly rotate SNs to take the role of CHs has been used to organize sensors into clusters for energy conservation purposes, as illustrated in Fig. 1. A CH is elected in each cluster. The function of a CH is to manage the network within the cluster, gather sensor reading data from the SNs within the cluster, and relay data in response to a query. The clustering algorithm is executed periodically by all SNs in iterations in which:

   - An SN announces its role as a CH candidate with probability p;
   - The announcement message carrying the candidate CH's residual energy information is broadcast with the time to live (TTL) field set to the number of hops bounded by the cluster area size predetermined at design time;
   - Any non-CH SN overhearing the announcement can select a CH with the highest residual energy to join a cluster, and will report its location to the candidate CH;
   - This announcement and join process is executed in iterations such that a tentative CH can change its role to an SN if it overhears a CH candidate having a higher residual energy in a subsequent iteration;
   - If a non-CH SN does not hear any CH announcement, p is doubled in the next iteration

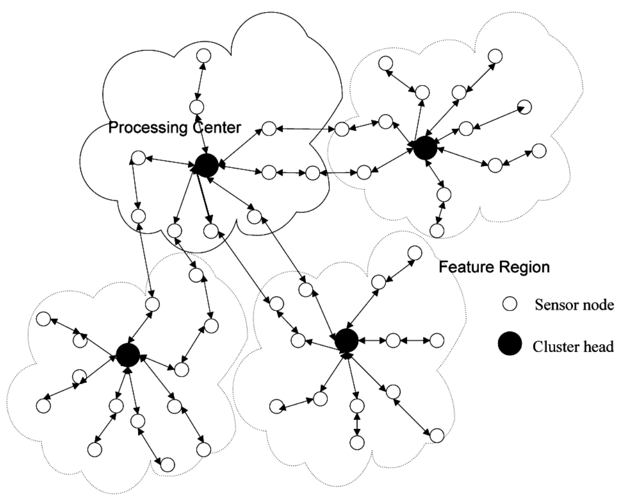

Fig. 1. Cluster-based WSN Architecture.

A clustering algorithm as described above can be proven to converge within a finite number of iterations, and, in effect, could randomly rotate the role of a CH among SNs in a cluster so that sensors consume their energy evenly [1]. With random rotation, the cluster size, n_s, would be equal to $1/p$ [11]. Note that to deal with uneven SN distribution, this CH-rotating probability doubles in a subsequent iteration until it becomes 1; so, in the worst case when an SN cannot find any CH to join a cluster, it will eventually form a cluster by itself with probability 1. This unbalanced clustering behavior occurs rarely when the WSN is dense. When the WSN is sufficiently dense and the target cluster area size is the same, it is shown that clusters are balanced in practice [1]. The total energy expended by the system depends on the period ($T_{clustering}$) over which the clustering algorithm is executed and the energy expended per execution ($E_{clustering}$). The clustering algorithm is executed with the rate of $1/T_{clustering}$ to balance energy consumption of SNs within a cluster. We determine the clustering interval $T_{clustering}$ for satisfying the assumption of fair rotation among SNs by simulation. Note that by our clustering protocol, an SN will select another SN to be the CH only if they are connected possibly through multiple hops; so, for the case in which an SN is $2r$ apart but it is still connected to a candidate CH because there are intermediate SNs around, the candidate CH can still be the CH for the SN. In the worst case in which there is no candidate CH around, an SN will elect itself as the CH. However this case is extremely rare because of the massive deployment of SNs with high density.

6. To save energy, the transmission power of an SN even when it is a CH is reduced to a minimum level to enable the SN to communicate with its neighbor SNs within one-hop radio range denoted by r. Thus, every SN needs to use a multihop route (i.e., passing through a number of other SNs) for it to communicate with another SN distance away. When the WSN becomes less dense as time progresses due to sensor node failures, the one-hop radio range can be increased dynamically to allow the WSN to continue its function at the expense of energy consumption.
7. The unreliable transmission failure behavior of the wireless medium in WSNs due to noise and interference is characterized by a transmission failure parameter. This parameter varies among sensors, depending on the node density and the packet transmission frequency of SNs within radio range. Let e_j denote the transmission failure probability of $\mathrm{SN_j}$. Note that e_j varies dynamically in response to network dynamics.
8. Users on a flying airplane or a moving vehicle can issue queries through any CH, which we call a processing center (PC), as labeled in Fig. 1. Assume that queries arrive at the system in accordance with a Poisson process with rate λ_q. A query may involve all or a subset of clusters, say, k clusters, to respond to the query for data sensing and retrieval. These requested clusters are termed *source clusters*. The CH of a source cluster receives m_s packets carrying the same data content from m_s SNs within its cluster because of source redundancy but it will only relay the first packet it receives to the PC. The CH could also aggregate data and return summarized information in terms of the *min*, *average*, or *max* of sensor readings received from m_s SNs. We assume queries are issued by the user who is on the move. Thus, the timeliness requirement may be tight, i.e., on the order of second. The WSN does not have a base station. Also, sensors in a cluster will rotate to be the CH in their cluster. As a result, the notion of higher energy consumption by critical nodes [3] for relaying messages to a base station or to a CH does not exist.
9. Routing in the WSN is based on geographic forwarding (e.g., [8]). No path information needs to be maintained by individual SNs to conserve energy. Essentially, only the location information of the destination SN needs to be known by a forwarding SN for any source-destination communication. We note that when a CH is elected periodically, the location information is broadcast to the WSN to let other CHs know its location. Also, SNs within a cluster know the location of their CH, and vice versa, as part of the election process.
10. A source CH must relay sensor data information to the PC in response to a user query, and thus, can consume more energy than an SN within its cluster. The energy consumed by the system for data forwarding in response to a query depends on the total length (in terms of the number of hops) of the paths connecting m_s SNs within a cluster to the source CH for source redundancy, and the total length of the m_p paths connecting the source CH and the processing center (the destination CH) for path redundancy. As the clustering algorithm in effect rotates sensor nodes within a cluster fairly evenly to assume the role of the CH, each sensor node would consume energy at about the same rate. Thus, instead of considering each individual sensor energy

level, we can consider the system energy whose initial energy level is given by $E_{initial} = nE_o$. When the energy level of the system falls below a threshold value, say $E_{threshold}$, the WSN is considered as having depleted its energy.

11. To save energy, SNs operate in *power saving* mode. At this mode, an SN operates either in active mode, i.e., transmitting or receiving, or in sleep mode. The radio module of a modern sensor [15], [16] allows it to shut off while in sleep mode. Essentially, in sleep mode, an analog block stays awake and acts as the radio detector. When the analog block detects a radio signal, the signal is converted to a control signal which, in turn, is sent to power control electronics to wake up the radio module. With the state-of-art technology, energy consumed by the analog block is very small. Also, the current technology can achieve the transient time between active and sleep mode of 5 μs [15]. The energy consumed for turning on/off radio while an SN is in power saving mode is also very small. Thus, we only consider the energy consumed while an SN is transmitting or receiving in active mode. For the energy model, we adopt the radio model in [1]. The energy dissipation to run the transmitter and receiver circuitry is denoted as E_{elec}. The energy used by the transmit amplifier to achieve an acceptable signal to noise ratio is denoted as E_{amp}. Also, there is an r^2 energy loss due to channel transmission under the assumption that the WSN is relatively free of obstacles, where r is the radio range. Thus, the energy spent by an SN to transmit a data packet of size n_b bits for a distance of r is given by

$$E_t = n_b(E_{elec} + E_{amp}r^2). \quad (1)$$

The energy spent to receive a message is given by

$$E_R = n_b E_{elec}. \quad (2)$$

We define the system *lifetime* or the *mean time to failure* (MTTF) as the total number of queries the system can answer correctly until it fails to delivery query results either due to channel or sensor faults, or when the system energy reaches the energy threshold level $E_{threshold}$. We define a query's QoS requirements in terms of its *reliability* and *timeliness* requirements, denoted as R_{req} and T_{req}. The system must deliver query results within T_{req} and the reliability of data delivery must be at least R_{req}. Our objective is to determine the best path and source redundancy levels to satisfy QoS while maximizing MTTF.

4 Probability Model

The adaptive fault-tolerant QoS control (AFTQC) algorithm developed in this paper takes two forms of redundancy. The first form is path redundancy. That is, instead of using a single path to connect a source cluster to the processing center, m_p disjoint paths may be used. The second is source redundancy. That is, instead of having one sensor node in a source cluster return requested sensor data, m_s sensor nodes may be used to return readings to cope with data transmission and/or sensor faults. Fig. 1 illustrates a scenario in which $m_p = 2$ (two paths going from the CH to the processing center) and $m_s = 5$ (five SNs returning sensor readings to the CH).

Below, we derive analytical expressions for R_q (query reliability) and E_q (energy consumption per query) resulting from the use of AFTQC. We first derive MTTF for the case in which only one source cluster is required to answer a query and only the reverse traffic is considered. Later, we generalize the result to the case in which the forward traffic for query dissemination is considered and in which multiple source clusters are required to answer a query.

4.1 Query Reliability

Let d_{inter} be a random variable denoting the distance between a source CH and the PC and d_{intra} be a random variable denoting the distance between an SN to its CH. Then, the number of hops between the PC and the source CH, denoted by h, is given by

$$h = \left\lceil \frac{d_{\text{inter}}}{r} \right\rceil. \quad (3)$$

With the user being mobile, a query can be issued from the user to *any* CH which serves as the PC for that query. Thus, the location of the processing center varies on query by query basis. For derivation convenience, without loss of generality, let the PC be located in the center of the sensor area with the coordinate at (0, 0) and the source CH be randomly located at (X_i, Y_i) in the square sensor area with $-A/2 \le X_i \le A/2$ and $-A/2 \le Y_i \le A/2$. Then, the expected value of d_{inter} is given by

$$\begin{aligned} E[d_{\text{int}\,er}] &= \int_{-A/2}^{A/2}\int_{-A/2}^{A/2} \sqrt{(X_i^2 + Y_i^2)}\left(\frac{1}{A}\right)\left(\frac{1}{A}\right) dX_i dY_i \\ &= 0.3825A. \end{aligned} \quad (4)$$

The same final expression for $E[d_{inter}]$ would result if we had taken the coordinate of the processing center to be (X_c, Y_c) in the square sensor area and put two more integrals, one for X_c and the other for Y_c with $-A/2 \le X_c \le A/2$ and $-A/2 \le Y_c \le A/2$, because of symmetric properties. For notational convenience, let N_{inter}^h represent the average number of hops to forward sensor data from a source CH to the processing center

$$N_{\text{inter}}^h = \lceil E[h] \rceil = \left\lceil \frac{0.3825A}{r} \right\rceil. \quad (5)$$

Since a sensor becomes a CH with probability p and all the sensors are distributed in the area in accordance with a spatial Poisson process with intensity λ, the CH and non-CH sensors will also be distributed in accordance with a spatial Poisson process with rates pλ and (1-p)λ, respectively. Non-cluster-head sensors thus would join the cluster of the closest CH to form a Voronoi cell [4] corresponding to a cluster in the WSN. It has been shown that [5], [11] the average number of non-cluster-head sensors in each Voronoi cell is $(1-p)/p$ and the expected distance from a non-cluster-head sensor to the CH is given by

$$E[d_{\text{intra}}] = \frac{1}{2(p\lambda)^{1/2}}. \quad (6)$$

If this distance is more than per-hop distance r, a sensor will take a multihop route to transmit sensor data to the CH. The average number of intermediate sensors (including the sensor itself) is the quantity above divided by per-hop distance r. Let N^h_{intra} denote the average number of hops to forward sensor data from an SN responsible for a reading to its CH. Then, N^h_{intra} is given by

$$N^h_{\text{intra}} = \left\lceil \frac{1}{2r(p\lambda)^{1/2}} \right\rceil. \tag{7}$$

A query response is transmitted from an SN performing sensing to the PC through the CH hop-by-hop. The total delay must be lower than the imposed deadline requirement T_{req} for the user to accept the query result. To achieve this, as a query response is relayed along a path, we choose a forwarding node that satisfies the "minimum transmission speed" requirement. Since the distance separating a sensing SN from the PC is $d_{inter} + d_{intra}$ and the maximum time for the query result to reach the PC is T_{req}, the minimum transmission speed, denoted by X_{set}, to satisfy the imposed deadline requirement is given by

$$X_{set} = \frac{d_{\text{inter}} + d_{\text{intra}}}{T_{req}}. \tag{8}$$

Plugging in the expected values of d_{inter} and d_{intra}, the expected minimum transmission speed is given by

$$E[X_{set}] = \frac{0.3825A + \frac{1}{2(p\lambda)^{1/2}}}{T_{req}}. \tag{9}$$

As a query result is forwarded hop-by-hop through geographical routing, the expression above represents the minimum per-hop transmission speed to transmit the query results from an SN to the PC in order not to miss the deadline. Here, we note that queuing delay is ignored because not much cross-traffic is anticipated in a query-based WSN; so, queuing delay is considered small compared with transmission delay.

Let $Q_{t,jk}$ be the probability that if a packet is forwarded to SN_k from SN_j, the speed requirement would be violated. To calculate $Q_{t,jk}$, we need to know the transmission speed S_{jk} from SN_j to SN_k. This can be dynamically measured by SN_j following the approach described in [8]. If S_{jk} is above $E[X_{set}]$, then $Q_{t,jk} = 0$; otherwise, $Q_{t,jk} = 1$. In general, S_{jk} is not known until runtime. If S_{jk} is uniformly distributed within a range $[a, b]$, then $Q_{t,jk}$ can be computed as

$$Q_{t,jk} = cdf\big(S_{jk} \leq E[x_{set}]\big) = \frac{E[X_{set}] - a}{b - a}. \tag{10}$$

Other than speed violation failure, a node may also fail to relay sensor data because of either a sensor failure or a transmission failure, or both. Let $Q_{r,j}$ be this failure probability of an SN, say, SN_j. Then, $Q_{r,j}$ is given by

$$Q_{r,j} = 1 - [(1 - q)(1 - e_j)]. \tag{11}$$

Here, for sensor failure, we have only considered hardware failure. Later, in Section 5, we will extend this to the case in which SN software faults are possible.

We develop a hop-by-hop data delivery scheme to implement the desired level of redundancy to achieve QoS. For path redundancy, we want to form m_p paths from a source CH to the processing center, as illustrated in Fig. 2. This is achieved by having m_p nodes on the first hop relay the data, and only one single node relay the data per receiving group in all subsequent hops. For source redundancy, we want each of the m_s sensors to communicate with the source CH through a distinct path. Here, we note that a WSN is inherently broadcast-based. However, an SN can specify a set of SNs in the next hop (that is, m_p in the first hop and 1 in a subsequent hop) as the intended receivers and only those SNs will forward data.

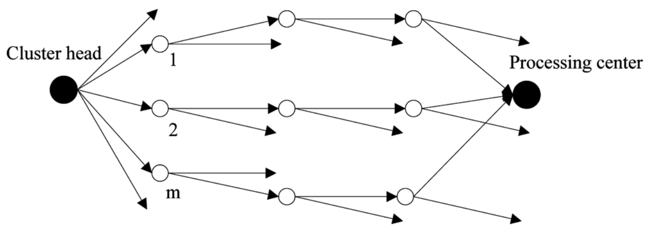

Fig. 2. Hop-by-hop data delivery in AFTQC.

It has been reported that the number of edge-disjoint paths between nodes is equal to the average node degree with a very high probability [6]. Therefore, when the density is sufficiently high such that the average number of one-hop neighbors, n_k, calculated as $\lambda\pi r^2$, is sufficiently larger than m_p and m_s, this hop-by-hop data delivery scheme can effectively result in m_p redundant paths for path redundancy and m_s distinct paths from m_s sensors for source redundancy.

The probability of SN_j failing to relay a broadcast packet to a one-hop neighbor SN_k because of either sensor/channel failures, or speed violation, denoted by $Q_{rt,jk}$, is given by

$$Q_{rt,jk} = 1 - [(1 - Q_{r,j})(1 - Q_{t,jk})]. \tag{12}$$

The probability that *at least* one next-hop SN (among the one-hop neighbors) of SN_j along the direction of the destination node is able to satisfy the speed requirement and receive the broadcast message is given by

$$\theta_j = 1 - \prod_{k=1}^{f \times n_k} Q_{rt,jk}. \tag{13}$$

Here, n_k is the number of neighbors; f is the fraction of neighbors that would forward data based on geographical routing, e.g., $f = 1/4$ meaning only the sensors along the quadrant toward the direction of the target node will do data forwarding. Note that while SN_j forwards data to its one-quadrant neighbors SN_ks, if one of SN_ks is the destination node, then the probability that the destination SN fails to receive the message due to sensor/channel failures or speed violation is exactly equal to $Q_{rt,jk}$, as given in (12).

Below, we derive the probability that a path is successfully formed for hop-by-hop data delivery between the source CH and the processing center. Since there are N^h_{inter} hops between the source CH (the first SN with index 1), and the processing center (the last SN with index $N^h_{inter} + 1$), a path is formed for data delivery if in each hop there is *at least* one next-hop sensor along the direction of the target node that is able to satisfy the speed requirement and receive the broadcast message, *and* also that the destination node is able to satisfy the speed requirement and receive the message. Thus, the probability that a path of length N_{inter}^h is formed successfully for hop-by-hop data delivery is given by

$$\Theta\left(N_{\text{inter}}^h\right) = \left(\prod_{j=1}^{N_{\text{inter}}^h - 1} \theta_j\right) \times \left(1 - Q_{rt,N_{\text{inter}}^h\left(N_{\text{inter}}^h+1\right)}\right), \tag{14}$$

where $Q_{rt,}N_{inter}^h(N_{inter}^h+1)$ is from (12) for the probability that the PC node (the last SN with index N_{inter}^h+1) fails to receive the message due to sensor/channel failures or speed violation. Here, we adopt the convention that if the upper bound is smaller than the lower bound in the product term, then the product term evaluates to 1.

We create m_p paths between the source CH and the PC based on the hop-by-hop data delivery scheme discussed earlier. The source cluster will fail to deliver data to the PC if one of the following happens:

1. None of the SNs in the first hop receives the message. The probability for this case is $1-\theta_1$.
2. In the first hop, $i(1 \le i < m_p)$ SNs receive the message, and each of them attempts to form a path for data delivery. However, all i paths fail to deliver the message because the subsequent hops fail to receive the broadcast message. The failure probability for this case is

$$\sum_{|I|<m_p}\left\{\left[\prod_{i\in I}\left(1-Q_{rt,1i}\right)\right]\left(\prod_{i\notin I}Q_{rt,1i}\right)\right\}$$
$$\left\{\prod_{i\in I}\left[1-\Theta_i\left(N_{inter}^h-1\right)\right]\right\},$$

where I stands for a set consisting of first-hop SNs that receive the message and $|I|$ is the cardinality of set I. The first term is the probability that i SNs from the set of fn_k nodes in the first hop successfully receive the message, and the second term is the probability that all i paths fail to deliver data. Note that a subscript i has been used to label Θ_i to refer to path i (i.e., the path that starts from a particular SN with index i). Also, the argument to Θ_i is only N_{inter}^h-1 because there is one less hop to be considered in each path.
3. In the first hop, at least m_p SNs receive the broadcast message from the source CH from which m_p SNs are randomly selected to forward data, but all m_p paths fail to deliver the message because the subsequent hops fail to receive the broadcast message. The probability for this case is

$$\sum_{|I|\ge m_p}\left\{\left[\prod_{i\in I}(1-Q_{rt,1i})\right]\left(\prod_{i\notin I}Q_{rt,1i}\right)\right\}$$
$$\left\{\prod_{\substack{i\in M,\\ M\subseteq I,\\ |M|=m_p}}\left[1-\Theta_i\left(N_{\text{int}\,er}^h-1\right)\right]\right\},$$

where M is a subset of I with cardinality of m_p. The second term in the above expression is the probability that all m_p paths fail to deliver data.

Thus, the probability of the source cluster failing to deliver data to the processing center is given by

$$Q_{fp}^{m_p} = 1-\theta_1+\sum_{|I|<m_p}\left\{\left[\prod_{i\in I}\left(1-Q_{rt,1i}\right)\right]\left(\prod_{i\notin I}Q_{rt,1i}\right)\right\}$$
$$\left\{\prod_{i\in I}\left[1-\Theta_i\left(N_{\text{int}\,er}^h-1\right)\right]\right\}+\sum_{|I|\ge m_p}\left\{\left[\prod_{i\in I}\left(1-Q_{rt,1i}\right)\right]\right.$$
$$\left.\left(\prod_{i\notin I}Q_{rt,1i}\right)\right\}\left\{\prod_{\substack{i\in M,\\ M\subseteq I,\\ |M|=m_p}}\left[1-\Theta_i\left(N_{\text{int}\,er}^h-1\right)\right]\right\}. \tag{15}$$

For source redundancy, instead of using one SN, we assign m_s SNs in each cluster to return sensor readings to their CH to cope with channel/sensor faults. To implement source redundancy, SNs also use hop-by-hop data delivery based on geographical routing to send sensor data to their CH. For a path of N_{intra}^h from an SN to the CH, again assign an index of 1 to the SN and an index of N_{intra}^h+1 to the CH. Then, following a similar derivation, the probability that a path is formed successfully from the SN to the CH for data delivery is given by

$$\Theta\left(N_{\text{intra}}^h\right) = \left(\prod_{j=1}^{N_{\text{intra}}^h-1}\theta_j\right)\times\left(1-Q_{rt,N_{\text{intra}}\left(N_{\text{intra}}^h+1\right)}\right). \tag{16}$$

For source redundancy, m_s SNs are used for returning sensor readings. So, the failure probability that all m_s SNs within a cluster fail to return sensor reading to the CH is given by

$$Q_{fs}^{m_s} = \prod_{i=1}^{m_s}\left[1-\Theta_i\left(N_{\text{intra}}^h\right)\right]. \tag{17}$$

Note that in each of the m_s paths, distinct e_j and S_{jk} exist along each path depending on each path's traffic condition. Combining results from above, the failure probability of a source cluster not being able to return a correct response, because of either path or source failure, or both, is given by

$$Q_f = 1-\left(1-Q_{fp}^{m_p}\right)\left(1-Q_{fs}^{m_s}\right). \tag{18}$$

Therefore, the query success probability is given by

$$R_q = 1-Q_f. \tag{19}$$

4.2 Query Processing Energy Consumption

Next, we calculate energy consumed per query. For source redundancy, in response to a query, an SN assigned would transmit a data packet to its source CH. Since the average number of hops between an SN and its CH is given by N_{intra}^h, as derived above, and a query requires the use of m_s SNs for source redundancy, the total energy required for these m_s SNs to forward sensor readings to the CH is given by

$$E_s = m_s N_{\text{intra}}^h\left[E_T+\lambda(\pi r^2)E_R\right]. \tag{20}$$

For path redundancy, let E_{ch} be the total energy consumed by the WSN to transmit sensor data from the source CH to the PC with m_p paths connecting the CH to the processing center. The source CH would broadcast a copy of the data packet and all first-hop neighbors would receive. Then, among the first-hop neighbors, m_p nodes would broadcast again and all 2nd-hop neighbors would receive. In each of the subsequent hops on a path, only one node would broadcast and the neighbors on the next hop would receive. Consequently, E_{ch} is given by

$$E_{ch} = E_T + \lambda(\pi r^2)E_R + m_p(N^h_{\text{inter}} - 1)[E_T + \lambda(\pi r^2)E_R]. \quad (21)$$

The total amount of energy spent by the system, E_q, to answer a query that demands a source cluster to respond, using m_s SNs for source redundancy and m_p paths for path redundancy, is given by

$$E_q = E_{ch} + E_s. \quad (22)$$

4.3 Energy Consumption Due to Clustering

For clustering, the system would consume energy for broadcasting the announcement message and for the cluster-join process. Since p is the probability of becoming a CH, there will be pn SNs that would be broadcasting the announcement message. This announcement message will be received and retransmitted by each SN to the next hop until the TTL of the message reaches the value 0, i.e., the number of hops equals N^h_{intra}. Thus, the energy required for broadcasting is $pn[N^h_{\text{intra}}\lambda(\pi r^2)(E_T + E_R)]$. The cluster-join process will require an SN to send a message to the CH informing that it will join the cluster and the CH to send an acknowledgement to the SN. Since there are pn CHs and $(n - pn)$ SNs in the system, the energy for this is $n(E_T + E_R)$. Let the size of the message exchange be n_l. E_R and E_T will be calculated from (1) and (2) with n_l in place of n_b. Let $N_{iteration}$ be the number of iterations required to execute the clustering algorithm. Then, the energy required for each execution of the clustering algorithm, $E_{clustering}$, is given by

$$E_{clustering} = pnN_{iteration}\left[N^h_{\text{intra}}\lambda(\pi r^2)(E_T + E_R)\right] + n(E_T + E_R). \quad (23)$$

4.4 System Lifetime—Mean Time to Failure

Our objective is to find the best redundancy level represented by m_p and m_s that would satisfy the query reliability and timeliness requirements while maximizing MTTF, when given a set of system parameter values characterizing the application and network conditions. That is, if T_{req} and R_{req} are the timing and reliability requirements of a query, then we determine the best combination of (m_p, m_s) such that the MTTF is maximized, subject to the constraint

$$R_q > R_{req}. \quad (24)$$

Note that the constraint given above implies that the timing requirement T_{req} is also satisfied because we consider the probability of minimum transmission speed being satisfied when we derive R_q in (19).

From a user's perspective, if the user does not see a response returned within the specified real-time constraint, the system is considered as having failed. We define a metric called the MTTF of the sensor system that considers this failure definition. Specifically, we define the MTTF of a sensor data system as the average number of queries that the system is able to answer correctly before it fails, with the failure caused by either channel or sensor faults (such that a response is not delivered within the real-time deadline), or energy depletion.

When m_p paths and m_s SNs are used to achieve R_q in order to satisfy condition (24), the amount of energy consumed is given by E_q in (22) above. Consider for the time being that the system fails due to energy depletion only. Then, the system fails when the system's energy falls below $E_{threshold}$. Let the potential maximum lifetime of the system be denoted by T_{life}. There are two sources of energy consumption: query processing and periodic clustering. Also, consider the case in which queries arrive at the system as a Poisson process with rate λ_q. The energy consumed due to query processing is given by $E_q\lambda_qT_{life}$, where λ_qT_{life} is the maximum number of queries the system can possibly process during its lifetime. On the other hand, the energy expended due to the execution of the periodic clustering algorithm is given by $E_{clustering}T_{life}/T_{clustering}$, where $T_{life}/T_{clustering}$ is the number of times the clustering algorithm is executed during the system lifetime. Thus, T_{life} can be calculated as follows:

$$\lambda_qE_qT_{life} + E_{clustering}\frac{T_{life}}{T_{clustering}} = E_{initial} - E_{threshold}. \quad (25)$$

The maximum number of queries that the system is able to sustain before running out its energy, denoted by N_q, is given by

$$N_q = \lambda_qT_{life} = \frac{\lambda_q(E_{initial} - E_{threshold})}{\lambda_qE_q + (E_{clustering}/T_{clustering})}. \quad (26)$$

Since the system is able to answer N_q queries before energy depletion, each with the reliability of R_q, the MTTF of the system is the expected number of queries that the system can answer without experiencing a failure with the upper bound of N_q, i.e.,

$$MTTF = \sum_{i=1}^{N_q-1} iR_q^i(1 - R_q) + N_qR_q^{N_q}. \quad (27)$$

This MTTF metric can be translated into a more classic "system lifetime" metric with the unit of time, i.e., mean lifetime to failure (MLTF), as follows:

$$MLTF = \frac{MTTF}{\lambda_q}. \quad (28)$$

4.5 Generalization

Certain assumptions have been made in the paper to simplify the mathematical analysis. Below, we discuss how these assumptions can be relaxed to generalize the model.

4.5.1 Query Involving Multiple Clusters for a Response

The analysis can be easily extended to the case where multiple source clusters are demanded. Let $P_q(k)$ be the probability that a query requires k source clusters to respond. Let $R_q(k)$ be the query success probability for a query that requires k source clusters to respond, and $E_q(k)$ be the energy consumption of the system to answer a query that requires k source clusters. The expressions for $R_q(k)$ and $E_q(k)$ can be easily derived from those based on a single source cluster, i.e., through (19) and (22), respectively, based on the application requirements (e.g., the query is considered successful, if all k source clusters must return sensor readings). Then, E_q would be given by the expected value of $E_q(k)$ as

$$E_q = \sum_{k=1}^{np} E_q(k) P_q(k). \tag{29}$$

The success probability of a query, R_q, would be given as

$$R_q = \sum_{k=1}^{np} R_q(k) P_q(k). \tag{30}$$

The same analytical expression for the MTTF, as given by (27), with new E_q and R_q, given in (29) and (30), then can be used to analyze the effect of $P_q(k)$.

4.5.2 Concurrent Query Processing with Distinct QoS Requirements

Our analysis can also be extended to scenarios in which queries arrive at the system concurrently, say, by multiple users, by simply measuring e_j (transmission failure probability experienced by $\mathrm{SN_j}$) and S_{jk} (progressive speed if the packet is forwarded from $\mathrm{SN_j}$ to $\mathrm{SN_k}$) to properly account for the interference and noise introduced due to simultaneous transmission of data packets by SNs. The reason is that the MTTF metric is based on the number of queries that the system is able to service before it fails, so it does not matter whether queries are processed sequentially or concurrently, as long as the interference and noise introduced due to simultaneous transmission of data packets by SNs have been properly accounted for in calculating the query success probability (R_q) and energy consumption (E_q).

In reality, queries may be in different service classes, and thus, have different QoS requirements. The analysis can be extended to handle this more general case by considering the probability of a query being in a particular QoS class and computing the weighted R_q and E_q of a query, and consequently, the MTTF of the system. For example a timeliness requirement can be (1 sec, 5 sec, 10 sec) and a reliability requirement can be (0.999, 0.99, 0.9), so there will be nine QoS classes. For each QoS class, say, class i, we apply the analysis method to calculate $R_{q,i}$ and $E_{q,i}$ for class i only. Then, given knowledge of the probability that a query is in class i, $PQoS_i$, we can calculate the expected reliability and energy consumption per query, $\overline{R_q}$ and $\overline{E_q}$, as

$$\overline{R_q} = \sum_i PQoS_i \times R_{q,i}, \tag{31}$$

$$\overline{E_q} = \sum_i PQoS_i \times E_{q,i}. \tag{32}$$

Then, the MTTF calculation can use $\overline{R_q}$ and $\overline{E_q}$ instead of R_q and E_q.

4.5.3 Software Fault

For source redundancy, m_s SNs are used for returning sensor readings. If we consider both hardware and software failures of SNs, the system will fail if the majority of SNs does not return sensor readings (due to hardware failure), or if the majority of SNs returns sensor readings incorrectly (due to software failure). Assume that all SNs have the same software failure probability, denoted by q_s. Also assume that all sensors that sense a given event make the same measurements unless experiencing software failure. Then, to account for software failure, (17) can be replaced with (33) below

$$\begin{aligned} Q_{fs}^{m_s} = & \sum_{|I| \geq \lceil \frac{m_s}{2} \rceil} \left\{ \prod_{i \in I} \left[1 - \Theta_i\left(N_{\text{intra}}^h\right)\right] \right\} \left\{ \prod_{i \notin I} \Theta_i\left(N_{\text{intra}}^h\right) \right\} \\ & + \sum_{|I| \geq \lceil \frac{m_s}{2} \rceil} \left\{ \prod_{i \notin I} \left[1 - \Theta_i\left(N_{\text{intra}}^h\right)\right] \right\} \left\{ \prod_{i \in I} \Theta_i\left(N_{\text{intra}}^h\right) \right\} \\ & \times \left\{ 1 - \left[\sum_{j=\lceil \frac{m_s}{2} \rceil}^{|I|} \binom{|I|}{j} (1-q_s)^j q_s^{(|I|-j)} \right] \right\}. \end{aligned} \tag{33}$$

Here, the first expression is the probability that the majority of m_s SNs failing to return sensor readings due to hardware failure, and the second expression is the probability that the majority of m_s SNs returning sensor readings but no majority of them agrees on the same sensor reading as the output because of software failure. Here, we note that sensor reading errors may be resulting from software faults or from falsified readings by a compromised sensor node having been attacked and that the majority voting mechanism proposed can cope with both types of sensor reading errors.

4.5.4 Data Aggregation

The analysis performed thus far assumes that a source CH does not aggregate data. The CH may receive up to m_s redundant sensor readings due to source redundancy but will just forward the first one received to the PC. Thus, the data packet size is the same. For more sophisticated scenarios, conceivably the CH could also aggregate data for query processing and the size of the aggregate packet may be larger than the average data packet size. We extend the analysis to deal with data aggregation in two ways. The first is to set a larger size for the aggregated packet that would be transmitted from a source CH to the PC. This will have the effect of favoring the use of a smaller number of redundant paths (i.e., m_p) because more energy would be expended to transmit aggregate packets from the source CH to the PC. The second is for the CH to collect a majority of sensor readings from its sensors before data are aggregated and transmitted to the PC. The analysis of data aggregation thus, in effect, is the same as the one we have performed for SN software faults in Section 4.5.3 requiring a majority of sensors to return correct sensor readings.

4.5.5 Forward Traffic

The analysis performed in the paper considers only the reserve traffic for response propagation from SNs to the PC but neglects the forward traffic for query dissemination from the sink to the CH and SNs. The reliability and energy consumption of the forward traffic due to hop-by-hop query delivery can be calculated by following a similar analysis as for the reverse traffic. The success probability (R_q) would be adjusted by considering the forward traffic and reverse traffic together as a series system. The energy consumption of a query (E_q) would be used to calculate the maximum number of queries the system can possibly process. This, along with R_q, would allow MTTF to be calculated.

TABLE 1
Parameter Default Values

Parameter	Default Value	Parameter	Default Value
m_p	**[1 – 10]**	A	**400m**
m_s	**[1 – 10]**	n_b	**50 bytes**
n	**1000**	E_{elec}	**50 nJ/bit**
n_s	**100**	ε_{amp}	**10 pJ/bit/m^2**
q	**10^{-6}**	E_o	**10 Joule**
e	**[0.0001 – 0.1]**	$E_{threshold}$	**0 Joule**
r	**40 m**	$N_{iteration}$	**3**
f	**¼**	$T_{clustering}$	**[5 – 20] sec**
λ	**10 nodes/(40 x 40 m^2)**	T_{req}	**[0.3 – 1.0] sec**
λ_q	**1 query/min**	B	**200Kb/s**

TABLE 2
Optimal (m_p, m_s) with Varying e and T_{req}

T_{req}	e=0.0001	0.001	0.01
0.4 sec	5,5	5,5	5,6
0.5 sec	3,3	4,4	4,4
1.0 sec	2,2	3,3	4,4
2 sec	1,1	2,1	2,3
5 sec	1,1	1,1	2,3

5 Evaluation

In this section, we present numeric data to demonstrate the tradeoff between R_q and E_q and that there exists an optimal (m_p, m_s) set that would maximize the MTTF of the sensor system while satisfying (24). Table 1 lists the parameters used along with their default parameters. Our WSN consists of 1,000 sensor nodes distributed according to a Poisson process with density λ in a square area of 400 m by 400 m. Each SN has a transmission radio range of 40 m. The initial bandwidth of the wireless channel is 200 Kb/s. Each SN has an initial energy of 10 J. The energy parameters used by the radio module are adopted from [1], [2]. The energy cost to run the transmitter/receiver radio circuitry per bit processed (E_{elec}) is chosen to be 50 nJ/bit. The energy used by the transmit amplifier to achieve an acceptable signal to noise ratio (ε_{amp}) is chosen to be 10 $\mathrm{pJ/bit/m^2}$.

While in reality e_j (transmission failure probability experienced by $\mathrm{SN_j}$) and S_{jk} (progressive speed if the packet is forwarded to $\mathrm{SN_k}$ from $\mathrm{SN_j}$) vary depending on network traffic, we consider $e_j = e$ and S_{jk} being uniformly distributed with parameters [a, b] to simplify the analysis. We vary other key parameters to study their effect on optimal (m_p, m_s) and MTTF.

5.1 MTTF Analysis

Table 2 summarizes the optimal (m_p, m_s) set that would maximize the MTTF of the sensor system under the environment characterized by the set of parameter values listed in Table 1. Other parameter values may generate different (m_p, m_s), but the trend remains the same. We see that as wireless transmission failure probability e increases, the system tends to use more redundancy to satisfy (24) and to maximize the MTTF of the sensor system. Also, as the real-time deadline increases, the system tends to use less redundancy. In the special case in which the network is extremely reliable and the deadline is not stringent, the optimal (m_p, m_s) is at (1, 1). We observe that there always exists an optimal (m_p, m_s) set that would maximize the MTTF of the sensor system.

Fig. 3 shows a snapshot of the MTTF of the sensor system as a function of (m_p, m_s) with $T_{req} = 1.0$. Two 3D graphs are shown in Fig. 3 to show the effect of e. The top 3D graph is for the case in which $e = 0.0001$, where the optimal (m_p, m_s) set is (2, 2) at which the MTTF is maximized. The bottom 3D graph is for the case in which $e = 0.001$ for which the optimal (m_p, m_s) set is (3, 3). We see from these two 3D diagrams that either inadequate or excessive redundancy is detrimental to the MTTF of the sensor system.

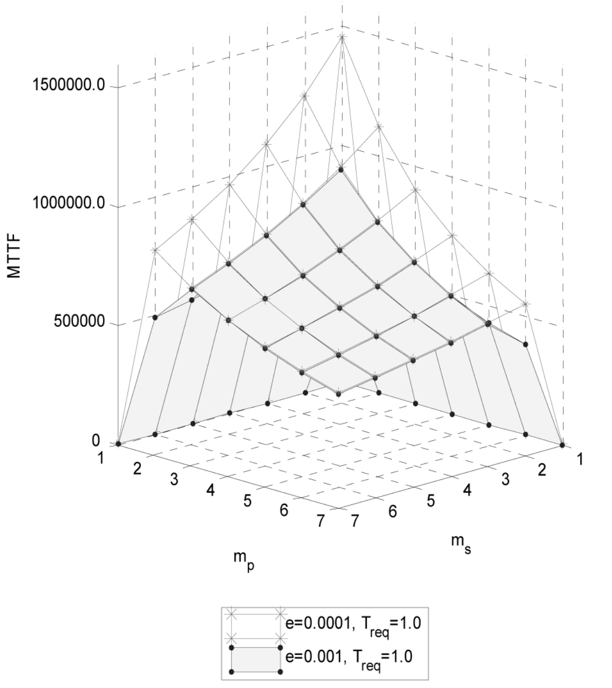

Fig. 3. MTTF versus (m_p, m_s) with $T_{req} = 1$ sec, $e = [0.0001 - 0.001]$.

The existence of the optimal (m_p, m_s) set can be best understood by seeing the trade-off between R_q and E_q as a function of (m_p, m_s). Fig. 4 shows R_q versus (m_p, m_s) as a function of (m_p, m_s). When either m_p or m_s increases, R_q increases. In particular, R_q is more sensitive to m_p because in the environment tested, the distance between the processing center and a CH (N^h_{inter}) is longer than that between the CH and an SN within a cluster (N^h_{intra}). Consequently, incorporating path redundancy (represented by m_p) greatly improves R_q compared with source redundancy (represented by m_s).

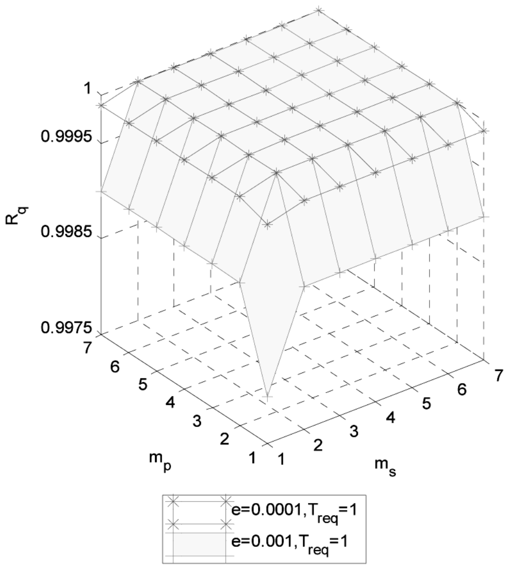

Fig. 4. R_q versus (m_p, m_s) with $T_{req} = 1$ sec, $e = [0.0001 - 0.001]$.

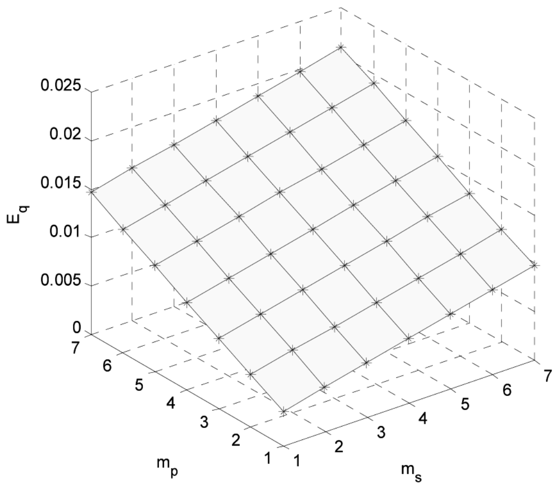

Fig. 5. E_q versus (m_p, m_s).

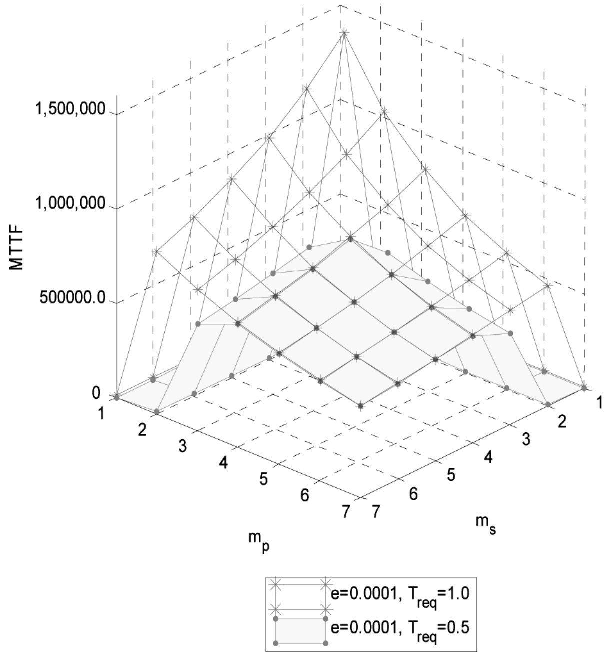

Fig. 6. MTTF with $e = 0.0001$, $T_{req} = [0.5 - 1.0]$ sec.

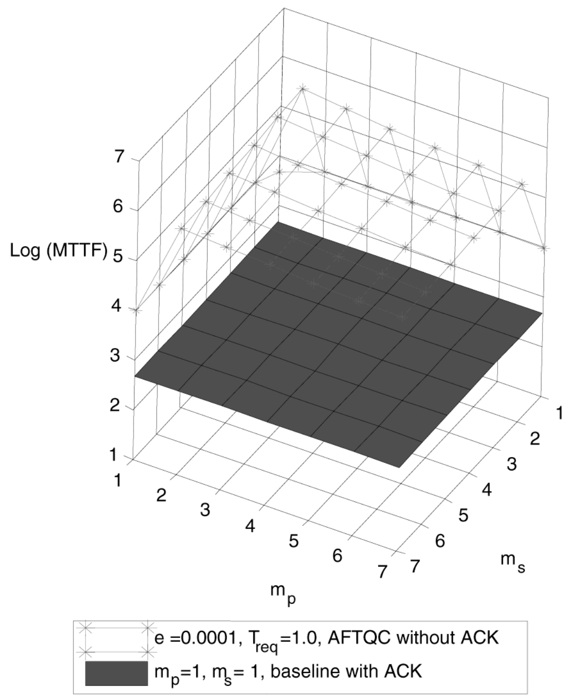

Fig. 7. AFTQC versus baseline with $T_{req} = 1$ sec, $e = 0.0001$ in logarithmic scale.

Correspondingly, Fig. 5 shows the energy consumption as a function of (m_p, m_s). We see that the energy consumption per query is monotonically increasing as either m_p or m_s increases. Therefore, if more redundancy is used to answer a query, on one hand, the MTTF would increase due to a higher R_q (to satisfy (24)), but, on the other hand, the MTTF would decrease due to a high E_q. As a result, an optimal redundancy level in terms of optimal (m_p, m_s) exists.

Next, we test the effect of the real-time deadline on MTTF. Fig. 6 shows a snapshot of the MTTF of the sensor system as a function of (m_p, m_s) with $e = 0.0001$ with varying T_{req}. The top 3D graph is for the case in which $T_{req} = 1.0$ for which the optimal (m_p, m_s) set is (2, 2) at which the MTTF is maximized. The bottom 3D graph is for the case in which $T_{req} = 0.5$ for which the optimal (m_p, m_s) set is (4, 4). In general, we observe that as T_{req} increases (less stringent real-time deadline constraints), the MTTF increases. Also, the system would select less redundancy to maximize the MTTF of the system.

5.2 Comparison of AFTQC versus Baseline

We compare our design with a baseline design in which there is no redundancy and the classic "acknowledgement and retransmission on timeout" mechanism is used for data transmission. Figs. 7 and 8 show a snapshot of the MTTF of the sensor system as a function of (m_p, m_s) in logarithmic scale in order to more vividly show the baseline design case. Fig. 7 is for the case in which the channel transmission reliability is relatively high, i.e., $e = 0.0001$. The top 3D

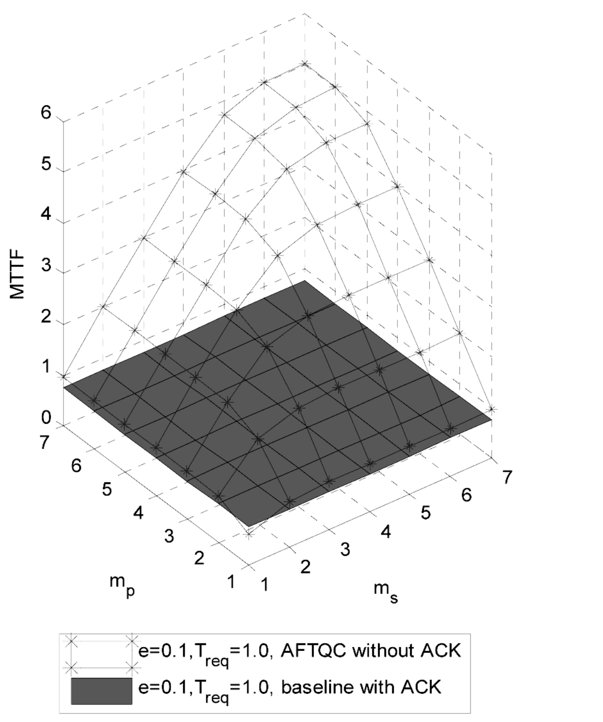

Fig. 8. AFTQC versus baseline with $T_{req} = 1$ sec, $e = 0.1$ in logarithmic scale.

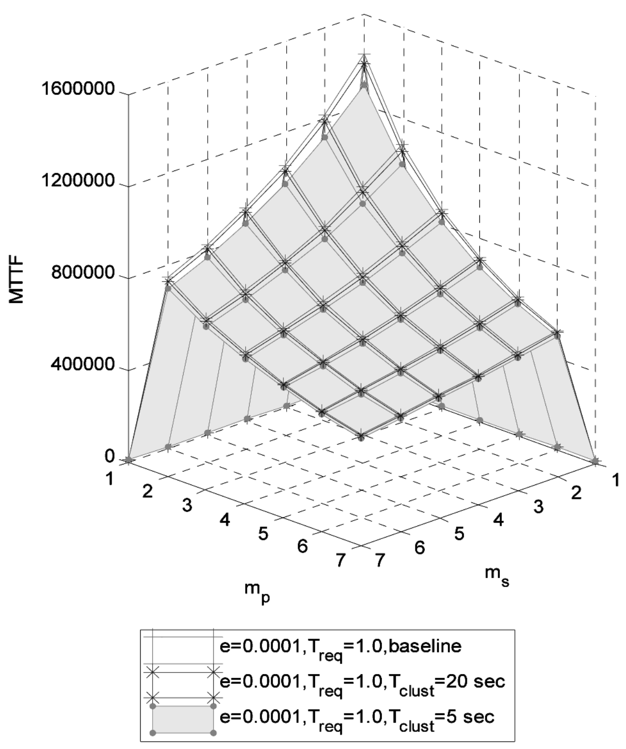

Fig. 9. Effect of clustering intervals on MTTF with $e = 0.0001$, $T_{req} = 1.0$ sec.

graph shows the MTTF under AFTQC. The bottom 3D graph shows the MTTF using the baseline design (labeled as $m_p = 1$, $m_s = 1$ with ACK). We observe that AFTQC (without ACK) greatly increases the MTTF compared with the baseline design under this set of parameter values characterizing the WSN. We also observe that when the WSN is extremely reliable, i.e., when e is extremely small, the optimal (m_p, m_s) is at (1, 1), AFTQC still yields a higher MTTF than the baseline system because no acknowledgement is being used by AFTQC which saves energy.

Next, we consider a case in which the channel transmission reliability is relatively low, i.e., $e = 0.1$. We observe that when the network is not reliable, the baseline scheme only marginally performs better than AFTQC when (m_p, m_s) is set to (1, 1) to run AFTQC. In all other settings, AFTQC significantly outperforms the baseline scheme, the effect of which is especially pronounced at the optimal $(m_p, m_s) = (7, 7)$. Summarizing the results observed from Figs. 7 and 8, we conclude that AFTQC operating under the optimal (m_p, m_s) set always outperforms the baseline scheme and that properly utilizing redundancy would prolong the system lifetime while satisfying QoS requirements of queries.

5.3 Effect of Clustering on MTTF

In this section, we analyze the effect of clustering on the proposed algorithm. We also analyze the effect of different clustering intervals on the system MTTF.

Fig. 9 shows a snapshot of the MTTF of the WSN system as a function of (m_p, m_s) with $T_{req} = 1.0$, e $= 0.0001$ to show the effect of clustering. All 3D graphs show the optimal (m_p, m_s) set of (2, 2) at which the MTTF is maximized. The top 3D graph shows the ideal AFTQC baseline case in which the energy used for clustering is zero, i.e., $E_{clustering} = 0$. The second 3D graph is for the case when the clustering interval $T_{clustering} = 20$ sec. The third 3D graph is the case when the clustering interval $T_{clustering} = 5$ sec. The energy consumed $E_{clustering}$ by the last two cases is calculated by (23).

We see that when the clustering interval is short ($T_{clustering} = 5$ sec), the MTTF values are lower than that under the ideal baseline case. This is because the energy consumption by the clustering algorithm is significant in this case. When the clustering interval is sufficiently long ($T_{clustering} = 20$ sec), the system achieves about the same MTTF value as the ideal baseline case. In this case, the energy consumption by the clustering algorithm is small and does not significantly affect the system MTTF.

Finally, we note that the MTTF curves for all three cases show the same trend with respect to (m_p, m_s) with the optimal set at (2, 2) and that the optimal (m_p, m_s) set is relatively insensitive to the energy used by the clustering algorithm. This is due to the assumption that clustering is executed frequent enough to maintain perfect rotation of CHs, so the frequency of clustering will only affect the total energy consumed but will not affect the optimal (m_p, m_s) set selected. In Section 6, we will conduct a simulation study to identify the frequency of clustering under which the assumption is justified, and compare simulation versus analytical results.

5.4 AFTQC with Software Failure

Finally, here we analyze the effect of software faults on MTTF. Figs. 10 and 11 show a snapshot of the MTTF of the sensor system as a function of (m_p, m_s) with $T_{req} = 1.0$ after applying (33) derived in Section 4.5.3 for modeling

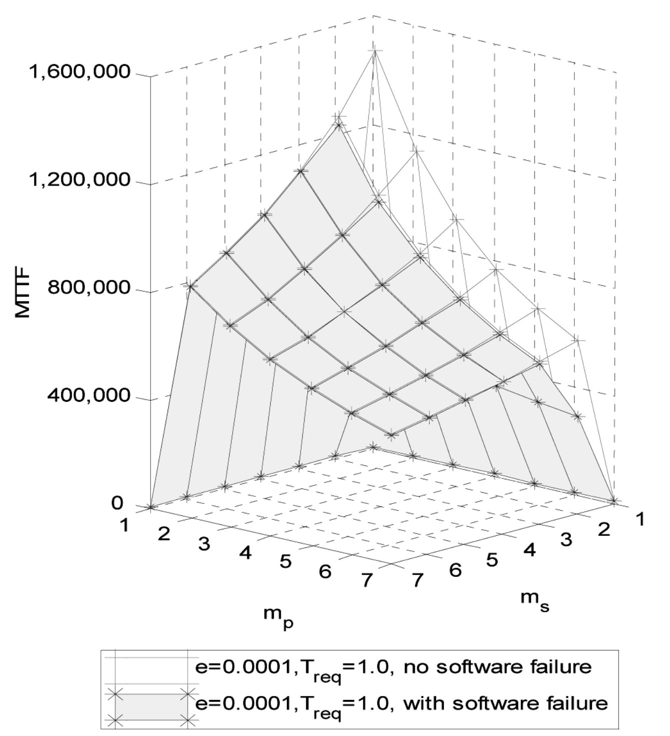

Fig. 10. AFTQC with/without software failure with $e = 0.0001$, $T_{req} = 1.0$ sec.

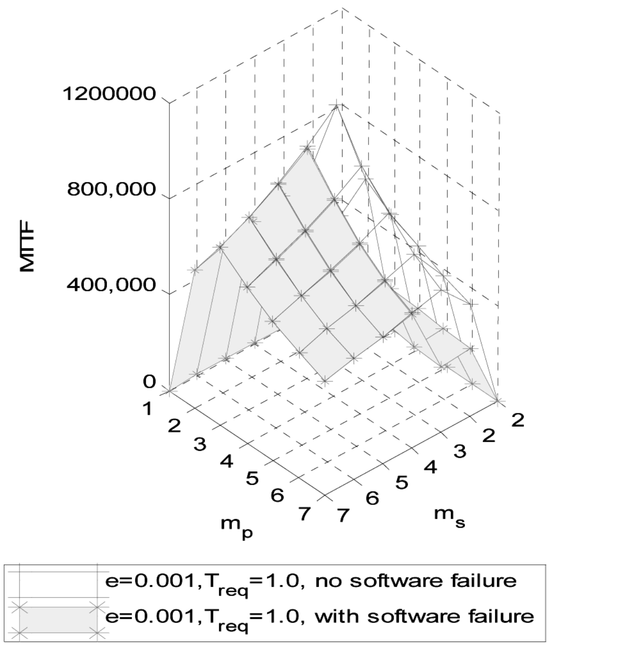

Fig. 11. AFTQC with/without software failure with $e = 0.001$, $T_{req} = 1.0$ sec.

software failure in the calculation. Figs. 10 and 11 show the shift of the optimal (m_p, m_s) when software failure is included compare with the case when there is no software failure. Fig. 10 is for the case in which $e = 0.0001$, $T_{req} = 1.0$. The top 3D graph is for the case when we do not include software failure in the analysis. For this case, the optimal (m_p, m_s) set is (2, 2) at which the MTTF is maximized. The bottom 3D graph is for the case when we include software failure in the analysis. For this case, the optimal (m_p, m_s) set is (2, 3). We see that when software failure is included in the analysis, the optimal (m_p, m_s) is changed from (2, 2) to (2, 3). This reflects the fact that when software faults are possible, the system tends to choose a larger number of sensor nodes to increase the probability that the majority agrees on the same sensor reading, e.g., in this case optimal m_s is changed from two to three. Fig. 11 is for the case in which $e = 0.001$, $T_{req} = 1.0$. In this case, the optimal is changed from (3, 3) to (3, 4). Again, we see that the system choses a larger number of sensor nodes to cope with software failure.

6 Simulation

In this section, we present simulation results to compare with analytical results for the purpose of validation. Table 3 lists a default set of parameter values used in the simulation. We use J-Sim as our simulation framework. We consider a small-scaled WSN so we could obtain simulation results with statistical significance. In our simulation environment, SNs are distributed in a square terrain area of size A^2 in accordance with a population distribution function. We consider two population distribution functions, uniform distribution versus homogeneous Poisson, and analyze the sensitivity of simulation results with respect to SN population distributions. SNs use stateless nondeterministic geographic routing, as described in [17]. To simulate geographic routing, we utilize the *Node Position Tracker* implemented in J-Sim. We use the S-MAC protocol [18] in our simulation of the sensor MAC layer. A query is considered as not being executed successfully if one of the following conditions happens:

- If all m_s SNs fail to deliver sensor readings to the source CH, due to a combination of link failure, SN energy depletion or SN hardware/software failures;

TABLE 3
Parameter Values in Simulation

Parameter	Value	Parameter	Value
m_p	**[1 – 4]**	n_b	**50 bytes**
m_s	**[1 – 4]**	n_q	**10 bytes**
N	**600**	E_T	**0.0000264 J**
n_s	**100**	E_R	**0.00002 J**
q	**0.0001**	E_o	**0.05 J**
e	**0.0001**	$E^s_{threshold}$	**0.0000264 J**
r	**40 m**	$T_{clustering}$	**5-20 sec**
f	**½**	T_{req}	**1.0 sec**
λ_q	**1 query/sec**	B	**200Kb/s**
A	**400m**		

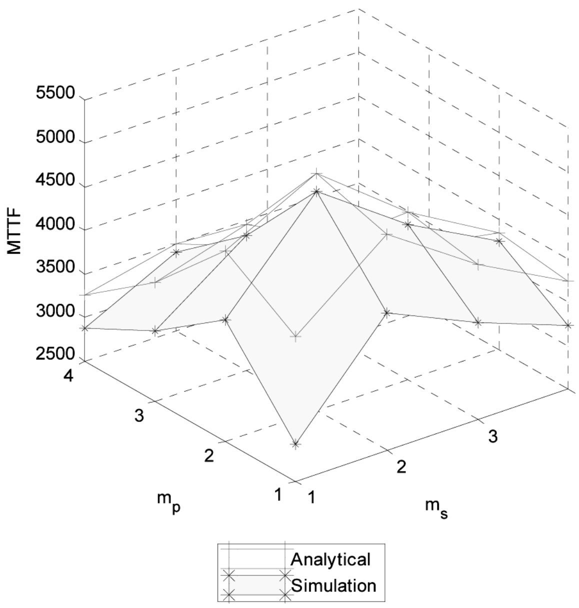

Fig. 12. Comparison of analytical and simulation results for MTTF versus (m_p, m_s).

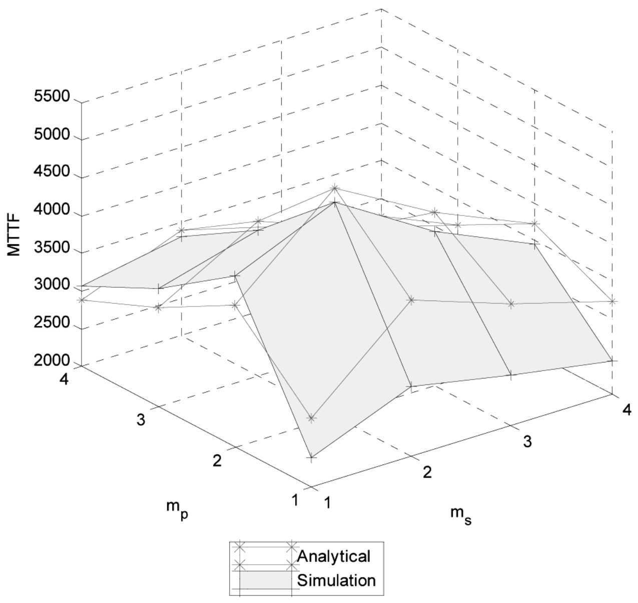

Fig. 13. Simulation versus analytical results in MTTF versus (m_p, m_s) when network dynamics are considered.

- All paths between the CH and the PC are broken, due to a combination of link failure, SN energy depletion, and SN hardware failure;
- The query result is not returned within the deadline requirement T_q. We accumulate the time it takes to propagate the results back based on the progressive speeds of the SNs chosen to forward data. For each segment (from an SN to the CH and from the CH to the PC), we use the transmission time of the first path that returns the query result to get the total response time. If SN measurement software faults are considered, the transmission time for all the SN-CH paths to return sensor readings to the CH is considered instead.

The simulation runs in rounds. In each round, we record the number of queries processed successfully, which is recorded as an instance of the system MTTF. We use the *batch mean analysis* technique to obtain MTTF, treating each MTTF obtained from a simulation run as a data point in order to obtain the average MTTF within a specified confidence interval and accuracy. We run the simulation until we archive 95 percent confidence level and 10 percent accuracy. To achieve this, we collect observations in batches with 1,000 observations in each batch. In one batch, we obtain a batch mean out of 1,000 observations collected. We run at least 10 batches to get a minimum of 10 batch means from which we calculate the grand mean and estimate the difference of the grand mean from the true mean with 95 percent confidence. If the accuracy obtained is greater than 10 percent, we run more batches and collect more observations until the specified 10 percent accuracy requirement is met. We run the simulation for the optimal (m_p, m_s) and other nonoptimal (m_p, m_s) values. The results are used to draw a 3D graph representing MTTF based on m_p and m_s against which analytical results are compared and validated. Below, we compare simulation results obtained with analytical results under identical parameter value sets.

Fig. 12 compares simulation results obtained versus analytical results for a query-based WSN operating under the set of parameter values listed in Table 3. We see that the simulation and analytical MTTF curves correlate very well, with the same optimal (m_p, m_s) at (2, 2). We have also conducted a simulation study that considers changes in the network conditions. Specifically, in addition to simulating transmission failure and transmission speed violation, we also simulate sensor node hardware failure and energy depletion. Fig. 13 compares simulation results versus analytical results when such network dynamics are considered. Again, the results show good correlation. Both simulation and analytical results confirm that in a WSN characterized by the set of parameter values in Table 3, the system can better tolerate sensor failures due to hardware or energy depletion when proper source and path redundancy are employed, especially at the optimal (m_p, m_s) identified.

Next, we conduct simulation experiments to determine the minimum clustering interval under which the assumption of fair rotation among SNs as the CH is justified and also to validate analytical results for the effect of clustering intervals on MTTF. The simulation results (bottom three curves) shown in Fig. 14 confirm that using a short clustering interval ($T_{clustering} = 5$ sec versus 20 sec versus 5 min) will result in a smaller MTTF since more energy would be consumed when the clustering algorithm is executed more often. The simulation results also reveal that the system can achieve a perfect rotation with $T_{clustering} = 5$ sec and near-perfect rotation with $T_{clustering} = 20$ sec under the given workload condition. The analytical results at $T_{clustering} = 20$ are also shown in Fig. 14 (top curve) which correlate well with simulation results. Consequently, we conclude that the assumption of fair rotation in the analytical model is justified.

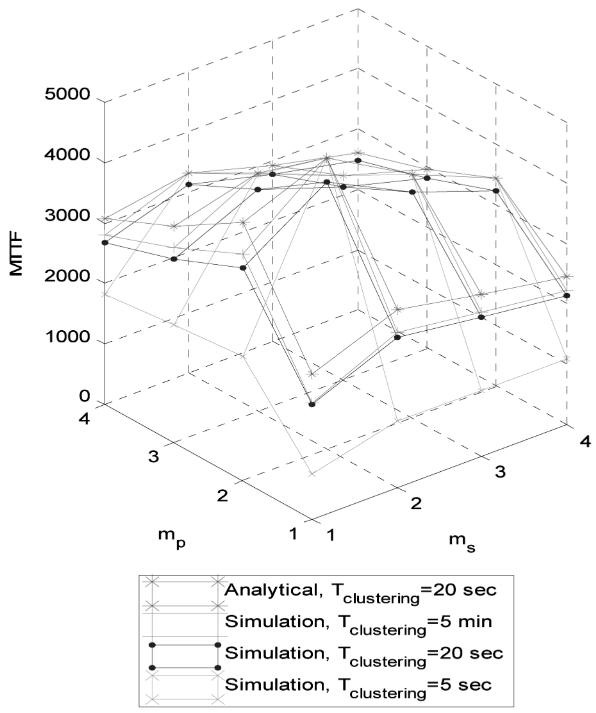

Fig. 14. Simulation results for the effect of clustering intervals on MTTF versus (m_p, m_s).

Finally, we have conducted simulation studies to compare the case when SNs are distributed according to a homogeneous Poisson process versus the case when SNs are distributed uniformly to test the sensitivity of simulation results with respect to SN population distribution. Fig. 15 shows that the simulation results are insensitive to these two types of distribution used with the mean percentage difference between them being only 0.69 percent.

7 Applicability and Future Work

In this paper, we have developed an adaptive fault-tolerant QoS control (AFTQC) algorithm which incorporates path and source redundancy mechanisms to satisfy query QoS requirements while maximizing the lifetime of query-based sensor networks. We discussed how these mechanisms can be realized using hop-by-hop packet data delivery and derived the probability of successful data delivery within a real-time constraint (R_q), as well as the amount of energy consumed (E_q) per query. When given a set of parameter values characterizing the operating and workload conditions of the environment, we identified the optimal (m_p, m_s) setting that would maximize the MTTF while satisfying the application QoS requirements.

To apply the results derived in this paper, one could build a table at design time listing MTTF as a function of (m_p, m_s) covering a perceivable set of parameter values. Dynamic parameter values such as e_j and distribution of S_{jk} can be predicted by using local measurements, or, alternatively, collected either proactively or reactively by the CHs at the expense of energy consumption. Then, a simple table lookup could be performed at runtime to determine the optimal (m_p, m_s) that could satisfy the QoS requirements and maximize the MTTF.

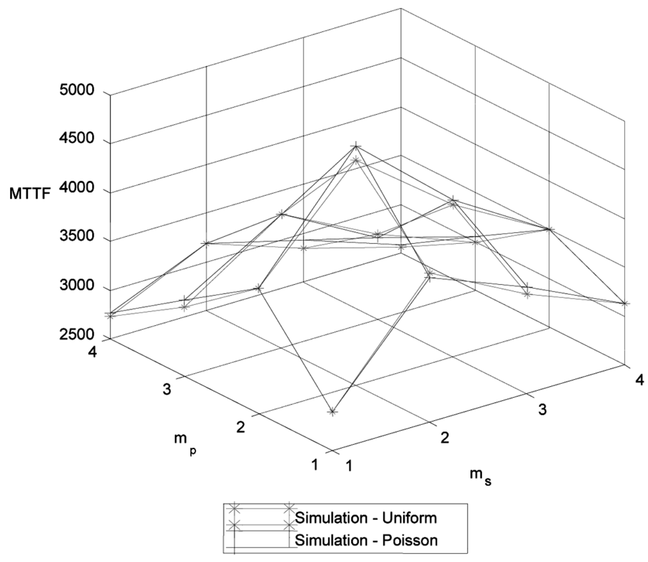

Fig. 15. Comparison of simulation results between Poisson and uniform distribution of SNs.

In the future, we plan to provide a more detailed analysis of the effect of network dynamics on MTTF, such as more energy may be consumed by some SNs over others or some SNs may fail earlier than others. This affects the number of SNs in a cluster as time progresses and makes several key parameters such as r, p, e_j, and S_{jk} as a function of time. Finally, we plan to consider the use of acknowledgment and timeout mechanisms in our hop-by-hop data delivery scheme at various levels, such as hop-by-hop or end-to-end, and identify the optimal (m_p, m_s) that minimizes MTTF, as well as conditions under which no-ACK is better than ACK-based data delivery schemes, or vice versa.

References

[1] O. Younis and S. Fahmy, "HEED: A Hybrid Energy Efficient, Distributed Clustering Approach for Ad Hoc Sensor Network," *IEEE Trans. Mobile Computing,* vol. 3, no. 3, pp. 366-379, Oct.-Dec. 2004.
[2] W. Heinzelman, C. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Comm.,* vol. 1, no. 4, pp. 660-670, Oct. 2002.
[3] P. Mhatre et al., "A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint," *IEEE Trans. Mobile Computing,* vol. 4, no. 1, pp. 4-15, Jan./Feb. 2005.
[4] D. Chen and P. Varshney, "QoS Support in Wireless Sensor Networks: A Survey," *Proc. Int'l. Conf. Wireless Networks,* pp. 21-24, June 2004.
[5] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocol for Self-Organization of a Wireless Sensor Network," *IEEE Personal Comm.,* pp. 16-27, Oct. 2000.
[6] B. Deb, S. Bhatnagar, and B. Nath, "ReInForM: Reliable Information Forwarding Using Multiple Paths in Sensor Networks," *Proc. 28th Ann. IEEE Conf. Local Computer Networks,* Oct. 2003.
[7] M. Perilo and W. Heinzelman, "Providing Application QoS through Intelligent Sensor Management," *Proc. First IEEE Int'l. Workshop Sensor Network Protocols and Applications,* May 2003.
[8] E. Felemban, C.G. Lee, and E. Ekici, "MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 6, pp. 738-754, June 2006.
[9] R. Iyer and L. Kleinrock, "QoS Control for Sensor Networks," *Proc. IEEE Conf. Comm.,* May 2003.

[10] P. Gupta and P.R. Kumar, "Critical Power for Asymptotic Connectivity in Wireless Networks," *Stochastic Analysis, Control, Optimizations, and Applications,* W.M. McEneaney, G. Yin, and Q. Zhang, eds., Birkhauser, 1998.

[11] S. Bandyopadhyay and E. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," *Proc. IEEE INFOCOM,* pp. 1713-1723, Apr. 2003.

[12] Y. Sankarasubramaniam, O.B. Akan, and I.F. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks," *Proc. Fourth ACM MobiHoc*, pp. 177-188, June 2003.

[13] J.A. Gutierrez, E.H. Callaway, Jr., and R.L. Barrett, Jr., *Low-Rate Wireless Personal Area Networks.* IEEE Press, 2004.

[14] O. Younis, S. Fahmy, and P. Santi, "Robust Communication for Sensor Networks in Hostile Environments," *Proc. 12th IEEE Int'l Workshop Quality of Service,* pp. 10-19, June 2004.

[15] G. Bravos and A. Kanatas, "Energy Consumption and Trade-Offs on Wireless Sensor Networks," *Proc. IEEE 16th Int'l Symp. Personal, Indoor and Mobile Radio Comm.,* vol. 2, pp. 1279-1283, Sept. 2005.

[16] Q. Shi, "Power Management in Networked Sensor Radios—A Network Energy Model," *Proc. IEEE Sensors Applications Symp.,* pp. 1-5, Feb. 2007.

[17] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A. Stateless Protocol for Real-Time Communication in Sensor Networks," *Proc. 23rd Int'l Conf. Distributed Computing Systems*, pp. 46-55, May 2003.

[18] V. Tippanagoudar, I. Mahgoub, and A. Badi, "Implementation of the Sensor-MAC Protocol for the JIST/SWANS Simulator," *Proc. IEEE/ACS Conf. Computer Systems and Applications,* pp. 225-232, May 2007.

Ing-Ray Chen received the BS degree from National Taiwan University, Taipei, and the MS and PhD degrees in computer science from the University of Houston. He is a program director and professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, security, pervasive computing, multimedia, distributed systems, real-time intelligent systems, and reliability and performance analysis. He has served on the program committee of numerous conferences in his research areas. He was an associate editor for the *IEEE Transactions on Knowledge and Data Engineering* during 2000-2004 and currently serves as an editor for *IEEE Communications Letters*, *IEEE Transactions on Network and Service Management*, *The Computer Journal*, *Wireless Communications and Mobile Computing*, *Wireless Personal Communications*, *Security and Communication Networks*, and the *International Journal on Artificial Intelligence Tools*. He is a member of the IEEE and the ACM.

Anh Phan Speer received the BS degree in telecommunications management and the MS degree in economics from Moscow Technical University of Communication and Informatics in 1997, and the MS and PhD degrees in computer science from Virginia Polytechnic Institute and State University (Virginia Tech) in 1999 and 2008, respectively. She is currently a senior development lead architect at Greystone CES in Tampa, Florida. Her research interests include wireless communications, wireless data management, sensor networks, fault tolerance, and mobile computing.

Mohamed Eltoweissy received the BS and MS degrees in computer science and automatic control from Alexandria University, Egypt, in 1986 and 1989, respectively, and the PhD degree in computer science from Old Dominion University in 1993. Dr. Eltoweissy is Chief Scientist for Cyber Security Research at Pacific Northwest National Laboratory. His research interests include the areas of information assurance and trust, networking and security in large-scale, ubiquitous cyberphysical systems, and group communications. His contributions include concern-oriented architecture and cell-based network model, dynamic key management in sensor networks, and elastic sensor-actuator networks. He has more than 130 publications in archival journals and respected books and conference proceedings. He is a senior member of the IEEE and a senior member of the ACM, the ACM SIGBED, and the ACM SIGSAC.