Quality of Service Analysis and Control for Wireless Sensor Networks

James Kay and Jeff Frolik
University of Vermont
jkay@uvm.edu, jfrolik@emba.uvm.edu

Abstract

This paper investigates wireless sensor network spatial resolution as a measurement of Quality of Service (QoS). We seek to control the network in such a way that sensors participate equally in the network while conserving energy and maintaining the desired spatial resolution. This work provides an analytic solution of a sensor network QoS control strategy demonstrated recently through simulation. General conclusions about the selection of parameters to control network performance, specifically the mean and variance of the QoS are presented. We show that there is a tradeoff between the static and dynamic QoS performance, as well as energy usage and conclude by presenting the several potential applications that are enabled by ability to control both the mean and variance of network QoS.

I. Introduction

Wireless sensor networks are rapidly increasing in size and complexity due to recent advances in radio frequency, computing and sensing technologies. These technologies are posed to enable applications utilizing hundreds or thousands of sensor nodes. Many of these applications will require the deployed network to operate autonomously, in remote or inaccessible environments, precluding maintenance and requiring the network to robustly compensate for node failure or addition. Examples include aircraft systems, remote applications, and hazardous area monitoring.

These types of applications favor system architectures that utilize many low cost, redundant sensing nodes so that the system can tolerate expected sensor losses over the fielded lifetime. The system decision to utilize many low cost rather than fewer higher cost sensing nodes typically results in nodes with significant reliability and energy constraints. In addition, the sheer number of the sensors may strain the bandwidth and computing resources of the network with redundant information resulting in sub-optimal performance. These systems benefit from control strategies in which the base station (or clusterhead) can dynamically adjust the number of active sensors to optimize the required system figures of merit at any given time.

Energy conservation, ad-hoc configuration, and information routing for these networks is an active area of research. However, this paper was motivated by recent work [1] that proposes to define wireless sensor network Quality of Service (*QoS*) in terms of how many of the deployed sensors are active. This measure is intuitively applicable to the class of problems described, where the expectation is that the number of sensors deployed exceeds the minimum number required for system functionality.

This definition of *QoS* requires some extensions if we are to use it as a criteria to support the goal of controlling the network in such a way that sensors participate equally in the network while conserving energy and maintaining spatial resolution. In particular, in this paper we propose adding an additional *diversity* measure which allows comparison of control strategies based upon how often any given node becomes active. Minimization of *turn-time* will be important in networks that require some level of node latency control. A low *turn-time* also ensures that spatial resolution is maintained and the sensor nodes are not *trapped* in inactive states for long periods of time.

This paper presents an analytic solution for a control strategy demonstrated recently through simulation [2]. Our analytic solution allows one to draw general conclusions about the selection of parameters to control network performance, which are then verified by simulation. We show that this strategy allows a tradeoff between the static and dynamic *QoS* performance, as well as energy usage, that can be used to tailor the network for the specific application. We also show that the protocol handles the dynamic addition and reduction of sensors robustly. We conclude by presenting several applications that are enabled by this strategy's ability to control both the mean and variance of the network *QoS*.

The remainder of this paper is organized as follows: Section II presents the control architecture investigated in this paper and discusses previous results. Section III presents an analysis of this architecture using Markov processes. A direct formulation of the problem is presented first, followed by a formulation that significantly reduces the dimensionality of the problem. Section IV presents the key results of using this analysis, and simulation. We conclude in Section V with proposed extensions to this work.

II. Sensor Automaton and Control Strategies

This work was motivated by two recently presented *QoS* control strategies. Again, we consider *QoS* as being defined in terms of average spatial resolution or equivalently the number of sensors active in a randomly deployed network. Specifically we consider the Gur strategy proposed in [1,3] and the ACK strategy proposed in [2]. Both of these strategies control *QoS* for a single-hop cluster. In addition, both strategies have sensors that use finite state automata. We will briefly discuss both strategies to give the reader an appreciation for the differences and similarities of these techniques. This will be followed by a detailed analysis of the ACK strategy.

In the following description we will denote the number of sensor nodes by N, and the number of possible node automaton states by G. We distinguish between the state of each automata and the state of the entire network by using the terms *automata state* and *system state*.

In both techniques one sensor is active as the clusterhead node. This sensor receives information from all other sensors in the network. It is assumed that all sensors have sufficient transmission strength to cover the entire area (i.e. cluster). These techniques are applicable to a wide range of networks including band limited applications suitable for low-cost monitoring (e.g. environmental), as well as high bandwidth applications (e.g. target tracking). This makes this technique compatible with various communication protocols including low cost ALOHA, and carrier sense multiple access (CSMA) methods.

A. Gur Game

In the Gur strategy the remaining sensors are in either a STANDBY or ON state as shown in Fig. 1. Sensors in the ON state are active in sending data to the clusterhead and counted to determine the network *QoS*. Sensors in the STANDBY state do not send data.

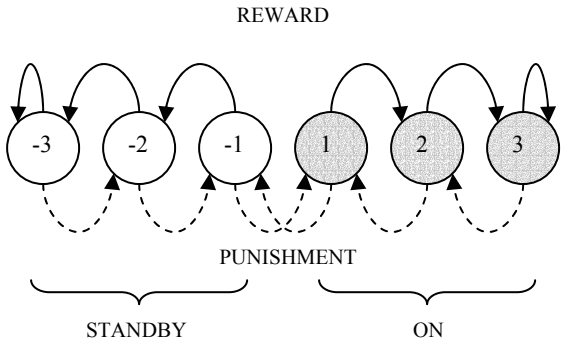

Figure 1. Gur game automaton.

Sensors move from one state to another based on being rewarded for their behavior (ON or STANDBY). Each sensor determines whether to reward or punish itself based upon a threshold, R, between 0 and 1 received from the clusterhead. Each node locally generates a random number, between 0 and 1, and rewards itself if this number is less than R, otherwise the node punishes itself. The value of R is calculated once each epoch by the clusterhead and broadcasted to ***all*** sensors*, including those* in the STANDBY state.

The threshold R is calculated from a function whose domain is the *QoS* calculated above and whose range is 0 to 1. [1,3] have shown that for this architecture the mean of the resulting *QoS* will tend to the maximum of this reward function for a very broad class of functions, allowing control of the network *QoS*. However, expressions for second and higher order statistics for this technique have not been calculated.

The Gur control strategy has two parameters that can be used to control the network performance, the number of automata states, and the particular form of the reward function. [1,3] have shown that 3 ON and 3 STANDBY states provide adequate control. We are unaware of work on selection of the reward function to optimize the dynamics (e.g. settling time, stability, and variance) of the *QoS*.

B. ACK Automaton.

The ACK protocol was developed as an alternative to the Gur game [2]. In this technique, rather than automata states being ON or STANDBY they are in varying states of being ON. This is accomplished by assigning a transmit probability T_i to each of the automata states. During each epoch, each sensor independently generates a random

number between 0 and 1. If this number is less than T_i, the node transmits, otherwise it is silent. The clusterhead acknowledges each transmitting sensor individually and provides a single bit of coupling information indicating whether the current *QoS* is greater than the desired value, Q_0. Sensors transition between states based on this single bit of information. They punish themselves if the *QoS* is greater than the desired value by transitioning toward the left in Fig. 2. Conversely they reward themselves by moving right if the QoS is less than or equal to the desired value. Note that in this technique sensors that do not transmit can be in a very low power state since they *do not need to listen* for control information from the clusterhead.

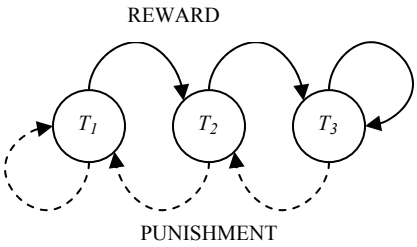

Figure 2. ACK feedback automaton

C. Comparison of Strategies

Comparison of these two techniques shows some similarity. They both rely on automata to provide the *QoS* control, and both use a global network parameter to provide coupling between the automata. They both also provide parameters that can be used to control the static and dynamic performance of the network. In the case of the Gur game these consist of the size of the automata, and the selection of the reward function. Although it is possible to determine network performance for particular selections of the reward function, variational methods to determine an optimal function have not been developed. In the ACK protocol the performance is determined by a finite set of continuous parameters, the automata state transmit probabilities T_i. Table 1 summarizes the parameters that can be varied for these methods.

Review of the previous work on these techniques [2] indicates several benefits to the ACK strategy:

- Network lifetime is extended versus the Gur strategy, Fig 3.
- A-priori knowledge of the sensor lifetime or specific reward function is not required.

Limitations of the Gur strategy include:

- The number of ON sensors is limited to 50% of the available sensors due to the structure of the Gur automata
- All sensors must receive control information each epoch. This is a significant energy penalty since receiver energy costs are often nearly equivalent to transmit energy costs.

Table 1: Comparison of degrees of control parameters for the GUR and ACK strategies.

Technique	Description	Quantity
Gur Game	Reward function	1
	Number of automata states	G
ACK	Desired *QoS*	1
	Transmit probabilities for automata states	*G*

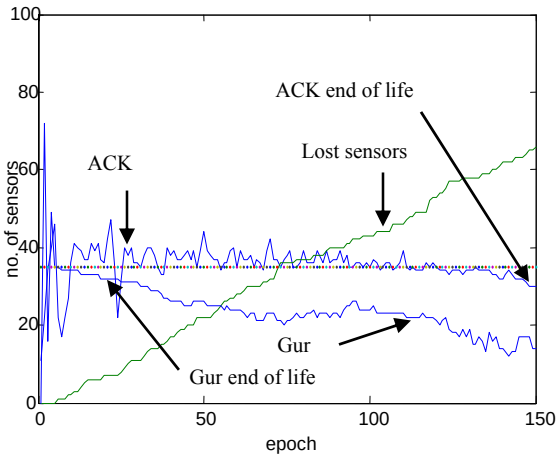

Figure 3: *QoS* performance vs. epoch for the ACK and Gur strategies. Network life is extended more than 5x using the ACK scheme[3]

The ACK technique admits analysis, which along with the benefits discussed motivated the analytic work in this paper. This analysis begins to develop the tools needed to determine control parameter values that provide desired network performance.

III. Analysis of ACK Strategy

As noted each of the *N* sensor nodes is modeled as a finite state automaton (Fig. 2). Each automata state is assigned a transmit probability, T_i, where *i* is the index of the automata state that can take on values

from 1 to G. Therefore, in any epoch a sensor node in automata state i has probability T_i of transmitting.

A. State Transition Rules

The N sensors in the network receive an acknowledgement (ACK) when they transmit. This single bit of information provides the ***only*** coupling of the global network state to the individual automata state. In the proposed *QoS* control strategy transitions between automata states are controlled by the following rules:

Rule 1: A transmitting node goes to the next higher automata state if QoS$\leq Q_0$. If a transmitting node is already in the highest automata state, it remains in the highest state.
Rule 2: A transmitting node goes to the next lower automata state if QoS$>Q_0$. If a transmitting node is already in the lowest automata state, it remains in the lowest state.
Rule 3: Non-transmitting nodes do not change automata states.

These rules along with the automata structure allow modeling the network as a Markov process for which steady state and dynamic solution techniques are well known[4,5]. Note that the transition probabilities are functions of the global network parameter Q_0 which prevents modeling each node independently of the others.

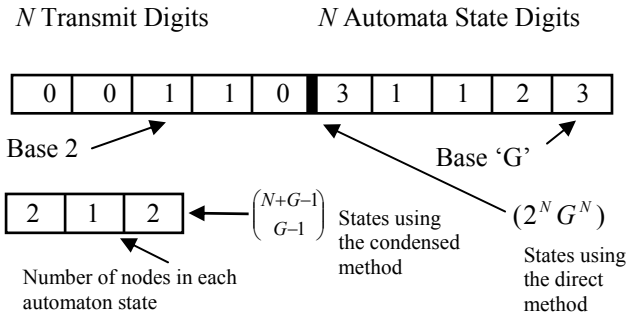

Figure 4: State enumeration, top is direct method, bottom is condensed method. The example shown is for 5 nodes, each with 3 automaton states.

B. Direct Markov Modeling of System

The assignment of the Markov states for this formulation is shown in the top half of Fig. 4. The states are enumerated by N transmit digits and N automata state digits. The transmit digits will always be binary digits, while the base of the automata state digits will be equal to G, the number of automata states. This assignment results in $2^N G^N$ states. Examination of this state assignment shows that every combination of automata state and network state can be represented. For example, Fig. 4 shows the case when we have five nodes (N=5) and 3 automaton states (G=3). In the specific network state shown nodes 3 and 4 are transmitting, and the other nodes are not. In addition we see that nodes 1 and 5 are in automaton state 3, nodes 2 and 3 are in automaton state 1, and node 4 is in automaton state 2.

This state enumeration provides knowledge of the network *QoS* (the sum of the Transmit Digits) as well as the automaton state of each node. It is straightforward to calculate the probability of transitioning from any state, to any other state using the following rules. We use S_j to designate the current network state (i.e. the combination of the transmit digits and the automaton digits) and S_i to designate the next state of the system. Using this notation the Markov state transition probabilities between any two states can be calculated using the following two step process:

Step 1: Determine which transitions have non-zero probabilities.

i. If a node's transmit digit is 0, then it remains in its current automaton state, since it will not receive any ACK information from the BASE. Any network state transition for which this is not true has 0 probability.
ii. If a node's transmit digit is 1, the automaton state for that node is increased by 1 if the sum of the transmit digits in state $S_i \leq Q_0$ (reward), and decreased by 1 if $S_i > Q_0$ (punish). Exceptions to this rule occur at the end states (i.e. when the automaton state is either 1 or G). If a node is in the 1 state and is *punished*, it remains in the 1 state. Similarly if a node is in automaton state G and is *rewarded* it remains in state G. Any network state transition for which this rule is not followed has 0 probability.

Step 2: Determine the transition probabilities for the non-zero cases.

i. The transition probability between S_j and S_i is a product consisting of N terms, one for each node. To determine the term corresponding to a node first identify for state S_i, the automaton state, k for that node, and the transmit state for that node. If the transmit state for a node is 1, then choose the transmit probability, T_k for the product term for that node. If

the transmit state for a node is 0, choose $1\text{-}T_k$ for the product term.

S_j=	1	1	0	0	0	1	2	1	1	1

S_i=	1	0	0	1	1	2	3	1	1	1

Figure 5: Network states S_i and S_j, for N=5, G=3, Q_0=3

To illustrate the calculation of transitional probabilities consider the example of Fig. 5 where we assume Q_0=3. In this case, the transition from S_j to S_i will be a reward transition, since the QoS of S_j is 2 which is less than or equal to Q_0. We also apply step 1 and determine that the transition probability is not 0. We then form the expression for the transition probability between states S_j and S_i which we denote P_{ji} as:

$$P_{ji}=T_2Q_3Q_1T_1T_1 \qquad (1)$$

Where $Q_3=1\text{-}T_3$, $Q_1=1\text{-}T_1$.

Once the transition probabilities between each state are determined the solution for the probability of the network being in any state S_i, which we denote $P(S_i)$ can be determined using standard Markov process solution techniques, e.g. [4,5]. Since the QoS of each S_i is known we can then determine the probability distribution function for the random variable QoS by weighting each value of QoS by the sum of the state probabilities for all states with that value of QoS.

The analysis above is valid when $Q_0<N$, and when all the $T_i>0$, in which case the process is completely ergodic and therefore has limiting state probabilities [4], and can be solved analytically for the probability density function (pdf) of QoS for each value of N and G. Knowledge of the pdf allows the calculation of any statistical moment for QoS. Results for N=2, G=2 and Q_0=1 are given in (2), where we use a normalizing factor, D, to simplify the expression.

$$P(QoS=0)=(2T_1-2T_1^2-6T_1T_2+6T_1^2T_2+3T_2^2 -3T_1^2T_2^2-2T_2^3+2T_1T_2^3)/D \qquad (2a)$$

$$P(QoS=1)=(4T_1T_2-4T_1^2T_2-4T_1T_2^2+4T_1^2T_2^2 +2T_2^3-2T_1T_2^3)/D \qquad (2b)$$

$$P(QoS=2)=(2T_1T_2^2-T_1^2T_2^2)/D \qquad (2c)$$

$$where\ \ D=2T_1^2(T_2-1)+3T_2^2-2T_1(T_2^2+T_2-1)$$

The results for the mean and variance of QoS for this case are shown in Fig. 6. In the top figure the height of the surface is the expected value of QoS. We have highlighted that region for which $|E(QoS)-Q_0|<0.01$ by setting the surface height to 0 in this case (as indicated by the arrow). We also plot the variance of QoS as a function of T_1 and T_2 along this highlighted region, in the lower half of Fig. 6.

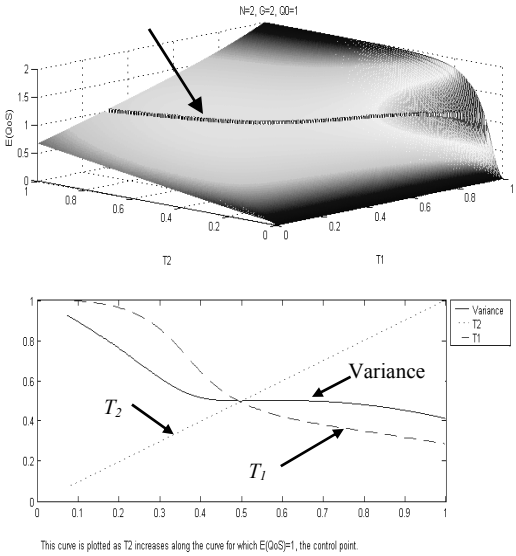

Figure 6: Results for N=2 G=2, Q_0=1

Observations from Fig. 6 are:

1. When $T_1=T_2$=0.5 the expected value is exactly 1, and the variance is exactly 0.5. In this case, G really is reduced from 2 to 1 since the probability of transmit does not depend on the automaton state of that node. In this case we have a Bernoulli process [2] and we may write the mean and variance as:

$$E(QoS)=NT=2\times 0.5=1$$
$$Var(QoS)=NT(1-T)=2\times 0.5(1-0.5)=0.5 \qquad (3)$$

This result is consistent with Fig. 6.

2. Along the curve for which $E(QoS)=Q_0$=1 the minimum variance occurs when T_2=1, $T_1=\frac{1}{2}\left(2-\sqrt{2}\right)$, and has value $\sqrt{2}-1\approx 0.414$ which is less than the case of G=1 for which the variance is 0.5 Results will be presented later in this paper showing additional control of the variance as we increase G.

Although we now have an analytic solution for the network, its usefulness is limited. It suffers from the curse of dimensionality, since the number of states

grows as $2^N G^N$. Even for a simple network consisting of two nodes with two automata states we must solve a set of sixteen linear equations.

C. Reducing Analytic Complexity

This rapid state growth in the formulation above provides motivation for reformulating the problem in a way that significantly reduces the number of states. Our approach is to rewrite the system equations in terms of conditional probabilities. In the previous formulation we noted that there are G^N distinct automaton states in the Markov process. However, due to symmetry, network states with the equal numbers of nodes in each automaton state will have the same limiting state probabilities and can be represented as a single state. This results in

$$C = \begin{pmatrix} N+G-1 \\ G-1 \end{pmatrix}$$

unique automaton states (see Fig. 4). A comparison of the dimensionality for the two formulations is shown in Table 2, and provides motivation for condensing the states.

Table 2: Comparison of Markov states for the direct and condensed formulations

N	G	Markov States	
		Direct	Condensed
2	2	16	3
3	3	216	10
4	3	1296	15
7	3	2.8E3	36
8	3	1.67E6	45
20	3	3.6E15	231

We represent these state probabilities by $P(S_i)$ where the subscript takes on C values. These state probabilities obey the following equation:

$$P(S_i) = \sum_j P(S_i^+ / S_j) P(S_j) \qquad (4)$$

where:

S_i=The event that the network is in automaton state i. We order the G digits of this network state from 1 to G, with the G'th digit containing the number of network nodes in the highest reward state, G-1 the next highest reward state, down to 1, which is the lowest, most punished state. The bottom half of Fig. 4 illustrates the case of 5 nodes, with 2 in the lowest state, 1 in the second state, and 2 in the highest state.

S_i^+ =The event that the network transitions to network state S_i.

In equation (4) the $P(S_i)$ are unknowns. The remaining conditional probabilities, $P(S_i^+ / S_j)$'s, are known given the transition rules presented earlier. It is convenient to define an activity measure between automaton states to facilitate writing an analytic expression for the $P(S_i^+ / S_j)$'s. We define:

$A_k(S_i,S_j)$ = The number of nodes in the k'th digit of the state vector S_j that must transmit in order for the network to transition to state S_i.

The domain of $A_k(S_i,S_j)$ is defined to be those ordered pairs of states, (S_i,S_j) which are consistent with the state transition rules presented earlier, and for which $i \neq j$. Those transitions that do not obey these rules have 0 probability of occurrence, and we treat the case of $i=j$ as a special case. The domain of A_k can be further divided into pairs representing *reward*, and *punish* transitions. We designate these subdomains as D^r, D^p respectively. Due to the hard limiting of rewarding and punishing at the G'th and 1'st automaton states we leave $A_k(S_i,S_j)$ undefined when $k=G$ for *reward* states and $k=1$ for *punish* states. We provide an example showing S_i, S_j, and A_k for a *reward* transition in Fig. 7, for a network with 6 automata states. As discussed above A_k is not defined for $k=6$ in this case.

$k=$	1	2	3	4	5	6
$S_j=$	2	1	2	4	1	0
$S_i=$	1	1	3	3	1	1
$A_k=$	1	1	2	1	1	X

Figure 7: Example calculation for A_k

Using these definitions we may rewrite (4) as:

$$P(S_i) = \sum_{(S_i,S_j)\in D^r} P(S_i^+ / S_j) P(S_j) + \sum_{(S_i,S_j)\in D^p} P(S_i^+ / S_j) P(S_j) + P(S_i^+ / S_i) P(S_i) \qquad (5)$$

This partitioning allows modeling each of the conditional probabilities in (4) as a set of independent Bernoulli processes. This independence allows us to express these probabilities as the product of the probabilities of the independent processes. We

define the following notation for the Bernoulli pdf and cumulative distribution function (cdf):

$$b(w;n,p)=\binom{n}{w}p^{w}(1-p)^{n-w}$$

= the probability of w successes in n independent Bernoulli trials with success probability p. If n=0 we define $b(w;n,p)=1$ when $w=0$, and 0 elsewhere. If w<0 we define $b(w;n,p)=0$.

$$B(w;n,p)=\sum_{i=0}^{w}b(i;n,p)$$

= the probability of w or fewer successes in n independent Bernoulli trials with success probability p. If w<0 we define B(w;n,p)=0.

In addition we define:

S_i^m = the value of the m'th digit of the state S_i.

Referring to the lower half of Fig. 4, we see:

1. Digits can range in value from 0 to N.
2. There are G digits in each network state S_i,.
3. The value of m can range from 1 to G.
4. $\sum_{m=1}^{G}S_i^m=N$ for each value of i

We now formulate the conditional probabilities as:

If $(S_i,S_j)\in D^p$ (punish case):

$$P(S_i^+/S_j)=\{1-B(Q_0-\sum_{k=2}^{G}A_k(S_i,S_j);S_j^1,T_1)\}$$

$$\times\prod_{m=2}^{G}b(A_m(S_i,S_j);S_j^m,T_m) \qquad (6)$$

If $(S_i,S_j)\in D^r$ (reward case):

$$P(S_i^+/S_j)=B(Q_0-\sum_{k=1}^{G-1}A^k(S_i,S_j);S_j^G,T_G)$$

$$\times\prod_{m=1}^{G-1}b(A_m(S_i,S_j);S_j^m,T_m) \qquad (7)$$

If i=j (special case):

$$P(S_i^+/S_j)=\{1-B(Q_0;S_i^1,T_1)\}\times\prod_{m=2}^{G}b(0;S_i^m,T_m)$$

$$+B(Q_0;S_i^G,T_G)\times\prod_{m=1}^{G-1}b(0;S_i^m,T_m) \qquad (8)$$

(6-8) are explained by reference to the activity measure, A_k. Consider the reward case (7), (i.e. $(S_i,S_j)\in D^r$). We wish to determine the conditional probability of transition from state S_j to S_i. For this transition to occur, we require a specific number of the total number of sensors in each of the G automaton states to transmit. The number required to transmit in the k'th automaton state is $A_k(S_i,S_j)$. Since the total number of sensors in the k'th automaton is S_j^k, the probability that $A_k(S_i,S_j)$ of them will transmit is $b(A_k(S_i,S_j);S_j^k,T_k)$.

For a given S_i, the number of sensors transmitting in each automaton state is only a function of the number of sensors in that state. This independence allows determination of the total probability of transition as the product of the individual probabilities, which is represented by the product term of (7). A special case occurs at automaton state G for the reward case and state 1 for the punish case. The number of sensors transmitting in these limiting (i.e. k=1 or G) states is not unique for all transitions, which is why A_k is left undefined for these cases. In the case of a reward transition the number transmitting in the G'th automaton state, plus the sum of the sensors transmitting in the other states must be less than or equal to Q_0. The probability of this occurring is: $B(Q_0-\sum_{k=1}^{G-1}A^k(S_i,S_j);S_j^G,T_G)$.

In the case where i=j we require that only sensors in the limiting states can transmit. The first term in (8) accounts for the punish case, where only sensors in state 1 are allowed to transmit. The second term accounts for the reward case, where only sensors in state G are allowed to transmit.

We now have a set of C linear equations for the C unknown $P(S_i)$'s. These determine the $P(S_i)$'s to a multiplicative constant [4]. We add the additional constraint that the state probabilities must sum to 1, i.e.

$$\sum_i P(S_i)=1$$

We now derive the probability density function for QoS given these $P(S_i)$'s. This pdf can then be used to calculate desired QoS statistics, in particular the mean and variance.

E. Calculation of the Probability Distribution Function for QoS

Given the $P(S_i)$ we may write the pdf of QoS as a sum of conditional probabilities as follows:

$$P(QoS=x)=\sum_{S_i} P(QoS=x/S_i)P(S_i) \qquad (9)$$

We now define:

$f_i^k(x) = b(x; S_i^k, T_k)$

= the probability that x of the nodes in automaton state k of network state S_i are transmitting.

The f defined above are independent of each other and can each be viewed as the pdf of a random variable x. The sum of these random variables for a given network state is the *QoS* for that network state. The pdf of the sum of independent random variables is the convolution of their pdf's [2]. Therefore the pdf of *QoS* may be written as the discrete convolution $(*)$ of these independent pdf's giving:

$$P(QoS = x/S_i) = f_i^1 * f_i^2 * f_i^3 ... f_i^G \qquad (10)$$

Given we can determine the pdf of *QoS*, *any* statistical moment can be calculated, in particular the mean and variance.

IV. Key Results

Investigations of our analysis have yielded several key results.

Result 1. The mean and variance of QoS can be controlled by selection of the transmit probabilities.

An example is shown in Fig. 8 for the case when N=5, G=3, Q_0=3. In this example we have set T_3=1 and allowed T_2 and T_1 to vary. In the left hand plot we have highlighted the region for which |E(*QoS*)-Q_0|<0.03 in black. There are actually two curves for which this condition holds, the first is the easily seen *funnel* shaped curve, the second one is a narrow strip that occurs when T_1 approaches 0 (as indicated by the arrow). Note that the funnel mouth widens as T_2 approaches 1. This feature allows control of the variance while maintaining a mean near the control point. We illustrate this in the right hand side of Fig. 8, where we fix T_2 at 0.8, and keep T_3=1.0, while we vary T_1 to control the variance. By allowing T_1 to vary from 0 to 0.45 we can choose a variance between 0 and 1.2. This ability to control the variance while keeping the mean relatively constant was observed in all cases analyzed. In the case of G=3 the range over which the variance can be controlled can be set by appropriate selection of T_2, and the operating point for the variance controlled by selecting T_1. This allows tuning the network performance to the application. Wider ranges of mean and variance control are possible as we increase the network size, N, and the number of automaton states, G.

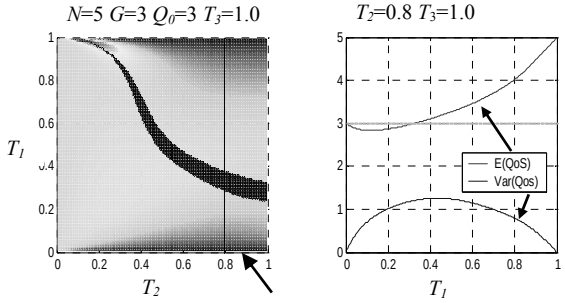

Figure 8. Mean and variance as a function of the transmit probabilities.

Result 2. There is a tradeoff between variance and network diversity.

We define *diversity* loosely as equality of participation among sensors. We will develop this concept by briefly revisiting our analysis, followed by the introduction of a more formal diversity measure we name *turn time*.

One of the results of condensing the number of Markov states is that we lose individual node information, while retaining network level information. Thus, our solution allows calculation of all statistics for the *QoS*, but does not allow us to determine the specific performance of any individual node. The result is that systems designed using the *QoS* analysis alone may behave in undesired ways.

This is illustrated by using the example of N=5, G=3, and Q_0=3 presented in Fig. 8. Let's assume our task is to design the system with the minimum variance in the *QoS*, while maintaining the mean close to Q_0=3. Using Fig. 8, we decide to select T_1=0, T_2=0.8, and T_3=1.0. For this selection we see that the mean is 3.0 and the variance is 0. We met our goals perfectly! The issue is that the network has a low *diversity*. For this particular selection of T_i's we can analyze the solution in closed form and see that in this case 3 of the nodes will eventually reach the highest automaton state and 2 will eventually reach the lowest automaton state. Once the network reaches this configuration no additional automaton state transitions will occur thus each sensor is *trapped* into its final state and some will not transmit. The specific sensors trapped in this lowest state will depend upon the starting state of the network, and how it evolves based on the stochastic rules presented earlier. Note that this trapping behavior occurs anytime we set the lowest

automaton's transmit probability to 0. The problem arises should one of the transmitting nodes fail, thus causing the *QoS* to drop to 2 without possibility of achieving Q_0 subsequently. (Note that our original analysis specifically excluded this case, although it is valid for values arbitrarily close to 0).

The discussion above begs the question of how to trade off variance for diversity in this *QoS* control strategy. We propose introducing a metric for diversity measure, *turn time,* defined as the time, measured in network epochs, that it takes a sensor in the network to transition from the lowest automaton state to the highest automaton state. *Turn time* being a random variable can be described by its statistics. However, the condensed problem formulation trades off dimensionality for granularity, and cannot distinguish individual sensor performance. Therefore simulation of the network is used to obtain *turn time* statistics.

Fig. 9 and Fig. 10 illustrate the tradeoff between variance and turn time. The two cases are identical except for the value of T_1 which is 0.1 in Fig. 9, and 0.01 in Fig. 10. Both cases maintain the mean of *QoS* near the desired control point of Q_0=3. The variance in Fig. 9 is ~10 times greater than that in Fig. 10, but is still relatively small (~5% of the mean). However, the two networks have widely different *turn times*. Note that the scale on Fig. 10 is multiplied by 10^5. The simulation for Fig. 10 was run with 10^6 epochs, and still only had a total of 6 turns in the entire simulation. The network of Fig. 10 would respond extremely slowly to sensor failures.

We conclude discussion of this result by reiterating that *trapping* states can be used to minimize variance, but produce low diversity. An additional example of this is shown in Fig. 11, where we show the mean and variance of *QoS* as a function of the number of sensor nodes. This system was simulated using T_1=0.001, T_2=0.5, T_3=1.0 and T_4=1.0, and Q_0=35. The system *QoS* tracks the number of deployed sensors until $N=Q_0$ at which point the system controls the mean at Q_0, with very little variance. This demonstrates the ability of this control method to robustly respond to the addition or loss of sensors, since the expected value of Q is maintained close to the desired value Q_0=35 over a wide range of N.

However, as demonstrated, a system such as Fig. 11 has slow response to sensor losses since the value T_1 is very small. It may be possible to extend this algorithm to use time varying values of the transition probabilities. For example, we may set T_1=0.001 as in Fig. 11, but allow it to vary to a higher value (e.g. 0.1) periodically (e.g. every 50 epochs). This is a subject of future work.

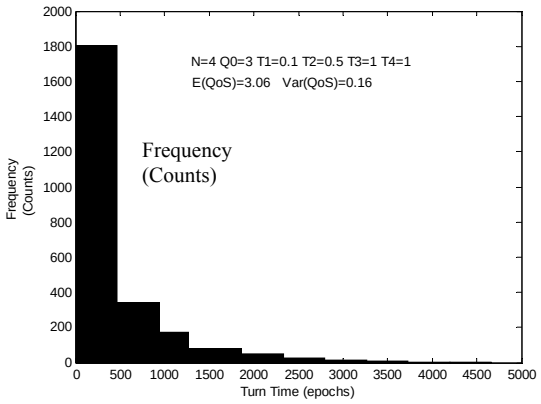

Figure 9: Turn time for N=4, Q_0=3, T_1=0.1, T_2=0.5, T_3=1, T_4=1

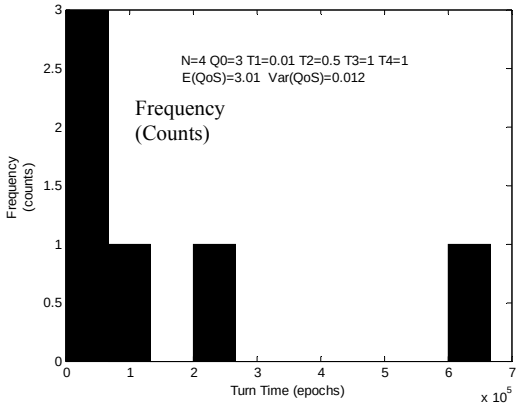

Figure 10: Turn time for N=4, Q_0=3, T_1=0.01, T_2=0.5, T_3=1, T_4=1

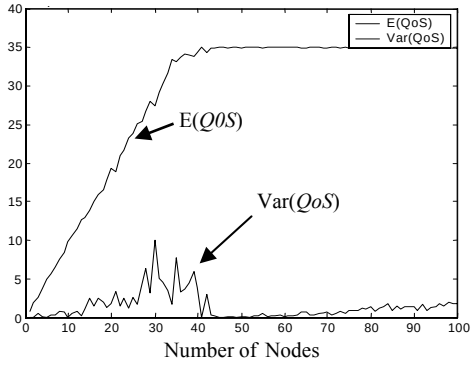

Figure 11: Reduction of variance using trapping states

Result 3. Network Health Monitoring

Up to this point we have been trying to reduce the variance of *QoS*, while maintaining the desired mean, and minimizing the *turn time*. However, variance can be used to estimate N for applications that can

tolerate a larger range of variance. This is one means of performing built in test (*BIT*) in energy constrained systems without having to individually talk to each sensor. This technique has both energy and bandwidth conservation benefits. Fig. 12 shows an example for this application with T_1=0.01, T_2=0.5 and T_3=1, note that this case differs from the Fig. 11 case in that T_1 is ten times larger, and there is one less automaton state (i.e. G=3 versus *G=4*). These changes cause a larger range in the variance, which varies from from approximately 0 to 40, as the number of sensor nodes varies from 0 to 300. Notice that the desired *QoS* is 35, so only a small fraction of sensors are active in any epoch when *N* is large, and yet we are able to infer *N*, the number of deployed sensors from the measured variance! We could use this information to send a request for replacement sensors, or for modifying our value of Q_0, to prolong the network lifetime, with some reduction in performance.

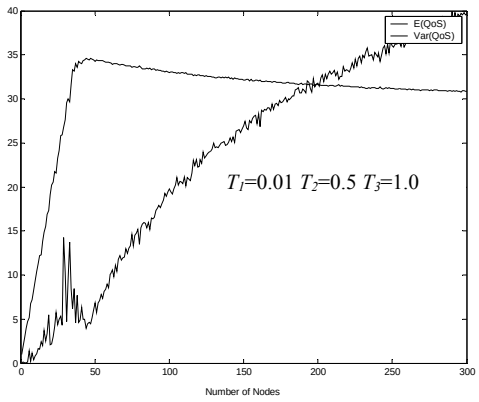

Figure 12: Inferring sensor count from Var*(QoS)*

V. Conclusions and Future Work

The results to date are promising for use of the ACK control method in a wide variety of sensor network applications. The control algorithm requires very little processing at both the sensor nodes and the clusterheads. A timer and the ability to generate a random number are the major function requirements for implementing this protocol. This allows this technique to be implemented in low cost networks. In addition, this method performs well in terms of energy conservation, since only the active sensors need to communicate with the base.

We see these characteristics as beneficial in systems that require dynamic control of the number of active sensors while minimizing energy usage. Examples include target tracking systems, structural health monitoring systems and remote sensing systems where the number of active sensors may need to be increased or decreased based upon events (e.g. target acquisition, structural load exceedance, rapid temperature increase).

The high degree of control over the mean and variance of *QoS* with this method motivates future work on evolving the analysis to provide tools for network optimization. Solutions of the transient behavior of this technique are also ripe for analysis, and will be needed in order to optimize the system performance. Further work on using variance for health monitoring is also intriguing.

REFERENCES

[1] Iyer, R. and L. Kleinrock, "QoS Control for Sensor Networks," presented at the IEEE International Communications Conference (ICC 2003), Anchorage, AK, May 11-15.

[2] Frolik, J. "QoS Control for wireless sensor networks", presented at the 2003 Wireless Communication and networking conference (WCNC), Atlanta, GA, March 21-25.

[3] Chung Y. *"Distributed control using finite state automata"*, PhD thesis, Univ. of Calif. Los Angeles, 1994.

[4] Drake, A. W.: "Fundamentals of applied probability theory", McGraw-Hill, New York, New York, 1967.

[5] Howard, R. A.: *"Dynamic Programming and Markov Processes,"* The M.I.T. Press, Cambridge, Mass., 1960.