QoS Control For Sensor Networks

Ranjit Iyer and Leonard Kleinrock
UCLA Computer Science Department
4732 Boelter Hall
Los Angeles, CA 90095-1596

***Abstract*—Sensor networks are distributed networks made up of small sensing devices equipped with processors, memory, and short-range wireless communication. They differ from conventional computer networks in that they have severe energy constraints, redundant low-rate data, and a plethora of information flows. Many aspects of sensor networks, such as routing, preservation of battery power, adaptive self-configuration, etc., have already been studied in previous papers, e.g., [1, 2, 3]. However, to the best knowledge of the authors, the area of sensor network quality of service (QoS) remains largely open. This is a rich area because sensor deaths and sensor replenishments make it difficult to specify the optimum number of sensors (this being the service quality that we address in this paper) that should be sending information at any given time. In this paper we present an amalgamation of QoS feedback and sensor networks. We use the idea of allowing the base station to communicate QoS information to each of the sensors using a broadcast channel and we use the mathematical paradigm of the Gur Game to dynamically adjust to the optimum number of sensors. The result is a robust sensor network that allows the base station to dynamically adjust the resolution of the QoS it receives from the sensors, depending on varying circumstances.**

I. INTRODUCTION

Sensor networks of the future are envisioned as thousands or more of inexpensive wireless nodes. Operating unattended, each of these sensors will be equipped with some computational power and sensing ability (e.g., sonar, radar, seismic, etc.). They are intended for surveillance applications such as those found in the military, environment, and deep space [4, 5, 6, 7]. The hardware technologies for these networks—low-cost processors, miniature sensing and radio modules—are available today, with future improvements in cost and capabilities expected within this decade. Wireless sensor networks improve sensing accuracy by providing distributed processing of vast quantities of sensing information. When networked, sensors can aggregate such data to provide a more complete view of the environment. Sensor networks can also focus their attention on "important events" detected by other sensors in the network (e.g., a person walking). Sensor networks are robust in that they can continue to provide information despite the failure of individual sensors. Sensors are envisioned to be interchangeable, and a significant amount of the data gathered will tend to be redundant [1].

Sensor networks are very different from conventional computer networks. First, because sensors have a limited supply of energy, energy-conserving forms of communication and computation are essential to wireless sensor networks. Second, since sensors have limited computing power, they may not be able to run sophisticated network protocols. Third, since the bandwidth of wireless links connecting sensor nodes is often limited, inter-sensor communication is further constrained [1].

Although the study of wireless sensor networks is still a burgeoning field, many aspects of sensor networks, such as routing, preservation of battery power, adaptive self-configuration, etc., have already been studied in previous papers, e.g., [1, 2, 3]. In this paper, however, we explore the area of sensor network quality of service (QoS). This is a rich area because sensor deaths (e.g., as a result of battery failure) and sensor replenishments (e.g., more sensors being dropped from an airplane or recharging of batteries) make it difficult to control the optimum number of sensors that should be sending information at any given time.

In this paper we present an amalgamation of QoS feedback and sensor networks. We use the idea of allowing the base station to communicate QoS information to each of the sensors using a broadcast channel and we use the mathematical paradigm of the Gur Game to dynamically adjust to the optimum number of sensors. The result is a robust sensor network that allows the base station to dynamically adjust the number of sensors being activated, thereby controlling the resolution of QoS it is receives from the sensors, depending on varying circumstances.

The remainder of this paper is organized as follows: Section II presents the problem description, and Section III surveys some of the previous work in wireless sensor networks. In Section IV we present a description of the Gur Game paradigm and Section V describes the network model that we use and the QoS control algorithm for sensor networks. Section VI shows the results of simulation for the algorithm, and Section VII concludes the paper.

II. PROBLEM DESCRIPTION

Imagine a future where stimuli, running the gamut from infantry moving across a battlefield to seismic information gathering on the surface of Mars, is collected by a host of small sensing devices. Such information is as voluminous as it is diverse, and the protocols instrumenting these sensor networks of the near-future are already being developed today. However, one area in this exciting vision remains rather unexplored. This is the area of sensor network QoS.

What is sensor network QoS? There are a variety of definitions possible, but for the purposes of this paper, we define it to mean *sensor network resolution*. Specifically, depending on the different stimuli present in the sensor network, we define it as the *optimum number of sensors* *sending information toward information-collecting sinks, typically base stations*. This is a very important issue, because in any sensor network we want to accomplish two things: (1) maximize the lifetime of the sensor network by having sensors periodically power-down to conserve their battery energy, and (2) have enough sensors powered-up and sending packets toward the information sinks so that enough data is being collected (we define a live sensor to be a sensor that has not run out of battery power). Note that the information sinks need a certain amount of information gathered from the different sensors, but sensors in close proximity to each other allow many of those sensors to be powered-down. This is the optimization problem we address.

What makes this problem hard? Let us assume for the moment that we have a "naïve" sensor network up and running, and that there is one base station with a broadcast channel to all the sensors that knows the optimal number of sensors that should be powered-on and sending packets at any given time. Then at time t, we could broadcast the probability $p^*(t)$ = (optimal number of sensors turned on at time t / total number of live sensors at time t) to all the sensors and have each sensor power-up with probability $p^*(t)$. One would think that this would get, on average, the optimum number of sensors powered-up at time t. However, this requires us to know the total number of live sensors at any given time. This is a very difficult number to calculate because sensor networks will likely consist of thousands of sensors randomly thrown about a geographic area. Further, as time progresses, sensors will likely expire (e.g., due to battery failure, being blown up by tanks, etc.) and new sensors may well be redistributed (e.g., dropped by airplane, battery power regenerated via solar power, etc.), making the population highly dynamic.

In this paper, we present an algorithm that addresses our two goals (stated earlier in this section), and that, at the same time, is robust enough to adapt to these changes in the network and simple enough to be run on small, and possibly disposable, sensors.

III. PREVIOUS WORK

We now briefly survey some of the previous work in the field:

Reference [1] presents a family of adaptive protocols called SPIN (Sensor Protocols for Information via Negotiation) that efficiently disseminate information among sensors in an energy-constrained wireless sensor network. Nodes running a SPIN communication protocol name their data using high-level data descriptors, called meta-data. They use meta-data negotiations to eliminate the transmission of redundant data throughout the network. However, there are no means for data sinks (base stations) to allow for QoS negotiations.

Reference [9] presents the paradigm of directed diffusion. In this scheme, a sink requests data by sending interests for named data. Data matching the interest is then drawn toward that node. Intermediate nodes can cache or transform data, and may direct interests based on previously cached data. However, once again, there is no provision made for actively regulating QoS.

Reference [7] puts forth the paradigm of data aggregation. The idea is to combine the data coming from different sources en route—eliminating redundancy, minimizing the number of transmissions, and thus saving energy. This paradigm shifts the focus from the traditional address-centric approaches for networking (e.g., finding short routes between pairs of addressable end-nodes) to a more data-centric approach (finding routes from multiple sources to a single destination that allows in-network consolidation of redundant data). Again, however, no provisions are made for fine-tuning QoS at the information sinks.

Perhaps [10] is the most relevant work to the present study as it actively probes the question of QoS that the base stations are receiving from the sensors. However, it defines QoS as total coverage in a static fashion. That is, it does not allow a data sink to dynamically alter the QoS it is receiving from the sensors, depending on varying circumstances (e.g., troops marching across a battlefield). The present work involving the Gur Algorithm, on the other hand, allows the number of powered-up sensors to change dynamically, thereby adjusting the QoS seen by the base station.

IV. THE GUR GAME

Our algorithm, which we describe in the Section V, uses the mathematical paradigm of the Gur Game. We describe the basic idea of the Gur Game in the following paragraphs.

Let us introduce the Gur Game with a simple example [8]. Imagine that we have many players, none of whom are aware of the others, and a referee. Every second, the referee asks each player to vote *yes* or *no*, then counts up the *yes* and *no* answers. A reward probability $r = r(k)$ is generated as a function of the number k of players who voted *yes*. We assume that $0 \le r(k) \le 1$. A typical function is shown in Fig. 1. Each player, regardless of how he or she voted, is then independently rewarded (with probability r) or penalized (with probability 1-r). For instance, let us assume that at some point the number of players voting *yes* was k_1. The reward probability would be $r(k_1)$. Each player is then rewarded with probability $r(k_1)$. Note that the maximum of the example occurs at $k^* = 35$. We can show the following: No matter how many players there are, we can "construct" them in such a way that approximately k^* of them (in this case, 35) vote *yes* after enough trials. This "Gur Game" property holds for almost any kind of function—whether or not it is discontinuous, multimodal, etc. Note further that the individual automata know neither the number k nor the reward function r(k).

Moreover, each player plays solely in a greedy fashion, each time voting the way that seems to give the player the best payoff. This is somewhat unexpected. Greed affects outcomes in an unpredictable manner. An example of greed leading to significantly sub-optimal outcomes is the famous prisoner's dilemma. In this scenario, two entities (the prisoners) greedily optimize their own behavior, but together they produce a globally sub-optimal result. This effect is common in greedy solutions. However, we will see that the method used here does not have this property because the players do not attempt to predict the behavior of the other players. Instead, each player performs by trial and error and simply preferentially repeats those actions that produce the best result for that player.

The natural question then becomes: "How do we construct players such that this remarkable property holds?" The answer (as shown by Tsetlin in [13]) is to allow each player to have a memory of his previous trials. Specifically, we associate with each player j, a finite discrete-time automaton M_j. The finite state automaton represents the player's memory. It is a single (nearest-neighbor) chain of

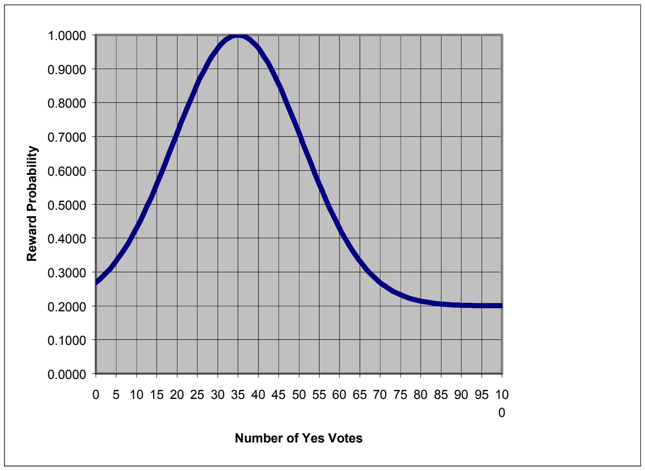

Figure 1. Typical Gur Reward Function

consecutive states where the total size of the memory is 2N, for some arbitrary N. Starting with the leftmost state, we number the states from –N to –1, then from 1 to N (see Fig. 2). Note that this partitions the chain into a left half (with negative numbered states) and a right half (with positive numbered states). The player is allowed to be in only one state at any given time. Transitions exist between states j and j+1 and j-1 (i.e., the player can transition only to adjacent states). If j happens to be N, then the transitions allowed are only to state N-1 and N (i.e., a self-loop). An analogous case exists when j happens to be –N.

The player votes *yes* when he is in a positive numbered state, and *no* when he is in a negative numbered state. When in a negative numbered state, he transitions leftward if he is rewarded by the referee and rightward when he is punished. Analogously, when in a positive numbered state, he transitions rightward when rewarded by the referee and leftward when he is punished. In other words, "center seeking" behavior is for punishment, and "edge seeking" behavior is for reward.

With this setup, it has been proven [13] that the Gur game property holds.

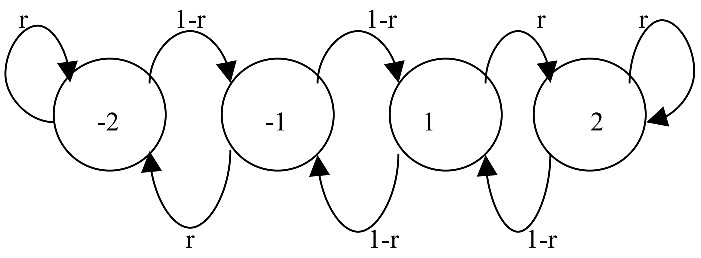

Figure 2. Gur Memory of Size N=2.

We give a new application of the Gur Game in the present work. Also, in Section VI, we study parameters involving the Gur Game and networks not previously considered.

V. NETWORK MODEL AND ALGORITHM

Let us assume that we have a collection of m sensors S_1 through S_m and one base station B. Time is divided into discrete intervals—each of one-second duration. Each sensor S_i is a distance d_i from base station B. We interpret this to mean that a packet is sent reliably from S_i to B and takes d_i seconds to reach B. B, on the other hand, is assumed to have a broadcast channel to all the sensors. For the purposes of this paper, we focus on the delay from sensor to base station while assuming that feedback given from the base station arrives instantaneously to all the sensors.

We associate with each sensor S_i a finite state automaton M_i. The finite automaton, of fixed size N, will be of the standard Gur Game form described in the previous section. The sensor S_i will power-up when it is in a positive numbered state, and power-down when it is in a negative numbered state. We assume that in a power-down state, a sensor can still receive and react to low-level signals. This is similar to the sensor paging situation described in [14].

At each second, if a sensor is powered-up, it will send a data packet containing sensor information toward the base station. In this paper, we do not address issues involving routing. If a sensor is powered-down, it simply "sleeps." Note that this type of sensor network assumes that the base station wants information from its sensors regardless of whether there are active stimuli or not. An example of this situation would be sensors deployed on the surface of Mars sending seismic information toward the base station.

The base station B desires optimal QoS from the sensor network at each time t. The tricky point here is what we consider to be optimal QoS. In Section II we defined optimal QoS to mean an optimal number of sensors powered on at time t. However, in order to accomplish this, we needed to know the total number of live sensors at time t. This is non-trivial, as was previously explained. Instead of this, let us assume that the base station wants information uniformly distributed from all the sensors (like gathering information from a geographic region). Then we can redefine optimal QoS at time t to mean receiving an optimal number of packets at time t. Assuming a "well behaved" QoS protocol that has been running for a sufficiently long period, we can assume that receiving k packets at time t means that approximately k sensors, distributed over the total geographic area, are powered-on at time t. It is a subtle point to redefine QoS in this way, but it does relieve us from the burden of trying to calculate the total number of live sensors at time t.

Obviously the base station will not necessarily receive the optimum number of desired packets at each time t. The question is, then, what to do about it? This is where our algorithm comes in. We associate with the base station a Gur reward function r(k). At each time t, the base station counts the number of packets k_t it has received from the sensors. It then calculates the Gur reward probability $r(k_t)$. Finally, it broadcasts this probability to all the sensors. Each sensor, in turn, independently rewards itself with probability $r(k_t)$ and punishes itself with probability $1-r(k_t)$. This corresponds to playing the Gur Game with the sensor network where a *yes* vote by the sensors means being powered-on and sending a packet, and a *no* vote by the sensors means being powered-off. The base station simply counts the number of *yes* votes in terms of the number of packets it has received and independently rewards or punishes the sensors accordingly.

VI. SIMULATION

We simulated the logical behavior of the Gur Game and abstracted away the details of the radio channel itself. We begin with a simple example. We assume that the memory size N is equal to 1 and that we have 100 sensors in the network with no sensor failures or renewals. Each sensor picks a random state as its initial state. We assume that the base station desires a rate of 35 packets received at each time t. (It was noted in simulations that obtaining a rate of 50 packets per second was relatively easy—undoubtedly because when the voting is skewed one way or the other, even a small possibility of punishment rebalances the votes by shifting votes from the majority to the minority. In order to stress the system, we choose a desired optimal sufficiently far from 50—in this case, 35.) The reward function used by the base station is $0.2 + 0.8\,e^{v}$ where $v = -0.002\,(k_t - 35)^2$ and k_t = the number of packets received at time t (this was the plot we showed in Fig. 1). In Fig. 3 we show a trace of the number of packets received versus time for a sample run of 2000 seconds for this control case based on the above parameters.

As one can see, the number of packets received by the base station fluctuates in the beginning but quickly converges to the optimal. Once there, it locks on as each sensor is rewarded with probability 1 and feedback is instantaneous. From the control case, we next simulate a more interesting set of parameters. The result is shown in Fig. 4. All the parameters are the same as before except that we now assume that d_i is distributed uniformly from 0 to 5 seconds, and we now allow the birth and death of sensors.

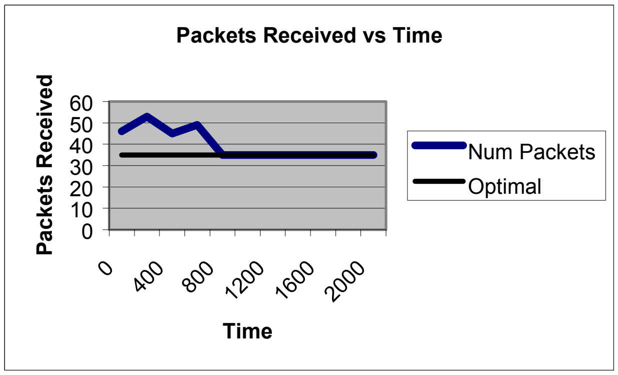

Figure 3. Active Sensors for the Simple Case.

Specifically, we start off with an initial population of 100 sensors, but new sensors are born into the system with exponentially distributed times between births with mean 100 seconds, and each sensor remains alive for an exponentially distributed time with mean 101 seconds. This model corresponds to a population of sensors that die, say, because of battery failure, and having sensors revive after they reenergize with solar energy. The simulation was run for 10,000 seconds.

As one can readily see, packet delay, sensor births and sensor deaths cause network fluctuations despite the optimal being obtained numerous times. It was noted in other simulations (that could not be included in this paper due to space limitations) that: (i) packet delay alone (no births or deaths) would cause fluctuations until the optimal was locked on for a time greater than $\max(d_i)$, from which point it locked on to the optimum (as we saw in Fig. 3); (ii) sensor births and deaths alone (no packet delays) would contain long periods of being locked on to the optimal (with a duration on the order of a small multiple of the birth and death interarrival times) until a birth or death perturbed the population; (iii) together, both (delays plus births and deaths) would cause the occurrence of an occasional birth or death effect to be dramatically amplified

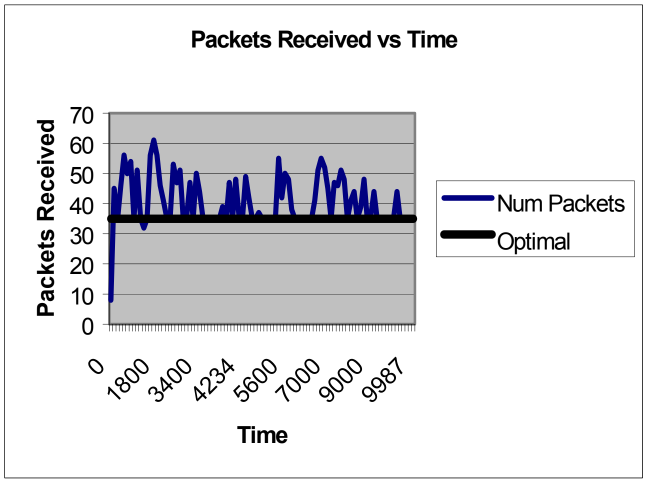

Figure 4. Active Sensors for a More Realistic Case.

by the packet delays, which is the behavior shown in Fig. 4. However, the algorithm is relatively robust and continually drives to the optimum number of 35, despite these problems.

The last thing we study is the way memory size N affects these fluctuations; specifically, we measure the standard deviation from the optimal. All the parameters are the same, as in the last case, except for the varying size of N. The simulation was run for 10,000 seconds, and each point averages five runs. We observe that a minimal value for the standard deviation is obtained for a relatively small size of N. Since sensors only have modest memory capacities, our algorithm is well suited for such networks.

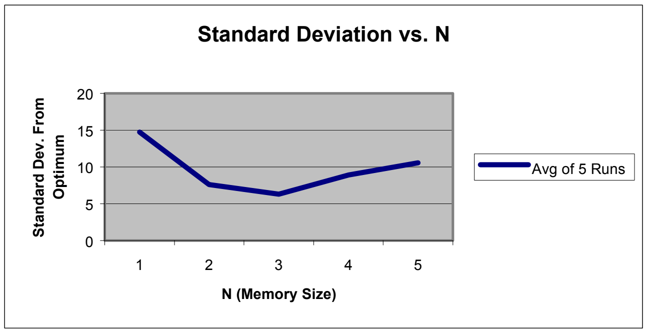

Figure 5. Effect of Varying Memory Size.

VII. CONCLUSION

Sensor networks are an exciting area with very real applications in the near future. Although many aspects of sensor networks have been studied before, quality of service (QoS) for sensor networks remains largely open. As was indicated, it is a non-trivial problem to specify the optimum number of sensors that should be sending information at any given time. In this paper we presented an algorithm using the Gur Game paradigm that allowed the base station to specify the optimal number of sensors from which it wanted information in the face of delays, births and deaths; thus it was able to adjust the QoS it desired from the sensors. The algorithm appears to be robust and is able to tolerate delay and sensor births and deaths reasonably well. In this paper we have reported the results of a study over a very small portion of the design space, and in this area, much future work remains.

REFERENCES

[1] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proc. 5th ACM/IEEE Mobicom Conference,* Seattle, WA, August 1999.

[2] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable Coordination for Wireless Sensor Networks: Self-Configuring Localization Systems," *Proceeding of the Sixth International Symposium on Communication Theory and Applications (ISCTA 2001)*, Ambleside, Lake District, UK, July 2001.

[3] D. Estrin, "Comm'n Sense: Research Challenges in Embedded Networking Sensing," *Presented at the UCLA Computer Science Department Research Review*, April 27th 2001.

[4] A. Cerpa, et al., "Habitat Monitoring: Application Driver for Wireless Communications Technology," *2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Costa Rica, April 2001.

[5] G. Pottie, W. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, Vol. 43, No. 5, pp. 551-8, May 2000.

[6] J. Warrior, "Smart Sensor Networks of the Future," *Sensors Magazine*, March 1997.

[7] B. Krishnamachari, D. Estrin,, and S. Wicker, "The Impact of DataAggregation in Wireless Sensor Networks," submitted to the *2002 International Workshop of Distributed Event-Based Systems*.

[8] B. Tung, and L. Kleinrock, "Distributed Control Methods," *Proceedings of the Second International Conference on High Performance Distributed Computing*, 1993.

[9] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (Mobicom 2000)*, August 2000, Boston, Massachusetts.

[10] S. Meguerdichian, K. Farinaz, P. Miodrag, M. Srivstava, "Coverage Problems in Wireless Ad Hoc Sensor Networks," *ICC-IEEE International Conference on Communications*, 2001.

[11] B. Tung, and L. Kleinrock, "Using Finite State Automata to Produce Self-Optimization and Self-Control," *IEEE Transactions in Parallel and Distributed Systems*, Vol. 7, No. 4, pp. 439-448, 1996.

[12] B. Tung, *Distributed Control Using Finite State Automata*, PhD dissertation, UCLA, 1994.

[13] M. Tsetlin, *Finite Automata and Modeling the Simplest Forms of Behavior*, PhD thesis, V.A. Steklov Mathematical Institute. 1964.

[14] D. Estrin, L. Girod, G. Pottie, M. Srivastava, "Instrumenting the World with Wireless Sensor Networks," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah. May 2001.