# An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks

Kemal Akkaya and Mohamed Younis
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250
kemal1 | younis@cs.umbc.edu

## Abstract

*Recent advances in wireless sensor networks have led to many new routing protocols specifically designed for sensor networks. Almost all of these routing protocols considered energy efficiency as the ultimate objective in order to maximize the whole network lifetime. However, the introduction of video and imaging sensors has posed additional challenges. Transmission of video and imaging data requires both energy and QoS aware routing in order to ensure efficient usage of the sensors and effective access to the gathered measurements. In this paper, we propose an energy-aware QoS routing protocol for sensor networks which can also run efficiently with best-effort traffic. The protocol finds a least-cost, delay-constrained path for real-time data in terms of link cost that captures nodes' energy reserve, transmission energy, error rate and other communication parameters. Moreover, the throughput for non-real-time data is maximized by adjusting the service rate for both real-time and non-real-time data at the sensor nodes. Simulation results have demonstrated the effectiveness of our approach for different metrics.*

## *1. Introduction*

Recent advances in miniaturization and low-power design have led to active research in large-scale, highly distributed systems of small-size, wireless unattended sensors. Each sensor is capable of detecting ambient conditions such as temperature, sound, or the presence of certain objects. Over the last few years, the design of sensor networks has gained increasing importance due to their potential for some civil and military applications such as combat field surveillance, security and disaster management. These systems process data gathered from multiple sensors to monitor events in an area of interest. In a disaster management setup a large number of sensors can be dropped by a helicopter. Networking these sensors can assist rescue operations by locating survivors, identifying risky areas and making the rescue crew more aware of the overall situation. On the military side, applications of sensor networks are numerous. For example, the use of networked set of sensors can limit the need for personnel involvement in the usually dangerous reconnaissance missions and

can provide a more civic alternative to landmines. Security applications of sensor networks include intrusion detection and criminal hunting.

Routing of sensor data has been one of the challenging areas in wireless sensor network research. It usually involves multi-hop communications and has been studied as part of the network layer problems [1,2,3,4]. Despite the similarity between sensor and mobile ad-hoc networks, routing approaches for ad-hoc networks proved not to be suitable to sensors networks. This is due to different routing requirements for ad-hoc and sensor networks in several aspects. For instance, communication in sensor networks is from multiple sources to a single sink, which is not the case in ad-hoc networks. Moreover, there is a major energy resource constraint for the sensor nodes. As a consequence, many new algorithms have been proposed for the problem of routing data in sensor networks. These routing mechanisms can be classified as data-centric [1], hierarchical [2] or location-based [3]. Current research on routing of sensor data mostly focused on protocols that are energy aware to maximize the lifetime of the network, scalable for large number of sensor nodes and tolerant to sensor damage and battery exhaustion. Since the data they deal with is not in large amounts and flow in low rates to the sink, the concepts of latency, throughput, delay and jitter were not primary concerns in sensor networks. However, the development of video and imaging sensors requires the consideration of quality of service (QoS) in sensor networks, which magnifies the difficulties associated with the energy efficiency and awareness.

QoS protocols in sensor networks have several applications including real time target tracking in battle environments, emergent event triggering in monitoring applications etc. Consider the following scenario: In a battle environment it is crucial to locate, detect and identify a target. In order to identify a target, we should employ imaging and/or video sensors. After locating and detecting the target without the need of imaging and video sensors, we can turn on those sensors to get for instance an image of the target periodically and send to the base station or gateway. Since, it is a battle environment; this requires a real-time data exchange between sensors and controller in order to take the proper actions. However, we should deal with real-time multimedia data, which requires certain bandwidth with minimum possible delay, and jitter. In that case, a service differentiation mechanism is needed in order to guarantee the reliable delivery of the real-time data.

Energy-aware QoS routing in sensor networks will ensure guaranteed bandwidth (or delay) through the duration of connection as well as providing the use of most energy efficient path. To the best of our knowledge, no previous research has addressed QoS routing in sensor networks. In this paper, we present an energy-aware QoS routing mechanism for wireless sensor networks. Our proposed protocol extends the routing approach in [17] and considers only end-to-end delay. The protocol looks for a delay-constrained path with the least possible cost based on a cost function

defined for each link. Alternative paths with bigger costs are tried until one, which meets the end-to-end delay requirement and maximizes the throughput for best effort traffic is found. Our protocol does not bring any extra overhead to the sensors.

In the balance of this section we describe the sensor network architecture that we consider and summarize the related work. In section 2, we analyze the complexity of the QoS routing problem in sensor networks and describe our approach. Section 3 includes simulations and evaluations of the protocol. Finally we conclude the paper in section 4 and outline our future research.

### *1.1 Sensor Network Architecture*

We consider the sensor network architecture depicted in Fig. 1. In the architecture sensor nodes are grouped into clusters controlled by a single command node. Sensors are only capable of radio-based short-haul communication and are responsible for probing the environment to detect a target/event. Every cluster has a gateway node that manages sensors in the cluster. Clusters can be formed based on many criteria such as communication range, number and type of sensors and geographical location [27][28]. In this paper, we assume that sensor and gateway nodes are stationary and the gateway node is located within the communication range of all the sensors of its cluster. Clustering the sensor network is performed by the command node and is beyond the scope of this paper. The command node will inform each gateway node of the ID and location of sensors allocated to the cluster.

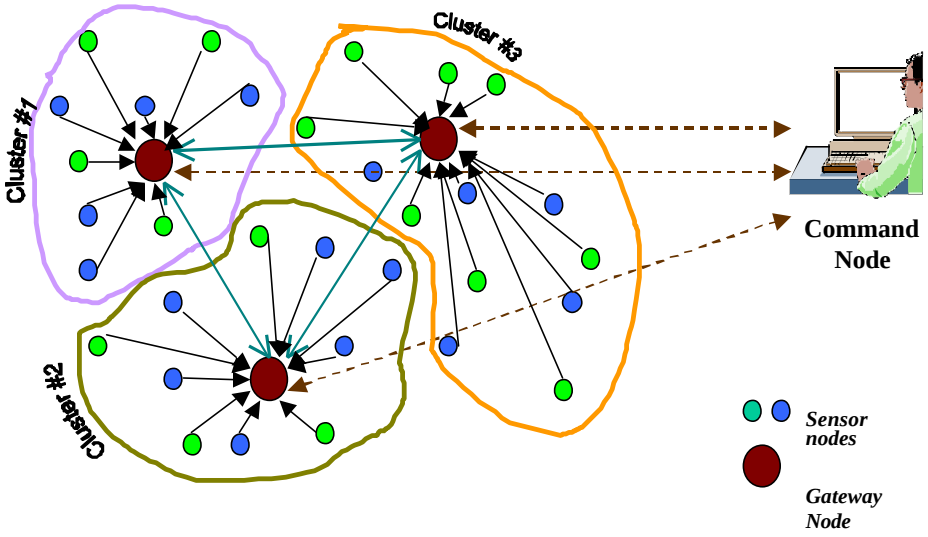


**Fig. 1:** Multi-gateway clustered network sensors

Sensors receive commands from and send readings to its gateway node, which processes these readings. Gateways can track events or targets using readings from sensors in any clusters as deemed by the command node. However, sensors that belong to a particular cluster are only accessible via the gateway of that cluster. Therefore, a gateway should be able to route sensor data to other gateways. Gateway nodes interface the command node with the sensor network via long-haul communication links. The gateway node sends to the command node reports generated through fusion of sensor readings, e.g. tracks of detected targets. The command node presents these reports to the user and performs system-level fusion of the collected reports for an overall situation awareness.

Although the multi-gateway architecture raises many issues such as cluster formation, cluster-based sensor organization and management, and inter-gateway communication protocol, this paper only focuses on the QoS routing of data within one particular cluster.

### *1.2 Related Work*

In traditional best-effort routing throughput and average response time are the main concerns. QoS routing is usually performed through resource reservation in a connection-oriented communication in order to to meet the QoS requirements for each individual connection. While many mechanisms have been proposed for routing QoS constrained real-time multimedia data in wire-based networks [5,6,7,8,9], they cannot be directly applied to wireless sensor networks due to the limited resources, such as bandwidth and energy, that a sensor node has.

On the other hand, a number of protocols have been proposed for QoS routing in wireless ad-hoc networks taking the dynamic nature of the network into account [10,11,12,13,14]. However, none of these protocols consider energy awareness along with the QoS parameters. Some of the proposed protocols consider the imprecise state information while determining the routes [10,11]. In our model, we do not have the problem of imprecision since the state of sensor nodes are maintained by the gateway node.

CEDAR is another QoS aware protocol, which uses the idea of core nodes (dominating set) of the network while determining the paths [12]. Using routes found through the network core, one can search for a QoS path easily. Since, the data flow in our sensor network architecture is many-to-one; there is no need to find a core of the network. Moreover, if any node in the core is broken, it will cost too much resource to reconstruct the core. Lin [13] and Zhu et al. [14] have proposed QoS routing protocols specifically designed for TDMA-based ad-hoc networks. Both protocols can

build a QoS route from a source to destination with reserved bandwidth. The bandwidth calculation is done hop-by-hop using distributed algorithms.

The only protocol for sensor networks that includes the notion of QoS in its routing decisions is Sequential Assignment Routing (SAR) [15]. The SAR protocol creates trees routed from one-hop neighbor of the sink by taking QoS metric, energy resource on each path and priority level of each packet into consideration. By using created trees, multiple paths from sink to sensors are formed. Furthermore, one of the paths can be selected according to the energy resources and QoS on each path. In our approach, we not only select a path from a list of candidate paths that meet the end-to-end delay requirement, but maximize the throughput for best effort traffic as well. In addition, the SAR approach suffers the overhead of maintaining the node states at each sensor node. Our protocol does not require sensor's involvement in route setup.

## *2. Energy-aware QoS Routing*

Our aim is to find an optimal path to the gateway in terms of energy consumption and error rate while meeting the end-to-end delay requirements. End-to-end delay requirements are associated only with the real-time data. Note that, in this case we have both real-time and non-real-time traffic coexisting in the network, which makes the problem more complex. We not only should find paths that meet the requirements for real-time traffic, but need to maximize the throughput for non-real time traffic as well. This is because most of the critical applications such as battlefield surveillance have to receive for instance acoustic data regularly in order not to miss targets. Therefore it is important to prevent the real-time traffic from consuming the bulk of network bandwidth and leave non-real-time data starving and thus incurring large amount of delay.

The described QoS routing problem is very similar to typical path constrained path optimization (PCPO) problems, which are proved to be NP-complete [18]. We are trying to find least-cost path, which meets the end-to-end delay path constraint. However, in our case there is an extra goal, which is basically to maximize the throughput of non-real-time traffic. Our approach is based on associating a cost function for each link and used a K least cost path algorithm to find a set of candidate routes. Such routes are checked against the end-to-end constraints and the one that provides maximum throughput is picked. Before explaining the details of proposed algorithm, we introduce the queuing model.

## *2.2 Queuing Model*

The queuing model is specifically designed for the case of coexistence of real-time and non-real-time traffic in each sensor node. The model we employ is inspired from class-based queuing model [16]. We use different queues for the two different types of traffic. Basically, we have real-time traffic and non-real-time (normal) traffic whose packets are labeled accordingly. On each node, there is a classifier, which checks the type of the incoming packet and sends it to the appropriate queue. There is also a scheduler, which determines the order of packets to be transmitted from the queues according to the bandwidth ratio "*r*" of each type of traffic on that link. The model is depicted in Fig. 2.

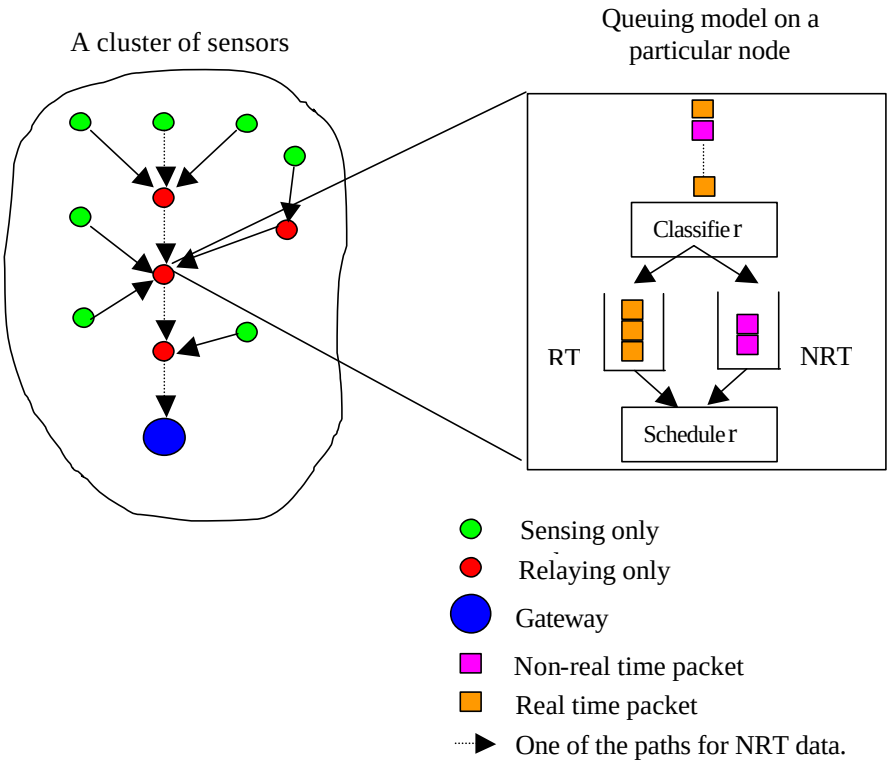


**Fig. 2**: Queuing model in cluster-based sensor network

The bandwidth ratio *r*, is actually an initial value set by the gateway and represents the amount of bandwidth to be dedicated both to the real-time and non-real-time traffic on a particular outgoing link in case of a congestion. Moreover, both classes can borrow bandwidth from each other when one of the two types of the traffic is non-existent or under the limits. As indicated in Fig. 3, this *r*-value is also used to calculate the service rate of real-time and non-real-time traffic on that particular

node, with $r_i m$ and $(1-r_i)m$ being respectively the service rate for real-time and non-real-time data on sensor node $i$.

Since the queuing delay depends on this $r$-value, we cannot calculate the end-to-end delay for a particular path without knowing the $r$-value. Therefore we should first find a list of candidate least-cost paths and then select one that meets the end-to-end delay requirement.

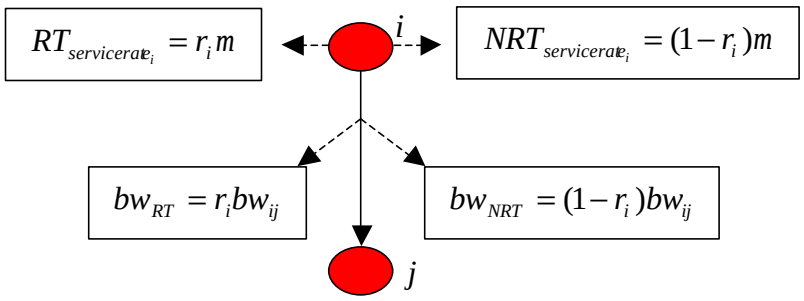


**Fig. 3**: Bandwidth sharing and service rates for a sensor node

Our approach is based on a two-step strategy incorporating both link-based costs and end-to-end constraints. First we calculate the candidate paths without considering the end-to-end delay. What we do is simply calculate costs for each particular link and then use an extended version of Dijkstra's algorithm to find an ascending set of least cost paths. Once we obtain these candidate paths then we might further check them to see which one meets our end-to-end QoS requirements by trying to find an optimal $r$-value that will also maximize the throughput for non-real-time traffic.

### *2.3 Calculation of link costs*

We consider the factors for the cost function on each particular link separately except the end-to-end delay requirement, which should be for the whole path (i.e. all the links on that path). We define the following cost function for a link between nodes $i$ and $j$:

$$\cos t_{ij} = \sum_{k=0}^{6} CF_k = c_0 \times (dist_{ij})^l + c_1 \times f(energy_j) + c_2 / T_j + c_3 + c_4 + c_5 + c_6 \times f(e_{ij})$$

where,

- $dist_{ij}$ is the distance between the nodes $i$ and $j$,
- $f(energy_j)$ is the function for finding current residual energy of node $j$,

- $T_j$ is the expected time under the current consumption rate until the node j energy level reaches the minimum acceptable threshold.
- $f(e_{ij})$ is the function for finding the error rate on the link between *i* and *j*.

The factors $CF_k$ are defined similar as in [17], however the cost function is further extended for error rate cost. The end-to-end delay is modeled as a constraint on the whole path and includes the propagation delay. Hence, it's not part of the cost function. Cost factors are defined as follows:

- $CF_0$ (Communication Cost)= $c_0 \times (dist_{ij})^l$, where $c_0$ is a weighting constant and the parameter *l* depends on the environment, and typically equals to 2. This factor reflects the cost of the wireless transmission power, which is directly proportional to the distance raised to some power *l*. The closer a node to the destination, the less its cost factor $CF_0$ and more attractive it is for routing.
- $CF_1$ (Energy Stock)= $c_1 \times f(energy_j)$. This factor reflects the remaining battery lifetime, which favors nodes with more energy. The more energy the node contains, the better it is for routing.
- $CF_2$ (Energy Consumption Rate)= $c_2 / T_j$, where $c_2$ is a weighting constant and $T_j$ is the expected time under the current consumption rate until the node j energy level hits the minimum acceptable threshold. The factor $CF_2$ makes heavily used nodes less attractive, even if they have a lot of energy.
- $CF_3$ (Relay enabling cost)= $c_3$, where $c_3$ is a constant reflecting the overhead required to switch an inactive node to become a relay. This factor favors relay-enabled nodes to be used for routing rather than the inactive nodes.
- $CF_4$ (Sensing-state cost)= $c_4$, where $c_4$ is a constant added when the node *j* is in a sensing-state. This factor does not favor selecting sensing-enabled nodes to serve as relays. It's preferred not to overload these nodes in order to keep functioning as long as possible.
- $CF_5$ (Maximum connections per relay)= $c_5$. Once this threshold is reached, we add an extra cost $c_5$ to avoid setting additional paths through that relay. This factor extends the life of overloaded relay nodes by making them less favorable.

- $CF_6$(Error rate)= $c_6 \times f(e_{ij})$where $f$ is a function of distance between nodes $i$ and $j$ and buffer size on node $j$ (i.e. $dist_{ij} / buffer\_size_j$ ). The links with high error rate will increase the cost function, thus will be avoided.

### *2.4 Calculation of end-to-end delay for a path*

In order to find a QoS path for sending real-time data to the gateway, end-to-end delay requirement should be met. Before explaining the computation of the delay, which consists of queuing delay and propagation delay for a particular path *P*, we introduce the following notation:

| | | |
|---|---|---|
| $l_{RT}$ | : | Real-time data generation rate for imaging sensors |
| $r_i m$ | : | Service rate for real-time data on sensor node *i* |
| $(1-r_i)m$ | : | Service rate for non-real-time data on sensor node *i* |
| $p_i$ | : | The number of sensing-only neighbors of node *i* on path *P* |
| $q_i$ | : | The number of relaying-only neighbors of node *i* on path *P* |
| $l_{RT}^{(i)}$ | : | Total real-time data rate on sensor node *i* |
| $TQ_{RT}^{(i)}$ | : | Total queuing delay on a node *i* for real-time traffic |
| $T_E$ | : | End-to-end queuing delay for a particular path *P* |
| $T_p$ | : | End-to-end propagation delay for a particular path *P* |
| $T_{end-end}$ | : | Total end-to-end delay for a particular path *P* |
| $T_{required}$ | : | End-to-end delay requirement for all paths |

Total real-time data rate by $p_i$ nodes will be $p_i l_{RT}$ and total real-time data rate by $q_i$ nodes will be $\sum_{j=1}^{q_i} r_j m$ (since every relaying-only node produces real-time data by the rate $r_j m$ ).

Then total real-time data load on a sensor node is:

$$l_{RT}^{(i)} = p_i l_{RT} + \sum_{j=1}^{q_i} r_j m$$

Hence, total queuing delay on a node is:

$$TQ_{RT}^{(i)} = l_{RT}^{(i)} / r_i m$$

The end-to-end queuing delay for a particular path is:

$$T_E = \sum_{i \in Path} TQ_{RT}^{(i)}$$

$$= \sum_{i \in Path} \left( \frac{p_i l_{RT} + \sum_{j=1}^{q_i} r_j m}{r_i m} \right)$$

$$= \left( \frac{l_{RT}}{m} \right) \left( \sum_{i \in Path} \frac{p_i}{r_i} \right) + \sum_{i \in Path} \left( \frac{\sum_{j=1}^{q_i} r_j}{r_i} \right)$$

$$= \left( \frac{l_{RT}}{m} \right) \sum_{i \in Path} \left( \frac{p_i + \sum_{j=1}^{q_i} r_j}{r_i} \right)$$

The end-to-end propagation delay for the path is:

$$T_p = \sum_{i, j \in Path} c \times dist_{ij}$$

where **c** is a constant, which is obtained by dividing a weighting constant by the speed of wireless transmission.

Hence, total end-to-end delay will be:

$$T_{end-end} = T_E + T_p$$

$$= \left(\frac{l_{RT}}{m}\right)\sum_{i\in Path}\left(\frac{p_i + \sum_{j=1}^{q_i} r_j}{r_i}\right) + \sum_{i,\, j\in Path}\left(c \times dist_{ij}\right) \qquad (1)$$

### *2.5 Algorithm*

While we generate a formula for calculating the end-to-end delay for a particular path, finding the optimal *r*-values for each link as far as the queuing delay is concerned, will be very difficult optimization problem to solve. Moreover, the distribution of these *r*-values to each node is not an easy task because the each value should be unicasted to the proper sensor node rather than broadcasting it to all the sensors which might bring a lot of overhead. Therefore, we follow an approach, which will eliminate the overhead and complexity of the problem. Basically, we define each *r*-value to be same on each link so that the optimization problem will be simple and this unique *r*-value can be easily broadcasted to all the sensors by the gateway.

If we let all *r*-values be same for every link then the formula will be simplified as:

$$T_{end-end} = \left(\frac{l_{RT}}{rm}\right)\sum_{i,\, j\in Path} p_i + \sum_{i,\, j\in Path}\left(q_i + c \times dist_{ij}\right)$$

Then the problem is stated as an optimization problem as follows:

$$Max\left(\sum_{i\in Path}((1-r)m)\right)$$

subject to;

$T_{end-end} \le T_{required}$ and $0 \le r < 1$

By considering the above optimization problem, we propose the algorithm as shown in Fig. 4 to find a least-cost path, which meets the constraints and maximizes the throughput for non-real-time data.

```
1 Calculate cost_ij, ∀i, j ∈ V
2 Find the least cost path for each node by using Dijkstra's shortest path algorithm.
3 for each imaging sensor node i do
4 begin
5       compute r from T_end−end(p_i) = T_required
6       if (r is in range [0,1)) then
7               Add r to a list corresponding to node i
8       else
9               Find K least cost paths (P_i^K) to the gateway by extended Dijkstra.
10              for each k ∈ K do
11                      Recompute r from T_end−end(p_i^k) = T_required
12                      if (r is in range [0,1)) then
13                              break;
14              if no appropriate r is found
15                      Reject the connection
16      end
17      Find max r from the list
```

**Fig. 4:** Pseudo code for the proposed algorithm

The algorithm calculates the cost for each link in line 1 of Fig. 4, based on the cost function defined in section 2.3. Then, for each node the least cost path to the gateway is found by running Dijkstra's shortest path algorithm in line 2. Between lines 5-15, appropriate *r*-values are calculated for paths from imaging sensors to the gateway. For each sensor node that has imaging capability, an *r*-value is calculated on the current path (line 5). If that value is not between 0 and 1, extended Dijsktra algorithm for K-shortest path is run in order to find alternative paths with bigger costs (line 9). K different least-cost paths are tried in order to find a proper *r*-value between 0 and 1 (lines 10-13). If there is no such *r*-value, the connection request of that node to the gateway is simply rejected. The algorithm might generate different *r*-values for different paths. Since, the *r*-values are stored in a list; the maximum of them is selected to be used for the whole network (line 17). That *r*-value will satisfy the end-to-end delay requirement for all the paths established from imaging sensors to the gateway.

In order to find the K least cost paths (i.e. K shortest paths), we modified an extended version of Dijkstra's algorithm. Finding K shortest paths is a classical network-programming problem, which has been studied extensively in [19, 20, 21, 22]. In [22], a generalization of Dijkstra's algorithm to solve K shortest path problem is given. The algorithm uses Dijkstra's concepts but it does not use relaxation at each node to find the shortest distance from the source. Instead, the algorithm records every path from the source to a particular node by adding new elements to the set

V. It is a labeling algorithm, which assigns labels to each node in the graph and then finds the paths. A labeling function $h$ is used to map numbers (labels) to the nodes. And the reverse function $h^{-1}$ returns the labels for a particular node. The algorithm creates a tree like structure for every path it has reached and keeps the cost of each path from source to any node $i$ in $d[i]$. Whenever the number of paths from source to target ($count_t$) reaches *K*, the algorithm terminates and returns the list of K shortest paths from source to target.

Since, the algorithm faces with loops during execution; we modified the algorithm in order to avoid loops and ensure simplicity and efficiency. Each time a new path is searched for a particular node, only node-disjoint paths are considered during the process. This might also help finding a proper *r*-value easily since that node-disjoint path will not inherit the congestion in the former path. Interested reader is referred to [22] for further information.

## 3. Experimental Results

The effectiveness of the energy-aware QoS routing approach is validated through simulation. This section describes the performance metrics, simulation environment, and experimental results.

### 3.1 Performance Metrics

We used the following metrics to capture the performance of our QoS routing approach:

- *Average lifetime of a node*: This gives a good measure of the network lifetime. A routing algorithm, which maximizes the lifetime of network, is desirable. This metric also shows how efficient is the algorithm in energy consumption.
- *Average delay per packet*: Defined as the average time a packet takes from a sensor node to the gateway. Most energy aware routing algorithms try to minimize the consumed energy. However, the applications that deal with real-time data is delay sensitive, so this metric is important in our case.
- *Network Throughput*: Defined as the total number of data packets received at the gateway divided by the simulation time. The throughput for both real-time and non-real-time traffic will be considered independently.

### 3.2 Environment Setup

In the experiments the cluster consists of 100 randomly placed nodes in a 1000×1000 meter square area. The gateway position is determined randomly within the cluster boundaries. A free space

propagation channel model is assumed [24] with the capacity set to 2Mbps. Packet lengths are 10 Kbit for data packets and 2 Kbit for routing and refresh packets. Each node is assumed to have an initial energy of 5 joules. The buffers for real-rime data and normal data have default size of 15 packets [25]. A node is considered non-functional if its energy level reaches 0. For the term $CF_1$ in the cost function, we used the linear discharge curve of the alkaline battery [23].

For a node in the sensing state, packets are generated at a constant rate of 1 packet/sec. This value is consistent with the specifications of the Acoustic Ballistic Module from SenTech Inc. [26]. The real-time packet generation rate ( $l_{RT}$ ) for the nodes, which have imaging/video capability is greater than the normal rate. The default value is 3 packets/sec. A service rate ( $m$ ) of 5 packets/sec is assumed. Each data packet is time-stamped when it is generated to allow the calculation of average delay per packet. In addition, each packet has an energy field that is updated during the packet transmission to calculate the average energy per packet. A packet drop probability is taken to be 0.01. This is used to make the simulator more realistic and to simulate the deviation of the gateway energy model from the actual energy model of nodes.

We assume that the cluster is tasked with a target-tracking mission in the experiment. The initial set of sensing nodes is chosen to be the nodes on the convex hull of sensors in the cluster. The set of sensing nodes changes as the target moves. Since targets are assumed to come from outside the cluster, the sensing circuitry of all boundary nodes is always turned on. The sensing circuitry of other nodes are usually turned off but can be turned on according to the target movement. We also assume that each sensor node is capable of taking the image of target to identify it clearly and can turn on its imaging capability on demand. During simulation, a small subset of current active nodes, which are the closest nodes to the target, are selected to turn on their imaging capability. Therefore, the imaging sensor set may change with the movement of the target.

The packet-sensing rate for imaging sensors is bigger than the normal sensors; hence more packets are generated when imaging sensors are employed. These packets are labeled as real-time packets and treated differently in sensor nodes. The r-value is initially assumed to be 0 but it's recalculated as imaging sensors get activated. The default end-to-end delay requirement for a QoS path is assumed to be 10 seconds, which is a reasonable amount of time to get image data periodically in a real-time target tracking application. Targets are assumed to start at a random position outside the convex hull. Targets are characterized by having a constant speed chosen uniformly from the range 4 meters/s to 6 meters/s and a constant direction chosen uniformly depending on the initial target position in order for the target to cross the convex hull region. It is

assumed that only one target is active at a time. This target remains active until it leaves the deployment region area. In this case, a new target is generated.

### *3.3 Performance Results*

In this section, we present some performance results obtained by the simulation. Different parameters are considered for end-to-end delay, buffer size, packet drop probability and real-time data generation capture the effects on the performance metrics defined earlier in this section.

Effect of end-to-end delay and real-time date generation rate on network *r*-values

In order to see how the algorithm behaves under stringent conditions, we varied the end-to-end delay and monitored how this change affects the network *r*-value. The results are depicted in Fig. 5. The network *r*-value goes down while the end-to-end delay requirement gets looser. Since the delay is not too strict, most of the nodes will be able to find a QoS path.

On the other hand, while we congest the network with more real-time data packets by increasing the real-time data generation rate, more bandwidth will be required for real-time packets. This will cause the *r*-value to increase so that each node can serve more real-time packets (See Fig. 6).

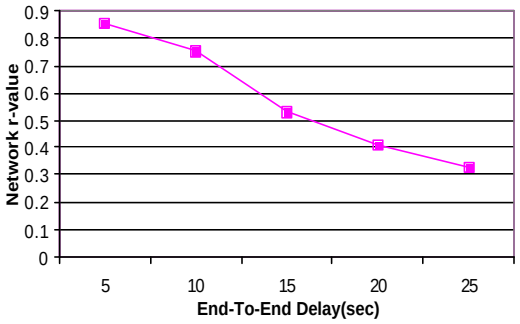


**Fig. 5:** Network r-value with different end-to-end delay values

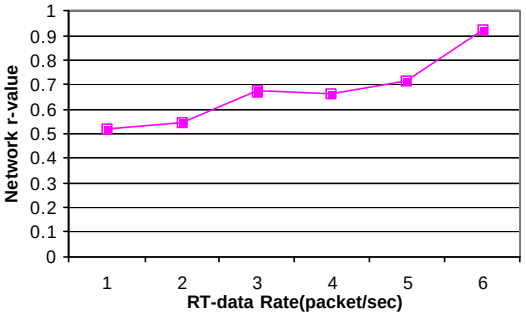


**Fig. 6:** Network r-value with different real-time data rates

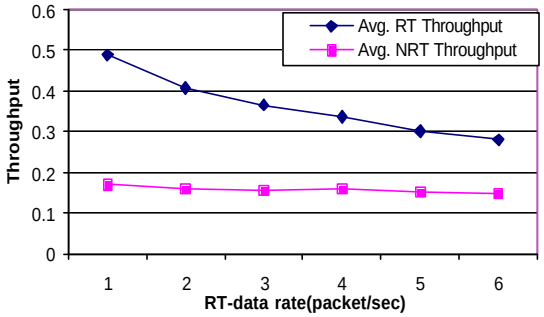


**Fig. 7:** Effect of rt-data rate on throughput

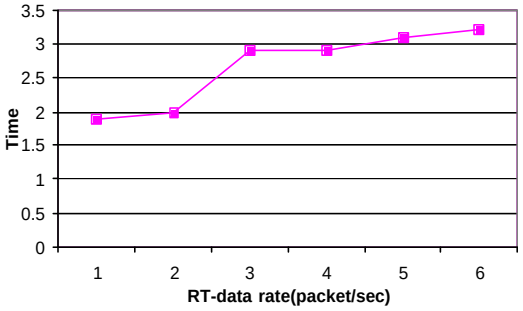


**Fig. 8:** Effect of rt-data rate on average delay for a packet

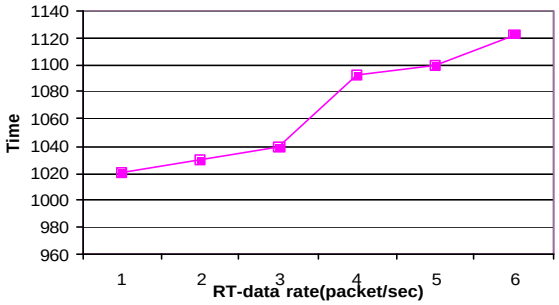


**Fig. 9:** Effect of rt-data rate on average lifetime of a node

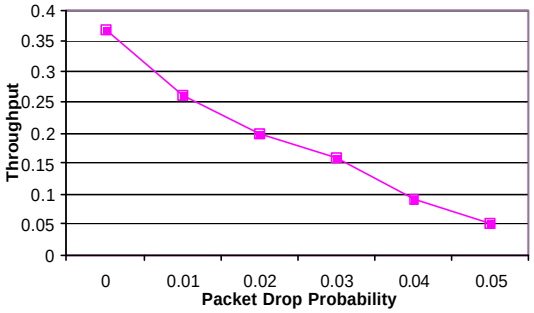


**Fig. 10:** Effect of packet drop prob. on throughput

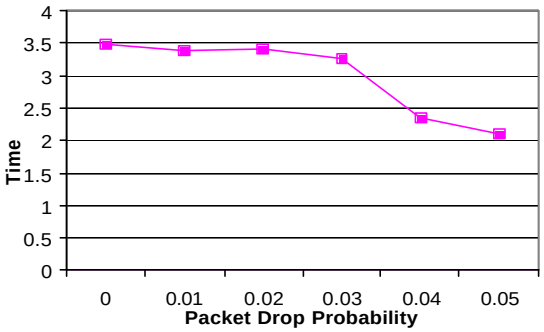


**Fig. 11:** Effect of packet drop prob. on average delay per packet

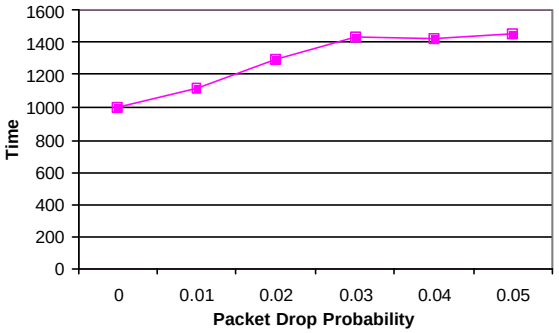


**Fig. 12:** Effect of packet drop prob. on average lifetime of a node

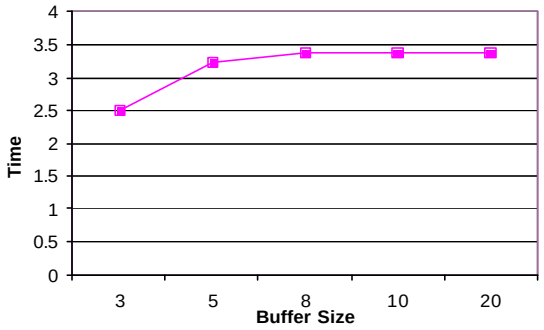


**Fig. 13:** Effect of buffer size on average delay per packet

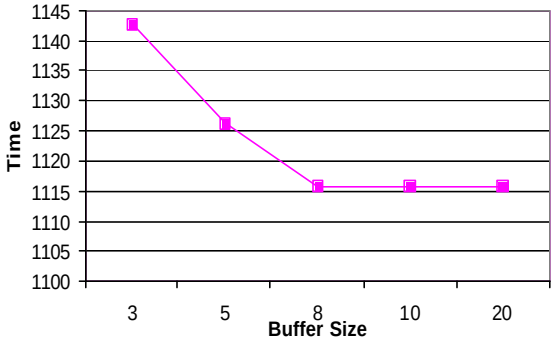


**Fig. 14:** Effect of buffer size on average lifetime of a node

Effect of real-time data rate on performance

In order to see the performance of the algorithm for different real-time data rates, we ran simulation for different values of real-time packet data rates. The results are depicted in Fig. 7, 8 and 9. We looked at the real-time and non-real-time data throughput. While the number of real-time packets increase, it gets more difficult to satisfy increasing number of QoS paths. Hence, this can cause some rejection or packet drops for real-time data causing throughput for real-time data to decrease. However, the throughput for non-real-time data does not change much since there is already a constant dedicated bandwidth for such data, ensured by the *r*-value. We restricted *r*-value to be strictly less than 1 causing the throughput for non-real-time data ($(1-r)m$) always greater than 0. The algorithm does not sacrifice the throughput for non-real-time data for the sake of real-time data.

Figure 8 shows the effect of real-time data rate on average delay per packet. The delay increases with the rate since packets (especially real-time packets) incur more queuing delay and share the same amount of bandwidth. We also looked for the lifetime of a node in order to see the effect of real-time data rate on energy metric. Figure 9 shows that the average energy for a sensor node increases with the real-time data rate. The reason for this increase is that the throughput decreases, causing the number of packets arriving to the gateway to decrease. Therefore, fewer packets will be relayed by the sensor nodes, which will save energy from transmission and reception energy costs.

Effect of packet drop probability on performance

To study the effect of packet drop probability on performance, we varied the probability of packet drop from 0.01 to 0.05. The results are depicted in Figures 10, 11 and 12. The average delay per packet decreases with the increasing probability. This can be explained by noting that as the number of hops the packet traverse increases, the probability that it will be dropped increases. This means that the packets that arrive to the gateway are most probable to take a small number of hops and thus incurring less delay. As expected, the throughput decreases due to lost packets. The average node lifetime increases since not all packets reach their destination and thus the node energy is conserved.

Effect of buffer size on performance

Since, the queuing model we employed uses buffers in each node and there is a limit on the size of those buffers, we varied the buffer size to see if this has any effect on the performance of the

algorithm. The results are shown in Fig. 13 and 14. The average delay per packet increases with the buffer size since the throughput increases. Packets are not dropped when there is enough space in the buffers. This will increase the number of packets arriving to the gateway. The packets from far nodes will be also able to reach the gateway. More packets from far nodes mean more delay, which eventually increases the average delay per packet. The increasing number of packets arriving to the gateway will also increase the energy consumption by increasing the number of transmission and reception costs, therefore decreasing the average lifetime of a node.

## *4. Conclusion and Future Work*

In this paper, we presented a new energy-aware QoS routing protocol for sensor networks. The protocol finds QoS paths for real-time data with certain end-to-end delay requirements. Moreover, the selected queuing model for the protocol allows the throughput for normal data not to diminish by employing a network wide *r*-value, which guarantees certain service rate for real-time and non-real-time data on each link. The effectiveness of the protocol is validated by simulation. Simulation results show that our protocol consistently performs well with respect to QoS metrics, e.g. throughput and average delay as well as energy-based metric such as average lifetime of a node. The network *r*-value is adjusted accordingly in the case of big real-time data rate on the nodes or stringent end-to-end delay requirements. The results have also shown that real-time data rate, buffer size, and packet drop probability have significant effects on the performance of the protocol.

We are currently extending the model to allow different *r*-values can be assigned to sensor nodes and plan to compare the performance of such extended model with the energy-aware QoS routing protocol presented in this paper.

## References


[1] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, Boston, Massachussetts, August 2000.

[2] W. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks,” In *Proc. Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks," in *Proc. Hawaii Intl. Conf. System Sciences, Hawaii*, pp. 3005-3014, 2000.

[4] R. Shah and J. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks", in the *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'02)*, Orlando, FL, March 2002.

[5] W. C. Lee, M. G. Hluchyi and P. A. Humblet, "Routing Subject to Quality of Service Constraints Integrated Communication Networks," *IEEE Network*, July/Aug. 1995.

[6] Z. Wang and J. Crowcraft, "QoS-based Routing for Supporting Resource Reservation," *IEEE Journal on Selected Area of Communications*, Sept 1996.

[7] Q. Ma and P. Steenkiste, "Quality-of-Service routing with Performance Guarantees," in the *Proceedings of the 4th International IFIP Workshop on Quality of Service*, May 1997.

[8] L. Zhang et al., "RSVP: A New Resource ReServation Protocol," *IEEE Network*, Sept 1993.

[9] E. Crowley, R. Nair, B. Rajagopalan and H. Sandick, "A framework for QoS based routing in the Internet," Internet-draft, draft-ietf-qosr-framework-06.txt, Aug. 1998.

[10] R. Querin and A. Orda, "QoS-based routing in networks with inaccurate information: Theory and algorithms," in *Proc. IEEE INFOCOM'97*, Japan, pp. 75-83, 1997.

[11] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in ad-hoc Networks," *IEEE Journal on Selected areas in Communications*, Vol. 17, No. 8, August 1999.

[12] R. Sivakumar, P. Sinha and V. Bharghavan, "Core extraction distributed ad hoc routing (CEDAR) specification," IETF Internet draft draft-ietf-manet-cedar-spec-00.txt, 1998.

[13] C. R. Lin, "On Demand QoS routing in Multihop Mobile Networks," *IEICE Transactions on Communications*, July 2000.

[14] C. Zhu and M. S. Corson, "QoS routing for mobile ad hoc networks," *In the Proceedings of IEEE INFOCOM*, 2002.

[15] K. Sohrabi, J. Gao, V. Ailawadhi, G. J. Potie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, pp. 16-27, October 2000.

[16] S. Floyd and V. Jacobson, "Link Sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, Vol. 3 No. 4 pp.365-386, August 1995.

[17] M. Younis, M. Youssef and K. Arisha, "Energy-aware Routing in Cluster-Based Sensor Networks," in the *Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems(MASCOTS2002)*, Forth Worth, Texas, October 2002.

[18] G. Feng, C. Doulgeris, K. Makki and N. Pissinou, "Performance Evaluation of Delay-Constrained Least-Cost Routing Algorithms Based on Linear and Nonlinear Lagrange Relaxation," In the *Proceedings of the IEEE International Conference on Communications (ICC'2002)*, New York, April 2002.

[19] D. Shier, "On algorithms for finding the k shortest paths in a network," *Networks*, 9:195-214, 1979.

[20] J. Y. Yen, “Finding the k shortest loopless paths in a network,” *Management Science*, 17:712-716, 1971

[21] D. Epstein, “Finding the k shortest paths,” *SIAM J. Computing*, 28(2):652-673, 1998.

[22] Q.V.M. Ernesto, M.M. Marta and L.E.S Jose, “The K shortest paths problem,” *Research Report, CISUC*, June 1998.

[23] S. Singh, M. Woo and C. S. Raghavendra, "Power-Aware Routing in Mobile Ad-Hoc Networks", in the *Proceedings of ACM MOBICOM'98*, Dallas, Texas, October 1998.

[24] J.B. Andresen, T.S. Rappaport, and S. Yoshida, “Propagation Measurements and Models for Wireless Communications Channels,” *IEEE Communications Magazine*, Vol. 33, No. 1, January 1995.

[25] M. Gerla, G. Pei, and S.-J. Lee, “Wireless, Mobile Ad-Hoc Network Routing,” in the *Proceedings of IEEE/ACM FOCUS'99*, New Brunswick, NJ, May 1999.

[26] "Data sheet for the Acoustic Ballistic Module", SenTech Inc., http://www.sentech-acoustic.com/

[27] A. Buczak and V. Jamalabad, "Self-organization of a Heterogeneous Sensor Network by Genetic Algorithms," *Intelligent Engineering Systems Through Artificial Neural Networks*, C.H. Dagli, et. a.. (eds.), Vol. 8, pp. 259-264, ASME Press, New York, 1998.

[28] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE Journal on Selected Areas of Communications*, Vol. 15, No. 7, September 1997.