Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.

A Fault-tolerant Multipath Routing Protocol in Wireless Sensor Networks

Shih-Kang Huang Kuo-Feng Ssu Tzu-Ting Wu

Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan 701

Abstract-*As the need for applying wireless sensors to military and civil applications increases, the design of a reliable routing protocol that provides robustness and scalability becomes an important issue in wireless sensor networks. Previously proposed approaches were concentrated on only power failures and crash faults. This paper describes a routing scheme that not only handles crash failures but data processing errors and patterned faults. The fault-tolerant protocol can locally repair broken paths without invoking network-wide route discovery to tolerate failures. The techniques of multipath routing and majority voting are utilized to overcome data processing faults and to increase the reliability of data delivery. The experimental results show that the protocol can efficiently detect and correct errors with limited execution overhead.*

Keywords*: wireless sensor networks, routing, node-disjoint paths, fault tolerance.*

1 Introduction

Wireless sensor networking is an emerging technology that could be widely applied to many military and civil areas, such as combat field surveillance, environment monitoring, scientific measurement collection, and disaster management. A wireless sensor network is generally composed of a large number of distributed sensor nodes that organize themselves into a multi-hop wireless network. Each node is equipped with one or more sensors, embedded processors, and low-power radios, and is normally battery operated. Typically, the sensor nodes coordinate themselves to perform a common task. Data transmission through the unreliable wireless sensor network is prone to many types of failures due to environmental effects. For the reason, designing a robust routing scheme to ensure correctness of data delivery becomes an important issue. Previous work was focused on the design of single-path routing using low-rate and periodic flooding of events to avoid crashed nodes [1]. A fault-tolerant cluster scheme organized the networks into clusters to perform run-time recovery of the sensors when the gateway experienced faults [2]. A geographical routing made the use of energy awareness and geographical information to achieve failure avoidance [3]. Multi-path routing schemes, such as [4], utilized three paths for transmitting three copies of the same event from the $source$ (sensors) to $sink$ (data processing or human interface devices) to improve the resilience to node crash failures.

Some of other previous work studied fault-tolerant issues on crash failures or processing errors for cluster heads during data integration [2, 5]. Other reliable routing schemes tried to reduce the impact of link and node failures on information routing by presenting distributed and dynamic routing algorithms with lower computing overhead [6–8]. Ganesan $et\ al.$ utilized a braided multipath scheme for energy-efficient recovery from node failures in wireless sensor networks [4]. Alternate paths in a braid were partially disjoint from the primary path, not completely node-disjoint. For patterned failures, the braided multipath approach showed comparable resilience with the node-disjoint paths. However, with braid-like paths, sensor nodes near the primary path had higher probability to exhaust their power than other sensors. Moreover, the scheme did not handle failures at run-time.

This paper introduces $data\,processing\,failures$ where binary contents of data packets are changed before transmission. Bit errors in a data frame may occur before the frame's checksum is computed and appended, so the sink will not notice the errors through error-detecting codes. The data processing failures could be caused by incorrect data processing, memory leakage, and hardware/software defects. The data processing faults do not crash sensor nodes. However, they propagate erroneous information to the network and prevent the sink from receiving correct data. If the faulty sensors are responsible for battle field surveillance, the sink will not be aware of the real status of the environment, or even make incorrect decisions due to the wrong data. In the paper, a routing scheme is presented to tolerate data processing failures, crash failures, and patterned failures (transient and local crash faults) using multipath data transmission.

With the scheme, every event is duplicated and transmitted to the sink through different node-disjoint paths.

This research was supported by the Taiwan National Science Council (NSC) under contracts NSC 92-2213-E-006-042 and 93-2213-E-006-019.

The sink takes majority voting to determine if the obtained data is correct or not. Moreover, a run-time path recovery mechanism is also used to enhance path availability. Once a possible broken link is detected, the run-time recovery scheme will repair the path locally. The fault-tolerant routing protocol was successfully implemented with ns2 simulator. The performance of the protocol was evaluated with crash, relaying, and patterned failures in varying network topologies. The experimental results show that the scheme improved both data delivery and correctness. On the other hand, the fault-tolerant mechanism only incurred limited extra energy dissipation and end-to-end delay.

2 System Model

The system considered in the paper consists of n sensors and one sink, which collects all the data gathered by the sensors. Every individual sensor is capable of tracking events of interest, collecting data, and transmitting them to the sink through multi-hop wireless communication. A node which performs the sensing task is referred to as $sensing\ node$, and if the sensing node makes its decisions to send its gathered data to the sink, it will be called $source$. Thus, the environment is a multiple-source and single-sink sensor field in which communications are made via a packet radio network. All sensors are assumed to be functionally equivalent and the communication range, denoted by a radius R_t, is fixed for all sensors.

2.1 The Failure Models

The failure models are categorized into three types: data processing faults, crash faults, and patterned faults.

- **Data processing failures** represent the faults, in which binary contents in packets are modified while being processed inside sensor nodes. For example, if the data frame 1101011011 is encoded with a CRC code generator polynomial G(x)=$x^4 + x + 1$, its transmitted frame should be 11010110111110. Figure 1(a) shows the fault-free processing and delivery. However, if the last bit of the original data frame is changed to 0 before it is encoded and transmitted by the sender, the resulting frame by applying CRC code will become 11010110101101, which will still be considered correct by the receiver (Figure 1(b)). The data processing failures could be caused by noise interference, incorrect data processing, memory leakage, and so on. The definition of $data\ processing\ failure$ is described as follows. Node N receives a packet d_{pkt} and then relays it. A data processing failure occurs if the content of d_{pkt} received by node N is different from that transmitted by node N.

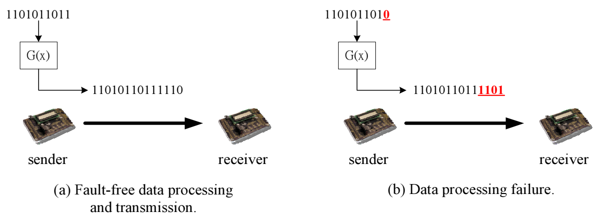

Figure 1. Data processing and transmission.

- **Crash failures** are permanent and occur independently [4]. The failure can stop a sensor node from working continuously. The sensor node is thus unable to detect events or communicate with neighboring nodes further. The crash faults may be the consequence of battery exhaustion or hardware defects due to natural disasters.

- **Patterned failures** capture geographically and transiently correlated failures. It is possible that all nodes deployed in a geographically correlated region lose connectivity with other sensors outside. The failures could occur due to environmental effects such as electromagnetic interference, natural phenomenon, and rain fades. Patterned failures in the paper are defined as following. For a circle with a radius R_p, communication capability for all sensors in the circle is disabled for a time interval.

2.2 Assumptions

The sensing nodes in the system are assumed to always produce correct data. The assumption can be achieved by a value fusion or a decision fusion [9]. The transceiver of the sink is fault-free. Each node is equipped with a positioning device such as GPS so location information can be obtained. The sensor nodes can also measure their own remaining energy. Both information can be piggybacked on either outgoing control or data packets.

3 Fault-tolerant Routing for Wireless Sensor Networks

A fault-tolerant scheme is developed to recover the three kinds of failures and increase correct data delivery. Initially, only two paths, the $primary\ path$ and $first\ alternate\ path$, are used for data transmission. The third path, called $second\ alternate\ path$, is established in advance but will not be used unless the sink receives inconsistent contents in the packets from the first two paths.

3.1 Multipath Routing Scheme

3.1.1 Multipath Mesh Construction

The first phase of our protocol is to create a mesh for modelling the network into levels according to the hop distance from the sink. The mesh construction is based on the method modified from [10]. In the beginning, the sink floods a broadcast RREQ message through the whole network. The RREQ message contains two additional fields, a hop count field and a sequence number. The hop count indicates the hop distance that the message has travelled since it left the sink. The hop count is initially set to 0 at the sink and is incremented by one every time when it is relayed. The sequence number serves as an identity of each different RREQ message generated by the sink and a larger sequence number means a more recent RREQ message. A node resides in level L if it is L hops away from the sink. For the initialization, the sink is considered at level 0 and all other sensors are defined at an infinite level. On receiving an RREQ message with hop count L, a node at level N performs one of the following actions.

1. $L < N$: The level of the node is reset to L and all entries in the routing table are deleted. The forwarder of the RREQ is inserted into the routing table as next hop back to the sink. The hop count in the RREQ is incremented by one and the message is further relayed.

2. $L = N$: The forwarder's id and energy are inserted into the routing table if no existing entry is found. Otherwise, energy information for existing entry will be updated. If the sequence number of the RREQ is greater than that recorded so far, the hop count will be incremented and the RREQ will be relayed. If not, the RREQ will be discarded.

3. $L = N{+}1$: The forwarder is in the same level. The forwarder is added as a new entry into the routing table to increase outgoing degree. The RREQ is discarded to prevent endless flooding.

4. $L > N + 1$: The RREQ is simply dropped.

As the RREQ message propagates, each node is able to determine its level and upstream neighbors.

3.1.2 Simple Aggregation

After the multipath mesh is constructed, all sensors are ready to collect data or help to forward data to the sink. Sensors that are put near may possibly sense or collect the same data within a short period of time. Energy and bandwidth will be wasted if all sensing nodes holding the identical data attempt to report the data to the sink. To solve the problem, a sensing node k is required to perform one-hop broadcast to inform its neighbors of the collected data and then wait for a time interval to receive possible one-hop broadcasts from other neighbors. If more than one node collects the same data, the following mechanism is used to determine which node should transmit the data.

$$C_k = \alpha L + (1-\alpha)\frac{1}{E_k} \tag{1}$$

Formula (1) is a simplified evaluation proposed in [11]. α is a tunable weight ranging from 0 to 1, L is the level which node k resides, and E_k is the remaining energy of node k. L and E_k are piggybacked onto the one-hop broadcast messages. The node with the smallest cost C_k sends its collected data to the sink. If two or more nodes have the same C_k, the node with lower ID will be selected.

3.1.3 Node Disjoint Multipath

The sensing node that is responsible for transmitting data chooses one of its neighbors from the forward routing table as its next hop based on Formula (1). The neighbor with the lowest cost is considered to be the best forwarder. By applying the rule hop by hop, the data packet will be eventually delivered to the sink. The resulting route is referred to as $primary\ path$. All data packets passing through the primary path have a flag, $primary\_path$, in the packet header.
Once the sink receives the data packet from the primary path, it will reply an acknowledgement to the source. As the acknowledgement is travelling back to the source node, every intermediate node attaches its location information on the packet. As a result, when the source receives the acknowledgement, all location information of the intermediate nodes on primary path can be recorded.
To tolerate patterned faults, each node in the first alternate path should keep a "threshold" distance away from the primary path. To generate the first alternate path, the source first examines if its one-hop closer neighbors have enough distance from the primary path. The qualified neighbor with the smallest distance is chosen as the next hop. The sink will acknowledge the source when the packet arrives. The source thus can record all intermediate nodes for the first alternate path.
Once the sink detects data processing failures, it notifies the source to activate the second alternate path for further data transmissions. The second alternate path needs to be created in advance so that the notification could be sent to the source through it in a reverse direction. To prevent the alternate path from being broken before it is activated, the intermediate nodes on the path are required to have multiple choices to forward data packets. The second alternate path is created by broadcasting a "diffuse packet", to which the location information of all the intermediate nodes on the primary and the first

alternate paths are attached. By propagating the diffuse packet towards the sink, nodes will either drop or relay this packet according to the attached location information. Nodes that are already on the primary path, first alternate path, or too close to them will simply drop the message; otherwise, the links will be recorded into the routing table as the second alternate path.

3.1.4 Majority Voting at Sink

Table 1. Results of Majority Voting

Primary Path	1st Alternate Path	2nd Alternate Path	Failure Detection	Failure Correction	Failure Occurrence
fault-free	fault-free	-	-	-	No
faulty/fault-free	loss	loss	Yes	No	Yes
fault-free	faulty	loss	Yes	No	Yes
fault-free	faulty	fault-free	Yes	Yes	Yes
faulty	fault-free	fault-free	Yes	Yes	Yes
faulty-1	faulty-2	all	Yes	No	Yes
faulty-1	loss	faulty-2	Yes	No	Yes
loss	faulty-1	faulty-2	Yes	No	Yes
faulty-1	faulty-1	-	No	No	Yes
loss	fault-free	loss	Yes	No	Yes
faulty-1	faulty-2	faulty-1/faulty-2	Yes	No	Yes
loss	loss	-	No	No	Yes

When a sensing node detects an interested event and is also responsible for reporting, the node will immediately generate a data packet (src, d, seq), where src, d, and seq represent the sensing $node's\ id$, $sensed\ data$, and $sequence\ number$, respectively. Another copy of the packet (src, d, seq) is then transmitted through the first alternate path. The sink compares data fields d in the packets with the same sequence number. If the two fields are identical, the data transmission is considered correct. Otherwise, it is obvious that one or two of the paths have encountered data processing failures. The sink notifies the source by sending a control message through reverse link of the second alternate path. The source then sends another copy using the second alternate path. On receiving data packets generated by the same source with the identical sequence number, the sink takes majority voting to determine the correct data. If the majority is reached, the failure is considered recovered; otherwise, (e.g., three different contents or lost packets), the data packets will be dropped. Possible results $R_{src,seq}$ of the majority voting is summarized in Table 1.

3.1.5 Failures on Next Hop

When a node's next hop encounters crash or patterned failure, our protocol will detour to the sink. There are two cases for a node N attempting to deliver packets.

1. **At least one neighbor is active:**

   Based on the instinct of MAC layer protocol, node N is aware of the transmission failure. Node N tries to forward the packet to a different neighbor

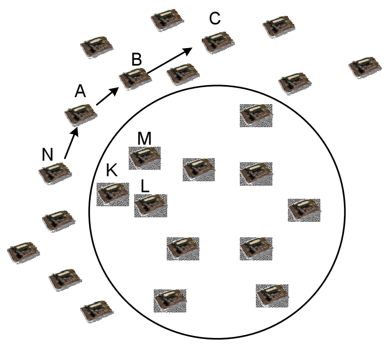

Figure 2. Detouring patterned failure.

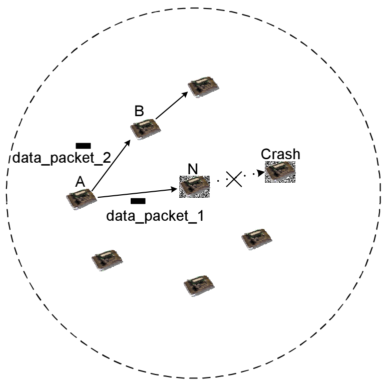

Figure 3. Avoiding crash failure.

   by looking up its forward routing table. Neighboring nodes that are closer to the sink and are not in other paths will be picked first. If there are no such nodes, those with the same level as node N will be used. As Figure 2 shows, node K, L and M are not able to help node N to forward data packets since they suffer from patterned failures. Node N forwards the data packets to node A instead and node A then forwards the data packets to node B. Therefore, the data packet could be routed progressively towards the sink but with potentially higher latency.

   The disconnected next hop neighbor is not immediately removed from the routing table. Each entry in the forward routing table has a counter named $failure\_count$. Node N uses the counter to record the number of times of failed transmissions for every neighbor. The counter is initially set to zero. The value is incremented for each transmission failure and is reset to zero if the link is recovered. To prevent node N from keeping transmitting to a permanently failed neighbor, a threshold is defined for the $failure\_count$. If the number of failed transmissions for a specific neighbor U reaches the threshold, the neighbor will be deleted from node N's routing table.

2. **No available neighbors exist:**

Node N has no neighbors to forward data so the packets sent to node N are dropped as shown in Figure 3. To prevent subsequent packets from keeping discarded, node N needs to warn its downstream neighbors about such kind of broken path. Node N generates a one-hop broadcast message called $remove\_me$. On receiving this message, a neighbor first searches its routing table and then removes node N from the routing table if found. The subsequent packet, $data\_packet\_2$, will be routed through node B instead of node N so the broken path could be avoided.

4 Performance Evaluation

4.1 Simulation Environment

The network simulator 2 (ns2) [12] was used for performance evaluation of the protocol. The distributed coordination function (DCF) of the IEEE 802.11 was utilized as the MAC layer protocol in the experiments. The IEEE 802.11 DCF uses Request-To-Send (RTS) and Clear-To-Send (CTS) control packets to provide virtual carrier sensing for unicast data packets. Each data transmission is followed by an acknowledgement (ACK). Broadcast packets are sent using CSMA/CA only and no acknowledgement is produced.

The sensor field, a 500m$\times$500m area, consisted of 600 sensor nodes that were uniformly distributed. The sink was randomly located in the field. Sensor nodes were functionally equivalent and had the fixed radio propagation range of 40 meters. The energy dispatching rates of battery power drain for transmission, reception, and standby were 660mW, 395mW, and 35mW, respectively.

4.2 Failure Injection

The simulation on crash failures was based on a Poisson distribution with parameter λ_c. Once the injection time arrived, an integer n_c was selected by a normal distribution with mean μ_c and standard deviation σ_c. n_c nodes on the sensor field were then selected according to Uniform distribution to simulate crash failures.

Data processing failures were injected based on a Poisson distribution with parameter λ_r. The way of faulty node selection was the same as simulating crash failures. The faulty nodes were given a data processing failure duration δ_r. If a sensor node with the data processing fault received a data packet, one randomly chosen bit in the data field (8 bits) was toggled before relaying.

The injection time of patterned failure was based on a Poisson distribution with a parameter λ_p. For each failure injection, n_p points on the field were randomly determined with μ_p and a standard deviation σ_p. All the nodes within a radius R_p (30 meters) of the selected points were turned off for an average duration δ_p. All parameter settings used for the simulation are shown in Table 2.

Table 2. Parameter Settings for Failure Injection

Parameters	Crash Failure	Data Processing Failure	Patterned Failure
Average Injection Time	λ_c=100 sec.	λ_r=400, 267, 200 and 160 sec.	λ_p=300 sec.
Mean Number of Failures/Injection	μ_c=20	μ_r=30	μ_p=3
Standard Deviation	σ_c=2	σ_r=2	σ_p=1
Duration for Transient Failures	-	δ_r=15 sec.	δ_p=20 sec.

Table 3. Failure Detection Performance

$(1/\lambda_r)$	N(Failures)	N(Failures)			N(Undetected Failures)
		Correctable	Uncorrectable	Total	
1/400	19.6	14.0 (71.4%)	5.0 (25.5%)	19.0 (96.9%)	0.6 (3.1%)
1/267	41.4	31.4 (75.9%)	8.4 (20.3%)	39.8 (96.2%)	0.7 (1.7%)
1/200	52.2	39.6 (75.8%)	12.1 (23.2%)	51.7 (99.0%)	0.5 (1.0%)
1/160	63.3	47.2 (74.6%)	15.5 (24.5%)	62.7 (99.1%)	0.6 (0.9%)

Table 4. Successful Delivery Comparison

Data Processing Failure Rate $(1/\lambda_r)$	Total Events	Fault Tolerance	Without Fault Tolerance	Improvement (%)
1/400	1055	943.1 (89.4%)	825.9 (78.3%)	14.2%
1/267	1080.7	956.3 (88.5%)	801.4 (74.2%)	19.3%
1/200	1071.2	941.2 (87.9%)	772.3 (72.1%)	21.8%
1/160	1063.7	910.3 (85.6%)	750.2 (70.5%)	24.6%

4.3 Experimental Results

Two routing protocols, with and without fault-tolerant mechanism, were compared in the simulation. With the non-fault-tolerant protocol, a source node utilized a randomly selected path to transmit data packets to the sink. No local route recovery was taken when a path ran into failures. The both routing schemes were executed on five different network topologies with the simulation time of 800 seconds.

Table 3 displays the performance for failure detection by applying fault-tolerant mechanism with four varying data processing failure rates. Failures were detected if the sinks did not receive at least two identical values for the same event. Only 1.7% of the failures were not detected. As the data processing failure rates increased, the ratio for failure detection also grew. The reason is that the probability of encountering data processing failures became higher, and thus the second alternate paths were more likely to be activated. With the help of local path repairing, the duplications of events could be routed to the sinks with fewer losses so fault detection was more effective.

The successful delivery means that the data generated by the source nodes are correctly received by the sinks.

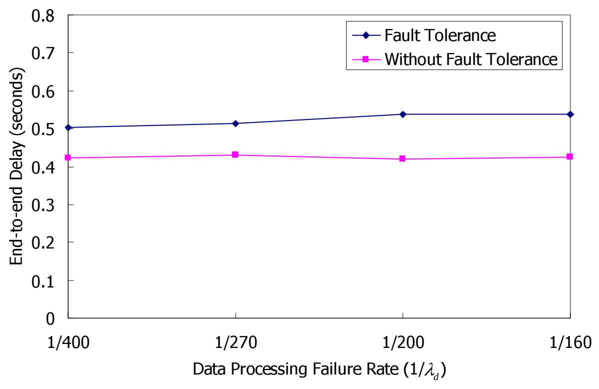

Figure 4. End-to-end delay between source and sink.

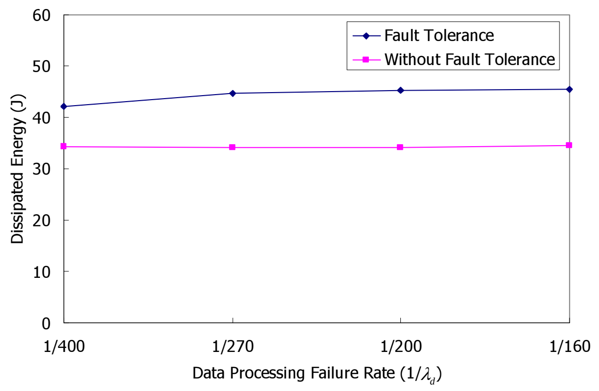

Figure 5. Average dissipated energy.

Table 4 compares the successful delivery between the fault-tolerant and non-fault-tolerant schemes. The fault-tolerant scheme accurately delivered about 88% of the events, whereas the average successful ratio for non-fault-tolerant counterpart was less than 75%. The fault-tolerant routing protocol had average improvement of 19.98% in the successful delivery ratio to the non-fault-tolerant scheme. The utilization of the second alternate paths and the avoidance mechanism for next-hop failures improved the likelihood of recovering data processing failures.

End-to-end delay and dissipated energy were used to measure the overhead for the fault-tolerant protocol. Figure 4 illustrates the end-to-end delay for both schemes. The fault-tolerant protocol spent 0.12 more second to transmit a packet compared to the original protocol. The delay was mainly from the detour around failures on next hop and random back-off for medium contention. The end-to-end delay did not increase dramatically with the higher data processing failure rates.

The difference of energy dissipation between the two routing schemes is shown in Figure 5. The fault-tolerant protocol required about 27.7% of the extra energy on average due to the redundant data transmission. The energy dissipation increased with the higher data processing failure rate. The phenomenon resulted from not only the utilization of the second alternate paths, but also the impact of disseminating diffuse packets, which were used to create the second alternate paths in advance.

5 Conclusion

This paper presented the use of run-time path recovery scheme to detour around crash and patterned failures. Utilizing node-disjoint multiple paths for routing duplicated collected data, the sink can take majority voting on them to detect most of the $data\ processing\ failures$. These mechanisms are important in a wireless sensor network that is composed of vulnerable micro sensors and has unreliable communication. The experimental results demonstrated that the fault-tolerant routing protocol detected 98% of the inconsistent data received at sink and corrected 76% of them. The protocol also improved the successful delivery ratio to 88% in the presence of failures. On the other hand, the performance overhead for transmission delay and energy consumption incurred by the fault-tolerant protocol was limited.

References

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed Diffusion for Wireless Sensor Networking,” *ACM/IEEE Transactions on Networking*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[2] G. Gupta and M. Younis, “Fault-tolerant clustering of wireless sensor networks,” *IEEE Wireless Communications and Networking*, pp. 1579–1584, Mar. 2003.

[3] Y. Yu, R. Govindan, and D. Estrin, “Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks,” *UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023*, May 2001.

[4] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, “Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks,” *ACM Mobile Computing and Communications Review*, vol. 5, no. 4, pp. 11–24, Oct. 2001.

[5] L. Prasad, S. S. Iyengar, R. L. Rao, and R. L. Kashyap, “Fault-Tolerant Sensor Integration Using Multiresolution Decomposition,” *Physical Review E*, vol. 49, no. 4, pp. 3452–3461, Feb. 1994.

[6] S. S. Iyengar, M. B. Sharma, and R. L. Kashyap, “Information Routing and Reliability Issues in Distributed Sensor Networks,” *IEEE Transactions on Computers*, vol. 40, no. 2, pp. 3012–3021, Dec. 1992.

[7] S. S. Iyenga, D. N. Jayasimha, and D. Nadig, “A Versatile Architecture for the Distributed Sensor Integration Problem,” *IEEE Transactions on Computers*, vol. 43, no. 2, pp. 175–185, Feb. 1994.

[8] C. S. Chiu, K. F. Ssu, and C. H. Chou, “A Reliable Multipath Routing Protocol in Mobile Ad Hoc Networks,” *National Computer Synposium*, pp. 892–899, Dec. 2003.

[9] T. Clouqueur, K. K. Saluja, and P. Ramanathan, “Fault Tolerance in Collaborative Sensor Networks for Target Detection,” *IEEE Transactions on Computers*, vol. 53, pp. 320–333, Mar. 2004.

[10] X. Hong, M. Gerla, H. Wang, and L. Clare, “Load balanced, energy-aware communications for Mars sensor networks,” *IEEE Aerospace Conference Proceedings*, pp. 9–16, Mar. 2002.

[11] M. Younis, M. Youssef, and K. Arisha, “Energy-Aware Routing in Cluster-Based Sensor Networks,” *Proceedings of IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pp. 129–136, Oct. 2002.

[12] The Network Simulator - ns-2. URL http://www.isi.edu/nsnam/ns/.