Ant Colony-Based *Many-to-One* Sensory Data Routing in Wireless Sensor Networks

Reza GhasemAghaei, ASM Mahfujur Rahman, Md. Abdur Rahman, Wail Gueaieb[1] and Abdulmotaleb El Saddik
Multimedia Communication Research Laboratory (MCRLab)
[1]Machine Intelligence, Robotics, and Mechatronics (MIRAM) Laboratory
School of Information Technology and Engineering
University of Ottawa, Canada
{reza, kafi, rahman, abed}@mcrlab.uottawa.ca, [1]wgueaieb@site.uottawa.ca

Abstract

An ant colony-based routing protocol is presented in this paper that is specifically designed to route many-to-one sensory data in a multi-hop Wireless Sensor Network (WSN). Because a many-to-one routing paradigm generates lots of traffic in a multi-hop WSN resulting in greater energy wastage, higher end-to-end delay and packet loss, the proposed routing protocol also comes with a lightweight congestion control mechanism, which is capable of handling both event-based and periodic upstream sensory data flow to the base station. The proposed protocol works in two-phases. During the first phase, the protocol uses ant-based intelligence to find and enforce the shortest path and in the second phase, when the actual many-to-one sensory data transmission takes place, the protocol combines the knowledge gained during the first phase with the congestion control mechanism to avoid packet loss and traffic while routing the sensory data. When compared with the related algorithms, the proposed algorithm shows promising results.

1. Introduction

Wireless Sensor Networks play an important role in many applications where it is not feasible to employ human beings. This spans monitoring remote facilities such as structural and industrial process, military applications, environment and habitat monitoring, health care applications, home automation, security threat observation, and traffic monitoring to name a few. In these applications, the sensed data needs to be propagated to the remote base station for further analysis. Most of such applications employ multi-hop WSN to deliver the sensory data from the sensing area to the base station. This requires efficient routing protocol to save unnecessary energy waste, to allocate bandwidth fairly, and to find the shortest path within a satisfactory delay. Many of the WSN applications such as intrusion detection, earthquake assessment, remote object tracking, to name a few, require that the captured event by *n* number of sensor nodes be transmitted to the base station within tolerable packet loss. Some literature calls this a *many-to-one* type of communication [1]. The traffic that needs to be handled by the underlying *many-to-one* routing protocol depends on the type of application. For example, a WSN employed to deliver certain sensory data periodically to the base station will pose different constraints on the underlying protocol than that used to deliver a huge burst of sensory data such as the one generated due to an earthquake or a forest fire. The underlying routing protocol must be designed to handle the traffic accordingly.

Among different routing approaches, ant-inspired intelligent algorithms show promising results in solving routing problems in sensor networks [2], [3], [4]. In this work, we propose an ant colony-based sensory data routing protocol called *Many-to-One* Improved Adaptive Routing (MO-IAR) protocol, which is able to route the upstream data flow through the shortest path by avoiding congestion. MO-IAR is capable of handling both event-based and periodic *many-to-one* sensory data flow. The proposed protocol operates in two phases. During the first phase, we employ forward ants and backward ants [3] to find the shortest route within a multi-hop WSN. What follows in the second phase is that the data ants route the actual sensory data through the shortest path. Data ants use a congestion control algorithm to avoid collision. When more than one neighborhood sensor tries to send data to a common next-hop neighbor, it results in collision over the wireless channel. Most popular congestion control mechanisms either follow the TDMA or CSMA approach. CSMA is preferred for its less complexity when the density of sensor nodes is low to

978-1-4244-1968-5/08/$25.00 ©2008 IEEE

moderate. Because MO-IAR only uses local neighborhood information for routing, we use a modified version of CSMA/CA to avoid complexity in implementing the congestion control mechanism.

The rest of the paper is organized as follows. In Section II we present some closely related works. We describe the proposed two phase routing protocol in Section III. In Section IV we present the simulation results and conclude the paper in Section V.

2. Related work

The MO-IAR was originally inspired by the research work explained in [3], [4]. The authors of [3] show an adaptive distributed, mobile agent-based basic ant routing algorithm that lays the basic foundation of the current work. However, the algorithms proposed there are not optimized for routing in wireless sensor networks and do not support *many-to-one* routing. Zhang et al. [4] proposed three ant-routing algorithms for sensor networks: the Sensor-driven Cost-aware Ant Routing (SC) algorithm, the Flooded Forward Ant Routing (FF) algorithm, and the Flooded Piggybacked Ant Routing (FP) algorithm. Although the algorithms show significant results, none of them are ideal for routing in wireless sensor networks for the above-mentioned reasons. Moreover, none of the three algorithms take into consideration the concept of reinforcement learning and *many-to-one* data routing.

STEER [5] is a geographic location-based routing protocol that offers an energy efficient and reliable upstream data delivery. Rather than choosing the next-hop neighbor by the sender for routing the packet (transmitter-oriented), a sender simply broadcasts a packet and one of the neighbors is elected to be the next-hop receiver (receiver-oriented) based on its closeness to the sink, which is termed as the temporal gradient. What follows is the real data communication from the sender to the newly elected next-hop. This process continues until the packet reaches the sink. However, STEER solely depends on the global sensor node localization system. The next-hop selection process of STEER might pose a significant amount of delay in case no suitable neighbor node is selected. Also, STEER does not provide *many-to-one* routing facilities and hence, congestion control is not provided. While STEER only chooses the next-hop based on closeness to the sink (farthest from the sender node), MO-IAR considers both the closeness from the sender to the next-hop node as well as the closeness from next-hop to the destination.

SELAR [6] is a location-based routing protocol that takes into account the location and energy level of neighboring nodes as well as the location of the sink node while routing a packet. In order to route the data packets destined to the sink, each sensor node first calculates the probable neighbors who fall within a predefined angular area to make sure that the packet is heading towards the sink. If more than one candidate node is found within the area, the packet is routed to the node with the highest energy level. In case, no routable node within that region is found, SELAR uses a gossiping technique for more robustness. The basic difference between SELAR and the proposed protocol is that MO-IAR uses the ant-based routing algorithm that converges very fast toward the destination and supports *many-to-one* data routing using effective congestion control mechanisms.

In summary, many protocols have been proposed, which independently survey different aspects of routing and congestion control mechanisms suitable for *many-to-one* sensory data. Some of them can be found in [6], [7]. However, this paper focuses on the cross layer optimization of adding a congestion control algorithm with a *many-to-one* ant-based routing protocol specially tailored for WSNs.

3. Proposed protocol overview

The proposed routing protocol relies on the localization information of each sensor node. The protocol is composed of two phases. The first phase is concerned with finding the best route using ant colony optimization and swarm intelligence. The second phase starts when the actual data routing takes place, which employs the congestion control mechanism to avoid possible collisions. Rather than working with collision detection, the algorithm behaves in a pro-active manner to mitigate the collision. Now we briefly describe each phase of the protocol.

Phase1: The protocol assumes that each sensor node knows its location and the location of the destination a priori. This can be achieved using GPS technology for example. Once the sensor nodes are initially deployed, each sensor node locally broadcasts a HELLO message to its neighbors to form the neighborhood table. In the rest of the paper we will assume that the neighborhood table is the same as the routing table. Once a sensor node hears any HELLO message from any of its neighbor, it records the ID, location of the neighboring node and distance between them. After a predefined time, depending on the density of the WSN, the protocol assumes that all the sensor nodes have identified their local neighbors and thus the protocol initiates phase 1. We assume that the sensor network has a static destination node and N

number of sensor nodes among which each sensor node sends n number of forward ants successively to find the shortest path between itself and the sink. Each forward ant uses the ant-routing algorithm detailed in [2] to find the best next-hop neighbor node who is both closer to itself and closest to the sink using probabilistic theory. Before leaving the current node, the forward ant updates the probability value of the chosen next-hop neighbor in the routing table as well. Details of the algorithm are presented later on. The protocol guides a forward ant toward the destination such that it visits the least number of nodes to find out the shortest path. Each forward ant also carries some global parameters in its packet header such as details of visited nodes, corresponding probability values, total number of hops visited, distance between each link, and neighbors of the visited nodes. The same process goes on until the destination is reached. After reaching the destination, the forward ant creates a backward ant, destroys itself, and hands over the global parameters to the backward ant. Algorithm 1 represents the basic operations of a forward ant.

Algorithm 1: MO-IAR Algorithm for a single Forward-Ant

```
Forward-Ant(source-node, current-node, destination-node)
begin
    if destination is reached then
        Create a new Backward-ant and Copy forward-list to backward-list
    else
        Add current-node to forward-list
    if Less than half of the sensor nodes visited then
        Choose the next neighbor according to the following probability
        P'_{i,d} = A_{i,d} × (P_{i,d} + β × C_{k,i}) / (1 + β × (|N| − 1))
        if selected neighbor visited then
            choose another neighbor node
    else
        Perish Forward-Ant
end
```

The backward ant follows the same trail followed by its parent forward ant, and reinforces the path by changing the probability values. A forward ant updates the probability value of the next-hop neighbor in the routing table of the current node. While, the backward ant first increases the probability value of the current node in its routing table if it was visited earlier by the parent forward ant and decreases the probability value of the non-visited neighbor nodes of the current node, as shown in algorithm 2. Upon reaching the source node back, the backward destroys itself. The similar shortest path exploration process goes on for the next (n-1) number of forward and backward ants. The shortest path between each source node and the given destination is calculated once the n^{th} backward ant reaches the source. A similar process takes place in the first phase for each of the (N-1) number of sensor nodes.

Algorithm 2: MO-IAR Algorithm for Backward-Ant

```
Backward-Ant(source-node, current-node, destination-node)
begin
    if source is reached then
        Perish Backward-Ant
    else
        for all neighbor nodes do
            if selected neighbor node visited by Forward-Ant then
                P_{i,d} = (P_{i,d} + ΔP) / (1 + ΔP)
            else
                P_{i,d} = P_{i,d} / (1 + ΔP)
            Remove first item of Backward-list
        Move Backward-Ant to selected neighbor node
end
```

Now we formally present the algorithms used by the protocol to find the shortest path. Algorithm 1 is used by the forward ant and algorithm 2 is used by the backward ant to find the shortest path between each sensor node and the sink with high energy efficiency, less latency, and a less packet loss rate. First we introduce the notations used in the algorithms. $|N_k|$ is the set of neighbors of node k, $C_{k,i}$ is the correction factor for adapting the cost of routing between the current node k and the next-hop node i. $A_{i,d}$ is a heuristic correction factor for adapting the cost of routing between the next-hop node i and the destination node d. $D_{k,i}$ is the distance between the current node k and the next-hop node i. $D_{i,d}$ is the distance between the next-hop node i and destination d. The distance D between two points is calculated using simple co-ordinate geometry. λ and γ are coefficient factors with values between 0 and 1. β is defined as the desirability of the correction factor $C_{k,i}$ with a value between 0 and 1. $d^{i}_{k,d}$ is the distance between the current node k and the destination node d via the neighbor node i, which is ($D_{k,i} + D_{i,d}$), and $P_{i,d}$ is the probability of choosing node i as the next-hop node by the current node k toward the sink/destination node d. However, more details can be found in [2].

Phase2: As soon as the shortest path between each of the (N-1) sensor nodes and the destination is known, the protocol initiates phase 2. This phase employs the data ants to route the actual data captured by $|N_S|$ number of sensor nodes, also referred to as source nodes, destined for the destination node. We define two extreme scenarios for this phase. One of the scenarios might arise if the sensory data generated by

the captured events do not follow any common shortest path. In this case, there will be no congestion at all and the results should be similar to the one proposed in [2]. This special case is illustrated in Figure 1, which shows the shortest paths between the source nodes 1, 4, 8, 9, 10, 26, and 34 to the destination node 18. The figure also shows that all the shortest paths have disjoint sensor nodes and thus, each source node can send its captured packets in parallel. In this scenario, the source nodes do not need to implement any congestion avoidance algorithms.

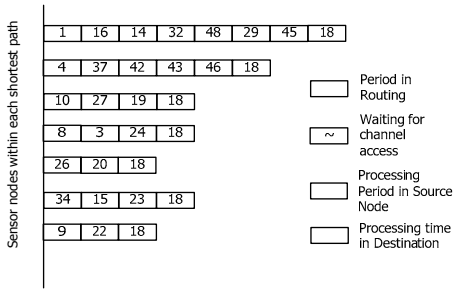

Figure 1. A *Many-to-one* routing scenario where none of the source nodes have a common next-hop neighbor node.

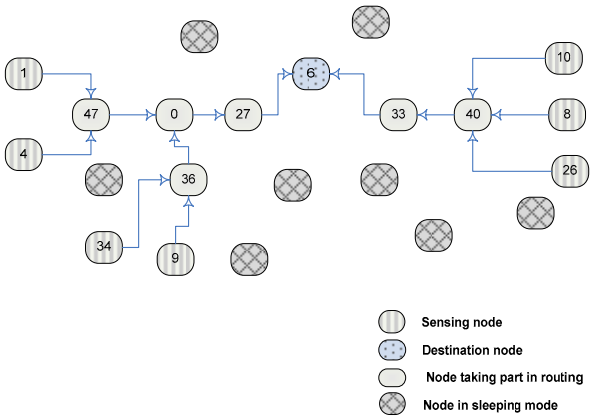

Figure 2. A many-to-one routing scenario where several sensor nodes sense an event at any given moment and try to route the sensory data using the shortest path to the sink.

The second scenario arises when two or more source nodes have at least one common neighbor in their shortest path. This scenario might be the result of periodic data captured by several sensor nodes or a number of events that simultaneously trigger several sensor nodes within the WSN. In this case, the protocol implements a lightweight congestion control mechanism to avoid collision. The novelty of this approach is that by mapping the shortest path with the neighborhood information, a data ant can easily guess the collision probability. Figure 2 illustrates how a data ant intelligently guesses the number of probable source nodes that might be contending to access the channel for sending their captured sensory data.

Looking at the shortest path provided by phase 1 and mapping it with the routing table, a data ant intelligently guesses the neighbors that might be under collision. As shown in Figure 2, 7 source nodes; node 1, 4, 8, 9, 10, 26 and 34 have sensing data that needs to be sent to destination node 6. Because node 1 and 4 have a common neighbor, which is 47, node 9 and 34 have a common neighbor 36, and node 8, 10 and 26 have a common neighbor 40, they need to run the congestion control algorithm to avoid any packet loss. Algorithm 3 serves as the congestion control mechanism implemented by each data ant experiencing congestion.

Algorithm 3: MO-IAR Algorithm for Data-Ant

```
Data-Ant(source-node, current-node, destination-node, shortest-path, packet)
begin
    if destination is reached then
        Perish Data-Ant
    else
        if current-node == source-node then
            Find the neighboring nodes' shortest paths from the routing table
            Determine probability of collision based on common next-hop neighbor
            Run binary exponential backoff algorithm to calculate sending delay
            Send packet to the next-hop neighbor
        else
            Enqueue route-thru packets in the buffer
            Dequeue to transmit packet to the next-hop neighbor in the shortest
            path
end
```

As shown in algorithm 3, the data ant has two salient states; the initial state when it remains within the source node and the routing state where it travels the route-thru nodes. In the first state, the congestion control algorithm implements the binary exponential backoff algorithm to regulate the channel access by each contending data ant and in the routing state the congestion control algorithm maintains the buffer to the route-thru traffic.

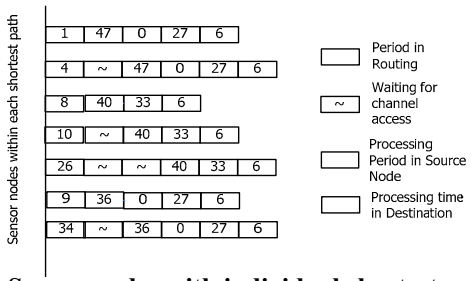

Figure 3. Source nodes with individual shortest path that gives the idea of common neighborhood node.

We use Figure 3 to explain the different states mentioned in algorithm 3. When the data ant is in the source node, it waits to access the channel to forward the captured sensory data to the next-hop neighbor. Each contending source node runs an exponential backoff algorithm to determine the delay offset. The protocol schedules the channel access in the following manner. The node having the lowest node index will send its packet first without any delay. The subsequent nodes use the binary exponential backoff algorithm to calculate their channel access time. For example, according to the proposed algorithm 3, between nodes 1 and 4, node 1 has the lowest index and thus sends its packet immediately. While node with index 1 is busy in transmitting, node 4 waits for the time defined by the binary exponential algorithm to avoid collision with node 1's transmission. The same occurs for the other source nodes, as shown in Figure 3. Node 1, 8 and 9 enjoy channel access without any delay while their contending counterpart(s) wait until the channel is clear. The process goes on until the contention is over. Once a packet enters a route, it enters the second state i.e. routing state. Each intermediate routing node maintains a queue similar to the one shown in Figure 4 where the algorithm implements a queue-based packet scheduling algorithm to service the route-thru packets.

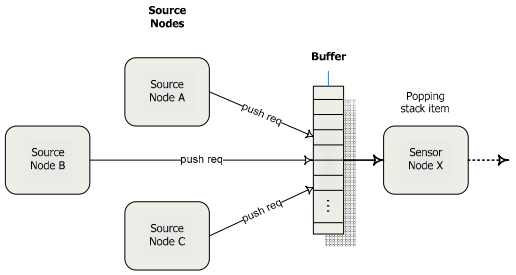

Figure 4. Buffer management scheme for servicing route-thru packets

Due to the convergence nature of the *many-to-one* routing paradigm, the shortest paths might merge or cross over at any intermediate node. Thus, the congestion control algorithm pushes the incoming route-thru packets to the queue and provides service according to the First Come First Serve (FCFS) principle. Regarding the destination node, we assume that it is not limited in buffer or bandwidth to handle incoming traffic from its neighboring nodes.

4. Performance evaluation

In order to test the performance of the proposed routing protocol, we have designed a java-based simulator. The simulator helps initializing the number of sensor nodes to be deployed, their spatial location and their connectivity. It also allows selecting the source nodes, the destination node, types of algorithm to compare with, and their parameters. It has two modes of operation: the nodes can be randomly generated or uniformly distributed. For simulations, we used networks with randomly distributed nodes as it is more realistic. Based on the connectivity and spatial layout of the sensor nodes, the traffic and the degree of congestion might vary. We have modified the works presented in [3], and [4] by adding our proposed *many-to-one* routing algorithm i.e. algorithm 3 to compare their performance with MO-IAR. Before presenting the simulation results, we first introduce several simulation parameters.

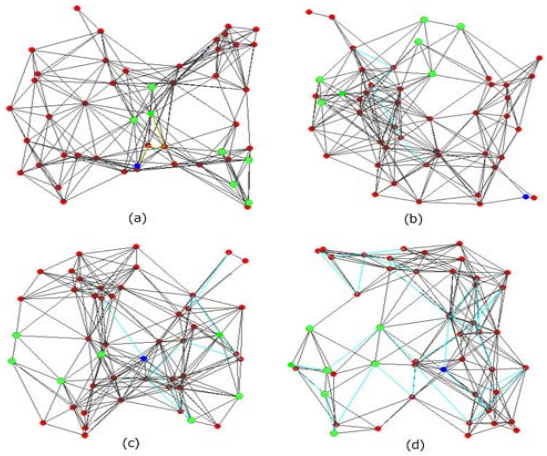

Figure 5. Four WSN models for evaluating MO-IAR

We chose the total number of sensor nodes N as 49, total number of source nodes N_S as 7 and the range of wireless transmission of each antenna as 10 meters. For phase 1, the simulation runs for a total of 200 seconds to find the shortest paths. The values of λ, γ, and β were chosen as 0.7, 0.5 and 0.1 respectively. We choose four different WSN testbeds (see Figure 5) to evaluate MO-IAR and compared it with the Basic, SC, FF, and FP algorithms. We evaluate MO-IAR and the Basic, SC, FF, FP algorithms in terms of the total time taken to transmit data from 7 source nodes to a particular destination and the congestion observed during the routing process.

In order to find the global percentage of total time taken by each algorithm, we first take each WSN shown in Figure 5, run each of the 4 simulations 10 times, calculate the total routing time of each algorithm for each simulation and finally, make the average of the total time. Figure 6 shows the percentage of total time taken by each algorithm. The time taken by each algorithm for each WSN testbed includes the 7 data

ants traveling from the source nodes to the destination by taking into account the routing time, waiting in queue, and the backoff time. From the figure, we conclude that our proposed MO-IAR algorithm requires the least amount of time (10.6%) in delivering the source packets to the destination.

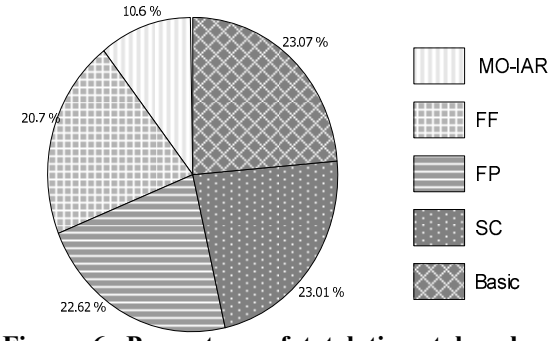

Figure 6. Percentage of total time taken by each algorithm during the simulation period

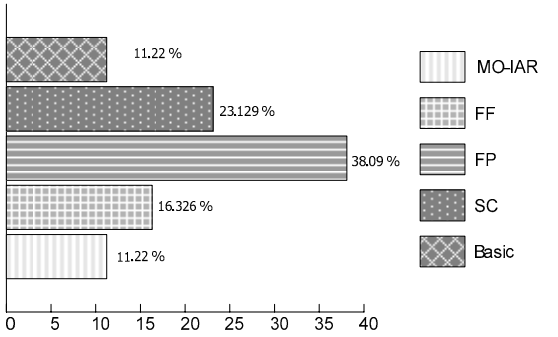

Figure 7. Percentage of total collisions observed by each algorithm during the simulation period

We now illustrate the congestion scenario of the algorithms. If the shortest paths generated during phase 1 are completely disjoint in terms of common neighbor, it results in no collision. Such a situation might stem from the shortest paths shown in Figure 1. However, the network architecture does not always ensure such a situation. Thus, we take the four WSN testbeds shown in Figure 5 to calculate the percentage of collisions suffered by each routing algorithm. Figure 7 shows the percentage of collision by taking into account all four networks during the simulation time. MO-IAR suffers lower collision than SC, FF and FP algorithms while it shows similar congestion behavior like the Basic algorithm. This is because when we extended the Basic algorithm [3] by incorporating our proposed *many-to-one* approach, it came out that the Basic algorithm uses long disjoint paths. This generates a high end-to-end delay as shown in Figure 6 but produces relatively less congestion. On the other hand, MO-IAR often finds the shortest paths in such a way that some closely located neighbors share some portions of the path depending on the network connectivity. This generates the least end-to-end delay, as shown in Figure 6, while causing similar congestion as that of the Basic algorithm [3].

5. Conclusion

In this paper we proposed an ant-colony based routing protocol that is specifically tailored for routing upstream *many-to-one* sensory data. The routing algorithm is coupled with a lightweight congestion control algorithm that helps in mitigating the collision. The protocol operates in two phases. During phase 1, it finds the shortest paths between each source node to a particular destination node and the 2nd phase uses these shortest paths to intelligently avoid the probable collisions. When compared with other related algorithms, MO-IAR outperforms them in terms of finding the shortest path within the least amount of overall time. The congestion behavior of MO-IAR was also satisfactory.

We envision several future projects. One of them is evaluating multimedia traffic over *many-to-one* routing protocol. In our future endeavors, we also plan to choose the shortest path by not only distance but also the remaining energy.

6. References

[1] C. T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *Proc. of the 2nd Int. conf. on Embedded networked sensor syst*ems, Baltimore, MD, USA, 2004, pp. 148-161.

[2] R. GhasemAghaei, M. A. Rahman, W. Gueaieb, and A. El Saddik, "Ant Colony-Based Reinforcement Learning Algorithm for Routing in Wireless Sensor Networks," *IEEE IMTC* 2007, Warsaw, Poland, May 2007.

[3] G. D. Caro, and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," *J. of Art. Intel. Research*, vol. 9, Dec. 1998, pp. 317-365.

[4] Y. Zhang, L. D. Kuhn, and M. P. J. Fromherz, "Improvements on Ant Routing for Sensor Networks," *Int. Workshop on Ant Colony Optimization and Swarm Intelligence*, Sep. 2004.

[5] M. Chen, T. Kwon, S. Mao, and V. Leung, "Spatial Temporal Relation-Based Energy-Efficient Reliable Routing Protocol in Wireless Sensor Networks," *Int. J. of Sensor Networks*, May 2007.

[6] G. Lukachan and M. A. Labrador, "SELAR: Scalable Energy-Efficient Location Aided Routing Protocol for Wireless Sensor Networks," in *Proc. of the 29th Annual IEEE Int. Conf. on Local Comp. Networks (LCN04)*, 2004.

[7] Y. G. Iyer, S. Gandham, and S. Venkatesan, "STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks," in *Proc. of IEEE ICCCN* 2005, San Diego, CA, Oct. 17–19, 2005.