A Lightweight Framework for Source-to-Sink Data Transfer in Wireless Sensor Networks

J. Jobin†, Zhenqiang Ye†,Honomount Rawat†,Srikanth Krishnamurthy†, Satish K. Tripathi‡
†*Dept. of Computer Science & Engineering*
University of California, Riverside, CA 92521
{*jobin,zye,hrawat,krish*}*@cs.ucr.edu*
‡*Dept. of Computer Science & Engineering*
University of Buffalo, SUNY Buffalo, NY 14260
tripathi@buffalo.edu

***Abstract*— Lightweight protocols that are both bandwidth and power thrifty are desirable for sensor networks. In addition, for many sensor network applications, timeliness of data delivery at a sink that collects and interprets raw sensor data is of great importance. In this work, we propose a lightweight framework for source-to-sink data transfer in a wireless sensor network that is geared towards achieving the above two objectives. Our integrated framework consists of three elements: 1) simple labels that eliminate complex addressing requirements, 2) implicit routing that provides an inherent robustness during sleep/wake schedules, and 3) MAC layer anycast to support routing. Our framework, in addition, facilitates the self-organization of sensor nodes into a network that efficiently relays information from the sources to the sink.**

The key idea of our framework is to associate each sensor node with a *hierarchical level* with respect to a sink and using MAC layer anycast to simply further packets to higher levels towards the sink. There are no explicit route tables created or maintained; this eliminates the overhead due to route queries or updates, the need for complex processing and the memory requirements for caching routing information. Furthermore, with our framework, the energy costs of data transmission are evenly distributed across the nodes, thereby improving the longevity of the network. Our MAC layer anycast mechanism not only facilitates routing, but also reduces the number of MAC layer back-offs incurred and, consequently, the waiting times for data transmission. This in turn, improves the timeliness of data delivery at the sink. To summarize, our framework is a) energy efficient, b) inherently robust, and c) conceptually simple. We qualitatively assess our scheme to show its efficiency in terms of power consumption, robustness to failure, ease of setup. The results from our simulations and assessments demonstrate the aforementioned benefits and the viability and potential of using our framework.

I. Introduction

Energy efficiency and timeliness of data delivery are two fundamental requirements for many wireless sensor network applications. In this paper, towards addressing the above requirements, we propose an integrated lightweight framework for transferring data from a sensing source to a data collecting sink in a wireless sensor network. Our scheme consists of three main elements: 1) classification of nodes with simple labels as opposed to providing them with unique but complex addresses (our scheme also facilitates simple self-organization), 2) provision of implicit dynamic routes that reduce the heavyweight routing overhead incurred with traditional routing schemes, and, 3) a MAC layer anycast method that facilitates routing and improves timeliness of data delivery by reducing waiting times and back-offs at the MAC layer.

Wireless sensor networks typically consist of tiny sensor nodes (referred to as *sources*) that detect an event and try to relay this information to higher-powered data aggregators, generally referred to as *sinks*. The transfer of information is to be done in a manner that takes into consideration several important factors such as: energy efficiency, robustness, flexibility, ease of implementation, etc. Traditional routing protocols from the wired network and wireless ad hoc network domains are far too complicated and expensive in terms of energy costs to be applicable directly to wireless sensor networks. This has been recognized by the research community and alternative paradigms such as data-centric mechanisms have been proposed to achieve the energy efficient transfer of information. Such schemes consider the network as being driven by data flows and the routing mechanisms are built so as to facilitate the flows. Data-driven routing architectures present a better alternative than traditional routing based ones. Nevertheless, they do not solve all the problems and are not necessarily suitable for all class of applications. The concept of being driven by data requires the network and, therefore, the nodes that form the network, to maintain state in terms of a data flow. This contributes to the incurred overhead in terms of storage and processing costs and this might in fact be costly for sensor networks.

We propose a lightweight framework that enables the sensor nodes to organize themselves into a network and relay sensed data to the sinks. The design objectives are to ensure energy efficiency while maintaining robustness and facilitating timely delivery in the presence of sleep/wake cycles of sensor nodes, typical in sensor networks. The initial setup, with our framework, consists of labels being assigned to the nodes to indicate their distance (in terms of hop count) from the sinks. We refer to these labels as *levels*. When a node has data to send to the sink, it simply directs it to a node that is at a higher level, i.e., is closer to the sink. The forwarding process continues until the information reaches the sink. All that a node needs to know is its own level. We, however, need a mechanism that facilitates the the transfer of information to the node's parent which is at a higher level. This is facilitated by our MAC layer anycast mechanism. There are no route tables maintained and there is no need for explicit addressing. Our scheme provides a multiplicity of routes for the data transfer.

Our contributions can be summarized as follows:

- We propose an integrated lightweight framework for source-to-sink data transfer in wireless sensor networks.
- Our framework eliminates routing overhead due to routing queries and/or updates, and other resource intense requirements such as caching and processing.
- The proposed scheme eliminates the need for explicit addresses in routing information and instead, uses dynamic paths that also achieve the distribution of energy costs across nodes.
- Our framework includes a MAC layer anycast scheme that facilitates routing. This has the additional advantage of reducing the number of back-offs and thus, the waiting times for data transmission.

Our simulations demonstrate the claimed benefits of using our framework.

The remainder of this paper is organized as follows: Section II provides the background to support the rest of the paper. In Section III, we provide a description of our framework. This is followed by an qualitative assessment of the performance of our framework in Section IV. We describe our experimental results in Section V and related work in Section VI. Finally, we conclude in SectionVIII.

II. Background

With sensor networks, numerous potential and existing applications abound in a broad variety of fields such as the environmental monitoring, national security, habitat monitoring, agriculture, atmospheric condition studies, health care, etc. Sensor networks differ from traditional networks in a variety of ways. Briefly, the size of the nodes that make up a network is of the order of a few square centimeters (e.g., the size of a credit card), the processor on the nodes operates at a few hundred MHz, and the available storage is a few megabytes. A primary difference is power - sensor nodes are severely constrained in terms of available power[1] and therefore all schemes and protocols employed must be energy-aware and energy-efficient. An extensive survey of sensor networks is provided by Akyildiz `et.al.`[2].

Two important areas of sensor network research are self-organization and routing. With self-organization, the nodes in the network would organize themselves (with little or no external help) into a fully functional network which can then relay the sensed information back to the data collecting sink nodes. This relaying of information is, in turn, is to be facilitated by routing schemes; these routing schemes have to take into account the energy constraints of the sensor nodes. We discuss several of these schemes, later, in Section VI. Our framework not only facilitates the aforementioned objectives in a lightweight manner, but also provides timeliness of data delivery, a key requirement of many sensor network applicatons via a tightly intercoupled MAC layer mechanism. Before we describe our framework, we discuss the type of applications that are relevant.

[1]The battery lives of unattended sensors need to be conserved.

A. Application Scenario

Since sensor networks are heavily driven by the application, the protocols developed are to be tailored, to a large extent, for the application that a particular network is built to address. Thus, a single solution does not work for all networks and it is important to delineate the specific application or class of applications that a network is being designed for.

We now describe the kind of application that our framework is meant to support by means of an example. Consider an airplane that flies over a certain geographic region and scatters hundreds of tiny sensor nodes. The geographic region is such that it prevents humans from going there physically and therefore, all information has to be collected by the sensor nodes. This kind of a scenario could easily be imagined in a battlefield that is covered with toxic elements, or inhospitable terrain such as marshlands or swamps. The information that needs to be collected could be the toxicity of the sand or air, temperature, humidity, presence and concentration of certain elements in the sand or air, etc. Moreover, we would also like the information to be conveyed as quickly as possible. So, ideally, if a node has information to send, it should send it to the nearest sink. We assume that all the sinks are connected to each other or to some central node where all the collected information can be collated and analyzed.

B. Motivation

The motivation for this work comes from the complexity in the currently existing solutions to information routing. Most routing protocols incur overhead in several areas: setup of a routing infrastructure, establishing route, maintaining route, route repair, state maintenance, etc. We will now discuss how these costs could potentially be reduced.

Among the tenets of the guiding philosophy behind our framework are:

- simple, one-time establishment of routing infrastructure
- elimination of explicit routing tables and state
- reduction of overhead due to routing queries and updates
- availability of multiple dynamic routes with little or no state information
- inherent robustness to route failure
- tighter coupling with and therefore improved efficiency at the MAC layer.

Consider Figure 1 which contrasts a single route with multiple dynamic routes. By a dynamic route, we mean that the route is not pre-specified. The circles represent nodes and the lines represent the links between the nodes. In the multiple route scenario, the ellipses represent a choice of two nodes for each link.

With sensor nodes either sleeping to conserve battery or ultimately dying out due to the depletion of battery resources links are bound to fail. Let the probability of a link failure in a considered time interval be x, where $x \leq 1$. If the single route on the left in Figure 1 is considered, the probability of successful delivery of a message that is sent in the considered time interval is $(1-x)^k$, where k is the number of links. In

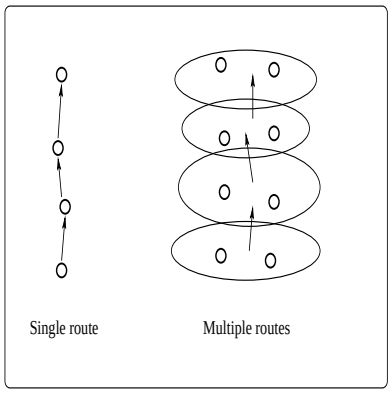

Fig. 1. Benefits of multiple dynamic routes

the multiple route scenario, assume that we have a choice of two nodes at each link. In this case, the link will fail only if both the nodes fail and therefore, the link failure probability is x^2, and consequently, the probability of successful transmission on the link is $1 - x^2$. Therefore, the probability of success for a message sent over k links is $(1 - x^2)^k$. The ratio $\frac{(1-x^2)^k}{(1-x)^k}$ represents the overall improvement factor by having multiple routes instead of a single route. Even a cursory assignment of values will show that this improvement is significant. For a generic case, where there are j nodes available at each level (instead of two), the improvement factor can be easily computed to be $\frac{(1-x^j)^k}{(1-x)^k}$. Thus, having a multiplicity of routes improves the chances of successfully transferring a sensor report to a sink by a significant factor.

Now, examine the same scenario in terms of energy costs. Recall that the nodes in a sensor network are typically energy-constrained. With a single route, each time a message is sent, a node is used once. However, with our multiple route scenario where there are j nodes at each level, each node is picked with a probability $1/j$, during the process of delivery of a message. Therefore, the energy costs are distributed across a larger number of nodes (by a factor of j) which in turn reduces the possibility of some of the nodes failing due to energy depletion caused by overuse.

III. A Framework for Source-to-Sink Data Transfer

As mentioned previously, our framework consists of three elements: a) a self-organization mechanism that assigns labels to nodes, b) an implicit dynamic routing policy and c) a tightly coupled MAC layer anycast mechanism. We now describe each of these elements in detail.

In order to organize the network in such a way that the nodes have a notion of their distances from the sinks, we propose a scheme that we call *simple hierarchical level based scheme.*

A. Self-organization: hierarchical level scheme

In the hierarchical level scheme, each node in the network belongs to a certain *level* with respect to a sink. The level represents the distance (in number of hops) from a sink. This concept is best explained by the use of a diagram.

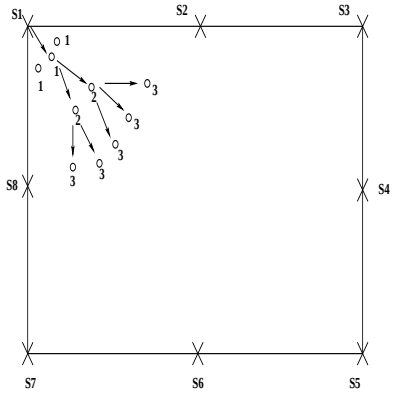

Fig. 2. The simple hierarchical level scheme.

Consider Figure 2 which depicts the layout of a sensor network. The sinks are shown by the X marks at the edges of the network and are labeled from $S1$ through $S9$ in a clockwise direction. The circles inside the box represent the sensor nodes.

The addressing process starts from each sink which broadcasts a message informing the sensor nodes in its vicinity indicating its identity; say, sink S1 would announce that it is S1. The nodes that hear this message assign themselves to level 1 (with respect to S1) and then each of them broadcasts a message indicating that they belong to level 1. All nodes that hear this message and do not have a level yet, increment the value of the received level by 1, assign themselves to this level, and then broadcast this new level. This process continues until all nodes belong to some level. Once a node has assigned itself to a level, it ignores any future broadcasts with level information[2].

B. Data transfer scheme

Once the above initial set up using levels is completed, the framework for data transfer is in place. The sensor nodes will use the levels to relay data to the sinks. When a node has data to be sent to a sink, it simply sends the data to a parent. Note that a parent of a node at level n is any node belongs to level $n - 1$ and is within the direct transmission range of the node. The parent, when it gets the message, sends it to its parent, and so on until the message reaches the sink which is at level 0. Note that no explicit route tables are created or maintained. There are no route queries or updates. For transferring data, the nodes do not have to specify explicit addresses; instead, they just hand the packets off to a parent node.

The question arises: How does a node choose a parent given that it can potentially have more than one? Broadcasting a message to all parents at each level could potentially lead to a broadcast storm and is inefficient in terms of energy usage. To address this issue, we now describe the third element of our framework - a MAC layer anycast that will result in the sender sending the data to only one parent at each level.

[2]This can be modified easily to accommodate cases wherein a node would choose to consider a broadcast that would potentially reduce its level.

C. MAC layer anycast

The data at each node has to be sent to a node that is at a higher level, i.e., one of the possibly many parent nodes. However, in order to prevent each message from being processed and forwarded by all of the node's parent nodes, we need to be able to ensure that *exactly* one of the parents performs the processing and forwarding.

A simple solution to this problem is that a node could maintain a list of its parent nodes and pick one a random when it has a message to be sent to a sink. However, this is not an efficient method. If the selected parent happens to be unavailable, the node will back-off and retry a few times before it decides to try another parent. Thus, for each successful transmission, a node could potentially have to try a few times and thus the overall waiting time of the data transfer could be adversely affected. Furthermore, the retransmissions result in a wastage of both bandwidth and energy. Our framework, in addition to facilitating routing, reduces the waiting time and the number of back-offs by incorporating the functionality of *anycasting* at the MAC layer.

When a node has a message that it wants to send to the parent nodes, it first broadcasts an RTS message. One or more of these parents will respond with a CTS. The sender node will pick one of these parents as the destination and then send the message directly to this parent node. Some of the parent nodes will send the CTS before the others. In this case, the other parent nodes might overhear the CTS messages and therefore, will desist from sending their own CTS messages. Even if there are one or more CTS messages received at the sender, the sender will ignore all but the first received CTS message. Thus, the node does not explicitly choose a parent node as a receiver; instead, the process would inherently cause one of the parents to be chosen to be the forwarding node for the sensed data. This parent node would typically be the node that is awake and is not interfered with (i.e., is not within the interference region of any other transmission). The probability that at least one of the neighbors is free from interference is larger than that of a particular chosen neighbor being free from interference. Thus, the scheme inherently provides robustness and reduces the possibility of back-offs at each link. Note here that this process does not cause the chosen routes to be longer. Since the chosen parent has to belong to the node's preceding level, at each step, the data moves one hop closer to the sink i.e., one of multiple *shortest* paths (if more than one are available) is chosen.

After a successful handshake, the node would need to identify the chosen parent uniquely in order to send the packet to the specific parent. We use the MAC address of the chossen parent as its identifier. At each instance when a parent node's CTS response is received, the MAC layer address is recorded in order to facilitate this. Note that for MAC layer packets, the Address Resolution Protocol (ARP) protocol typically provides a translation of the IP address to the MAC address. In our case, the ARP mechanism is modified appropriately to facilitate the above functionality.

D. Implementation details

Each node maintains a simple counter that indicates the level it belongs to. Note that this level is with respect to a specific sink and therefore, this information will also be encoded. The reason for maintaining sink information is that there could be multiple sinks. When the self-organization process starts, the nodes have no knowledge of how far they are from any given sink. So, it is possible that after a node assigns itself to a particular level with respect to a particular sink, it might then learn that it is in fact closer to another sink and may wish to forward information to this closer sink.

The counter is simply a bit register (0 and 1) and thus is easy to include on any sensor node. For instance, with a 5 bit representation of a level, one could have up to 32 levels from any given sink. The first part of the counter is used to represent the sink identifier and the second part to represent the level number with respect to that particular sink. Figure 3 shows a node with level number of 5 and associated with sink 3.

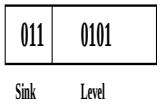

Fig. 3. Addressing for the simple hierarchical level based scheme.

Each node could potentially maintain more than one such counter. Each counter only requires a few bits of storage and so this does not cause a large storage overhead. Figure 4 shows the case where the address propagation begins from two sinks and reaches node B. B could associate itself with sink S1 or with sink S3 or both. If the node is allowed only one counter, it will always try to choose to associate itself with the sink with respect to which it is at the lower level since, sending to the nearest sink is preferrable in terms of resource utilization. So, in this case, node B will choose sink S1 with regard to which its level number is 4 instead of sink S3 with regard to which, its level number is 5.

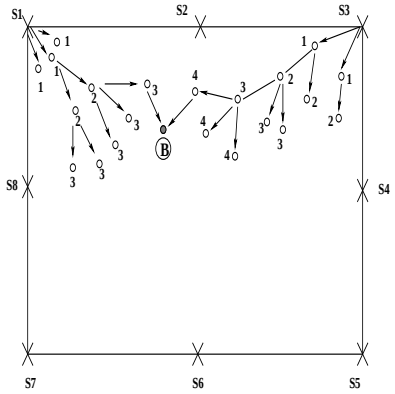

Fig. 4. Sensor nodes with multiple sink associations.

By maintaining multiple counters, the node could improve

its efficiency as well as that of the data transfer process. For example, in the above case, if node B decides to maintain associations with both sinks S1 and S3, it will maintain two counters with the corresponding sink and level numbers. Assume that at some point in time, it has a message to be sent. Normally, it would choose to send to S1 since it is closer. However, if the node knows that the congestion is higher in the direction of S1, or it is unable to find an awake parent that can forward the packet towards S1, it could make a decision to send it to S3.

Notice that we do not maintain any routing tables or route caches. Using a simple counter and minimal information with regard to the levels, the nodes can transfer data easily. We provide a brief description of our scheme in the form of pseudo-code. Recall that our scheme has two main parts - 1) the initial labeling phase wherein nodes obtian associated sink and level numbers, and 2) the data transfer phase when the nodes try to send data to the sinks.

Initial labeling:

variables:
sink-num: // *to store sink number*
level-num: // *to store level number*
self: // *node's pointer to itself*
bcast-recvd: // *broadcast packet received*
data structure:
level-info = (sink-num,level-num)

Sink node:
Broadcast *level-info* to nearby nodes.

Sensor node:
1. Listen to broadcast from other nodes
2. If (*self.level-info* is NULL)
{
Extract *bcast-recvd.level-info*
self.level-num = (bcast-recvd.level-num + 1)
self.sink-num = (bcast-recvd.sink-num)
}
else
Discard broadcast packet received
3. Create new data structure *packet-to-send*
packet-to-send = (*self.sink-num,self.level-num*)
4. Broadcast *packet-to-send*

Data transfer:

Send:
If node has data to send
{
Assign *parent.level-num* = (*self.level-num* - 1)
Assign *parent.sink-num* = (*self.sink-num*)
Let *parent.level-info* =
(*parent.sink-num,parent.level-num*)
Broadcast MAC layer RTS with *parent.level-info*
Wait for CTS from a parent node
Send DATA to the parent that responds
}

In order to keep the exposition of the algorithm simple, we have not included some details such as that a node could have multiple labels corresponding to different sink associations. However, these issues were described in the previous section and it is fairly straightforward to include them in the algorithmic representation.

IV. Qualitative Assessments

In this section, we conduct a detailed qualitative assessment of our schemes. We consider issues such as setup cost, latency, reliability, impact upon failure, and throughput.

A. Setup cost: number of messages

In our self-organization scheme, every node broadcasts its own level number once. The children nodes pick up this notification, determine their own levels accordingly, and broadcast their levels to their children. Thus, each node in the network broadcasts one message. Therefore, the total number of messages, M, to set up a hierarchy in a network of N nodes, is: $M = N$. Note that this is a one-time cost.

B. Storage costs

Each node stores one or more level counters that is associated with a sink. Depending on the size of the network and the density of the nodes, the number of bits needed for storing this level information will change. However, even with as little as 1 byte (8 bits), our framework can support up to $2^8 = 256$ levels, if there is a single sink. If each level has, on an average, four nodes, the network can have up to 1000 nodes. With 2 bytes (16 bits) and four nodes at each level, the network can have more than a quarter of a million nodes. Thus, the scheme is significantly economical in storage costs and therefore, scalable. Furthermore, we point out that routing is fairly simple in spite of such large network sizes.

C. Data transmission cost: number of messages

In order to send a data message to its parent node, a node uses the MAC layer anycast mechanism which involves a series of RTS/CTS exchanges. As described earlier this is to ensure that only one parent is chosen to deliver the message instead of all parents.

Thus each node has to broadcast an RTS message when it wants to send data. The parent nodes, when they hear the RTS, will try to respond with a CTS. If a parent overhears the CTS sent by another parent, it will desist from sending its own CTS. However, sometimes two parents can be *hidden* from each other, in which case, they both will send a CTS to the sender node. The sender node upon receiving the first CTS will send the data to the node that sent the CTS. It will then ignore all subsequent CTS messages from other parents by discarding them.

Once the RTS/CTS exchange is successful, the sender will send the data message and the receiver will reply with an ACK when it gets the data successfully. Thus, we have 3 control messages at each level for a data message to be sent. As mentioned earlier, due to topological artifacts additional CTS messages may be generated. However, the number of such CTS messages is at most 4 (easily computable by geometry); typically due to the random distribution of nodes, the additional CTS transmissions seldom occur and hence, one may consider such additional overhad to be negligible.

The methods provide significant savings in terms of transmission costs as compared with the typical routing schemes which construct and use explicit routes. With an explicitly defined route, a node will try to contact a specific next-hop destination. If that fails, it will retry a few times as dictated by the MAC protocol. If this fails, it will decide that the link has failed and initiate a route discovery or try to use another route that is in its cache. The retransmissions and repeated route query attempts may lead to a wastage of both power and bandwidth resources.

D. Latency

Once the routing infrastructure is in place, the nodes will start relaying information to the sink. The time taken to transfer the information, i.e., the latency incurred is important for many sensor network applications.

In the simple hierarchical level scheme, a sender has to negotiate with its parents using the RTS/CTS mechanism and pick one parent to send the data to. The time taken for the RTS/CTS would be typically equivalent to the duration of a single RTS/CTS exchange since the other nodes that send CTS messages after the first CTS has been received by the sender would be ignored. However, in case there is a CTS collision, this time could be higher.

Note that with an explicit routing scheme, retransmissions may be more common and this in turn would lead to back-offs. This may be due to the fact that the explicit parent being sought is either asleep or is within the interference range of an other transmission. Thus, typically one would expect the latency incurred to be lower with our framework.

E. Reliability

Our scheme offers an inherent reliability in two different aspects. The first aspect is the fault tolerance afforded by having multiple parents. Because each node has multiple parents, a node can be assured that even if one parent fails, there might be another parent that can relay the message. This happens transparently to the sender node, i.e., it does not need to know which of the parents have failed nor will have to take additional steps to recover from the failure.

The second aspect follows from the multiplicity of parent nodes. Note that each time a sender has to send data, it goes through an RTS/CTS exchange. Assuming a scenario where all nodes are homogeneous, it is likely that a different parent node will be selected each time. This implies that the energy costs of data transmission are uniformly distributed across the nodes. This prevents scenarios wherein some single parent node gets chosen more often than others, thereby draining its energy sooner than the other nodes and consequently increasing the possibility of its failure. Therefore, the overall reliability of the network is increased[3].

F. Savings as compared with traditional routing policies

To analyze the savings achieved by using our framework, we compare it with two generic routing schemes - a proactive scheme and a reactive scheme. We now briefly describe the various operational components of the schemes in order to facilitate a comparison in terms of the incurred costs.

A proactive routing scheme is typically based on maintaining routing information in a table and updating it on a regular basis. It usually provides only one path to a destination based on some variation of the shortest-path algorithm. However, the scheme requires route update messages to be sent periodically regardless of whether the network topology has changed.

A reactive scheme, also known as an on-demand scheme, does not maintain route tables. Instead, it tries to discover routes when they are needed. Typically the whole process consists of the following phases: route discovery, route maintenance, and route rediscovery (in case of failure).

Now we analyze the three schemes in terms of their costs for various operations - setup, communication, failure, etc. We will refer to the survey paper by Royer and Toh [8] for assessing some of these costs. Note that the paper considers routing protocols for ad hoc networks. These would need to be modified to be applicable to a sensor network where the emphasis is on power savings. Nevertheless, for comparison purposes, a generic protocol from each of the two types is sufficient to understand the rough quantum of savings that our scheme can provide. We represent the number of nodes in the network by N and the diameter of the network under consideration by d in the following discussion.

1) Cost of setup: The cost of setup involves setting up a route. [8] categorizes the cost into time complexity and communication complexity. Time complexity is the number of steps required to perform an operation and communication complexity is the number of messages required to complete an operation.

For a proactive protocol, the time complexity is $O(d)$ and the communication complexity is $O(x = N)$ where x is the number

[3]Note here that we can further improve this scheme by choosing back-off times prior to transmitting the CTS message based on a parent's residual energy level. Thus, nodes with higher residual energies would be more likely to be chosen for packet forwarding.

of nodes affected by a topological change. For a reactive protocol, the time complexity is $O(2d)$ and the communication complexity is $O(2N)$. In our scheme, each node broadcasts one message during the setup phase. Let there be j nodes at each level. In this case, at each step, j nodes could be transmitting simultaneously. Therefore, the time complexity is $O(N/j)$. The communication complexity is $O(N)$ since each node broadcasts one message.

2) Route maintenance: In a proactive protocol, route maintenance is done by sending update messages on a periodic basis. This happens even if there are no changes in the network topology and is therefore inefficient in such scenarios. In a reactive protocol, there are no such periodic updates, but a form of route maintenance occurs by letting the nodes cache multiple routes to the same destination. This results in storage overhead. In our scheme, no routes are explicitly maintained and there are no periodic updates. The only piece of information maintained is the level number associated with a sink. As described earlier, this information can be maintained in a few bits of data. Thus, our scheme is significantly better than a reactive or proactive scheme in terms of consuming lower overhead.

3) Route failure: In a proactive protocol, in case of a link failure, the time complexity for recovery is $O(d)$ and the communication complexity is $O(x = N)$. This is because each node maintains routing tables that get updated on a periodic basis. In a reactive protocol, there are no such tables and hence the cost increases. The time complexity associated with recovery from a link failure is $O(2d)$ and the communication complexity is $O(2N)$. In our scheme, a link failure does not lead to any additional messages because of the MAC layer anycast. By using this technique, instead of a sender choosing a destination, the destination chooses itself and notifies the sender that it is ready to receive data. Thus, a node always chooses a link that is available and link failures are transparent in most cases. One problem that could arise is if all the parent nodes of a node fail. In this case, the node can resort to using its siblings for message delivery. This issue is explored in more detail later when we discuss possible refinements to our framework.

The costs incurred in recovering from route failures can be particularly severe in the case of proactive protocols that simply erase the entire route when the particular route fails. This scenario will not occur in our scheme since no route is maintained explicitly. Even a protocol that uses local route repair techniques (instead of complete route erasure) has the cost of communicating with neighbors (time taken for recovery can be fairly large) in order to find a new link.

An especially detrimental scenario can arise when a link is deemed unavailable and the routing protocol either switches to a new route, or initiates route discovery. Often, it might happen that the node itself is not dead but the link fails due to external factors such as interference. Thus, even though the node is alive and might become available sometime in the future, cost intensive recovery methods are invoked. In our scheme, we do not erase or switch routes or initiate route discovery when a link fails; therefore the costs are are much lower.

4) Energy efficiency: The proactive and reactive protocols do not have any inherent notion of energy efficiency. For instance, if a route is established, it will continue being used. This could lead to energy depletion of the nodes on that route but will not be resolved unless there are specific mechanisms built to address this. Compare this to our scheme, where energy efficiency is built in. At each level, considering there are j nodes, each node has a $1/j$ chance of being chosen for delivering a message. Thus, the energy costs are now more evenly distributed across all the nodes instead of being restricted to nodes on a single route as in the case of a traditional routing scheme. This is because in our scheme, a route is not pre-defined and there are multiple routes available to the sink. A route is created dynamically and thus it is more energy efficient.

Note that nodes nearer to the sink are more likely to get used since they act as parents to a larger number of nodes. One could think of these nodes as bottleneck nodes. However this remains true even in a traditional scheme. Our scheme improves the energy distribution factor in the regions further away from the sink.

5) Reliability: In a proactive protocol, reliability is provided by the existence of multiple routes in the routing table. However, maintaining the routing tables incurs overhead in terms of periodic messages. In a reactive protocol, some reliability may be provided by letting the nodes maintain multiple routes. This, however, entails the overhead of storing these routes at each node. In our scheme, reliability is provided in the form of multiple dynamic routes. No route tables or routes are maintained or stored. Thus, there is no extra overhead other than the level information maintained by the nodes.

G. Assumptions

We assume that the nodes are uniformly scattered in the geographic region. We also assume that the sinks can be positioned at the edges of this region in such a way that they can establish communication with the sensor nodes. Naturally, this entails that the process cannot start until the sinks are in place. Once the nodes are scattered, they do not move. Thus, mobility is not considered. We would like to emphasize that even though we have described our framework in great detail and provided analysis for a variety of issues, extensive experimentation evaluation is required in order to fully understand the benefits of the framework. We now describe some experiments which is an initial step in the direction.

V. Simulations and Results

In order to further understand and analyze our schemes, we conducted simulations using the ns-2 simulator [1]. The ns-2 version was 2.27 and the simulations were conducted on a set of Linux machines. Table I shows the values of the simulation parameters used in the experiments. The values were averaged over ten runs.

In the experiments that follow, we refer to our scheme as the anycast scheme since it involves the anycast at the MAC layer. The generic routing scheme which uses a predefined route is referred to as a unicast scheme.

Parameter	Value
Number of nodes	100 to 400
Number of sinks	8
# of traffic generators	20
Simulation area	1250 X 1250 m^2 to 1600 X 1600 m^2
Simulation time	400 s

TABLE I
SIMULATION PARAMETERS

A. Robustness to failure

Earlier we described how our scheme distributes the task of data transfer among the nodes more evenly than a generic routing protocol. This leads to a decrease in the probability of some nodes failing before the others. Figure 5 shows how the nodes fail, i.e., die due to energy depletion.

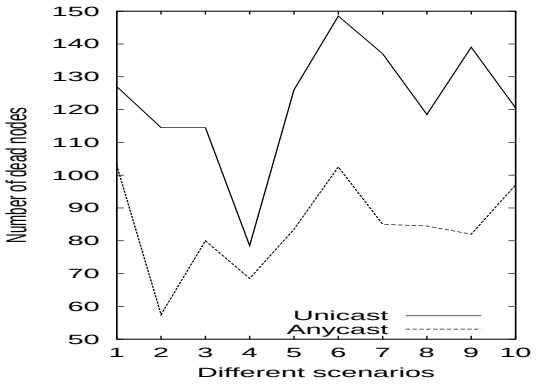

Fig. 5. Node failure in different scenarios for anycast and unicast

The x-axis shows different scenarios generated by the setdest utility in $ns-2$. The y-axis is the number of nodes that die. The simulation area is $1250X1250\,m^2$ and the number of nodes is 250. The graph shows the anycast always outperforms the unicast scheme in all scenarios. Fewer number of nodes die during the course of a simulation in the anycast scheme because of the even task distribution.

B. Failure over duration of time

Now we would like to see the dynamic nature of the node failure using the two schemes. Figure 6 shows the cumulative failure of the nodes in time. The x-axis shows time in simulation seconds. The y-axis shows the number of nodes that die due to overuse. In this scenario, we specify the life of a node as 1000 packets, i.e., a node can transmit 1000 packets before it dies. For the traffic generator nodes, we fix a higher lifetime than the other nodes. For this specific set of experiments, this lifetime was 5000 packets. The sink nodes are assumed to never die and their lifetimes are set as such.

Figure 6 shows the case for a $1250X1250\,m^2$ network with 100 nodes and Figure 7 is the same network with 400 nodes. Thus the density is four times more in the second case.

The graph shows that as time goes by a larger number of nodes die in the unicast scheme as compared to the anycast

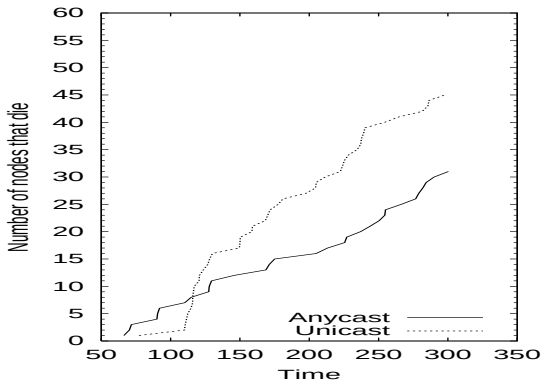

Fig. 6. Node failure with time for 100 nodes

scheme. This behavior is explained by the fact that the anycast scheme uses the nodes more evenly than the unicast scheme. Thus, the energy depletion over time, which ultimately leads to failure, is more uniformly distributed among the nodes in the anycast scheme.

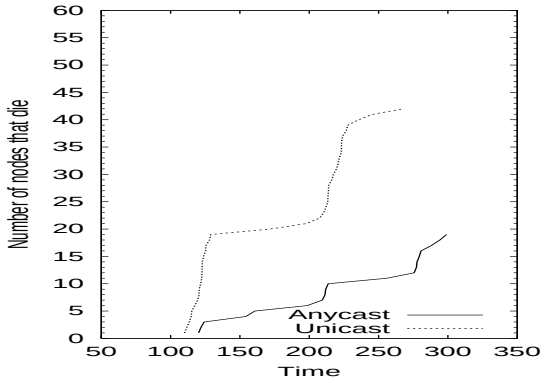

Fig. 7. Node failure with time for 400 nodes

Figure 7 shows the case where the density is more. The plot shows that anycast is still better than unicast in terms of a lower number of nodes that die. Moreover, it is also seen that the overall improvement in anycast is drastically better than unicast. The number of dead nodes in the unicast scheme decreases from 45 to 41 as the density increases; in the anycast scheme this number decreases from 31 to 19 which is a 38% drop. The reason is that as the network becomes denser, there are more dynamic paths available and hence the energy costs will be even more evenly distributed among the nodes thereby reducing the overall probability of node failure.

C. Energy distribution

We would now like to see how the energy costs are distributed. In our model, we assume that the most energy is consumed in transmitting and receiving data packets; control packets such as RTS and CTS are assumed to consume significantly less energy. As an indicator of the energy cost, we decided to track the number of packets sent and received.

Our scheme utilizes dynamic paths to transfer data. Thus, nodes are utilized randomly in a uniform manner. Figure 8 shows the usage of the nodes. The x-axis shows the total number of packets processed per node; this includes both transmitted and received packets. The y-axis is a cumulative fraction of the nodes up to 1, i.e., all nodes.

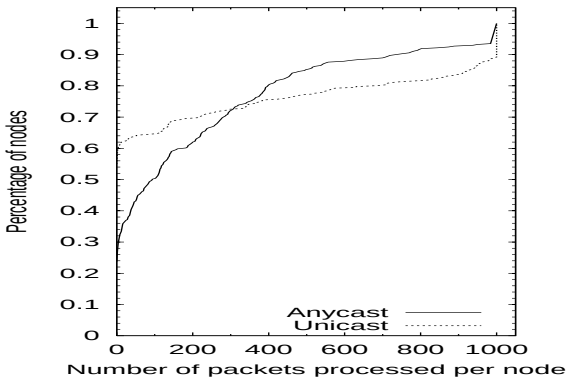

Fig. 8. Cumulative distribution of packets per node

The plot shows that up to 28% of the nodes do not process any packets in the anycast scheme. This figure is 60% in the unicast scheme. This means that all the processing is done by 72% of the nodes in the anycast scheme and 40% in the unicast scheme. Thus, the usage of the nodes is significantly more distributed in the anycast scheme. This implies that the chances of some nodes failing earlier than the others is less in the anycast scheme. Moreover, the plot also shows a significant rise in the curve at the 1000 packets mark on the x-axis. This means that significantly more nodes process a larger number of packets in the unicast scheme as compared to the anycast scheme.

D. Summary

Thus, our scheme is more robust to node failure than a generic routing scheme by a significant factor. The nodes are more evenly utilized because of the multiple dynamic routes and this reduces the probability of overall node failure.

VI. Related Work

Routing in wireless sensor networks has been the focus of significant research in the last few years [9], [7], [5], [6], [4], [3]. It should be noted that owing to the inherent nature of wireless sensor networks, there is no single routing scheme that fits all scenarios. Sensor networks are heavily driven by the application and consequently, the routing scheme applied to a particular network will depend on its purpose.

Lin et.al. propose the concept of *gossip routing* which is a probabilistic form of the simple broadcast scheme. The idea is that a node will send a message to some of its neighbors instead of all. This is determined probabilistically. This is primarily intended for small networks since for larger networks, the overhead is significant. Thus, for a energy-constrained wireless sensor network, this is probably not a good option.

Barrett et.al. propose the concept of *parametric probabilistic sensor network routing* [3]. Their scheme tries to combine the best features of limited flooding and information-sensitive path-finding protocols. The probability of transmitting to the neighbors is driven by a probability function that is stored at each node. This function incorporates the distance between source and destination, and the distance between the current node and the destination. Their scheme is similar to ours in the intention that the number of broadcasts has to be reduced, except that in our case, the nodes do not broadcast data back to the sink. Their scheme calculates distances by starting with an arbitrary estimate and then using an iterative process to refine this estimate. In contrast, we do not perform any such operations. Thus, in terms of computation and storage overhead, our scheme is more lightweight.

In [5], the authors discuss the concept of *directed diffusion* as a communication paradigm. This work is partly similar to ours in the motivation aspect - to relay information from the sources to the sinks in a robust, scalable, and energy-efficient manner. However, their approach is data-centric which is based on naming data and also requires that all nodes be application-aware. The nodes disseminate *interests* and set up *gradients* in order to facilitate the flow of information. The robustness comes from setting up multiple paths by using the gradients. They also discuss the concept of reinforcing certain paths which are better than others. Correspondingly, there is also the negative reinforcement concept to deal with low quality paths. The nodes have to maintain caches of interest entries that are a few bytes in size (36 bytes in their experiments).

In our scheme, we do not require the nodes to be application-aware. All that a node needs to maintain is a *level number* associated with a sink that indicates the distance (number of hops) from the sink. No explicit paths are set up and so there is no overhead of reinforcement or negative reinforcement. Unlike directed diffusion where any node in the network can be interested in information anywhere in the network, our application domain has only the sinks interested in information; the sensor nodes only collect and relay the information to the sinks. Our scheme is not meant for the case where any sensor node wants to communicate with any other sensor node; we assume that the communication is only from source-to-sink.

The concept of *rumor routing* has been proposed by Braginsky et.al [4]. The authors note that rumor routing is a logical compromise between flooding queries and flooding event notifications. The idea is based on the creation of paths leading to each event and when a query is generated, it can be directed on a random walk until it finds the event path instead of flooding it. However, the random walk concept suffers from the possibility of taking a long time before it finds the correct path. A similar protocol called the *Wanderer* has been discussed in [3] where they use it only for the sake of comparison against their protocol.

VII. Further Possible Refinements

Our proposed framework is only a first step towards our goal of developing a lightweight mechanism for source-to-sink data

transfer in wireless sensor networks. We now describe several refinements that could potentially enhance performance. We do not go into great detail here since this is a topic for future work.

A. Using siblings to route around failure

Consider the scenario shown in Figure 9. For ease of exposition, we use a uniform layout for the network. There are two levels - 3 and 4 and 6 nodes in each level named a through f. Each node can send data to three of its parents, e.g., in the diagram, node $4b$ can send to nodes $3a$, $3b$, and $3c$ whereas node $4d$ can send to nodes $3c$, $3d$, and $3e$. This is shown by the arrows.

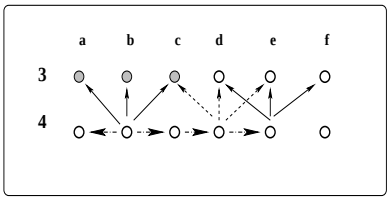

Fig. 9. Optimization of message delivery in a failure scenario.

Now consider a scenario where all the parent nodes within range of a node fail. So, for instance, node $4b$ has data to send and it finds that all three of its parents, i.e., nodes $3a$, $3b$, and $3c$ are not available. In this case, the node has the option of sending it to a sibling, i.e., a node that is in the same level. The sibling could potentially have parents that are available and thus the data could be sent to them.

Notice that in the figure, node $4b$ can send the data to either sibling $4a$ or $4c$. However, sending to $4a$ will not accomplish anything since it has the same three parents as $4b$, which are not available. So, it is preferable to send to node $4c$. An even better option is to send the message to a sibling that is two or more hops away since this would mean going away from the region close to the failed parents. For instance, sending to the sibling $4d$ would open up two extra possibilities for the parent nodes - $3d$ and $3e$.

B. Multicasting to multiple siblings

A related optimization in the case of unavailable parent nodes is to send to more than one sibling. Thus, the node will multicast the data to multiple siblings. This way, even if some of the multicast paths end in failure, there is a better probability that the data will still be able to reach the sink via some path.

For instance, consider Figure 10 which shows a node B that wants to send data. However, both its parents, P1 and P2, are unavailable due to some reason (e.g., interference, etc.) and therefore the choice is to send the data to a sibling. If B decides to multicast to two siblings C and D, the chances of the data transfer being successful increases. However, at the same time, more network energy resources will be utilized.

The challenge here is in identifying the degree of multicast. The more siblings we multicast to, the more the drain on the overall network energy resources. The tradeoff is between

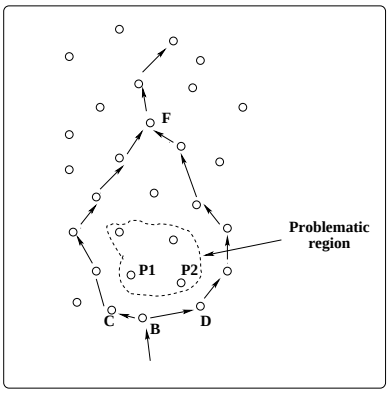

Fig. 10. Multicasting to multiple siblings

increased reliability and increased energy consumption. Additionally, we could also decide that once the data is multicast around the "problematic" region, it could then be aggregated at a higher level so as to reduce energy costs, like shown in the figure.

C. Soft state association with multiple sinks

As described earlier, each node can associate itself with one or more sinks. For each association, a node has to maintain the data structure shown in Figure 3. Depending on the storage constraints, a node might not want to maintain more than one or two such counters. In case a node finds that a new association is closer to a particular sink, it can choose to replace an older association with the new one. However, the simple act of dropping an association does not mean that it is not able to reach that particular sink. It still knows the identity of its neighbors and has the option of using a previously discarded association by merely querying its neighbors to see if they can reach the sink. Thus, these extra associations are maintained in a soft state. This is better explained by the Figure 11.

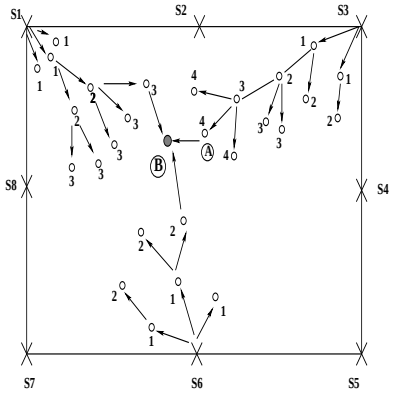

Fig. 11. Soft state association with multiple sinks

The figure shows node B in the middle of the network. Assume Node B has associations with two sinks - S1 at level 4 and S3 at level 5. After a while, it comes to know that sink S6 is only 4 levels. At this point, it decides to drop its

association with sink S3, i.e., it will discard the level counter (S3,5). However, notice that node A still is a neighbor of B and A is associated with sink S3. Therefore, if at any point in the future, if B has to contact S3, it can query its neighbors and find that it can reach S3 through node A.

D. MAC layer back-offs

In the MAC layer anycast process, it is possible that when a node sends an RTS to its parents, there might be multiple parents who decide to send a CTS simultaneously and this could result in CTS collisions. A simple alternative is to back-off for a random period and then try again. An improvement can be made by associating the back-off interval with the amount of remaining energy. So, for instance, a node that has lower energy remaining might back-off for a longer interval than a node with higher energy. Another option is that if a node is running low on energy, it will not respond will a CTS unless absolutely necessary. We intend to study these issues in greater detail as part of our future work.

E. Incorporating more state at MAC layer

Another possibility in optimizing the choice of parent is to incorporate the estimated "burden" on the parents. The parents nodes will, over a period of time, learn about the number of child nodes that they are linked to. Then, at the time of data transfer, along with the CTS replies, the parents will also include a counter that indicates the number of its children. The sender will then pick the parent which has the least estimated burden. However, this would be at the cost of incorporating more state at the MAC layer. The study of the tradeoffs involved is the subject of future work.

F. Data aggregation

If an event is detected by multiple sensors in a close vicinity, there is an opportunity to save energy by data aggregation. So, in this case, multiple detecting nodes will collaborate to select one sender and then the process of data transfer will proceed from that node. We intend to investigate this aspect in the future.

VIII. Conclusion

We presented a lightweight integrated framework that enables sensor nodes to self-organize themselves into a sensor network and transfer data from source to sink nodes. Our integrated framework combines three elements: a) labeling nodes, 2) implicit dynamic routes, and 3) MAC layer anycast.

Our contributions can be summarized as follows:

- We proposed a lightweight integrated framework to facilitate source-to-sink data transfer.
- Our scheme eliminates the overhead of routing queries and updates since it does not maintain explicit routing tables.
- Our scheme also reduces storage and processing overhead because the nodes do not need to maintain or specify explicit addresses or routes.
- We showed that the framework supports inherent reliability by making multiple dynamic paths available to the sink.
- We described a MAC layer anycast to reduce the MAC layer waiting times and the back-offs.
- We also showed that our framework is energy efficient.

As future work, we plan to investigate in greater detail the optimizations that we described earlier. We also intend to investigate the incorporation of mobility into our scheme. Mobility will make it more complicated to maintain the levels. We would like to see how to modify the scheme so as to still make it effective while retaining as much of the simplicity as possible.

References

[1] *ns-2*. http://www.isi.edu/nsnam/ns.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, volume 38, pages 393–422. Elsevier North-Holland, Inc. New York, NY, USA, 2002.

[3] Christopher L. Barrett, Stephan J. Eidenbenz, Lukas Kroc, Madhav Marathe, and James P. Smith. Parameteric probabilistic sensor network routing. *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 122–131, 2003.

[4] David Braginsky and Deborah Estrin. Rumor routing algorithm for sensor networks. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31, September 2002.

[5] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67, August 2000.

[6] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. Modelling data-centric routing in wireless sensor networks. *USC Computer Engineering Technical Report CENG*, pages 02–14, 2002.

[7] M. Lin, K. Marzullo, and S. Masini. Gossip versus deterministic fboding: Low message overhead and high reliability for broadcasting on small networks. *Technical Report: CS1999-0637*, 1999.

[8] Elizabeth M. Royer and C-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46–55, April 1999.

[9] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. *Proceedings of the first international conference on Embedded networked sensor systems*, pages 14–27, November 2003.