An Application of Path Sharing To Routing For Mobile Sinks In Wireless Sensor Networks

Okafor Friday Onyema+, Fagbohunmi Griffin Siji*

+Department of Computer Science
Michael Okpara University of Agriculture, Umudike Tel:+234-803-7172374 Email: Revmachi_4@Yahoo.Co.Uk
***Department of Computer Science, Abia State Polytechnic, Aba Tel:+234-706-4808382. email: oluwasijibomi1@hotmail.com**

Abstract

*Power Conservation is one of the most important challenges in wireless sensor networks. In this paper, we present a minimum-energy routing algorithm. Our main goal is to reduce power consumed and prolong the lifespan of the network. The protocol, named CODEXT: **Co**ordination-based **D**ata dissemination for Sensor Networks e**XT**ension, addresses the sensor networks consisting of mobile sinks. CODEXT which is an improvement over CODE protocol **Co**ordination-based **D**ata dissemination for sensor networks considers energy conservation not only in communication but also in idle-to-sleep state. Better informed routing decisions can often be made by sharing information among neighbouring nodes. To this end, we describe the CODEXT protocol, a generic outline for Wireless Sensor Network (WSN) protocols that focuses on locally sharing feedback with little or no overhead. This paper describes one instantiation of it, CODEXT protocol for optimizing routing to multiple sinks through reinforcement learning. Such a routing situation arises in WSNs with multiple, possibly mobile sinks, such as WSNs with actuators deployed in parallel to sensors. This protocol is based on GAF protocol and grid structure to reduce energy consumed. Our simulation results show that CODEXT gain energy efficiency and prolong the network lifetime.*

Keywords: Source, Sink, Coordination-based Data dissemination protocol, WSNs

---

1.0 Introduction

A wireless sensor network is randomly deployed by hundreds or thousands of unattended and constrained sensor nodes in an area of interest. These networking sensors collaborate among themselves to collect, process, analyze and disseminate data. In the sensor networks, a *data source* is defined as a sensor node that either detects the stimulus or is in charge of sensing requested information. The sources are usually located where environment activities of interest take place. A *sink* is defined as a user's equipment such as PDA, laptop, etc. which gathers data from the sensor network.

Limitations of sensors in terms of memory, energy, and computation capacities give rise to many research issues in the wireless sensor networks. In recent years, a bundle of data dissemination protocols have been proposed [3]. Most of these efforts focus on energy conservation due to the energy limitation and the difficulty of recharging batteries of thousands of sensors in hostile or remote environment. Generally, the power consumption of sensors can be used for three functionalities - the power consumed for the: (a) transmission of packets (b) reception of packets and (c) the power consumed when the network is idle. Besides, recent studies have shown that radio communication dominates energy consumption in the sensor networks, rather than computation [7]; therefore, power conservation is an especially

important challenge at the communication layers.

Each sensor network possesses its own characteristics to cater for different applications. An example of such applications is the monitoring and control of safety-critical military, environmental, or domestic infrastructure systems. Depending on each application, the sinks may be mobile while the sensors are stationary. On the other hand, the number of sinks may be large since many users may simultaneously access the sensor networks. In this paper, we propose an energy-efficient data dissemination approaches which have been built as an improvement over the CODE protocol. These protocols individually address the sensor networks consisting of mobile sinks and the sensor networks consisting of a large number of sinks.

The algorithm, Coordination-based Data Dissemination Protocol Extension (or CODEXT for short), addresses mobile sinks. The authors are motivated by the fact that handling mobile sinks is a challenge of large-scale sensor network research. Though many researches have been published to provide efficient data dissemination protocols to mobile sinks [9]; they have proposed how to minimize energy consumed for network communication, regardless of idling energy consumption. In fact, energy consumed for nodes while idling cannot be ignored [10], show that energy consumption for *idle:receive:send* ratios are *1:1.05:1.4*, respectively. Consequently, they suggest that energy optimizations must turn off the radio. Doing this not only simply reduces number of packets transmitted but also conserves energy both in overhead due to data transfer, and in idle state energy dissipation when no traffic exists, especially in sensor networks with high node density. In CODEXT, we take into account the energy for both communication and idle states. CODEXT provides an energy efficient data dissemination path to mobile sinks for coordination sensor networks. CODEXT is based on grid structure and coordination protocol GAF [13]. The key observation driving the CODEXT notion is that wireless communication between neighbouring nodes is not a private, point-to-point exchange between them, but rather it is broadcast, implying that it can be received by all nodes within range. Extensive amounts of local data exist on the single nodes in a wireless network, which, if shared, could improve the performance of the routing and or application levels. This data is usually small, such as residual energy, available routes to sinks, route costs to specific sinks, application role assigned to the node, link quality information, etc. When shared with neighbours, this information could be used for adjusting existing routes and taking routing decisions to minimize costs.

To better understand the rest of the paper, the authors first describe the general protocol design goals of sensor networks in Section 2. Then in section 3 and 4, we present the protocol and its performance evaluation.. The discussion about benefit of the proposed approach is given right after its evaluation. Section 5 concludes the paper.

1.1 Protocol Design Goals

The wireless sensor network has its own constraint that differs from adhoc networks. Such constraints make designing routing protocol for sensor networks very challenging [1]. Firstly, sensor nodes are limited in power, processing capacities and memory. These require careful resource management. Secondly, sensor nodes may not have global identifications (IDs). Therefore, classical IP-based protocol can not be applied to the sensor networks. Thirdly, sensor nodes might be deployed densely in the sensor networks. Unnecessary nodes should turn off its radio while guaranteeing connectivity of the entire sensor field. Fourthly, generated data traffic has significant redundancy in it since multiple sensors may generate same data within the vicinity of a phenomenon. Such redundancy needs to be exploited (through compression techniques) by the routing protocols to improve energy and bandwidth

utilization. This will be addressed in the clustering algorithm to be proposed later.

In order to design a good protocol for the sensor networks, such constraints should be managed in an efficient manner. In this paper, emphases was placed on three major design goals in data dissemination protocol for wireless sensor networks.

1.1.1 Energy Efficiency/Network Lifetime

Energy efficiency is the most important consideration due to the power constraint of sensor nodes. Recent studies have shown that radio communication is the dominant consumer of energy in the sensor networks. Most of recent publications mainly focus on how to minimize energy consumption for sensor networks. Besides, multi-hop routing will consume less energy than direct communication, since the transmission power of a wireless radio is proportional to the distance squared or even higher order in the presence of obstacle. However, multi-hop routing introduces significant overhead for topology management and medium access control [1]. Another characteristic of the common sensor networks is that sensor nodes usually generate significant redundant data. Therefore similar packets from multiple nodes should be aggregated so that the number of packets transmitted would be reduced [8]. Several work, [7], [11], suggest that unnecessary nodes should be turned off to conserve energy and reduce collision.

1.1.2 Latency

The user is interested in knowing about the phenomena within a given delay. Therefore, it is important to receive the data in a timely manner [5], [7].

1.1.3 Scalability

Scalability is also critical factor. For a large scale sensor network, it is likely that localizing interactions through hierarchical and aggregation will be critical to ensure scalability [5]. Keeping these design goals in mind, in this paper we propose a data dissemination protocols for large-scale sensor networks to achieve energy efficiency while guaranteeing a comparable latency with existing approaches.

1.2 CODEXT: A Coordination-Based Data Dissemination Protocols To Mobile Sink

CODEXT addresses the sensor networks consisting of mobile sinks. In CODEXT, we rely on the assumptions that all sensor nodes are stationary. Each sensor is aware of its residual energy and geographical location. Once a stimulus appears, the sensors surrounding it collectively process the signal and one of them becomes the source to generate data report. The sink and the source are not supposed to know any *a-priori* knowledge of potential position of each other. To make unnecessary nodes stay in the sleeping mode, CODEXT is deployed above *GAF-basic* protocol [10]. Fig.1 depicts CODE general model where the routing algorithm is implemented above the GAF protocol. In this paper, we only focus on CODEXT routing algorithm. Details of GAF algorithm can be referred in [13].

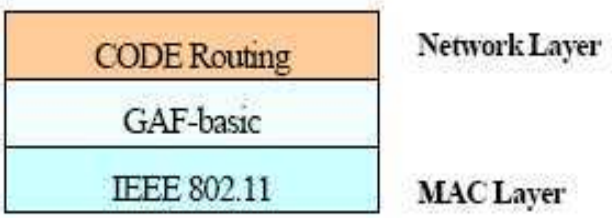


Fig.1.CODEXT System Model

The basic idea of CODEXT is to divide sensor field into grids. Grids are indexed based on its geographical location. According to GAF, each grid contains one coordinator which acts as an intermediate node to cache and relay data. CODEXT consists of three phases: *data announcement*, *query transfer* and *data dissemination*. As a stimulus is detected, a source generates a *data-announcement* message and sends to all coordinators using simply flooding mechanism. Each coordinator is supposed to maintain a piece of information of the source including the stimulus and the source's location. As a mobile sink joins in the network, it selects a coordinator in the same grid to act as its *Agent*. When it needs data, it sends a query to this *Agent*. The *Agent* is in charge of forwarding the query to the source based on the target's location and grid IDs. An efficient data dissemination path is established while the query traverses to the source. Receiving a query, the source sends the data to the sink along the data dissemination path. The Agent helps the sink to continuously keep receiving data from the source when the sink moves around. Periodically, the sink checks its location. If the sink moves to another grid, it first sends *cache-removal* message to clear out the previous data dissemination path and then re-sends a query to establish a new route.

1.3 CODEXT Theory

A. Grid Indexing

We assume that we have partitioned the network plane in virtual *MxN* grids (for example in Fig.2 that is 3x2 grids). Each grid ID which has a typed [CX.CY] is assigned as follows: at the first row, from left to right, the grid IDs are [0.0], [1.0], and [2.0]. Likewise, at the second row, grid IDs are [0.1], [1.1], and [2.1] and so forth. To do this, based on the coordinate (x, y), each node computed itself CX and CY:

$$CX = \left\lfloor \frac{x}{r} \right\rfloor, \quad CY = \left\lfloor \frac{y}{r} \right\rfloor \qquad (1)$$

where r is the grid size and [x] is largest integer less than x.

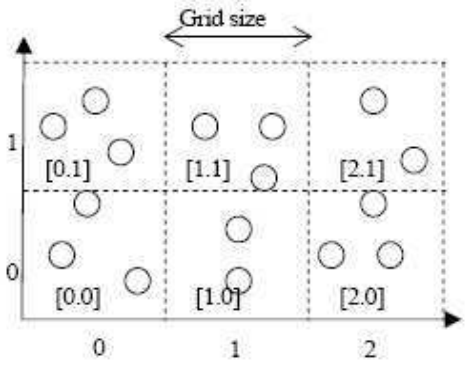


Fig.2.Grid Indexing

B. CODEXT Algorithms

a*) **Data Announcement***

When a stimulus is detected, the source propagates a *data-announcement* message to all coordinators using simply flooding mechanism. Every coordinator stores a few piece of information for the data dissemination path discovery, including the information of the stimulus and the source location. In this approach, the source location does not mean the precise location of the source, but its grid ID. Since the coordinator role might be changed every time, the grid ID is the best solution for nodes to know the target it should relay the query to. To avoid keeping data-announcement message at each coordinator indefinitely, a source includes a *timeout* parameter in *data-announcement* message. If this timeout expires and a coordinator does not receive any further *data-announcement* message, it clears the information of the stimulus and the target's location to release the cache.

*b) **Query Transfer***

Every node is supposed to maintain a *Query INformation Table* (hereafter called QINT) in its cache. Each entry is identified by a tuple of (*query, sink, uplink*) (*sink* is the node which originally sends the *query*; *uplink* is the last hop from which the node receives the query). By definition, two entries in QINT are identical if all their

corresponding elements are identical. For example in Fig.3, node n1 and node n2 receive a query from sink1 and sink2, therefore it maintains a QINT as Fig.4.

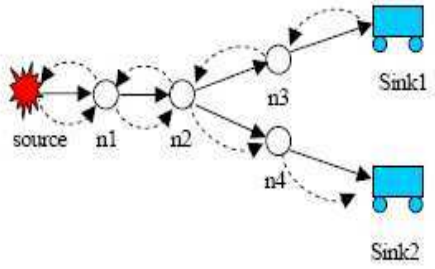


Fig.3.Query Transfer And Data Dissemination Path Setup

Node n1		
query	sink	uplink
A	sink1	n2
A	sink2	n2

Node n2		
query	sink	uplink
A	sink1	n3
A	sink2	n4

Fig.4.Query Information Table Maintained At Nodes n1 and n2

Receiving a query from an uplink node, a node first checks if the query exists in its QINT. If so, the node simply discards the query. Otherwise, it caches the query in the QINT. Then, based on target's location stored in each coordinator, it computes the ID of next grid to forward the query. This algorithm is described in Fig.5. In this figure, *NODE* is the current node handling the query packet and *src_addr* contains the target's location. If NODE is the source, it starts sending data along the data dissemination path. Otherwise, it finds the next grid which is closest to the target to relay the query. In case the next grid contains no node (so-called *void grid*) or the next grid's coordinator is unreachable, it tries to find a round path. To do this, it first calculates the disparity, δ_{CX}, δ_{CY}_..

$$\Delta_{CX} = p->src\_addr.CX - NODE.CX, \quad \delta_{CX} = \frac{\Delta_{CX}}{|\Delta_{CX}|}$$

$$\Delta_{CY} = p->src\_addr.CY - NODE.CY, \quad \delta_{CY} = \frac{\Delta_{CY}}{|\Delta_{CY}|}$$

The next grid will be
NextGrid.CX = NODE.CX δ_{CX}
NextGrid.CY = NODE.CY δ_{CY}

```
Find_Next_Grid(NODE, packet* p)
{
If (NODE is Source)
    NODE.send_data();
Else{
    Δcx = p->src_addr.CX - NODE.CX;
    Δcy = p->src_addr.CY - NODE.CY;
    δcx = (Δcx == 0) ?0: Δcx / |Δcx|;
    δcy = (Δcy == 0) ?0: Δcy / |Δcy|;
    NextGrid.CX = NODE.CX + δcx;
    NextGrid.CY = NODE.CY + δcy;
    If (lookup_neighbor_table(NextGrid) == TRUE)
        return NextGrid;
    Else
        find_round_path();
    }
```

Fig. 5. Pseudo-Code of Finding Next Grid ID Algorithm

Each node is supposed to maintain a *one-hop-neighbour table*. (i.e. information about its one-hop neighbours). If a node can not find the next grid's coordinator in this table, it considers that grid as a void grid

.

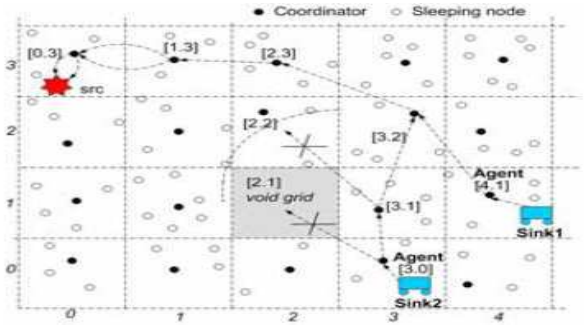


Fig.6.Multi-Hop Routing Through Coordinators

For example in Fig.6, the sink1 sends query to the source *src* along the path [4.1], [3.2], [2.3], [1.3], [0.3]. However, with the sink2, the grid [3.0]'s coordinator can not find grid [2.1]'s neighbour (due to void grid) and grid [3.1]'s coordinator also can not find grid [2.2]'s neighbor (due to unreachable node) in its one-hop-neighbour table. Therefore, it finds the round path as [3.1], [3.2], [2.3], [1.3], [0.3]. A data dissemination path is discovered by maintaining a QINT at each intermediate node. A query from a sink is re-transmitted when the sink moves to another grid. The path length of each neighbour for each sink are stored in a Neighbour Table, e.g.,

```
1:  init:
2:  CODEXT.init();
3:  routeData(DATA):
4:
CODEXT.updateFitness(DATA.Routing,
    DATA.Feedback);
5:  if (myAddr in Routing)
6:      if (explore)
7:  possRoutes =
        PST.getAllRoutes(DATA.Routing.si
        nks);
8:  route = explore.select(possRoutes);
9:  else
10: route =
        CODEXT.getBestRoute(DATA.Rout
        ing.sinks);
11: DATA.Feedback.value =
        CODEXT.getBestCost(DATA.Routi
        ng.sinks);
12: DATA.Routing = route;
13: sendBroadcast(DATA);
```

Fig. 7. CODEXT Pseudo Code Initialization And Processing Of One DATA Packet

c) Data Dissemination

A source starts generating and transmits data to a sink as it receives a query. Upon receiving data from another node, a node on the dissemination path (including the source) first checks its QINT if the data matches to any query to which uplinks it has to forward. If it finds that the data matches several queries but with the same uplink node, it forwards only one copy of data. Doing this reduces considerable amount of data transmitted throughout the sensor network. For example in Fig.4, node n1 receives the same query A of sink1 and sink2 from the same uplink node (n2). Therefore, when n1 receives data, it sends only one copy of data to n2. Node n2 also receives the same query A of sink 1 and sink 2 but from different uplink nodes (n3, n4).

Thus, it must send two copies of data to n3 and n4. Likewise, the data is relayed finally to the sinks.

2.0 Handling Sink Mobility

CODEXT is designed for mobile sinks. In this section, the authors describe how a sink keeps continuously receiving updated data from a source while it moves around within the sensor field.

Periodically, a sink checks its current location to know which grid it is located. The grid ID is computed by the formula (1). If it is still in the same grid of the last check, the sink does nothing. Otherwise, it first sends a *cache-removal* message to its old Agent. The *cache-removal* message contains the query's information, the sink's identification and the target's location. The old Agent is in charge of forwarding the message along the old dissemination path as depicted in Fig.8. After receiving a *cache-removal* message, a node checks its QINT and removes the matched query. When this message reaches the source, the whole dissemination path is cleared out, i.e. each intermediate node on the path no longer maintains that query in its cache. Consequently, the source stops sending data to the sink along this dissemination path. After the old dissemination path is removed, the sink re-sends a query to the target location. A new dissemination path is established as described in section (b) above. By doing this, the number of queries which is needed to be re-sent is reduced significantly compared with other approaches. Hence, collision and energy consumption is reduced. Also, the number of lost data packet is decreased. In case the sink moves into a void grid, it selects the closest coordinator to act as its Agent.

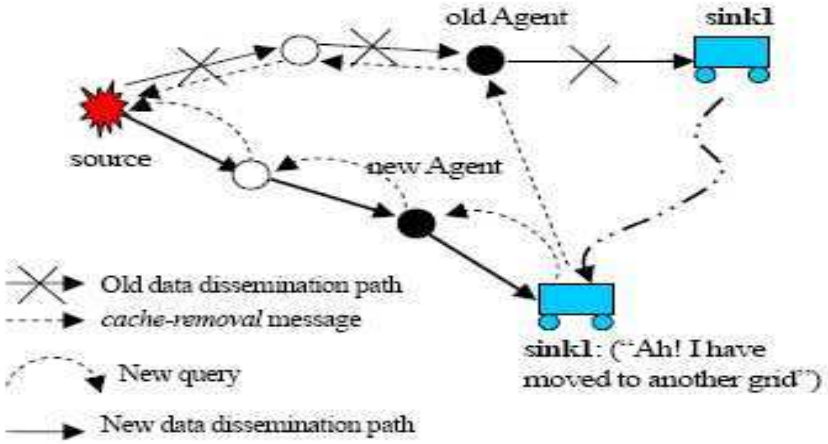


Fig.8.Handling Sink Mobility

2.1 CODEXT Performance

A. Simulation Model

Here, the authors developed a simulator based on OMNET++ simulator to evaluate and compare CODEXT to other approaches such as Directed Diffusion (DD) and CODE. To facilitate comparisons with CODE and DD, we use the same energy model used in n2 that requires about 0.66W, 0.359W and 0.035W for transmitting, receiving and idling respectively. The simulation uses MAC IEEE 802.11 DCF that OMNET++ implements. The nominal transmission range of each node is 250m, [13].

Our goal in simulating CODEXT is to examine how well it actually conserves power, especially in dense sensor networks. In the simulation, we take into account the total energy consumed for not only transmitting, receiving but also idling. The sensor network consists of *400* sensor nodes, which are randomly deployed in a *2000mx2000m* field (i.e. one sensor node per *100mx100m* grid). *Two-ray ground* is used as the radio propagation model and an omni-directional antenna having unity gain in the simulation. Each data packet has *64* bytes, query packet and the others are *36* bytes

long. The default number of sinks is 8 moving with speed *10* m/sec (i.e. the fastest human speed) according to *random way-point* model (David B, J and David A.M 1996). Two sources generate different packets at an average interval of *1* second. Initially, the sources send a data-announcement to all coordinators using flooding method. When a sink needs data, it sends a query to its Agent. As a source receives a query, it starts generating and sends data to the sink along the data dissemination path. The simulation lasts for *200* seconds. Four metrics are used to evaluate the performance of CODEXT. The energy consumption is defined as the total energy network consumed. The success rate is the ratio of the number of successfully received packets at a sink to the total number of packets generated by a source, averaged over all source-sink pairs. The delay is defined as the average time between the time a source transmits a packet and the time a sink receives the packet, also averaged over all source-sink pairs. We define the network lifetime as the number of nodes alive over time.

2.2 Performance Results

a) Impact of Sink Number

The impact of the sink number on CODEXT is first of all studied. In the default simulation, we set the number of sink varying from 1 to 8 with the max speed 10m/s and a 5-second pause time.

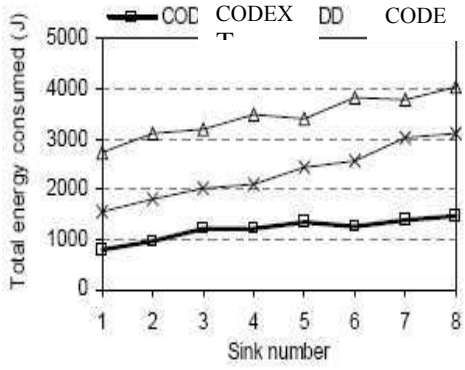


Fig.8.Energy Consumption For Different Numbers Of Sinks

Fig.8 shows total energy consumption of CODEXT. It demonstrates that CODEXT is more energy efficient than other source protocols. This is because of three reasons. Firstly, CODEXT uses QINT to efficiently aggregate query and data along data dissemination path. This path is established based on grid structure. Hence CODEXT can find a nearly straight route between a source and a sink. Secondly, CODEXT exploits GAF protocol, so that nodes in each grid negotiate among themselves to turn off its radio. Thirdly CODEXT uses the concept of SHARING TREE. The goal in CODEXT is to route the data to multiple sinks. Because standard routing tables show single sink routes, we need a new data structure to manage options for routing data through different combinations of neighbours to reach different combinations of multiple sinks. For this, we use the CODEXT Sharing Tree, a data structure that allows for easy selection of the next hop(s) for a given set of sinks. The name CODEXT sharing tree derives from the tree shape of the data structure, as well as our goal to allow a single packet to share a path as it travels to multiple sinks. This section outlines the key properties of the CODEXT [5].

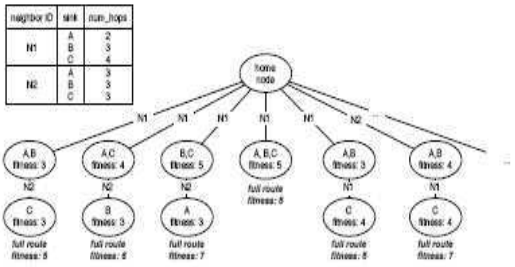


Figure 9: The Neighbour Table For A Sample "Home Node" And Part Of Its Corresponding CODEXT Sharing Tree

2.2.1 Functionality Of The CODEXT Sharing Tree

The CODEXT sharing tree is maintained at each node to keep all possible routes to all sinks through all combinations of neighbours. It is worth noting that each node, referred to as the home node in its CODEXT sharing tree, maintains only one sharing tree, independent of the number of sources, sinks,

and neighbours. Here we explore the CODEXT sharing tree through its interface. init(): The CODEXT sharing tree is initialized with data contained in the Neighbour Table. Here, we illustrate the CODEXT sharing tree contents through the small example in Figure 9 where the home node has 2 neighbours, N1 and N2, and the system has 3 sinks, A, B, and C. The intention is to use the CODEXT sharing tree to select the neighbours to serve as the next hop(s) for each of the destination sinks. As the goal is to share the routes as much as possible, the options of using a single neighbour to serve multiple sinks are considered. To illustrate the routing choices available, we observe that N1 can route packets toward any of the following neighbour combinations: {A}, {B}, {C}, {A,B}, {A,C}, {B,C}, {A,B,C}. The same subsets can be reached through N2. To select the next hops for all sinks, we must choose sets of these neighbour combinations, such that their union includes all desired sinks exactly once. For example, to route a packet to all three sinks, we could select {A,B}N1 and {C}N2, where the subscript indicates the neighbour to which the combination belongs. Alternately, {A,B,C}N1 is sufficient. The set of all possible routes for all three sinks is the brute force combination of all neighbour combinations. To structure these choices, a tree is constructed where each node is a neighbour combination. In this tree, a path from any leaf to the root represents a routing option to reach all of the sinks. For example, in Figure 9, the path from the first leaf to the home node (the tree's root) corresponds to the first selection above. The final initialization step annotates each node of the CODEXT sharing tree with it fitness value, update Fitness(route, f). As previously observed, fitness values are initial estimates which are updated as the system receives new fitness values through the feedback mechanism of the CODEXT FRAMEWORK.

Therefore, whenever a packet is overheard, its feedback values are used to update the corresponding neighbour combinations, the node(s) in the CODEXT sharing tree. update Tree(): Each time the Neighbour Table changes due to the insertion or deletion of a neighbour or sink, the CODEXT sharing tree must be updated. Since the fitness values are calculated only at initialization and updated later through feedback, it is important not to lose them during an update. Therefore, rather than rebuild the CODEXT sharing tree from scratch, an update function that makes only the required changes is provided.

GetAllRoutes(sinkSet)

Every packet carries the subset of sinks that it should be routed to by the receiving node. The CODEXT sharing tree has the responsibility to take this subset and enumerate the possible options for the next hop(s). These options can be visualized as a set of partial paths in the CODEXT sharing tree starting at the home node. Each path must include PST nodes, which union includes exactly the destination sinks. getBestRoute(sinkSet): During the stable phase of our CODEXT protocol, we rotate among all available best routes for a specified sink subset. For convenience, we place the responsibility for balancing the selection among multiple options inside the CODEXT sharing tree, providing a single function that returns only one route. Therefore, it reduces significantly energy consumption. In contrast, DD (Direct Diffusion) always propagates the new location of sinks throughout the sensor field in order for all sensor nodes to get the sink's location. In CODE, the new multi-hop path between the sink and the grid is rebuilt. Also, data dissemination path of CODE is along two sides of a right triangle.

Fig.10 demonstrates the average end-to-end delay of CODEXT. As shown in this figure, the delay of CODEXT is shorter than CODE and slightly longer than DD. In Fig.10, it shows that the success rate of CODEXT is always above 90 percent. It means that CODEXT delivers most of data successfully to the multiple sinks.

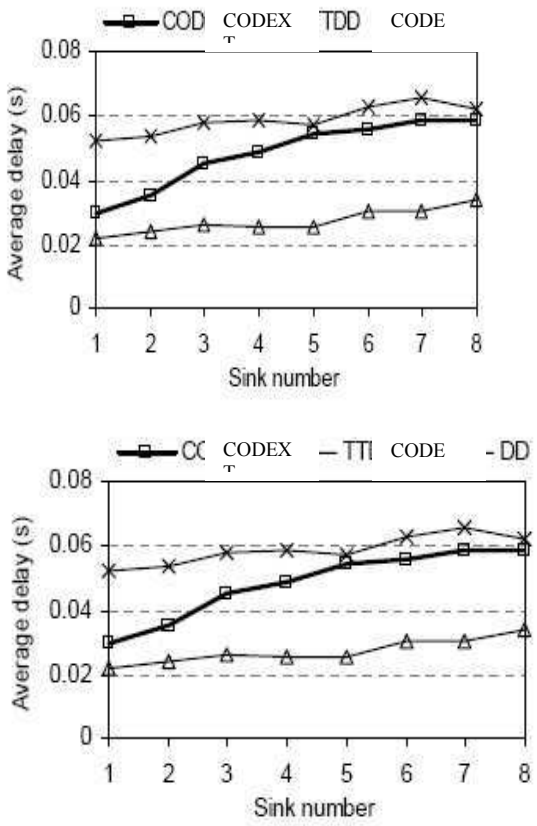


Fig.10 .Delay For Different Numbers Of Sinks

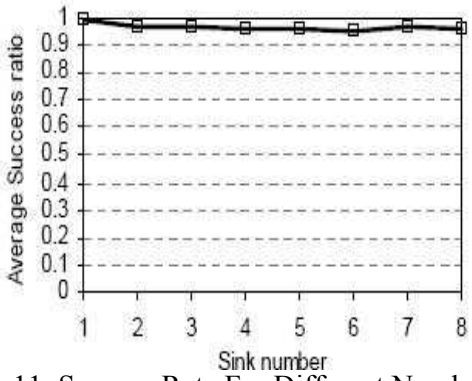
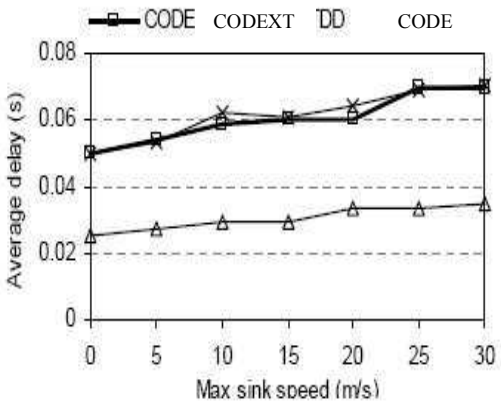


Fig.11. Success Rate For Different Numbers Of Sinks

2.2.2 Impact of Sink Mobility

In order to examine the impact of sink mobility, CODEXT is measured for different sink speeds (0 to 30 m/sec). In this experiment, the network consists of 8 mobile sinks and 400 sensor nodes.



13. Delay For Different Sink Speeds

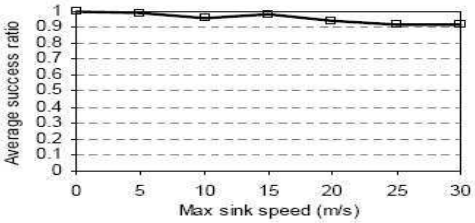


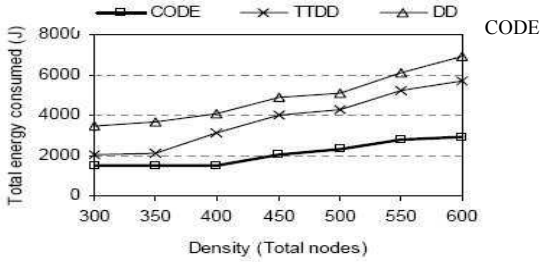
Fig.14. Success Rate For Different Sink Speeds

Fig.12 demonstrates total energy consumed as the sink speed changes. In both low and high speeds of the sinks, CODEXT shows the total energy consumed is better than other protocols, about twice less than CODE and three times less than DD. The reason is that, aside from above explanations, the mobile sink contact with the coordinator to receive data while it is moving. Thus, the query only needs to resend as it moves to another grid. Fig.13 shows the delay of CODEXT which is comparable with CODE and longer than DD. In Fig.14, the success rate is also above 90 percent. These results show that CODEXT handles mobile sinks efficiently.

2.2.3 Impact Of Node Density

To evaluate the impact of node density on CODEXT, we vary the number of nodes from 300 (1 node/cell on average) to 600 nodes (2 nodes/cell). Eight sinks move with speed 10m/sec as default. Fig.15 shows the energy consumption at different node densities. In this figure, CODEXT demonstrates better energy consumption than

other protocols. As the number of nodes increase, the total energy consumption slightly increases. This is because of turning off node's radio most of the time. Therefore, energy is consumed mostly by the coordinators. While in CODE and DD, nodes which don't participate in communication still consume energy in sleeping mode.



CODE

Fig.15 .Energy Consumption For Different Node Density

2.2.4 Network Lifetime

In this experiment, the number of sinks is 8 moving with speed 10 m/sec. The number of sensor nodes is 400. A node is considered as a dead node if its energy is not enough to send or receive a packet. Fig.15 shows that number of nodes alive of CODEXT is about 60 percent higher than CODE at the time 600sec. This is due to two reasons: The first is that CODEXT focuses on energy efficiency. The second is that rotating coordinators distribute energy consumption to other nodes, thus nodes will not quickly deplete its energy like other approaches. CODEXT concentrates on dissemination nodes to deliver data, therefore such nodes will run out of energy quickly. We do believe that when the node density is higher, the lifetime of CODEXT will be prolonged much more than other approaches.

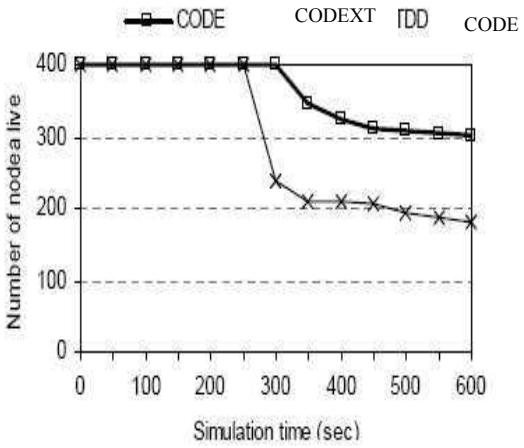


Fig.16. Number Of Node Alive Over Time

2.3 Conclusion

Many sensor network protocols have been developed in recent years. [2], [4], [12]. One of the earliest work, SPIN [3] addresses efficient dissemination of an individual sensor's observation to all the sensors in the network. SPIN uses meta-data negotiations to eliminate the transmission of redundant data. Directed Diffusion [3] and DRP [7] are similar in that they both take the data-centric naming approach to enable in-network data aggregation. In Directed Diffusion, all nodes are application-aware. This enables diffusion to achieve energy saving by selecting empirically good paths and by caching and processing data in-network. DRP exploits application-supplied data descriptions to control network routing and resource allocation in such a way as to enhance energy efficiency and scalability. GRAB [14] targets at robust data delivery in an extremely large sensor network made of highly unreliable nodes. It uses a forwarding mesh instead of a single path, where the mesh's width can be adjusted on the fly for each data packet. GEAR [14], uses energy aware neighbour selection to route a packet towards the target region. It uses Recursive Geographic Forwarding or Restricted Flooding algorithm to disseminate the packet inside the destination regions.

While such previous work only addresses the issue of delivering data to stationary sinks, other work such as CODE [6], SEAD [2] and SAFE [9], [3] target at efficient data

dissemination to mobile sinks. CODE exploits local flooding within a local cell of a grid which sources build proactively. Each source disseminates data along the nodes on the grid line to the sink. However, it does not optimize the path from the source to the sinks. When a source communicated with a sink, the restriction of grid structure may multiply the length of a straight line path by 2. Also, CODE frequently renews the entire path to the sinks. It therefore increases energy consumption and the connection loss ratio. SAFE uses flooding that is geographically limited to forward the query to nodes along the direction of the source. SAFE uses geographically limited flooding to find the gate connecting itself to the tree. Considering the large number of nodes in a sensor networks, the network-wide flooding may introduce considerable traffic. Another data dissemination protocol, SEAD, considers the distance and the packet traffic rate among nodes to create near-optimal dissemination trees. SEAD strikes a balance between end-to-end delay and power consumption that favors power savings over delay minimization. SEAD is therefore only useful for applications with less strict delay requirements.

CODEXT differs from such protocols in three fundamental ways. First, CODEXT exploits GAF protocol [13] to reduce energy consumption and data collision while the nodes make decision to fall into sleeping mode. Second, based on grid structure, CODEXT can control the number of transmitted hops and disseminates data along a path shorter than others such as CODE. Third, the number of re-transmitted queries is reduced by maintaining an Agent to relay data to the sink when it moves within a grid. In addition, CODEXT takes into account of query and data aggregation [5], [6] to reduce the amount of data transmitted from multiple sensor nodes to sinks like other approaches.

References

[1] Fan Ye et al (2002) "**Sensor Networks: A Two-Tier Data Dissemination Model For Large-Scale Wireless Sensor Networks**" *Proceedings of the Eighth Annual ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM 2002)*, Atlanta, GA.

[2] Hyung Seok Kim et al (2003) "**Dissemination: Minimum-Energy Asynchronous Dissemination To Mobile Sinks In Wireless Sensor Networks**" *Proceedings of the first international conference on Embedded networked sensor systems*.

[3] Intanagonwiwat C et al (2003: 2-16) "**Directed Diffusion For Wireless Sensor Networking**" *Networking, IEEE/ACM Transactions* Vol 11 Issue.1.

[4] Joanna Kulik et al (2002), "**Negotiation-Based Protocols For Disseminating Information In Wireless Sensor Networks**" *ACM Transaction* on Vol 8 , Issue 2.

[5] Krishnamachari B, Estrin D, and Wicker S . (2002) "**The Impact Of Data Aggregation In Wireless Sensor Networks**". *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*.

[6] Maddes S et al (2002) "**Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Network**". *IEEE Workshop on Mobile Computing Systems and Applcation.* .

[7] Nirupama B et al (2000:28-34), "**Gps-Less Low Cost Outdoor Localization For Very Small Devices**". *IEEE Personal Communications Magazine*, Vol 7.

[8] Pottie G J and Kaiser W J (2000:51-58). "**Embedding The Internet: Wireless Integrated Network Sensors**". *Communications of the ACM*, Vol 43.

[9] Sooyeon Kim et al (2003:228-234); "**A Data Dissemination Protocol For Periodic Updates In Sensor Networks**" *Workshops*, Proceedings. 23rd International Conference on *Distributed Computing Systems*.

[10] Stemm M and Katz R H. (1997) "**Measuring And Reducing Energy Consumption Of Network Interfaces In Hand-Held Devices**". *IEICE Transaction and communication*.

[11] Wendi B et al (1995) "**An Application-Specific Protocol Architecture For Wireless Microsensor Networks**" *IEEE transactions on wireless communications.*

[12] Wensheng Zhang et al (2003:305-314) "**Data Dissemination With Ring-Based Index For Wireless Sensor Networks**" *Proceedings. 11th IEEE International Conference* on Wireless Netwoking.

[13] Xu Y et al (2001), "**Geography-Informed Energy Conservation For Ad Hoc Routing**". *Proceedings . of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy.

[14] Yan Yu et al ,(2001) "**Geographical And Energy Aware Routing: A Recursive Data Dissemination Protocol For Wireless Sensor Networks",** UCLA Computer Science Department.