uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Chendong Xu
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Delay-constrained Power-efficient Data Aggregation Framework in Sensor-Actor Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. Stojmenovic
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Prof. A. Nayak
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. P. Flocchini

Prof. T. Kunz

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Delay-constrained Power-efficient Data Aggregation Framework in Sensor-Actor Networks

by

Chendong Xu

A thesis submitted to
the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario K1N 6N5, Canada

Your file Votre référence
ISBN: 978-0-494-59894-8
Our file Notre référence
ISBN: 978-0-494-59894-8

Acknowledgements

I would like to acknowledge my sincerest gratitude to my supervisor Prof. Dr. Ivan Stojmenovic of Ottawa-Carleton Institute for Computer Science. Without his enthusiastic and expert guidance during writing this thesis, it would not be possible to complete the thesis. I am grateful to my co-supervisor Prof. Dr. Amiya Nayak for the help in preparation of this thesis. I would like to thank Dr. Xu Li from Carleton University for the valuable advices and discussions in thesis writing and experiments. I would also like to thank my family and my wife Yanyan Liang in particular for their continuous understanding and support.

Contents

List of Figures

List of Table

Abstract

In *data aggregation* problems, sensor measurements from the whole sensory field are collected at the data sink periodically or on-demand as a single report using functions such as average, maximum, minimum, counts, deviation, etc. This thesis is to design a data aggregation framework applicable for real-time sensor-actor networks. Our goal is to set up a reporting tree that will minimize power consumption at individual nodes while preserving delay requirements.

Existing solutions to data aggregation problem usually use hop count as the energy cost metric and/or operate in a centralized fashion. The only known delay bounded power efficient localized algorithm [MGPA] sets up an initial power efficient tree (regardless of delay), and then dynamically changes the tree based on measured delay in ongoing traffic, with speed-ups and slow-downs achieved by using maximal/minimal transmission ranges at some nodes. We show here that the initial tree is closer to hop count than power optimal while the energy consumption per node is apparently unbalanced.

We propose to construct a power optimal delay bounded data aggregation tree, assuming delay is proportional to hop count, which is a reasonable approximation in low traffic scenarios. Our *desired hop progress* (DHP) scheme constructs a data aggregation tree rooted at a sink/actor using only edges of localized minimal spanning tree (LMST) over all sensors, if delay along this tree is acceptable and power consumption is to be near optimal, like in [TKS]. Otherwise, hop selection (along LMST) made at each step is subject to the ratio of potential delay to message lifetime. The main idea is to reduce the network's overall energy consumption and balance energy consumptions at nodes by applying approximately equal hop lengths along the tree. There are two variants of DHP: DHP with Area coverage algorithm (DHPA), and DHP with Area coverage algorithm and CDS construction algorithm (DHPAC). In DHPA algorithm, area coverage algorithm is applied first to select a subset of active sensors that monitors the same area as the original set, thus allowing the rest of sensors to sleep. Then DHP is applied on the set of active sensors. In DHPAC protocol, a connected dominating set (CDS) is constructed over active sensors. Active sensors not in CDS report to their nearest CDS neighbors (related delay counted at the parent node), and DHP is then applied over LMST of CDS nodes and links in CDS. Experimental results indicate that DHP can totally save up to 75% energy and extend up to 123% network lifetime in comparison with [MPGA]. Meanwhile, DHPA and DHPAC also present significantly better energy efficiency than the variant of [MPGA], where area coverage algorithm is also implemented.

Chapter 1: Introduction

1.1Background

Sensor devices have been well developed in recent years. Typically, sensor devices are small in size and are equipped with limited power supply, memory, and simple-function CPU. These low cost, low power, and multifunctional devices deployed over an area of interest can operate autonomously to detect concerned events, process data, and report sample result. Furthermore, sensor devices are able to self-organize to form a network, and communicate with each other using their wireless interfaces.

In most applications, sensors are expected to send their detected data to data sinks, which have no resource limitation. At the sinks, the collected information is further processed for end-user querying. When predetermined data sinks (actors) are out of its communicating range, a sensor will rely on other nodes in its network to forward its data packets. All these features make sensor network feasible and practical in real world. A data sink or actor collects required data from data sources (sensors). When data correlation exists in networks, data from different sources can be combined into a single report at intermediate nodes.

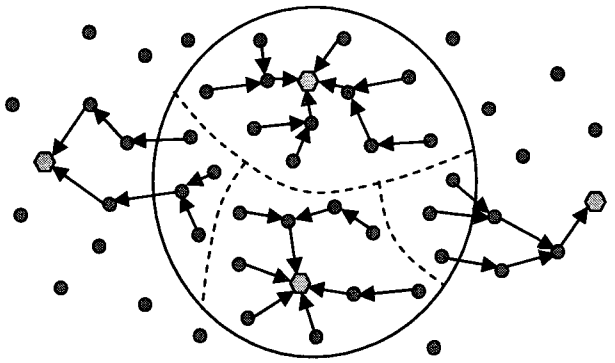


Figure 1. Finding nearest actor for each sensor from geocasting region

Sampling results collected at sources are sent to predetermined data sinks/actors. Reporting can be event-driven or request-driven. Event-driven reporting starts at sensors. When interesting events take place, sensors deployed in event range will generate related data reports and send them to predetermined data sinks (actors). In this thesis, we are

interested in the request-driven reporting, where all sensors report periodically upon request from a sink/actor. Data sinks/actors trigger reporting. We assume that they simultaneously flood the network. Sensors receiving a message originated from a sink/actor will retransmit it, so that neighboring sensors learn its path to them. Sensors will store the path to only the nearest actor/sink, and will not retransmit information received from sinks/actors further away from the nearest one to prevent over-flooding. Through a query request, data sinks/actors ask sensors in the entire network or from a certain area to report their sampling results. Broadcasting is applied if sensors from the entire network are to report, while geocasting can be used to activate only sensors from a region. This is illustrated in Figure 1, which shows paths from sensors to their nearby sink/actor. We will describe later how these paths are constructed and updated.

1.1 Problem Statement

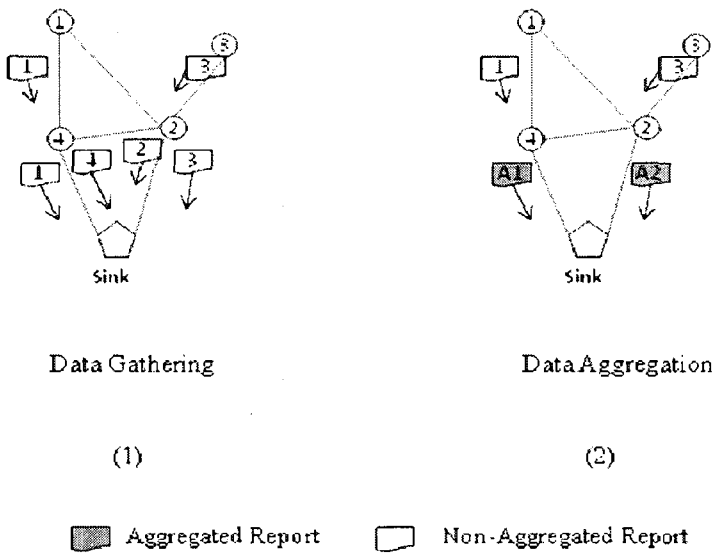


Figure 2. Data Gathering and Data Aggregation

In data gathering, each source independently sends data to its associated sink. For highly correlated data, such a reporting generates a lot of redundant traffic. In order to save power and bandwidth, data should be locally processed before being forwarded.

Data aggregation is recommended in this scenario. It reduces the number of packets to be retransmitted among sensors, and consequently reduces communication overhead and power consumption. The paradigm shifts the focus from the traditional address-centric approaches to a more data-centric approach. The problem can be stated more formally as follows.

Data Aggregation: The sources send data to the sink/actor, with aggregation/consolidation function applied over the data originated at multiple sources on their way to the sink/actor.

We assume that an aggregation function, g, is selected by a data aggregation application. One intermediate node, N, will implement aggregation, $g(f(x))$, before its packet forwarding, where $f(x)$ denotes data from a node set, x, including all N's children and itself. The data aggregation functions could be basic operations such as COUNT, MIN, MAX, AVERAGE, and SUM.

Figure 2 is a simple illustration of the difference between data gathering and data aggregation. In data gathering (without aggregation), all sources (1, 2, 3, and 4) send their data separately to the sink. In data aggregation, when data from source nodes 1 and 3 reach intermediate nodes 4 and 2, they are processed/compressed/combined. Then only aggregated data/reports like A1 (including data from nodes 1 and 4) and A2 (including data from nodes 2 and 3) will be forwarded. Obviously, with the application of aggregation, energy consumption is reduced because of fewer transmissions.

Data aggregation applications can be classified into *fully-aggregated* and *partially-aggregated* applications. In full aggregation, complete redundancy exists between data from different sources. Two pieces of data can be combined and reduced to the same size as those of the original pieces. As for partial aggregation, messages cannot be reduced to the same size as the original data size. Only a part of the information in each piece of data is redundant. For instance, in Figure 2(2), when full aggregation is performed by node 2, the generated packet will be as long as a single original packet. Once partial aggregation is performed in node 2, the length of aggregated packet A will be between the sum of all packets' lengths and one original packet length. All cases discussed in this thesis will apply full aggregation.

As an aggregated report includes data from multiple sources, the method of counting such report's experienced hops should be different from that used by traditional data gathering. The number of hops experienced by an aggregated report should be equal to the hop count from the farthest source node to the receiver, not from the report generator. For example, in Figure 2(1), the report generated by node 2 experiences one hop when it reaches sink. However, in Figure 2(2), the experienced hop count of report A (generated by node 2, too) should be two at the sink, because it includes data from nodes 1, 3, and 4.

The volume of energy consumed in one transmission will vary greatly with its transmission distance. We adopt the first-order radio model [RH] to compute a node's transmission power. The volume of a node's energy consumed in one transmission is equal to $\beta d^{\alpha} + c$, where β is a constant related to the transmission amplifier, d is transmission distance, $\alpha \geq 2$ is a signal attenuation factor, and c is the processing energy consumed by electronic units (of transmitter and receiver).

We are particularly interested in power optimality under constrained delay requirements. That is, reports should arrive at sink within certain time limits. Potential applications include a timely response to a chemical/biological attack in a large portion of or even whole field with subsequent updates.

As we are discussing data aggregation in low-rate sensor networks, we use the metric "hop count" for the delay measurement. Typically, a packet delay between two neighboring nodes is composed of queuing delay, processing delay, propagation delay, and transmission delay [ZSS]. Propagation delay is negligible compared to others. Due to aggregation, for each sensor, only one aggregated packet is sent in each round of data collection. In low traffic scenarios, delays are mainly caused by a MAC layer protocol applied at each node (hop) and are approximately equal. In fully-aggregated applications, data reports generated by different nodes should be of the same size. When bandwidth is identical for all sensors, transmission delay in each transmission should also be the same. With respect to processing delay, we believe that the time spent on handling one packet at different nodes is the same. Overall, the delay for any report transmission should be similar. We can say one report's experienced delay is proportional to its experienced hop count. Therefore, we are going to measure delay by hop count in this thesis.

The selection of an aggregation protocol for a certain network depends on the information available at the network nodes. In networks where each node is provided with full topology information and which are stable over time, a centralized solution can be applied to guide data aggregation in the entire network. It may lead to an optimal aggregation structure. However, if network nodes do not know the full network graph, centralized solutions will not work as expected. In contrast, localized solutions only require the information available from neighbors to make decisions. Although this type of protocols can only reach near-optimal results, in large scale sensor networks they will be more practical with reasonable communication overheads. Therefore, we prefer localized schemes to solve our data aggregation problem in sensor networks.

1.3 Existing Solutions

With the development of forwarding schemes with aggregation, various structured data aggregation schemes have been studied. Among them, there are some cluster-based approaches [HCB], where the cluster header nodes perform data aggregation. On the other hand, some solutions such as [TKS], [KEW], [ZSS], [KKK] and [MPGA] are tree-based where each node collects data from its downstream nodes (children) and sends aggregated data to its parent node.

[HCB] proposed a localized cluster-based aggregation protocol, named LEACH. By this protocol, the network is clustered and random cluster-headers are selected to aggregate collected data and transmit them directly to an actor. Through in-cluster aggregation, the protocol reduces the amount of information sent to remote actors, thereby saving significant energy. However, delay problems are not mentioned in [HCB].

[TKS] proposed a topology based energy efficient aggregation protocol. The protocol first computes a localized minimum spanning tree (LMST) over all network nodes, where transmission energy is considered as the costs of links. Then it constructs an aggregation tree through a flooding over the LMST. The aggregation tree resulting from this protocol should be energy efficient, but no delay constraint is discussed.

In [KEW], the authors studied energy consumption and delay in data aggregation. Authors recommended three centralized protocols to construct an energy suboptimal aggregation structure. Hop count is selected as a metric for energy consumption in

[KEW]. Although delay was discussed in [KEW], no delay-bound for data aggregation was mentioned. [ZSS] proposed a scheme to build a delay-bounded SMT based on the bounded shortest multicast algorithm (BSMA), a centralized solution. The aggregation tree resulting from the algorithm is a near-optimal structure considering delay bound. Like [KEW], the authors also use hop count to measure network energy consumption.

[KKK] introduced a structure called delay bound minimum degree spanning tree (DB-MDBT). The protocol pursues most per-node fairness (in energy consumption) considering delay bound. Its aggregation tree construction needs global information, and hop count is treated as the metric of network energy consumption.

[MPGA] presented a centralized scheme, named integer linear program (ILP, proved to be NP-hard), and a distributed solution, named multi-state (MS) framework, for applications in multi-actor sensor networks. Both protocols pursue minimum energy consumption in data aggregation considering delay bounds. The authors selected first-order radio model to compute energy costs between nodes, and measured delay by time. The MS framework sets an initial power efficient tree (regardless of delay), and then dynamically changes the tree based on measured delay in ongoing traffic, with speed-ups and slow-downs achieved by using maximal/minimal transmission ranges at some nodes. The framework expects to save as much energy as possible, while maintaining the rate of unexpired reports in an expected range. However, we show that its initial tree is not really optimized for energy consumption except for hop counts at nodes far from an actor and leads to energy imbalance in the entire network.

Among all mentioned algorithms, only the MS framework proposed by [MPGA] is a localized solution to data aggregation problem, which pursues network energy efficiency considering delay limit. It selects the first-order radio model in counting node transmission energy and hop counts to measure delay in simulations (although it measures delay by time in theoretical analysis), similar to our selection. Therefore, the MS framework is an alternative and competing solution to those we propose here.

1.4 Motivations and Objectives

In single-actor request-driven real-time networks, we are going to design a localized energy efficient solution to data aggregation problem with delay bound, where the first-order radio model is selected as energy model and delay is measured by hop count.

Most existing solutions to delay-bounded data aggregation problem select hop count as the energy cost metric and/or adopt centralized solutions. The MS framework [MPGA] is the only known localized delay-bounded power efficient data aggregation algorithm that uses the first-order radio model. However, it has been proven to be not as energy efficient as expected in both initialization and tree update phases.

This thesis intends to design a delay-constrained power-efficient (DCPE) framework to solve the data aggregation problem. The aggregation tree resulting from solutions in this framework is expected to have approximately minimum overall energy and balanced energy consumption, while its height (proportional to delay) is controlled in the given limit.

1.5 Assumptions

We assume data aggregation will take place in a wireless low-rate sensor-actor network. The network is composed of one actor and a collection of homogenous sensors. All nodes are static after being deployed. There is no void area in sensor networks. Any greedy geographic routing can work well in such networks. Sensors will detect the physical environment around them and report sampled data if required. All nodes are aware of their positions by means of GPS or a position determination algorithm [HB]. All sensors in the network are time synchronized by one of the existing synchronization algorithms [SBK].

Since sensors are energy constrained, we assume each sensor is aware of its residual energy. If a sensor's residual energy is less than a threshold E_{th}, it will broadcast a quit message in its neighborhood and then switch off.

Ideal MAC and physical layer are used. Packet transmissions have no collisions and no failures. Antennas are omni-directional. The network is modeled with the unit disk graph (UDG) where nodes may communicate directly if their distance is smaller than communicating radius. As a consequence, all links are bidirectional. Multi-hop transmission is used when a destination node is outside the transmission range of a source node. The sensor's sensing is also modeled with UDG. Typically, a sensor's communicating radius will be longer than or equal to its sensing radius. The ratio of both can be arbitrary.

Finally, we regard unexpired packets as reliable packets since discussed data aggregation is delay constrained. For ease of analysis, we use a metric, named *reliable ratio* (calculated at the actor), to evaluate protocols' performance in meeting a delay bound, where reliable ratio is equal to the number of reliable (valid) packets over the number of all received packets.

1.6 Contribution

The Delay-Constrained Power-Efficient (DCPE) framework includes one localized solution to delay-bounded data aggregation problem concerning network energy efficiency and two variants, which add sensor area coverage protocol and connected dominating sets construction protocol to improve network topology. The main idea is to optimize nodes' energy efficiency (overall consumption and distribution) by balancing each node's hop length (proportional to delay) along aggregation tree.

Our desired hop progress (DHP) scheme first constructs a localized minimal spanning tree (LMST) rooted at actor and connecting all sensors, only considering node energy consumption. Then, the algorithm adjusts the LMST to be an aggregation tree with a height no more than the given limit. When delay along LMST is acceptable, no change will happen and the LMST will be the final aggregation structure, similar to [TKS]. Otherwise, nodes detecting their paths along LMST can not meet delay requirement will attach to neighbors with desired hop progress to meet delay constraint.

As area coverage algorithm can generate a power efficient subnet, which can complete monitoring tasks as the original set. Thus, we combine our DHP protocol and the area coverage algorithm into a DHPA algorithm. In the DHPA algorithm, DHP protocol is implemented over active nodes selected by the area coverage algorithm. After that, we introduce a DHPAC algorithm, where a connected dominating set (CDS) is constructed over active sensors. Active sensors not in CDS do not follow DHP protocol to select parents. They just attach themselves to their nearest CDS neighbors. DHP is applied over LMST of CDS nodes.

1.7 Performance Analysis and Future Work

Three algorithms in the DCPE framework pursue network energy optimality considering delay limit. To evaluate their performance, we simulate these algorithms and compare them with the MS framework [MPGA]. In experiments we measure the average node energy consumption in 5 data aggregations and the number of iterations before the first node failure (call it network lifetime). Experimental results show that DHP can save energy by 25-75%. Also, the network lifetime is extended by 20-123% if the delay limit of data aggregations is not strict. In addition, we compared DHPA and DHPAC with MS combined with area coverage algorithm (MSA). We found, in comparison with MSA, in small delay limit data aggregations, DHPA consumed 15-50% less energy, but network life caused by it is close to MSA. If the assigned delay limit is large, DHPA can save 25-70% energy with 50-123% network life extension. DHPAC can keep evidently better energy efficiency than MSA, no matter what delay limit is: its energy consumption is 40-70% of that by MSA, and its network life is 133-250% of that of MSA.

Overall, our protocols are more energy saving than MS/MSA. Besides, compared to MSA, DHPAC is able to extend significant network life. As for DHP/DHPA, they also lead to considerable network life extension to MS/MSA if the assigned delay limit is not tight. However, if delay limit is low enough, the network life of MS/MSA may be close to that of DHP/DHPA in certain scenarios.

Future work include adjusting our DHP protocol to data gathering (without aggregation) problem, and considering dynamic energy balancing of DHP protocol when traffic is not low. In addition, we may apply our solutions in scenarios with realistic MAC and physical layer, where collisions and transmission failures occur.

1.8 Organization of the Thesis

This thesis is organized as follows: Chapter 2 gives a review of literature on area and node dominating set construction algorithms, LMST construction algorithm and LMST-based data gathering, and three delay-bounded energy efficient aggregation protocols; Chapter 3 proposes the Desired-Hop-Progress (DHP) data aggregation protocol; Chapter 4 describes DHP with area coverage algorithm (DHPA) and DHP with area coverage

algorithm and connected dominating set (DHPAC) algorithms; in Chapter 5, simulation results are presented followed by conclusion and future works in Chapter 6.

Chapter 2. Literature Review

2.1 Power Efficient Backbone Construction and Topology Control

A backbone is a subset of sensors sufficient for performing assigned tasks. It may have particular properties for different motivations or tasks. As power efficiency is one of the important issues explored in this thesis, the two backbones discussed in this chapter have a common characteristic: power-efficient. Both backbones, area and node dominating (coverage) sets, are expected to increase network energy efficiency somehow.

In our proposed protocol and framework, we are going to involve one topology control algorithm: local minimal spanning tree (LMST) where transmission power is regarded as link cost. The non-loop LMST used in this thesis is sparse (only n-1 links, where n is node number) and built in a distributed way. We can realize a network-wide broadcasting or data aggregation through this structure at approximately minimal energy cost.

2.1.1 Area Coverage Algorithm

Chips \ Status	Sleep	Idle	Receive	Transmit
CC2420 [BCDC]	144nw	712μw	35.28mw	30.67mw
ZMD44101[ZMD]	4.8μw	2.4mw	74.4mw	76.8mw
UZ2400[UZ]	3.6μw	13.68mw	34.2mw	41.4mw

Table 1 Some Zigbee Chips Power Consumption Measurements

Wireless sensor networks are often deployed over remote and severe environment. Sensors may be forced to quit for various reasons, like energy exhaustion or electronic unit failures. In order to prolong network lifetime, sensors are typically overly deployed in an interest area. Therefore, in many wireless sensor networks, there exist extra sensors.

From Table 1 [BCDC] [ZMD] [UZ], we find that a sensor in sleep mode spends much less energy than it does in other modes. In order to save energy, obviously, we should allow nodes that do not have monitoring tasks to sleep. One node that can be free from its monitoring task must satisfy the condition that its monitoring area is effectively covered by other active sensors. To check whether one sensor is qualified to sleep, we recommend an area coverage algorithm.

Gallais et al. [GCSS] proposed several localized area coverage protocols applicable for arbitrary ratios of communication and sensing radii. In these protocols, sensors can make decisions without knowledge about neighbors ahead, which consequently decreases communication traffic.

Sensors in the network are supposed to be time synchronized by one of the existing synchronization protocols [SBK]. At the beginning of each round, every sensor sets a random timeout (avoid simultaneously sending messages between two neighboring nodes), and listens to messages from neighboring nodes before timeout expires. When timeout expires, each node evaluates its coverage and makes a decision to be active or sleep. [GCSS] provided 4 schemes to inform neighboring nodes with 3 kinds of messages (positive message, negative message, and retreating message):

1). Positive message Only (PO): only nodes to be active will send a message.

2). Positive and Negative message (PN): all nodes will broadcast their decisions to direct neighbors. A positive message indicates that the node will stay in active status, and a negative message indicates that this node will switch to sleep mode.

3). Positive and Retreat message (PR): once timeout expires, a node evaluates its coverage, and only nodes deciding to be active send decisions. Then those active nodes keep listening to messages from other neighbors, and coverage evaluation is implemented when new announcements come. If subsequent messages show relating nodes can provide more support so that receivers can be free from monitoring tasks, those receivers that have already declared to be active may 'change' their modes to sleep. Such nodes will send a retreating message.

4). Positive, Negative, and Retreat message (PNR): when timeout expires, all nodes will evaluate their coverage and send a decision message. Active nodes continue listening. If subsequent messages show one active node can attain enough support to allow it to sleep, this node will send a retreat message.

In order to evaluate coverage, [SBK] recommended an intersection-based coverage algorithm based on theories in [H] as follows:

If there are at least two covering circles and any intersection point of two covering circles inside the sensing area is covered by a third covering circle, then the sensing area is fully covered (no uncovered area).

In other words, a disk O is fully covered by other disks if and only if every intersection point of disks O_1 and O_2 inside O is covered by another disk O_3. In addition, intersection points of any other disk O_1 with O must also be fully covered by another disk O_2.

In addition, when coverage evaluation is required, a sensor implements an intersection-based coverage algorithm based on knowledge extracted from received declarations, including position and status of neighbors. If already known neighbors, *NS*, cannot fully cover the sensor's sensing area, the sensor will decide to stay active. Otherwise, the connectivity of active ones in *NS* will be further checked. If connected, the sensor can sleep; otherwise, the sensor has to stay in active status.

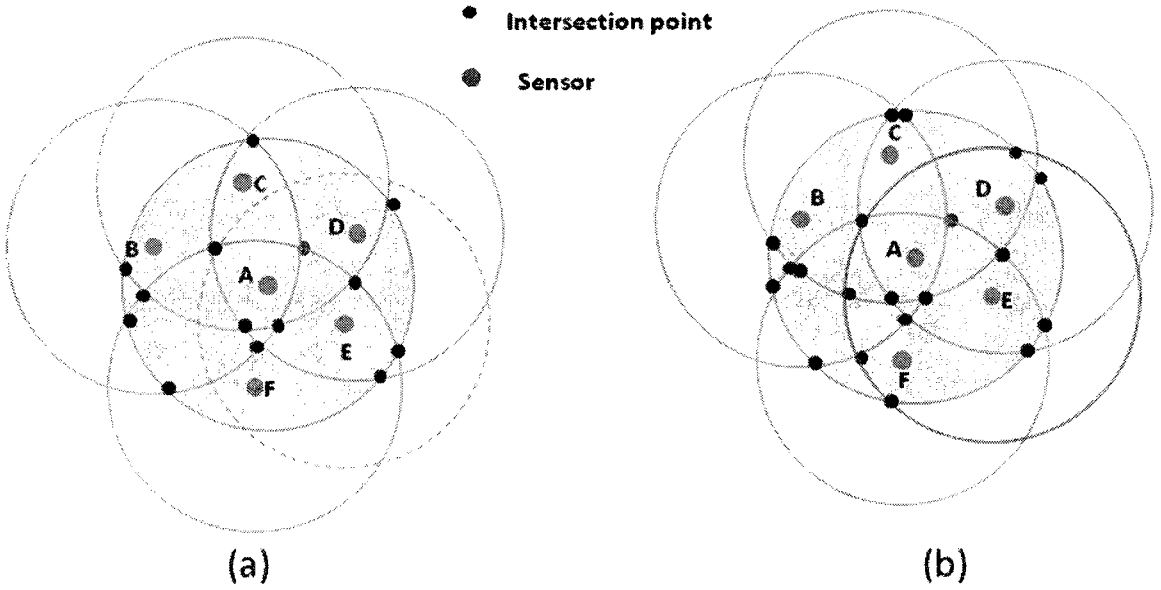


Figure 3. Sponsors' Connectivity

For instance, in Figure 3(a), in node A's sensing area (shadow area), any intersection point (black points) of two covering circles among covering circles of nodes B, C, D, and F as well as node A is covered by at least one covering circle other than two circles

generating the point. Hence, the sensing area of node A can be fully covered by nodes B, C, D, and F.

A sensor's communication range is usually no less than its sensing range. If the communication radius is no less than two times the sensing radius, nodes B, C, D, and F must be connected since distance between two points in one circle is less than the diameter of the circle (=communication radius). Under this situation, node A will switch to sleep mode after it receives decisions from nodes B, C, D, and F. However, once the communication radius r_c is in $[r_s, 2r_s)$, where r_s is equal to sensing radius, node E has to participate in the group that supports node A to sleep (we suppose one extreme $r_c = r_s$) as shown in Figure 3(b). It is because without the aid of node E, node F will be disconnected from nodes B, C, and D.

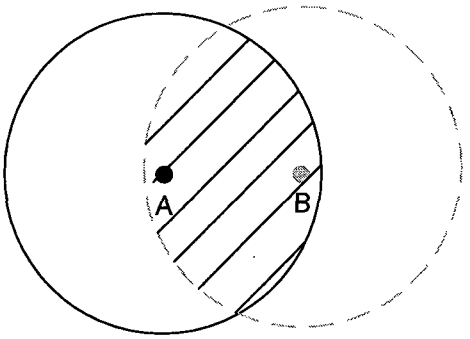


Figure 4. Negative Message Contribution

As mentioned, [GCSS] provides 4 schemes to declare a node's decision. Negative and retreat message broadcasting is not necessary although positive message broadcasting is. Actually, a negative or retreat message implies that the related node's sensing area is covered by other node(s) already. Therefore, in uncovered area deletion negative and retreat messages are as effective as positive messages.

For instance, in Figure 4, node A receives a negative advertising message from node B, which indicates that the sensing area of node B, including the overlapped (shadow) area between both nodes' sensing areas, is covered by other neighbors already. The negative message tells node A that the covering circle of node B can contribute to the removal of uncovered area, but it has to be neglected in connectivity check of active nodes. Therefore, if node A finds that its sensing area is fully covered by node B and a set

S of active neighbors, it chooses to sleep only if the nodes in set S are connected. Otherwise, node A will stay active.

A retreat message has same function as a negative message. An interesting simulation result in [GCSS] shows that a retreat message has a higher efficiency in active node cutting than a negative message while the total volume of messages sent is less.

It is proved in [CS] that an active node set resulting from the sensor area coverage protocol is connected, if the original network N is connected. The author proved that over the same network area-dominating set G includes node-dominating set F generated by coverage algorithm which will be proved in the next section to be connected. According to the definition of node-dominating set, we know that any node in set G but not in set F must be connected to one node in F.

Finally, when border monitoring is taken into account, the intersection-based area coverage protocol can resolve marginal node overwork problem. Normally, nodes located close to border have no chance to sleep, since no node is deployed at certain side of these nodes and there is always uncovered area for such nodes. In this scenario, if the uncovered area beyond the border can be ignored, marginal nodes will have chance to switch to sleep mode.

Example

In Figure 5, node 4 has 5 neighbors in its sensing area. Nodes' communicating radii are twice their sensing radii. In this case, all nodes in the figure can communicate with each other directly. Nodes do not have any knowledge about their neighbors at the beginning. Each node randomly sets a unique timeout before area coverage evaluation takes place in any node. In the process of evaluation, every node will listen to advertising messages from its neighbors before its timeout expires. When timeout expires, the node evaluates its uncovered area by intersection-based coverage algorithm, based on received advertising messages. If known neighbors cannot provide enough support, the node will decide to stay active and broadcast its decision (including node id and position) at most transmission power, such as nodes 1, 2, 3, 5, and 6 in Figure 5(2). Conversely, in Figure 5(3), node 4 receives 5 decision messages when its timeout expires, and the received messages show that neighbors can completely cover its sensing area according to

intersection-based coverage evaluation protocol: all intersection points in sensing area of node 4 (black points) are covered by third circle. Then node 4 decides to switch to sleep mode and informs its neighbors of its decision as shown in Figure 5(4).

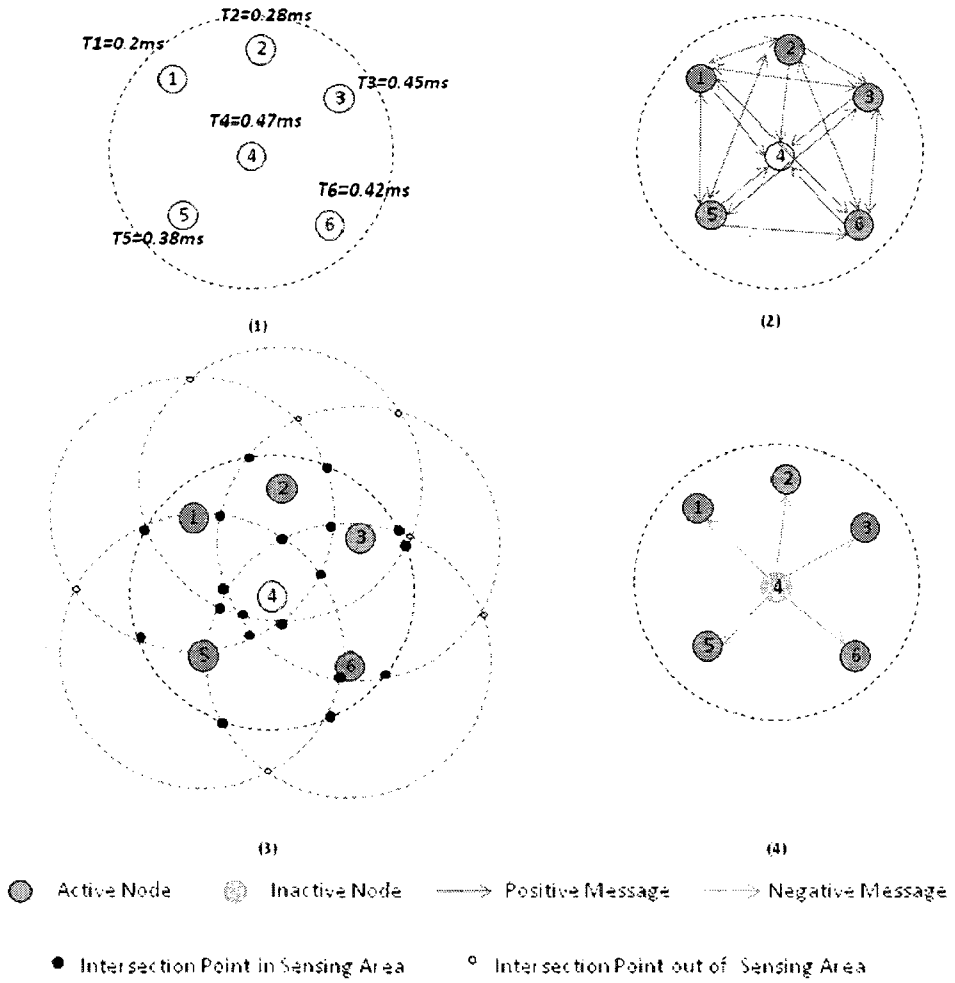


Figure 5. Area Coverage Evaluation

2.1.2 Connected Dominating Set (CDS) Construction

A subnet is called a dominating set if each node in the network either belongs to it or has at least one neighbor belonging to it. Nodes in the dominating set can be called dominants. In a connected (node) dominating set, all dominants are supposed to be connected.

There exist several CDS construction algorithms. Stojmenovic and Seddigh [SS] proposed a neighbor elimination algorithm in 2000. Wu and Li [WL] proposed a coverage algorithm to construct CDS, where considered coverage only points to those provided by two directly connected one-hop neighbors. Dai and Wu [DW] offered a more general coverage algorithm (generalized self-pruning rule), where coverage can be provided by an arbitrary number of connected one-hop neighbors. In [DW], authors proposed an enhanced dominating set by a coverage algorithm when two-hop topological knowledge is available.

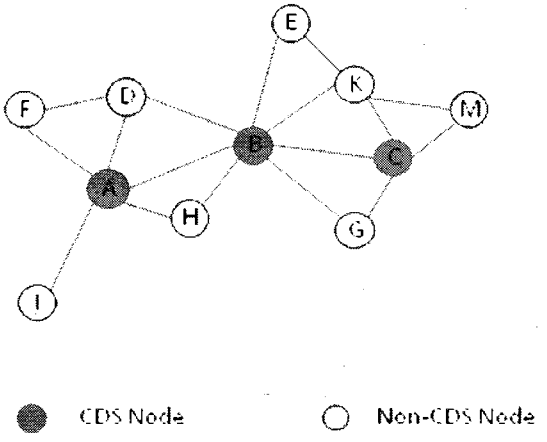


Figure 6. Connected Dominating Set of Network

Generalized Self-Pruning Rule by one-hop neighbors

Dai and Wu [DW] proposed Generalized Self-Pruning Rule by one-hop neighbors in 2003. A node is called an intermediate node if it has at least two one-hop neighbors that are not directly connected. The protocol first checks whether a node is intermediate. Non-intermediate nodes will never be dominant. Then for each intermediate node, they attempt to seek a connected subset of its direct neighbors, which can cover the node. If no such subset is found, this node will become dominant, otherwise, it is non-dominant. In this algorithm, if we say that node A is covered by node set S, it means: 1). any neighbor of node A either belongs to node set S or is a neighbor of at least one node in node set S; 2). all nodes in set S are connected and have higher priorities than node A, where priority

can be determined by node id or something else (in our project, it is determined by a key consisting of rest energy and node id). Then, CDS is a node set such that nodes in this set can cover the rest of the nodes in the network. In Figure 6, nodes A, B, and C make up the connected dominating set of the network. Other nodes are all covered by nodes in this set. For example, node K is covered by nodes B and C (lower node id means higher priority) because: 1) neighborhood of node K includes nodes B, E, C, and M. Nodes E and M are neighbors of nodes B and C, respectively. All neighbors of node K can be reached in one-hop by node set, {B, C}. 2) Nodes B and C are connected, and nodes B and C have higher priorities than node K.

Carle and Simplot-Ryl [CS] proposed an algorithm to build CDS similar to the one described by Generalized Self-Pruning Rule. First, a node checks itself whether to be an intermediate node. Then each intermediate node computes subgraph *HPN* of those Higher Priorities Neighbors. If this subgraph is non-empty and connected (checked by Dijkstra's shortest path algorithm [D]) and there is no neighbor of current node which is not neighbor of any node in *HPN*, the node is "covered" and non-dominant; otherwise, the node will be dominant. The sensors that are determined to be dominant make up CDS of the network. Non-intermediate nodes will not become dominant.

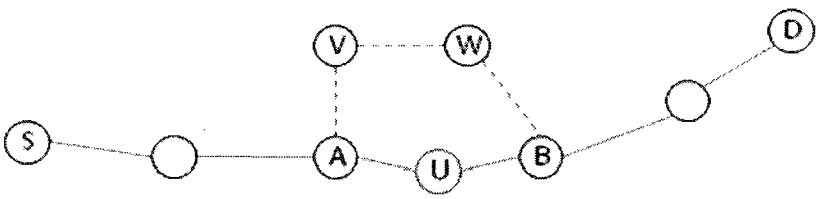


Figure 7. Node Connectivity in CDS

It can be proved that the dominants are connected if the network is connected.

Proof [DW]: Suppose there is a path S...A,U,B....D (shown in Figure 7) connecting nodes S and D in the original network. Suppose node U is covered by neighbors V and W, but is not in the connected dominating set F. Then nodes A and B must be connected to one of nodes V and W. The portion of A, U, B can be replaced by A, V, W, B. Thus there is still a path between nodes S and D in the generated subgraph made up of node set F.

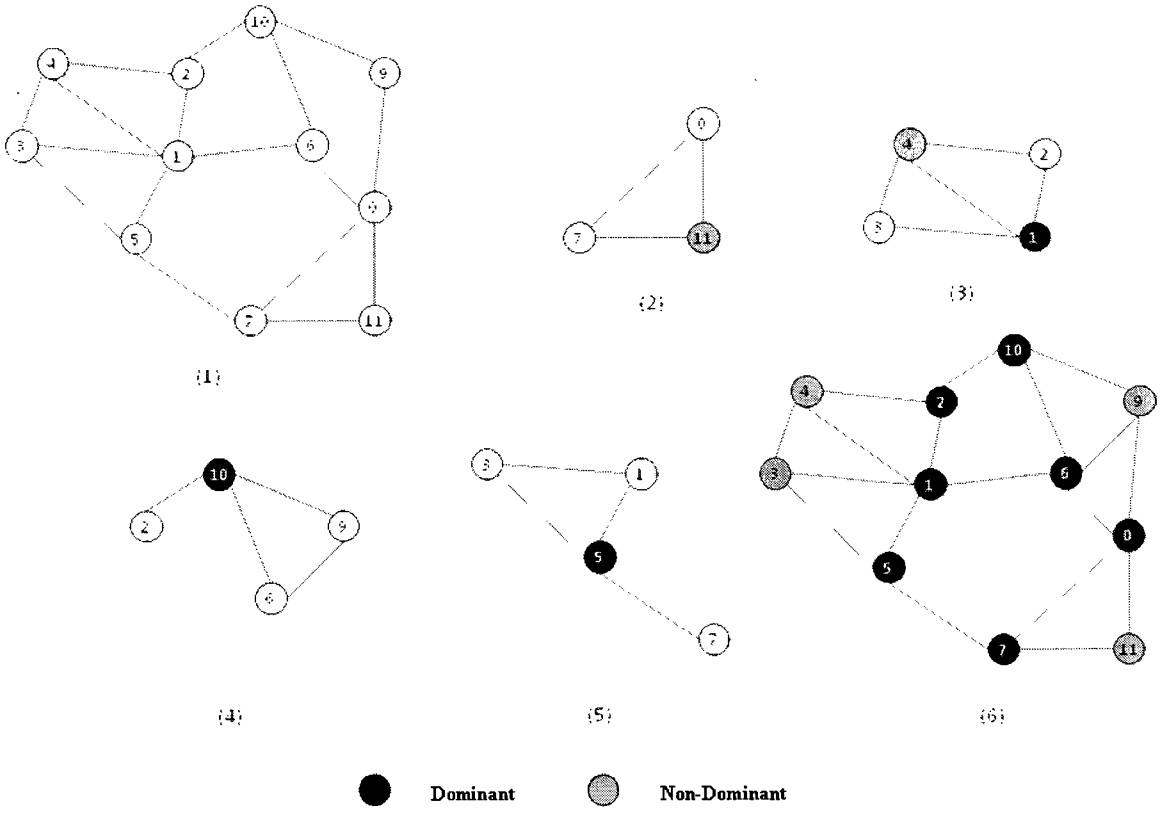


Figure 8. CDS Construction

Example

We assume that there is a network consisting of 11 nodes as shown in Figure 8(1), and every node has a priority according to its id (lower id means higher priority). Every node is supposed to know its neighbors' positions and priorities. One CDS will be built in the network by the algorithm proposed by [CS] as follows:

Node 11 (shown in Figure 8(2)) finds itself to be non-intermediate, thus not dominant. Node 4 is not dominant either, since it is dominated by node 1 as shown in Figure 8(3), and so are nodes 3 and 9. Nodes 0 and 1 decide to be dominant because they have the highest priority in their neighborhood. Some nodes such as node 10 in Figure 8(4) decide to be dominant since although they have neighbors with higher priorities, these neighbors are not connected. In Figure 8(5), node 5 has two higher-priority and connected neighbors, nodes 1 and 3, but none of them is connected to node 7, the low-priority

neighbor of node 5. In this case, node 5 decides to be dominant. Nodes 2 and 6 are similar to node 5. Eventually, the resulting CDS is a graph shown in Figure 8(6).

2.1.3 Local Minimal Spanning Tree (LMST)

Minimal spanning tree is a topology connecting all nodes in the network at the lowest overall cost of edges, where cost can be distance of an edge, bandwidth of a link, or expected power consumption in transmission through a link. MST is an optimal choice to connect all nodes in the network, but in order to construct such a tree, a global protocol must be implemented. To localize the construction process, LMST as a sub-optimal choice is recommended.

Li et al. [LHS] proposed a LMST construction protocol as follows: each node constructs a local MST in its one-hop neighborhood by Prim's algorithm and only keeps those outgoing edges from the node. Then all nodes exchange their computation results with neighbors. The resulting topology is directed. To convert the directed topology to be non-directed, there are two ways introduced in [LHS]: LMST+ and LMST-. In the former, all edges are kept in the final topology, and in the latter, only bi-directional edges are available in the final topology. Since LMST often has loops, Ovalle-Martínez et al. [OSGS] introduced a method to covert LMST to MST by breaking loops.

Example

We assume there is a network as shown in Figure 9(1). Every node in this network knows its neighbors' positions. In LMST construction, each node first computes a local MST in its neighborhood and keeps its outgoing edges. For instance, node 6 computes its local MST as shown in Figure 9(2) and picks the outgoing edges connected to it as shown in Figure 9(3). Then nodes exchange their selection results with neighbors. Figure 9(4) indicates the directed topology in the entire network. The final LMST (no matter LMST+ or LMST-) structure in the entire network would be the one shown in Figure 9(5).

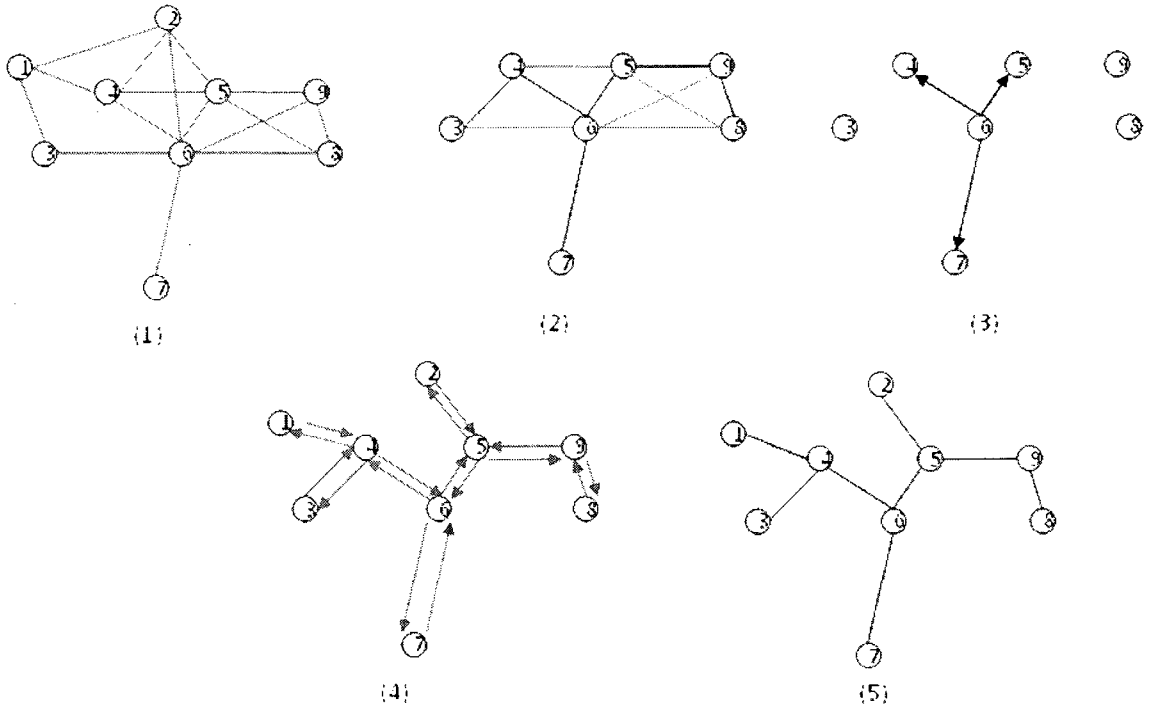


Figure 9. LMST Construction

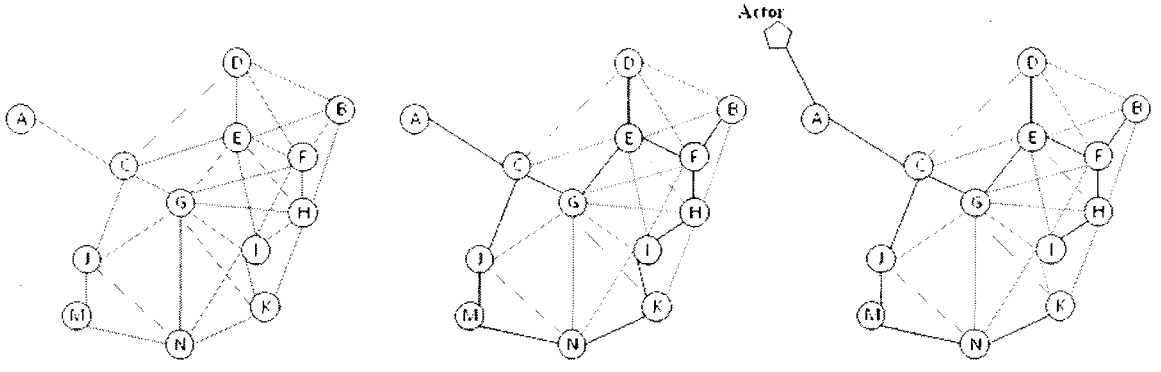


Figure 10. LMST-Based Data Aggregation Tree Construction

2.2 LMST-Based Data Aggregation Tree

Tan, Korpeoglu, and Stojmenovic [TKS] proposed a LMST-based data aggregation protocol for a multi-hop network. Data packets are supposed to be forwarded and aggregated along a LMST-based aggregation tree rooted at actor and spanning all sensors.

First of all, all nodes in the network compute their LMST edges and neighbors as described in Section 2.1.3. The aggregation tree construction is done via flooding starting from the actor. The actor initiates a tree-construction packet to find routes from all nodes towards itself. This packet includes one distance field to estimate the sender's cost of reaching the actor, where the cost of an edge is the power consumption in the edge (called the shortest weighted path method, **SWP**). A sensor receiving a construction message for the first time will update its parent id and distance with the values of the sender, and respond to the parent node (sender) by one notification. And then the sensor creates a new construction message reflecting its situation and sends the message through its LMST edges except the one connected to the parent. If a sensor receives multiple construction messages from different LMST neighbors, an update is necessary if a new route with a smaller cost is detected. After the update the sensor informs the new and old parents of the modification. Then a new construction message is created and sent out again. Otherwise, the sensor just drops the received message.

While the process is completed in the entire network, every sensor in the network should know its parent node to which its data packet will be sent and children nodes from which data packets will be received. In the resulting reporting tree, the path connecting one sensor to the sink is approximately most power-efficient.

Besides the SWP method, the authors also recommended two other methods, first parent path method (FP) and minimum hop path method (MH). In FP, no distance field is used. A node will set its parent as the first LMST neighbor from which the tree-construction packet was received. As for MH, the distance field points to the hop count from the sender to actor. For example, Figure 10(1) indicates one network's original topology. The LMST (according to the LMST- scheme) over this network is shown in Figure 10(2). The aggregation tree generated by the MH method proposed by [TKS] is shown in Figure 10(3).

Once a reporting tree is constructed, data packets will be forwarded along the tree when data aggregation is required. All internal nodes of the tree will implement aggregation before their data forwarding.

2.3 Energy Suboptimal Aggregation Tree

[KEW] studied energy and delay cost in multiple source nodes data aggregation problem. Authors proposed measuring energy consumption and delay by hops. With this assumption, Steiner Minimum Tree (SMT) [CBV] is well known as an energy optimal tree. However, the task of constructing SMT is NP-hard. Then authors proposed three suboptimal solutions: Center at Nearest Source (CNS), Shortest Paths Tree (SPT), and Greedy Incremental Tree (GIT).

Center at Nearest Source (CNS): In this data aggregation scheme, the source which is nearest the sink acts as the aggregation point. All other sources send their data directly to this source which then sends the aggregated information on to the sink.

Shortest Paths Tree (SPT): In this data aggregation scheme, each source sends its information to the sink along the shortest path between the two. Where these paths overlap for different sources, they are combined to form the aggregation tree.

Greedy Incremental Tree (GIT): In this scheme the aggregation tree is built sequentially. At the first step the tree consists of only the shortest path between the sink and the nearest source. At each step after that, the next source closest to the current tree is connected to the tree.

As mentioned, all algorithms in [KEW] adopted hops to count the energy consumption of aggregation tree. In data aggregation with few non-source nodes, the network energy consumption by CNS will consume approximate energy needed by data gathering. And global information will be required in shortest path searching by nodes applying SPT and GIT. Finally, although delay is discussed by [KEW], no delay bound has been taken into consideration in three algorithms.

2.4 Delay-bounded Steiner Minimum Tree

[ZSS] proposed a protocol to build an aggregating structure spanning all sources and actor with approximately minimal energy cost considering the given delay-bound.

Authors proposed evaluating the structure's energy consumption and delay with hop count.

When the energy consumption is measured by hop count, Steiner minimum tree [CBV] is well known as an optimal structure to collect data from all sources for fully-aggregated applications. If delay bound is required for data aggregation, [ZSS] proposed a delay-bounded near-optimal tree (called delay-bounded Steiner tree, DB-SMT) to be aggregation structure. Because single-actor network aggregation can be treated as a reverse process of multicast, [ZSS] turns to bounded shortest multicast algorithm (BSMA) [ZPG] to generate DB-SMT.

BSMA creates a minimum-energy-cost multicast tree with delay bound. It first creates a shortest delay multicast tree by Dijkstra's shortest path algorithm. Then through certain iterations of delay-bounded path switching, the energy cost of the multicast tree is gradually reduced until it is minimal while resulting trees do not violate the delay bound. In each path switching, the in-tree link with most energy cost is replaced by a non-tree link with least energy cost, which can maintain tree connectivity and does not violate tree's delay bound.

In cases without non-source nodes, every sensor implements one and only one report transmission. Therefore, the overall transmission number is equal to the number of nodes. Since hop count is treated as the metric of energy consumption in [ZSS], the overall network energy consumption cannot be controlled by the algorithm proposed by [ZSS]. Besides, [ZSS] constructs DB-SMT by BSMA that requires significant time and computational complexity in large scale networks since it implements Dijkstra's shortest path algorithm by O(nlog(n)) iterations in each tree construction.

2.5 Delay-Bounded Minimum Degree Spanning Tree (DB-MDST)

[KKK] focuses on network energy balance while respecting delay requirement. Like [ZSS], authors proposed measuring network energy consumption and delay by hop count. The algorithm proposed by [KKK] builds an aggregation tree with minimum degree and a height not beyond given limit, where the maximum node degree is considered as tree degree.

[BAS] considered the minimum degree spanning tree (MDST) to be an optimal aggregation structure in fully-aggregated applications, when node energy consumption balance is concerned. When full aggregation is implemented, the difference of energy consumption for all nodes merely comes from the packet reception cost, which is determined by the number of children. The node with more children consumes more energy in one collecting round. Therefore, MDST is considered as the optimal structure for energy balance, where the number of children for each node is reduced as much as possible.

Delay-bounded minimum degree spanning tree (DB-MDST) is generated by an approximated MDST algorithm [KKK]. Through iterations of edge-replacing, a random tree is modified to be an approximated MDST with a height not larger than given limit. In each edge-replacing, one link from the maximum degree node is removed and one link connecting to minimum degree node, which can maintain tree connectivity and does not violate tree's delay bound, is added.

Locating the maximum degree node and adding an edge as well as network connectivity check in each edge-replacing require global knowledge. Therefore, to construct a DB-MDST in large scale networks by the algorithm proposed by [KKK], vast communication overheads are required.

2.6 Multi-state Data Aggregation Framework

Melodia, Pompili, Gungor, and Akyildiz [MPGA] discussed the construction of data aggregation trees using a metric different from hop count, by using energy model, $u(d)=d^{\alpha}+c$, for the cost to communicate at distance d. It is assumed that sensor nodes initially know their own positions, positions of their neighbors, and positions of actors. It is also assumed that there are no void areas. Thus, there exist advance from given sensor to any actor, if direct transmission is not possible. They formulate integer linear program to construct a data aggregation tree to minimize overall energy spent for reporting. More precisely, a set of actors is first selected, and then a minimum energy tree toward these actors that meet required event reliability is constructed. Basically, several trees are created inside a geocasting region, and paths/trees are then constructed toward each actor, including those outside that region. Event reliability is the ratio of unexpired packets over

all received packets (data and notifications). The delay of a packet can be measured as the largest delay in a reporting aggregated tree (each sensor must wait for data from its 'children' sensors before aggregating and forwarding the result to its parent sensor).

Since integer linear programming based solutions are centralized algorithms with high time complexities, they are highly impractical. Authors [MPGA] then considered some distributed solutions (named as multi-state framework) using some localized heuristics and state based behavior of sensors. Each sensor alternates among four different states, namely *idle*, *start-up*, *speed-up*, and *aggregation states*. The main objective of state transitions is to reduce the number of hops when the reliability requirement is violated and to save energy when the reliability requirement is met. In the startup state, each sensor node selects its next hop based on two-hop rule. Let S be a sensor, N be one of its neighboring nodes, and A be one of actors. Sensor S selects neighbor N for which $u(|SN|)+u(|NA|)$ is minimized, over all neighbors and over all actors. Consider a candidate neighbor N so that $y=|SN|$ and $x=|NA|$, and candidate neighbor B so that $y+\Delta=|SB|$ and $x+\nabla=|BA|$. It can be shown that the difference between the two neighbors in considered criterion is $u(|SN|)+u(|NA|)-u(|SB|)-u(|BA|) \approx \alpha y^{\alpha-1}\Delta + \alpha x^{\alpha-1}\nabla$. When the sensor is far from the actor, $y<<x$ and the difference is $\approx \alpha x^{\alpha-1}\nabla$. This means that the selection is biased toward the neighbor that is closest to A, that is, the selection is close to the one made in greedy routing. This is not really a power saving selection. Meanwhile, for sensors located close to the actor, their selections can be energy saving ones. Obviously, start-up protocol will unbalance energy distribution in the entire network.

The objective of MS framework is to converge to a solution with reliability close to the event reliability threshold and minimal energy consumption. Sensor nodes associated with an actor base their state transitions on the reliability observed by the actor, which is broadcast at the end of each decision interval. The objective of the collaborative operation of nodes in the speedup state is to minimize the number of hops between sources and actors. This is achieved by applying the greedy routing scheme by Finn (cf. [F]), which minimizes hop count. If all nodes apply speedup procedure, too much advance can be made with excessive cost in energy. Authors [MPGA] then discussed aggregation state where nodes select nearest neighbors with advance toward an actor as

the next hop although details of this protocol remain unclear. However, the state transition scheme is not efficient. Its feedback proposed all nodes adopt common routing protocol to increase reliability or save energy. However, such proposals may not be suitable at individual nodes and lead to more state transitions in some cases. For example, if a group of nodes can maintain reliable ratios of 100% at any state, they do not have to switch to speed-up state to increase reliable ratio. On the contrary, for a group of nodes located far from the actor, their data are destined to be expired if all of them work in aggregation state. Such failures may lead to more requirements of transition towards speed-up state. Although authors adopted the setting of switching probability to make up the drawback, obviously, this random scheme cannot guarantee the sensor to make correct decision. Moreover, nodes will stay at start-up state, if the network reporting reliability is in acceptable range. In this scenario, aggregation state procedure, the most energy efficient choice, will not be considered even if it can achieve similar reliability. Hence, to optimize state transition scheme, the points that trigger transitions have to be configured precisely for different data aggregations.

MS framework is the only known localized solution to delay-bounded data aggregation problem, where power efficiency is concerned. Although none of three routing protocols used in the framework is delay sensitive, the framework relies on a state switching scheme to control routing protocol selection, thereby making data aggregation respect the given delay limit. For energy model selection, the authors adopt the first-order radio model to count energy consumption in transmissions. In terms of delay measurement, [MPGA] uses timestamp included in data report to count data experienced delay. As our analysis in the problem statement showed, in low rate networks the assumption of delay proportional to hop counts is reasonable. Actually, [MPGA] also accepts this assumption. In their simulation part, we can find they use hop counts to evaluate protocols' delay performance. Based on above reasons, we consider MS framework as the best competitor to our proposed framework, and it will be simulated and compared with our solutions in experiments.

Chapter 3 Desired Hop Progress (DHP) Data Aggregation Protocol

3.1 Multi-Step Solution and One-step Solution

Our DHP protocol is to first build an energy optimal aggregation tree in the network. Then we adjust the tree based on delay requirements at nodes so that it does not violate the given delay limit. Typically, one node following delay-bounded routing protocol will seek a new route with shorter delay when the known one cannot meet its delay requirement. In this situation, the node has two options in selecting new parent: attempt to make up for the delay gap at the current node (called *one-step solution*) or distribute the task to the current node and the rest nodes on its path to actor evenly (called *multi-step solution*). One node following one-step solution will try to seek a parent node which has the maximum progress in its neighborhood or has the minimum progress among all neighbors satisfying delay requirement. On the other hand, in multi-step solution, delay lag problem is proposed to be solved in a series of transmissions from the current node. The whole delay gap is divided into certain pieces and only one piece has to be caught by the current node.

Obviously, if one-step solution is performed, data reports will be faster forwarded to nodes whose known route can meet delay requirement well, which may lead to higher delivery reliability. However, this scheme is easy to cause energy imbalance. It is because aggregation tree adjustment is done in a bottom-up fashion, i.e., from leaf nodes towards the root (actor). Lower level nodes will start their parent fixup earlier. Normally, these nodes will detect delay lag problem and then attempt to solve it right away when they apply one-step solution. Consequently, when higher level nodes start their route evaluation, they may find their known routes can still meet their delay requirements due to lower level nodes' adjustment. Therefore, following one-step solution, the higher level nodes will have lower chance to change parents; moreover, even if parent change are needed, the expected progresses should be less than those located in lower levels. In the resulting aggregation tree, nodes located at lower levels of the initial tree often connect to parents by long edges and others tend to choose short edges. In multi-step solution, nodes are proposed to solve delay lag problem partially. Then, more nodes will be involved in searching for new parents and expected progresses at nodes will be quite even.

Consequently, we can greatly reduce the unfairness due to the sequence of decision making at nodes. Apparently, following multi-step solution, energy consumption at nodes will be better balanced. Furthermore, the solution may lead to the less overall energy consumption in certain cases. For instance, an energy optimal aggregation tree (Figure 11(1)) has been built in a network consisting of one actor (A) and four sensors (B, C, D, E) that array in a line and have equal distance (r) between any two neighboring nodes. All nodes in this network can contact each other. Every node knows its own and its neighbors' delays to actor along the tree. Now one data aggregation with delay bound=2 hops is expected in this network. We suppose that nodes following one-step solution will transmit reports to the closest neighbors meeting delay requirement, while those following multi-step solution just transmit reports to parents with one more hop progress than old ones. Then energy consumed at node N is EO_N by one-step solution and EM_N by multi-step solution. The difference between overall energy consumption by both solutions is:

$$
\begin{aligned}
E_{one-step} - E_{multi-step} &= (EO_E + EO_D + EO_C + EO_B) - (EM_E + EM_D + EM_C + EM_B) \\
&= [(3r)^\alpha + (2r)^\alpha + r^\alpha + r^\alpha] - [(2r)^\alpha + (2r)^\alpha + (2r)^\alpha + r^\alpha] \\
&= (3^\alpha + 1 - 2 \cdot 2^\alpha) r^\alpha \\
\because\ & 3^\alpha > 2 \cdot 2^\alpha (2 \le \alpha \le 4) \\
\therefore\ & E_{one-step} - E_{multi-step} > 0
\end{aligned}
$$

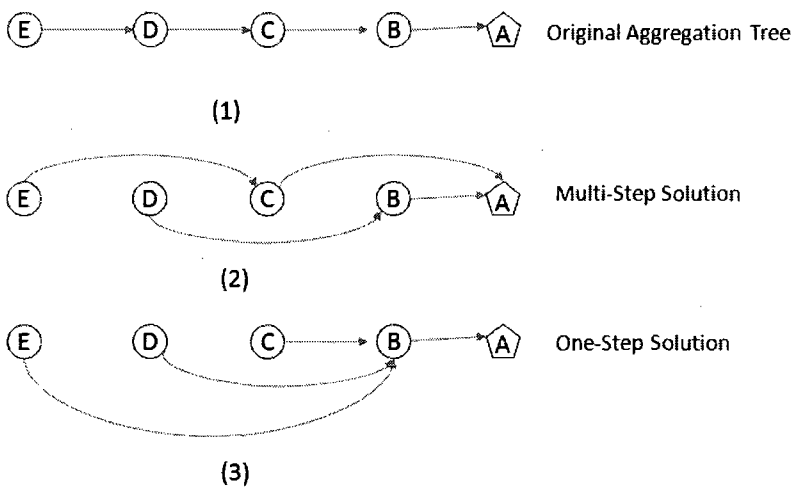


Figure 11. One-Step Solution and Multi-Step Solution

Considering energy balance and conversation, we adopt multi-step solution for nodes' parent switching in our DHP protocol. In other words, a node facing delay lag does not have to seek a parent with the maximum progress or a progress equal to the difference but one having an appropriate progress that we call desired hop progress.

3.2 DHP Data Aggregation Protocol

Desired-hop-progress (DHP) protocol is a delay-bounded power efficient data aggregation protocol. Aggregation tree resulting from this protocol will be significantly energy efficient (in energy balancing and saving), considering given delay limit.

We assume that every sensor in the network knows its neighbors' positions through previous 'hello' message exchanges. Therefore, according to positions, each node can locally compute its LMST neighbor candidate list, *LNCL*, where transmission power is treated as the cost of edge. If data collection is required, actor will trigger a LMST-based request flooding to inform all nodes to report, and a shortest path tree over the LMST is constructed like FP [TKS].

Definition 1: *UDG-distance* between two nodes is the minimum hop count between nodes in the original Unit Disk Graph (UDG).

Definition 2: *LMST-distance* between two nodes is the minimal hop count between two nodes along LMST.

For example, the UDG of a sensor-actor network and all edges' costs are shown in Figure 12(1). All nodes' UDG-distance and LMST-distance to actor are shown in Figure 12(2). If the request message includes one field for counting its experienced hop number, each sensor can estimate its LMST-distance to actor through request broadcasting.

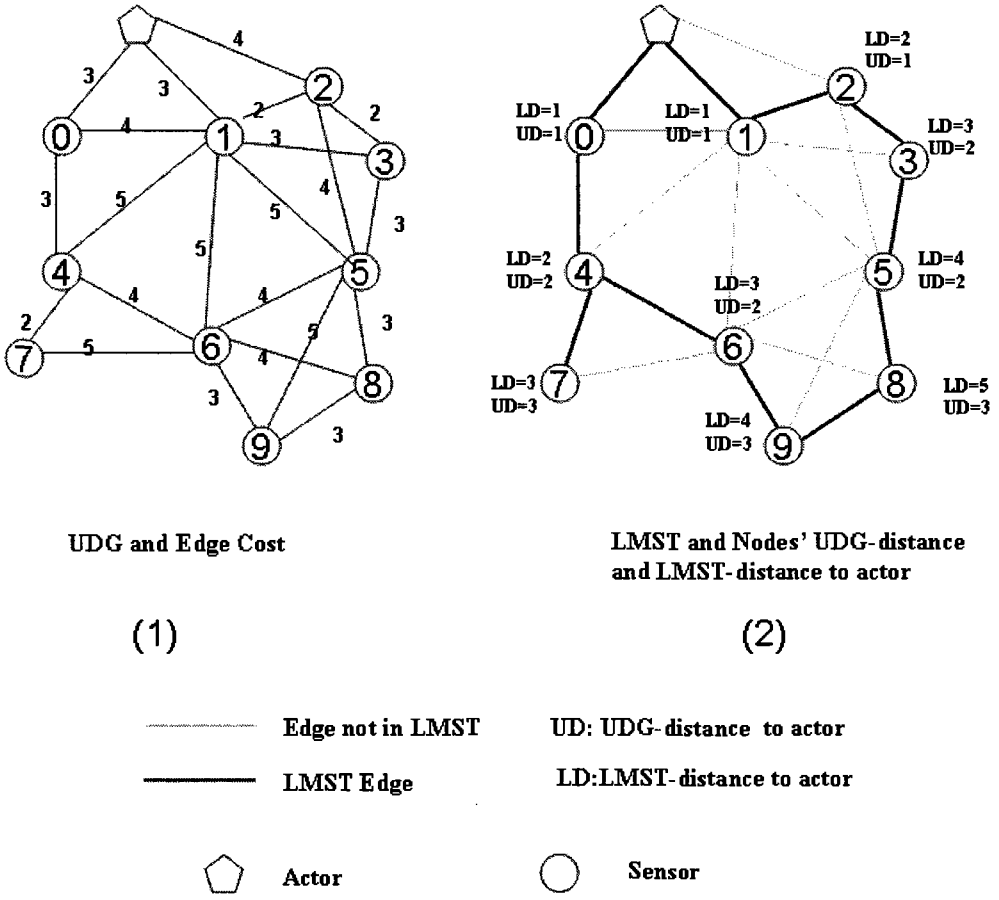


Figure 12. UDG-distance and LMST-distance

Definition 3. Suppose that a sensor A is k hops (UDG-distance) away from the sink, and the current delay limit is n. If a report is received with already experienced delay m then this report is expected when $m+k <= n$ and unexpected otherwise. In data aggregation, sensor will wait and aggregate all expected reports, but neglect those unexpected reports.

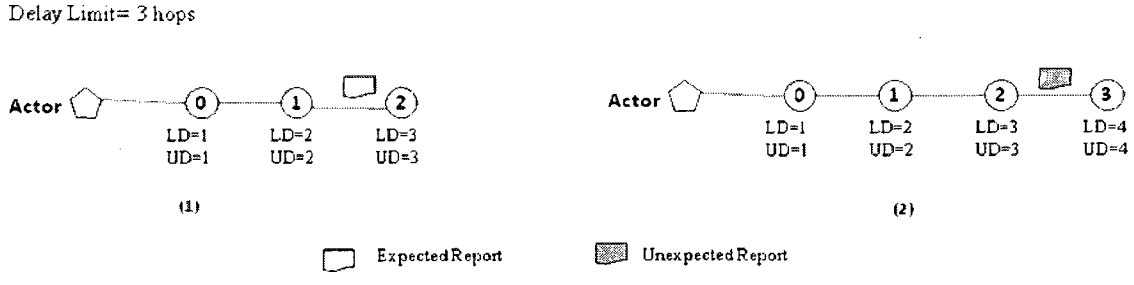


Figure 13. Expected Reports and Unexpected Reports

For instance, two data aggregations with delay limit= 3 hops are respectively implemented in networks shown in Figure 13(1) and (2). Both of them are applied from the rightmost node towards actor. In Figure 13(1), node 1 considers the report from node 1 as expected report, since it is 2 hops away from actor and can receive the report in the 1st reporting round. However, in Figure 13(2), the report from node 3 will be treated as unexpected report by node 2 as node 2 finds the sum of its known shortest delay to actor UD (3 hops) and the report's experienced delay (1 hop) is beyond the given delay limit (3 hops).

Definition 4: *desired hop progress* for one sensor is the ceiling of the sensor's LMST-distance to actor over its aggregated report lifetime (measured by hops):

$$DHP = \left\lceil \frac{LD}{DL - MED} \right\rceil,$$

where *DHP* denotes desired hop progress, *LD* denotes LMST-distance to actor, *DL* denotes Delay Limit, and *MED* denotes the Most Experienced Delay of all expected reports when reaching sensor (*MED*=0, if no expected report).

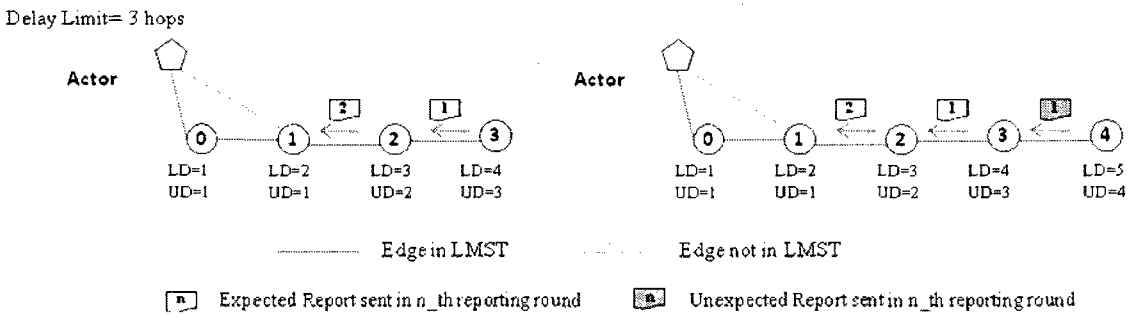


Figure 14. Desired-Hop-Progress Computation

Figure 14 shows two data aggregations with delay limit= 3 hops are implemented in both networks. In the left one, node 1 will send its report in the 3rd reporting round since it has to wait for expected report from its child, node 2. Thereby, its desired hop progress should be $\left\lceil \frac{LD}{DL - MED} \right\rceil = \left\lceil \frac{2}{3-2} \right\rceil = 2$ hops. In the right network, report from node 4 will be considered as unexpected reports by node 3. Hence, the desired hop progress at node 3

is $\left\lceil \frac{LD}{DL-MED} \right\rceil = \left\lceil \frac{4}{3-0} \right\rceil = 2$ hops. And then node 1 finds the Most Experienced Delay among its expected reports is still 2 hops and its desired hop progress should be $\left\lceil \frac{LD}{DL-MED} \right\rceil = \left\lceil \frac{2}{3-2} \right\rceil = 2$ hops too.

Due to the use of aggregation, the flooding (initial) tree is the aggregation structure with approximately minimal energy cost when no delay bound is considered. However, when data aggregation is subjected to given delay limit, the paths along LMST may not meet the delay requirement at sensors. Clearly, in such scenarios, sensors must seek shortcuts in the LMST to meet the delay requirements. DHP protocol will turn to desired hop progress computed at each node for shortcut seeking.

The process of aggregation request flooding and aggregation tree construction in DHP protocol will progress as follows:

First, actor initiates an aggregation request message as required. The message includes request id, delay limit, node *id*, *UD* (UDG-distance to actor), and *LD* (LMST-distance to actor) as well as LNCL. The request message is sent at maximum transmission power.

Each sensor receiving a request message updates the sender's request *id*, *UD* and *LD* in its neighbor table and updates its own *UD* (=*sender's UD+1*) if the request id is new to the receiver or the sender's *UD* less than all known neighbors' *UD*s. In addition, if the packet sender is sensor's LMST neighbor (according to LMST- scheme) and the received request id is new to sensor, sensor will update its request id, delay limit, and *LD* (=*sender's LD+1*) according to the received message. Meanwhile, it sets a flag, named construction flag, on (initially off), denoting that the sensor is in aggregation tree construction state. Once the sensor updates its request id, it will generate a new request message, and sends it to all its neighbors. The process is repeated from one sensor to another, until every node in the entire network completes its request message broadcasting. When request flooding completes in the entire network, one shortest path tree *T* over the LMST will be constructed from the actor to all sensors, and each sensor knows all neighbors' and its own *UD*s and *LD*s along *T*.

If it has known all neighbors' *LD*s, one sensor will pick up two kinds of candidates: parent candidates and child candidates, in its neighborhood, where the former's *LD* is smaller than sensor and the latter's *LD* is larger than sensor.

If sensor *S* has an empty child candidate set, which means it will never become others' parent, it starts aggregation tree parent (afterwards for short parent) selection right away. The sensor computes its desired hop progress, where *MED* is 0. Among all parent candidates (must be non-empty, as every non-actor node has parent node in LMST), the one having hop progress closest to the desired hop progress is chosen to be the parent node. If two parent candidates have same hop progress, the one having shorter Euclidean distance is chosen. If two parent candidates have different hop progresses, but two progresses have same difference to the desired hop progress, we prefer the one with more hop progress. Once S decides its parent node, it will declare this decision at maximum transmission power. This decision message includes node id, parent node id, as well as the evaluated node report delay at parent (for short evaluated report delay, which is equal to *MED+1*). If aggregation reliable ratio has to be computed, parent declaration of *S* should include the number of source nodes related to its aggregated report and the total number of nodes under aggregation tree rooted at node *S*.

If one sensor (*S*)'s child candidate set is not empty, it will wait for all child candidates' parent declaration. For each received decision message from child candidate, *S* will delete the sender from child candidate list, and add it to child list if S is its selected parent. *S* considers the evaluated report delay included in the declaration as experienced delay of reports from this child in data aggregation. According to this value, *S* evaluates whether data reports from this child will be expected and then marks this child to be expected or not (let us call the children sending expected report as expected children, and others as unexpected children). If the child is considered as expected child and its evaluated report delay is larger than *MED*, *S* will update *MED* with this delay. When child candidate list becomes empty, *S* starts its parent selection and then broadcasts the selection result as last paragraph, where *MED* would be a non-zero value if the sensor has expected children.

If one sensor has decided its parent and broadcasted the decision, it will switch its construction flag to off and clear off the parent candidate list.

Example

We assume a data aggregation by DHP protocol will be performed over the network shown in Figure 15. All nodes in the network finished their LMST computation before data collecting starts. When one data collecting is required, request message is forwarded along one LMST connecting all nodes in the network like Figure 16. Each node's LMST-distance and UDG-distance to actor is also indicated in the figure.

When delay limit for the data collecting is over 7 hops, LMST can satisfy delay requirement. Then, desired hop progress computed at each node is one hop. Consequently, the LMST becomes the final aggregation tree, same as the result of FP [TKS].

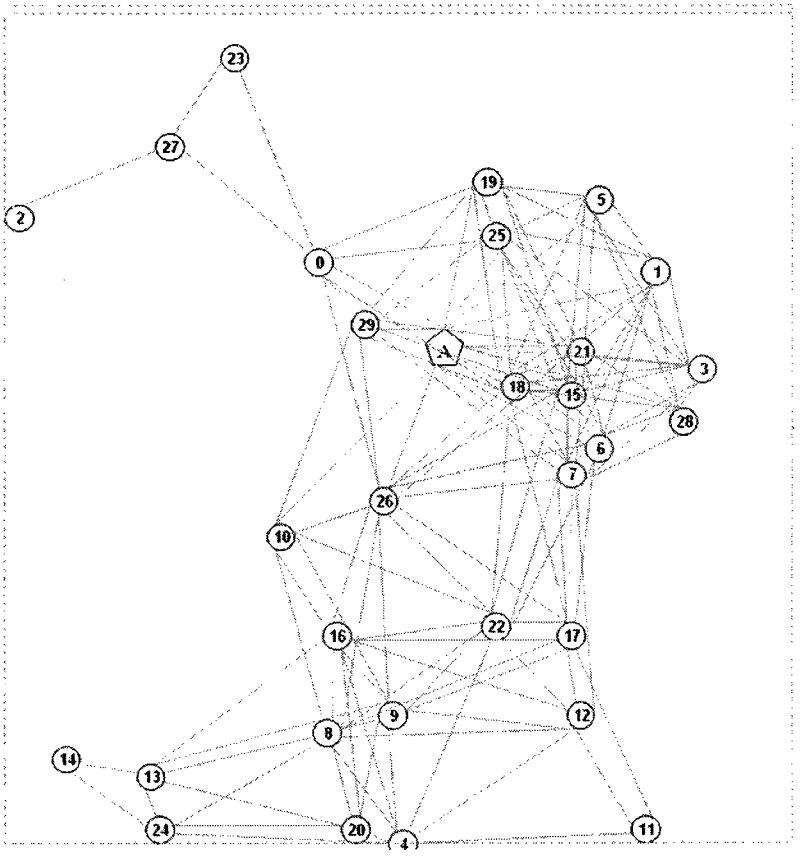


Figure 15. Network UDG

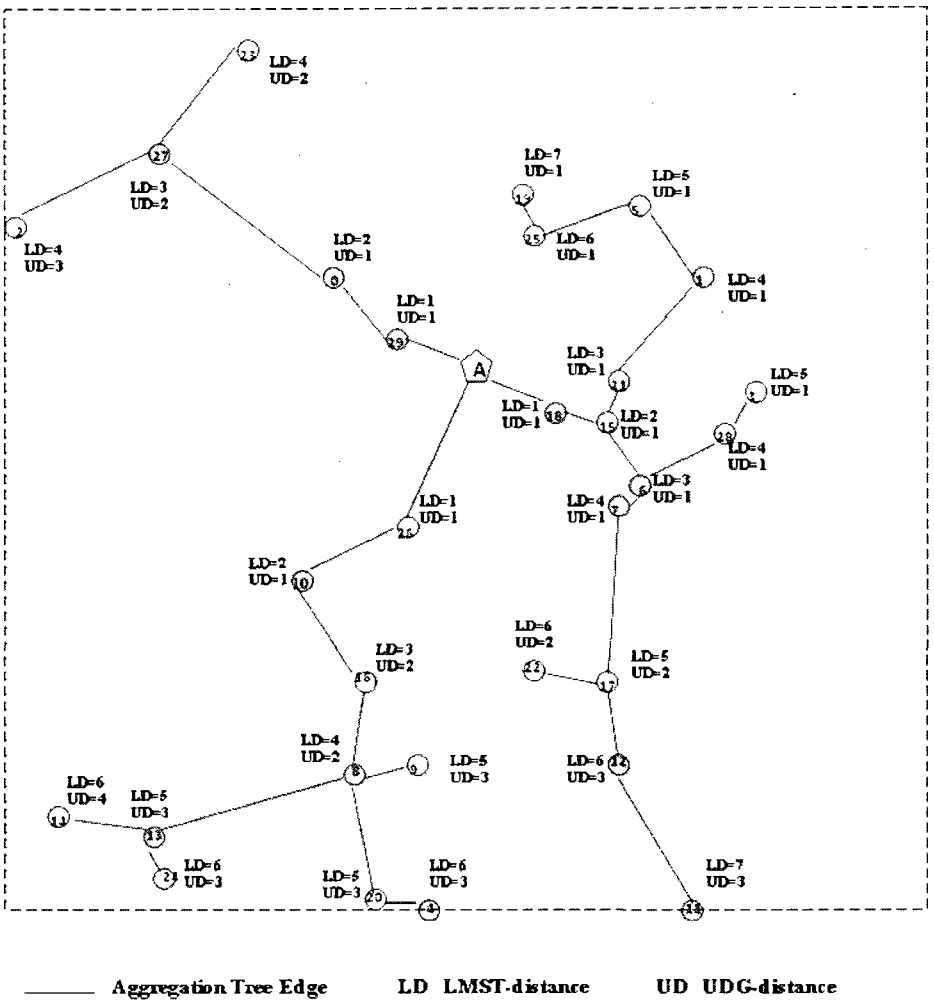


Figure 16. Initial Tree, LMST-distance and UDG-distance to the Actor

When delay limit is less than 7 hops, certain nodes in the network will seek new parents other than the ones in LMST as they cannot meet delay requirement. And desired hop progress at nodes will be various. Now we suppose delay limit for the data aggregation is 4 hops. DHP protocol will generate an aggregation tree like Figure 17. For example, nodes in the range surrounded by grey dashed lines will do their aggregation tree construction as follows: when they have received all neighbors' request messages,

node 14 and 24 are aware that no neighbor has LD less than them. Then both nodes compute their desired hop progress *DHP* and get the same result: $\left\lceil \frac{6}{4-0} \right\rceil = 2$ hops. According to the result, node 24 selects node 8 as parent, while node 14 selects node 13 (1 hop progress) as parent since it cannot access neighbor having more progress. When node 13 receives all child candidates' (nodes 14 and 24) parent decisions, it is aware that node 14 is expected child and its *DHP* should be $\left\lceil \frac{5}{4-1} \right\rceil = 2$ hops. Among all parent candidates, node 16 is the best parent choice. At the same time, node 4 receives parent declaration from node 11, its only child candidate; it computes *DHP* and gets a result, 2 hops. Then node 8 is chosen to be parent. Like node 4, when nodes 9 and 20 receive all child candidates' parent decisions, they find none of neighbors has chosen them to be parents. *DHP* computed at both nodes is 2 hops. Then both of them determine to attach to aggregation tree through node 16. After them, node 8 finds, among all child candidates, only nodes 4 and 24 select it as parent. Then it considers its *DHP* should be $\left\lceil \frac{4}{4-1} \right\rceil = 2$ hops and node 18 is the only parent candidate meeting such hop progress. Following node 8 is node 16 to start parent selection. Among all expected children, report from node 13 experiences the longest delay of 2 hops. Thus, its *DHP* is $\left\lceil \frac{4}{4-2} \right\rceil = 2$ and node 26 is chosen to be parent. When it receives node 16's parent decision, node 18 finds only node 8 selected itself as parent and expected report from the only child will experience 2 hops. As the delay is acceptable, LMST parent, node 26, is chosen to be parent. Finally, node 26 makes its decision. The only candidate, actor, becomes its parent.

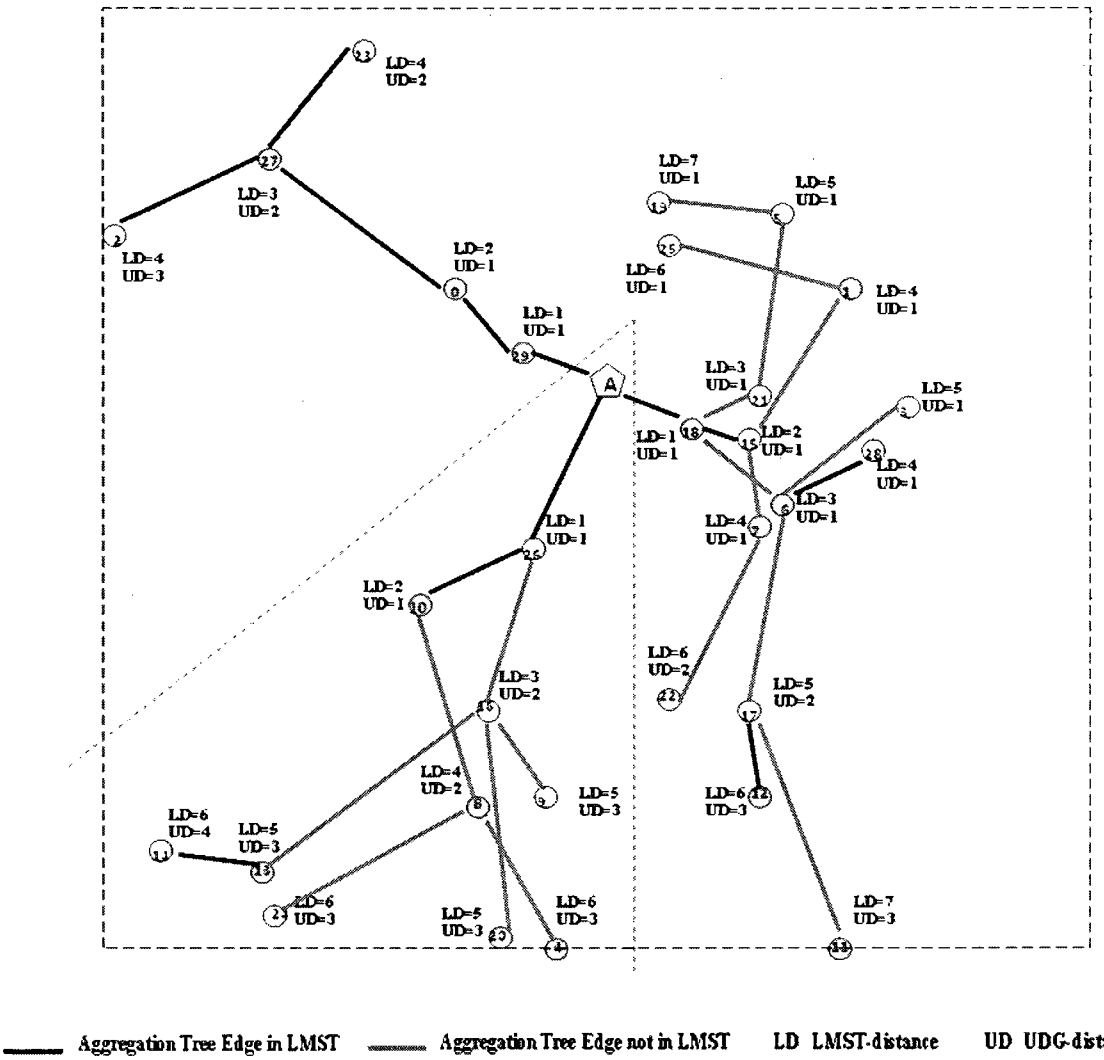


Figure 17. DHP Aggregation Tree with Delay Limit=4 hops

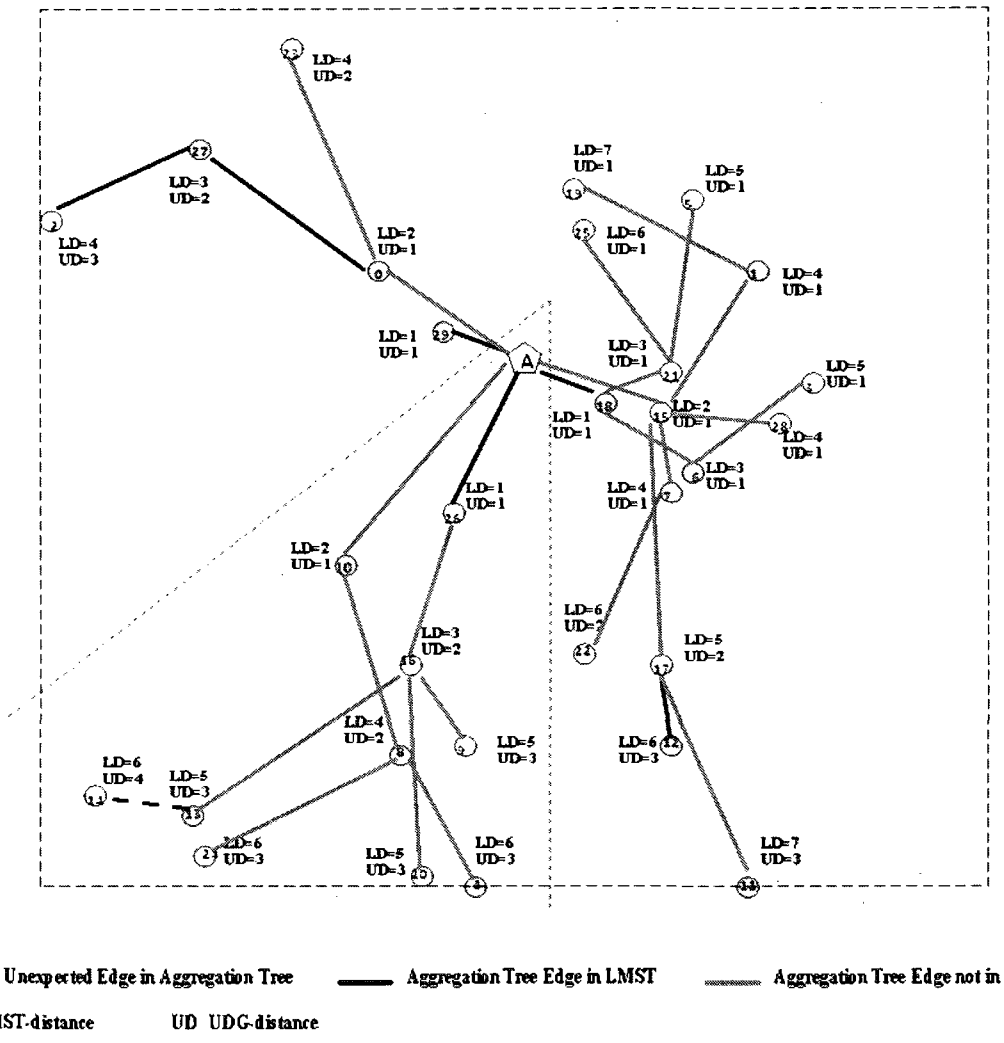


Figure 18. DHP Aggregation Tree with Delay Limit=3 hops

If the given delay limit is small enough like 3 hops, some nodes in the network may fail to find paths to actor meeting delay requirement. In this case, unexpected edge (edge between parent and unexpected child) will appear in aggregation tree generated by DHP protocol. Figure 18 shows aggregation tree generated by DHP protocol when delay limit is 3 hops. Similarly, we focus our discussion on nodes in the same area. First of all, nodes 14 and 24 decide their parents and send out decisions, where nodes 13 and 8 are respective parents. But node 13 considers node 14 to be unexpected child. And it computes its *DHP* (=2 hops) and selects node 16 to be parent. On the other hand, node 4 starts its parent selection. The computed *DHP* is two hops and node 8 is the chosen parent. After nodes 4 and 13, nodes 9 and 20 compute their *DHP* (=2 hops) and choose

node 16 as parent. When node 8 receives all child candidates' parent decisions, it finds reports from all expected children will experience at most one hop at it, then its *DHP* is set to $\left\lceil \frac{4}{3-1} \right\rceil = 2$ hops and node 10 is the best parent choice. Following it, node 16 also believes reports from all expected children only experience one hop. Thus, its *DHP* is $\left\lceil \frac{3}{3-1} \right\rceil = 2$ hops and its parent should be node 26. Then, node 18 makes parent selection. Because its only expected child will send report with a delay of 2 hops, *DHP* at node 18 is $\left\lceil \frac{2}{3-2} \right\rceil = 2$ hops. Thereby, the node selects actor as parent. Finally, node 26 determines actor to be its parent.

Data reporting along the constructed aggregation tree will start at leaf nodes and internal nodes without expected children. They just send out reports including their own data. And other nodes in the tree will wait until they have received all expected reports, and then apply aggregation and send aggregated report to parents. All unexpected reports will be abandoned by receivers.

Example

Now we suppose delays resulted from requests and parent declaration messages are negligible, compared to those incurred in report transmissions. As multi-reception is allowed by all nodes, in each round, sensors ready to report will send out their reports simultaneously.

Data aggregation along aggregation tree with delay limit=4 hops (shown in Figure 17) is indicated in Figure 19. Let's concentrate on the same area of nodes. In the first round, leaf nodes (nodes 4, 9, 14, 20, and 24) send their reports (only including their own sampling data) to respective parents. In the second round, nodes 8 and 13 apply aggregation and send out aggregated reports. In the third round, nodes 16 and 18 perform reporting. Finally, node 26 aggregates all sending data in one report and transmits it to actor. Delay over this reporting path is 4 hops, exactly the delay limit. Data aggregation along aggregation tree with delay limit=7 hops will be implemented similarly.

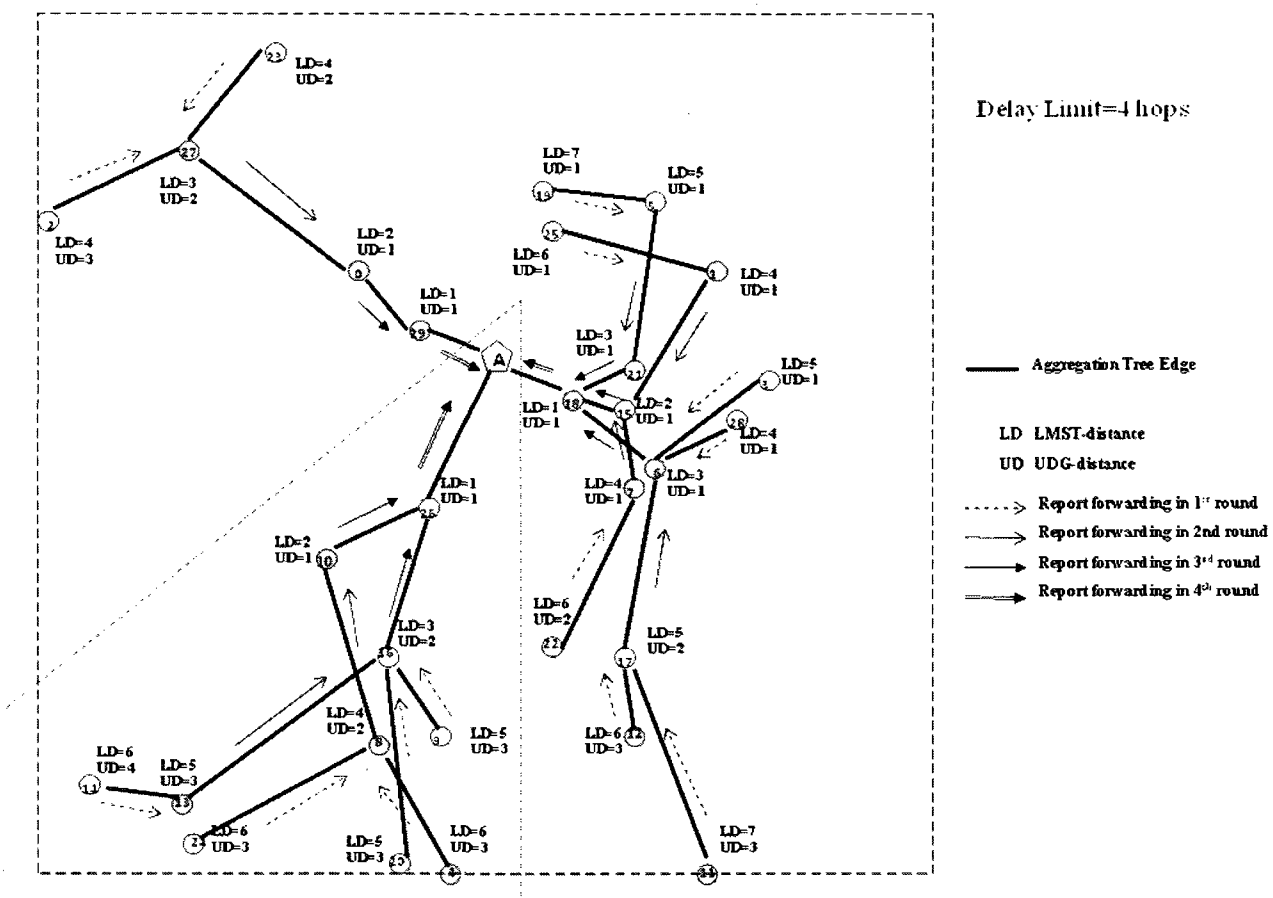


Figure 19. Data Aggregation (Delay Limit=4 hops) by DHP

Data aggregation along aggregation tree with delay limit=3 hops (shown in Figure 18) is indicated in Figure 20. As mentioned previously, internal nodes in aggregation tree do not wait for unexpected children's data packets. Therefore, data aggregation over nodes in the area surrounded by dashed lines will progress in the following way: In the first round, leaf nodes (nodes 4, 9, 14, 20, and 24) send out reports only involving their own data. Meanwhile, node 13 (internal node without expected child) also sends out its report, which neglects the unexpected report from node 14. In the second round, node 8 and 16 implement aggregation and send their aggregated reports. In the third round, nodes 10 and 26 send their aggregated reports to actor. Data in both reports experience a delay of 3 hops, which matches the delay limit exactly. However, data from node 14 is abandoned during this collection.

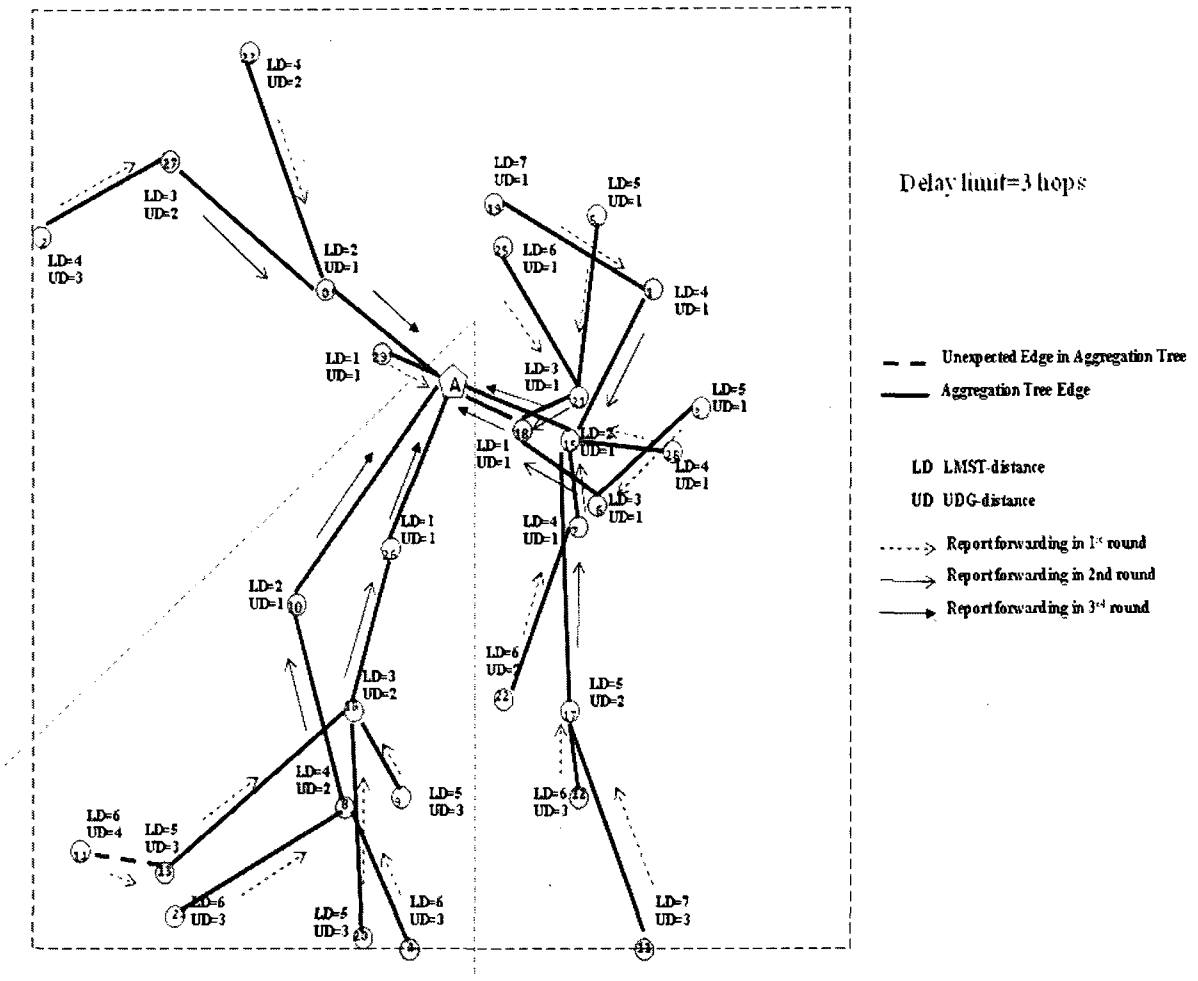


Figure 20. Data Aggregation (Delay Limit=3 hops) by DHP

Data aggregations following DHP protocol will use the same aggregation tree if same delay limits are required. Thus, one data aggregation will reuse the aggregation tree used by the previous one, if given delay limits for both tasks are same. In one data aggregation without aggregation tree construction, sensor receiving a new request message from a LMST neighbor just retransmits the message to other neighbors, and then starts to report or waits for expected reports. All reports will be forwarded and aggregated along the old aggregated tree.

3.3 Algorithm Pseudo Code

Suppose the network consists of sensor set V and actor A, and

N (v): all neighbors of node v.

Num (N): the number of nodes in node set N

UD: UDG-distance to actor

LD: LMST-distance to actor

DL: Delay Limit

ED: Evaluated Delay of report

MED: Most Experienced Delay of all expected reports

PC: parent candidate list of node

CC: child candidate list of node

ECL: expected children list

LNCL: sensor's direct neighbors in its neighborhood-wide MST connecting all neighboring dominants.

The task of DHP is to do a LMST-based data collecting request broadcasting in the entire network; build an aggregation tree to connect actor and all sensors, subject to given delay limit; and implement data aggregation along the built aggregation tree.

Algorithm 1. DHP, Actor A

1. Data collecting is required
2. A creates request packet p (request id, node id, A's UD, LD, DL, and LNCL)
3. A sends p to N(A)
4. ***for*** each report received by A
5. A saves report
6. ***end for***

Algorithm 1. DHP (Aggregation tree Construction Part), Sensor S

1. ***for*** each request packet p received by S
2. S updates sender's request id, UD, and LD in N(S)
3. ***if*** (S request id<sender request id) or (sender's UD+1<S's UD)
4. S's UD=sender's UD+1

5. *if* (S request id<sender request id) and (sender and S are LMST neighbors)
6. *if* new DL=old DL
7. S updates request id
8. *else*
9. S sets Aggregation-Tree-Construction-Flag ON
10. S updates request id, LD (=sender's+1), DL
11. *end if*
12. S creates a request packet p (request id, node id, S's UD and LD, DL, and LNCL)
13. S sends p to N(S)
14. *end if*
15. *if* Aggregation-Tree-Construction Flag ON and S has received all requests from N(v)
16. **CandidateClassify()**
17. *if* CC is empty
18. **ParentSelection()**
19. *end if*
20. *end if*
21. *end for*
22. *for* each parent declaration received by S
23. *if* *sender* $\in$ *CC*
24. CC=CC-{sender}
25. *if* S is selected parent
26. *if* (sender's ED+S's UD< DL)
27. ECL=ECL+{sender}
28. *if* sender's ED >MED
29. MED=ED
30. *end if*
31. *end if*
32. *end if*
33. *if* CC is empty

34. **ParentSelection()**

35. ***end if***

36. ***end if***

37. ***end for***

The task of procedure CandidateClassify is to separate neighbors into parent candidates and children candidates

---

Procedure: CandidateClassify (), Sensor S

---

1. PC= Φ
2. ECL= Φ
3. MED=0
4. PC={ $\forall n \in N(S)$, n having shorter LD than S }
5. CC={ $\forall n \in N(S)$, n having longer LD than S }

The task of procedure ParentSelection is to compute sensor's desired-hop-progress and select a parent among parent candidates according to the result of desired-hop-progress.

Procedure: ParentSelection (), Sensor S

---

1. DHP= $\left\lceil \frac{LD}{DL - MED} \right\rceil$
2. MPS={n | |n's hop progress – DHP| be minimum, $n \in PC$ }
3. ***if*** Num(MPS)=1
4. parent=MPS
5. ***else***
6. SPS= {n | n's LD be minimum, $n \in MPS$ }
7. ***if*** Num(SPS)=1
8. parent=SPS
9. ***else***
10. parent= {n | n's geographic distance to S be minimum, $n \in SPS$ }
11. ***end if***

12. ***end if***

13. ED=MED+1

14. S creates a parent declaration P(node id, parent id, ED)

15. S sends P to N(S)

16. S sets Aggregation-Tree-Construction-Flag OFF

Algorithm 1. DHP (Data Reporting Part), Sensor S

1. after updated sensor's request id in request broadcasting
2. N=|ECL|
3. ***do*** while N≠0
4. ***if*** S receives report from *sender* ∈ *ECL*
5. N=N-1
6. S saves data
7. ***end if***
8. ***end do***
9. S implements aggregation
10. S creates a report p (* report id, node id, data)
11. S sends p to parent

---

* report id= sensor current request id

Chapter 4. DHP Protocol with Area Coverage Algorithm and Connected Dominating Set

DHP protocol will create an energy efficient aggregation tree subject to delay bound, which connects all nodes in the network. As extra nodes may exist in networks consisting of randomly deployed nodes, reporting from these nodes is unnecessary and can be saved. To improve network energy efficiency, we propose an area coverage algorithm to select active nodes from the original network to monitor their surroundings, while others switch to sleep mode and stop monitoring and reporting; and then in data aggregation, DHP protocol will be implemented over selected active nodes. Since it not only saves enormous energy consumed in monitoring but also reduces considerable number of unnecessary transmissions, DHP protocol with Area coverage algorithm (DHPA) will be more energy saving than DHP protocol in networks with extra nodes.

Besides the area coverage algorithm, a connected dominating set (CDS) can be added in our delay-bounded energy-efficient data aggregation protocol. Let us call it DHPAC. In DHPAC, a CDS is computed over all active sensors. Then one LMST is constructed to connect all nodes in CDS (dominants) and actor. In aggregation tree construction, only dominants perform DHP protocol and the remaining active sensors just attach to the aggregation tree through their closest dominants. As data from non-dominants are collected and aggregated at nearby nodes and only dominants consider connecting to parents with expected progress, DHPAC is more energy efficient than that DHPA for most delay limits. However, if the delay limit is large enough and the generated aggregation tree is close to LMST, obviously, DHPA will outperform DHPAC.

Both area coverage algorithm and CDS construction algorithm will separate sensors into different roles: the former separates sensors into active and inactive sensors; the latter separates active sensors into dominants and non-dominants. As sensors in different roles will consume greatly different energy in aggregations, in order to balance energy distribution in the entire network, we recommend that sensor networks periodically implement backbone construction algorithm(s) to realize role rotation.

4.1 DHP with Area Coverage Algorithm (DHPA)

In Chapter 2, we discussed that the localized area coverage algorithm [GCSS] proposed by Gallais et al. would attain a subnet that is able to complete monitoring task as well as its original network while the rest of nodes fall asleep. Nodes following this algorithm can locally determine whether to be active and need a small amount of message exchanges between one-hop neighbors. Our DHPA algorithm is going to use this algorithm to select an active node set over which data aggregation is implemented.

In order to reduce simultaneous decision broadcastings between neighbors, [GCSS] proposed each node to start its intersection-based coverage evaluation [H] after a random timeout. Nodes following this algorithm do not rely on previous 'hello' message exchanges to get neighbors' information, but they do rely on decisions from them; thus, nodes that start area coverage evaluation early will find few neighbors that can cover its monitoring area. Consequently, these nodes have higher chances to stay active. To balance node energy in the entire network, nodes with more energy have more chance to be active nodes than others. To reach this goal, in DHPA algorithm the timeout at one node is $T1=\frac{c}{E_{rest}}$ rather than a random value, where E_{rest} is node residual energy and c is a constant: the more energy a node has, the earlier it starts to evaluate.

[GCSS] offers four optional methods for decision advertising. If retreat messages are used, the algorithm may result in sparser topologies than ones without retreat message. However, in such a node set, the average node degree will be too low; consequently, nodes in data aggregation will be hard to find neighbors with appropriate progress when needed. To balance collecting reliability and energy consumption, we select the PN scheme: area coverage evaluation is only implemented once at each node, and both positive and negative messages are broadcast.

Since there is no energy limit at actor, we assume that actor periodically sends out a "hello" message so that sensors around it can detect it. When area coverage algorithm is completed in the entire network, every active node will know all active neighbors around it. Then each of them can locally compute its LMST neighbor candidates from all active neighbors.

In DHPA, only active nodes take part in data aggregation. When one data collecting is required, DHP protocol is implemented over the LMST rooted at actor and connecting all active sensors.

Example

We assume data aggregation by DHPA algorithm will be implemented over the network shown in Figure 15. All sensors have the same initial energy. After area coverage algorithm implementation, sensors are divided into active and inactive node sets as shown in Figure 21. According to received status (active or not) decisions from neighbors, each active node selects LMST neighbor candidates from active neighbors. When one data collecting is required, a request message is forwarded along a LMST (like in Figure 22) from actor to all active nodes. The figure also indicates each node's UDG-distance in the original network and LMST-distance in the LMST connecting all active nodes.

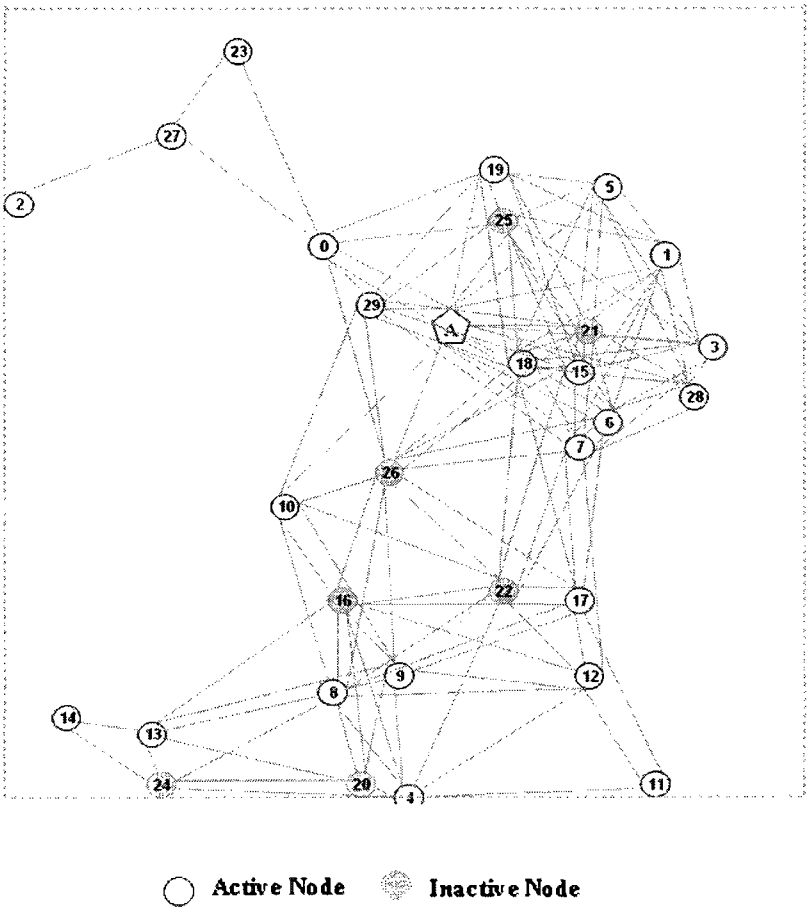


Figure 21. Network after Area Coverage Algorithm

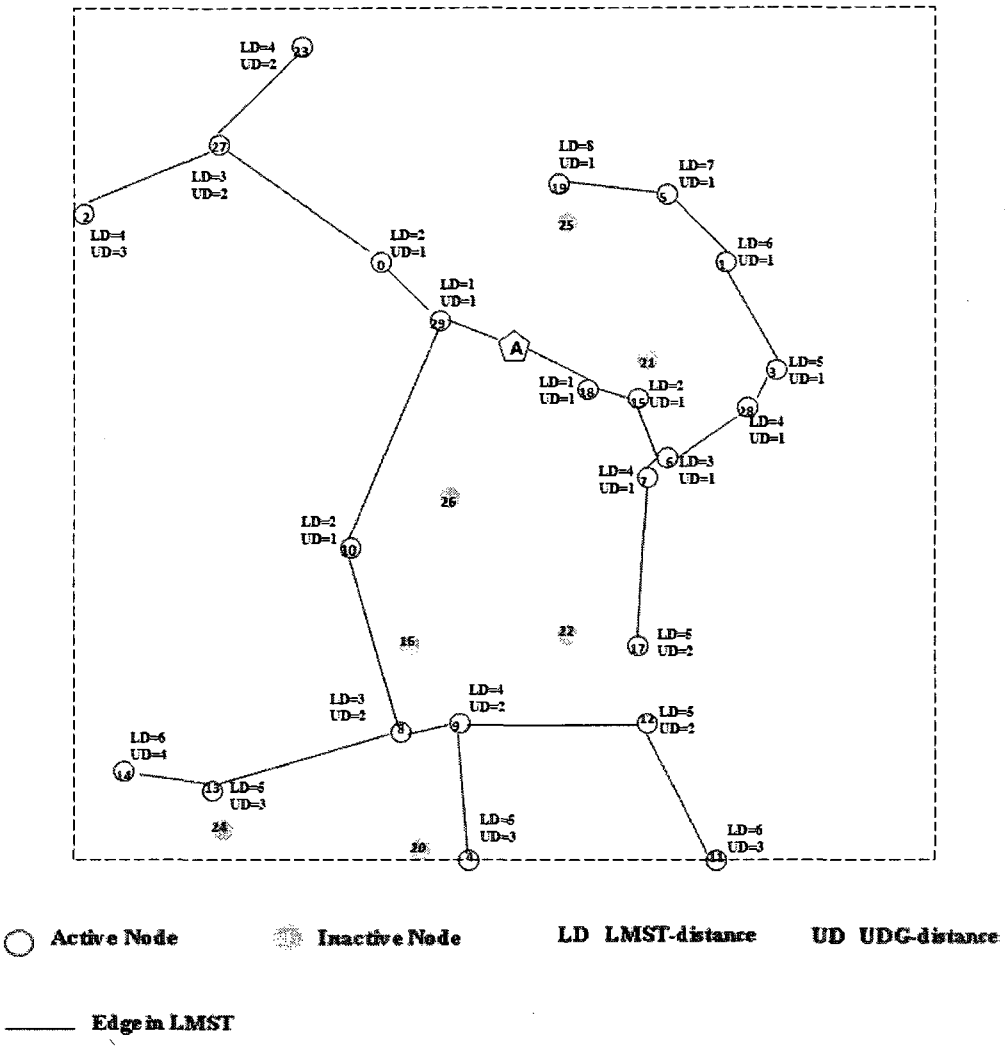


Figure 22. LMST, LMST-distance and UDG-distance over active nodes

The LMST (as in Figure 22) that emerges in request broadcasting will become the final aggregation tree when delay limit for data aggregation is larger than 8 hops since paths along the LMST can meet delay requirements at all nodes.

When delay limit is lower than 8 hops, nodes with detected delays along LMST that cannot satisfy the requirement will seek neighbors with appropriate progress to be parents in aggregation tree. Figure 23 indicates the aggregation tree generated by DHPA when delay limit is 4 hops. In the aggregation tree construction, every node will seek its parent based on *DHP* computed at it. For example, nodes in the area surrounded by dashed lines will proceed

as follows: first, nodes 14 and 11 find no neighbors with *LD*s less than theirs. Then they start their parent selection. Since there is no other parent candidate, node 14 has to select node 13 to be its parent even though its desired hop progress is $\left\lceil \frac{5}{4-0} \right\rceil = 2$ hops. Meanwhile, node 11 gets the same *DHP* and finds its parent with expected progress, node 9. Next, nodes 4, 12, and 13 perform parent selection. Similar to node 14, node 13 cannot access other nodes with more progress. Then its LMST parent (node 8) eventually becomes its parent choice. Nodes 4 and 11 get the same *DHP*= $\left\lceil \frac{5}{4-0} \right\rceil = 2$ hops and select node 8 as their common parent. When node 8 receives all child candidates' decisions, it detects that expected reports from children will experience at most 2 hops. Considering its LD, node 8 thinks its *DHP*= $\left\lceil \frac{3}{4-2} \right\rceil = 2$ hops. But the only neighbor with progress in its communication range is node 10 (one hop progress), which becomes the parent of node 8 eventually. And the aggregated report from node 8 will experience 3 hops when it reaches the selected parent. Due to this report, node 10 has to send its aggregated report to actor directly.

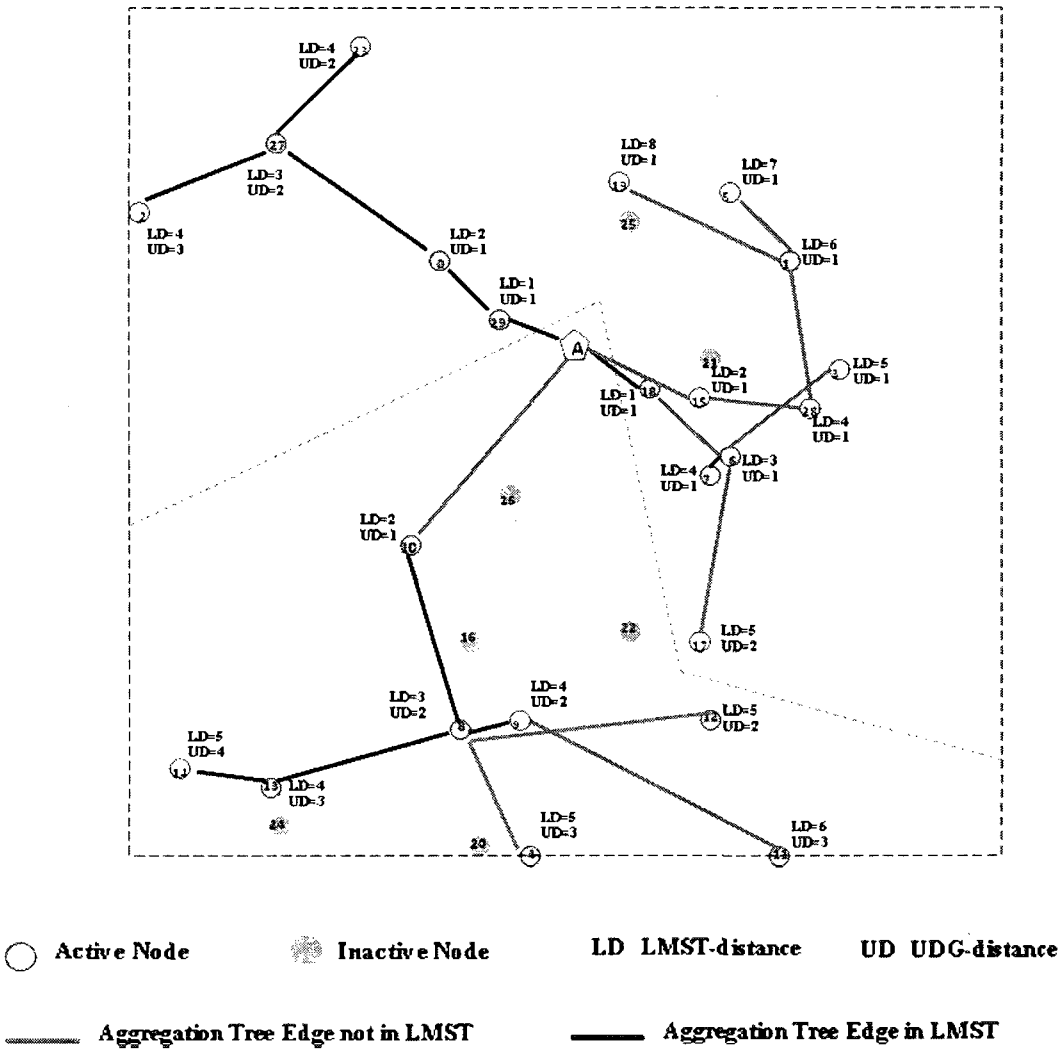


Figure 23. DHPA Aggregation Tree with Delay Limit=4 hops

When the delay limit is small enough like 3 hops, unexpected edges (edges between parent and its unexpected children) will appear in aggregation tree created by DHPA algorithm. Figure 24 shows an aggregation tree with delay limit= 3 hops. One unexpected edge (dashed edge) shows up in this aggregation tree. When node 8 receives parent selection result from node 13, it finds a report from this child will experience 2 hops. Since delay limit=3 hops and actor is not in its neighborhood, node 8 ensures this report will not reach actor before deadline. Thus, it considers node 13 as an unexpected child. As for other nodes in the area surrounded by dashed lines, nodes 4, 11, and 12 do not find any child, then compute their DHP and get a common result: 2 hops. According to the DHP, nodes 4 and 12

select node 8 as their parent, and node 11 chooses node 9. As node 8 excludes node 13's report from its aggregation, its *DHP* will be $\left\lceil \frac{3}{3-1} \right\rceil = 2$ hops. However, no neighbor can offer such progress. Thus, it has to choose the LMST parent node 10 as its parent. As for node 9, its *DHP* is 2 hops and, node 10 is the best parent choice among all parent candidates. Because of expected children nodes 8 and 9, node 10 selects actor as its parent.

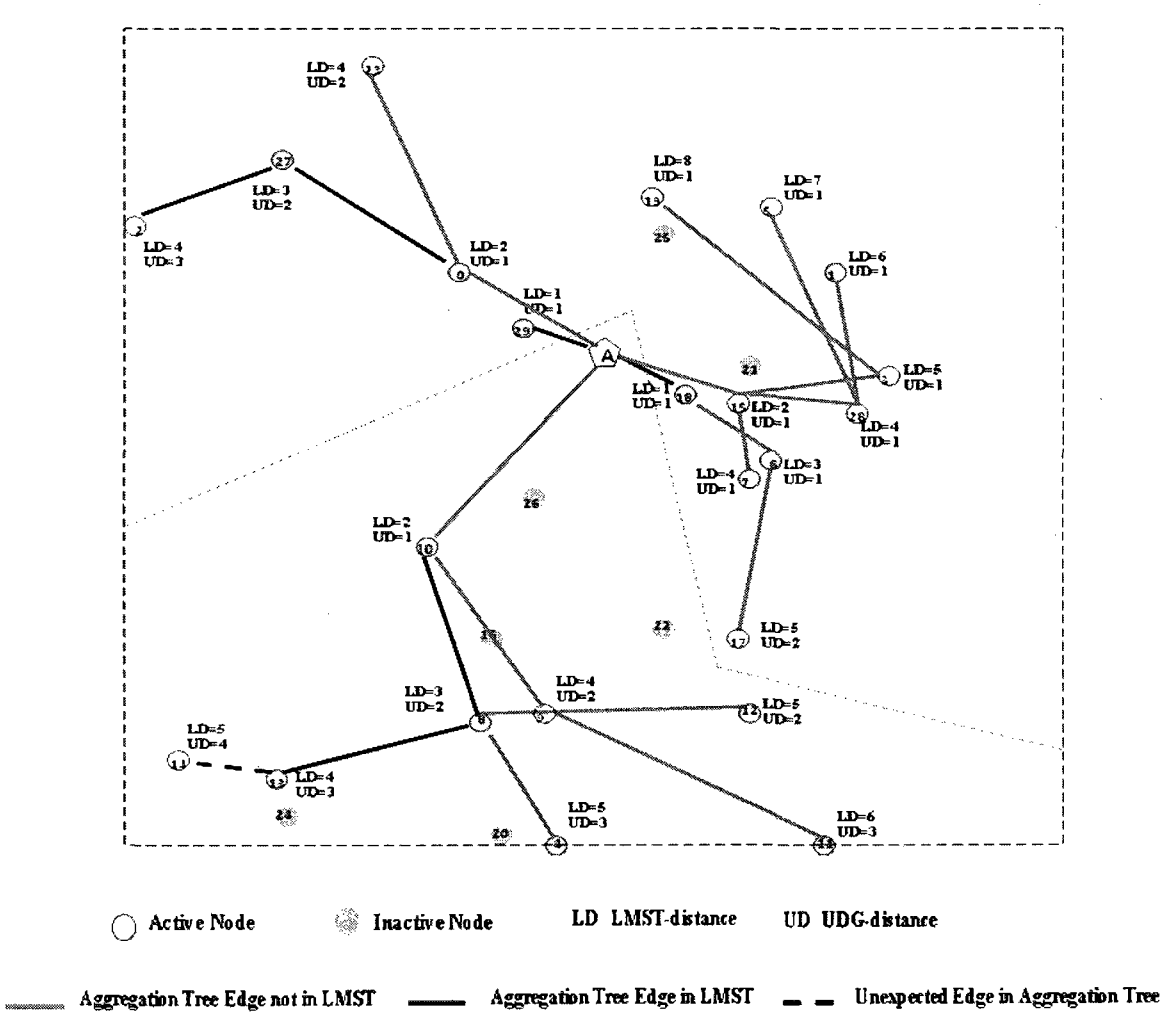


Figure 24. DHPA Aggregation Tree with Delay Limit=3 hops

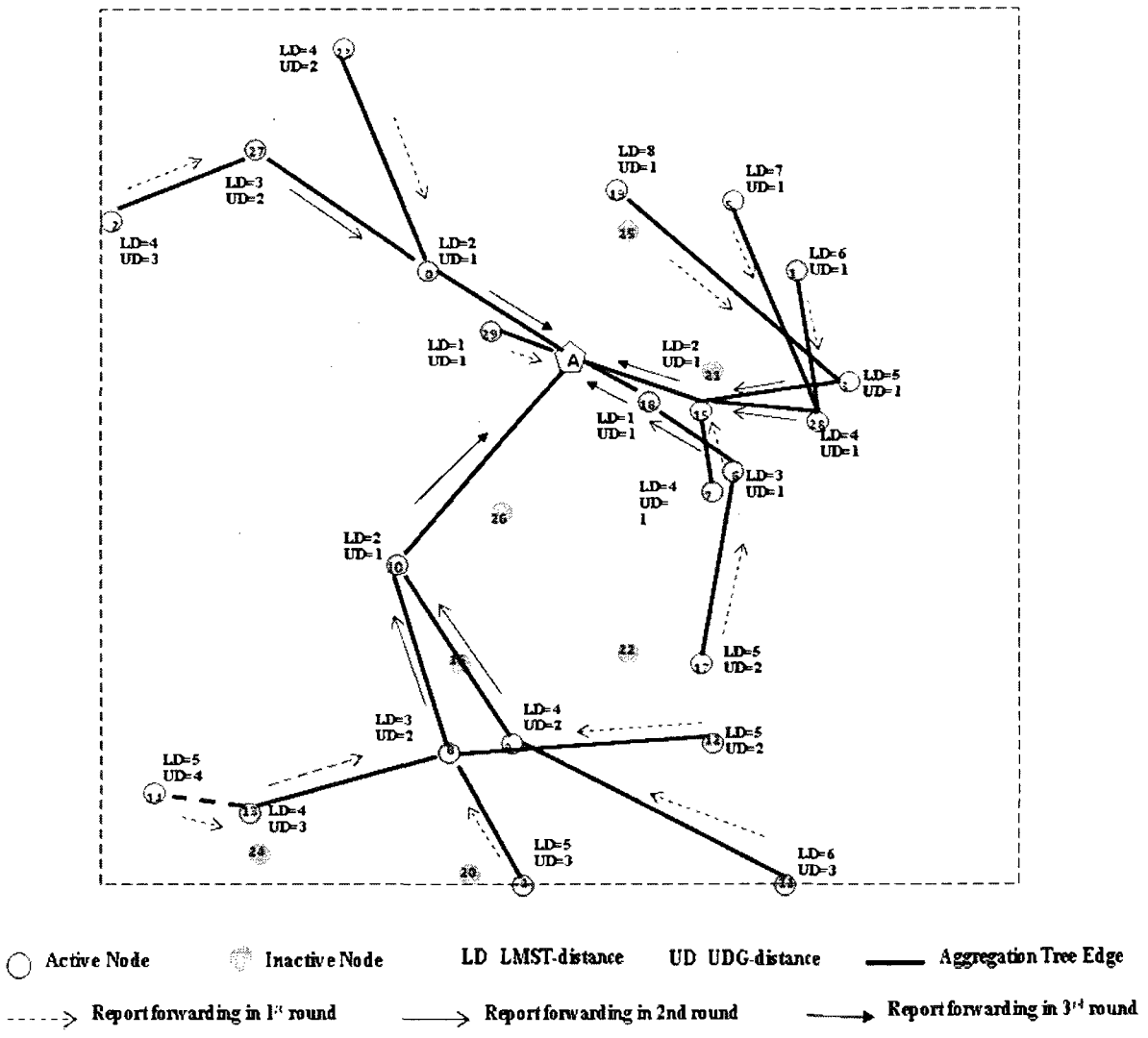


Figure 25. Data Aggregation (Delay Limit=3 hops) by DHPA

Data reporting by DHPA algorithm at nodes will proceed as it does by DHP protocol. First, leaf nodes send out their own sampling data. Internal nodes will wait until they receive all expected reports, then perform aggregation and send aggregated reports to parents. During this process, all reports from unexpected children will be neglected by parents.

For example, Figure 26 presents nodes' data reporting along aggregation tree with delay limit=3 hops. In the first round, leaf nodes (nodes 1, 2, 4, 5, 7, 11, 12, 14, 17, 19, 23, and 29) and internal node without expected child (node 13) send out their data reports.

During this round, node 13 does not wait for report from node 14 since it regards the report as unexpected one. In the second round, internal nodes (nodes 3, 6, 8, 9, 27, and 28) perform aggregation and send aggregated reports. In the last round, nodes 0, 10, 18, and 15 implement their aggregations and report transmissions. Data aggregation in the entire network lasts a delay of 3 hops.

4.2 Algorithm Pseudo Code

Sensors by DHPA algorithm first implement area coverage algorithm. Then only active sensors perform DHP protocol (code is offered in the last chapter) when data collecting is required.

Suppose that the network consists of sensor set V and actor A, and

N(v): neighbor table of node v

AN(v): active neighbor table of node v

UD: UDG-distance to actor

LD: LMST-distance to actor

DL: Delay Limit

LNCL: LMST Neighbor Candidate List

Algorithm 2. DHPA, Actor A

1. Data collecting is required
2. A creates request packet p (request id, node id, A's UD and LD, DL, and LNCL)
3. A sends p to AN(A)
4. ***for*** each report received by A
5. A saves report
6. ***end for***

Algorithm 2. DHPA, Sensor S

1. ***if*** Internal timer notice S to perform Area coverage evaluation
2. S implements Area Coverage Algorithm
3. ***end if***
4. ***if*** S is active
5. S implements *DHP protocol , where N(S) is replaced with AN(S)

6. end if

*use code presented in the last chapter

Algorithm 3. Area Coverage Algorithm, S

1. S sets T1= $\frac{c}{E_{rest}}$
2. S initiates N(S)
3. ***do while*** T1 isn't expired
4. ***if*** S receives working status decision P
5. According to p, S creates a neighbor record n (node id, node position, node energy)
6. S adds n to N(S)
7. ***end if***
8. ***end do***
9. S implements Intersection-based area coverage evaluation[H]
10. ***if*** S is covered
11. S working status is active
12. ***else***
13. S working status is inactive
14. ***end if***
15. S creates a working status decision P (node id, position, residual energy, working status)
16. S sends P at maximum transmission power
17. ***if*** S working status is inactive
18. S switches to sleep mode
19. ***end if***
20. ***for*** each received working status decision P by S
21. According to p, S creates a neighbor record n (node id, node position, node energy)
22. S adds n to N(S)
23. ***end for***

4.3. DHP with Area Coverage Algorithm and Connected Dominating Set Construction (DHPAC)

DHPAC algorithm implements not only a localized area coverage algorithm [GCSS] but also connected dominating set (CDS) construction before data aggregation. The CDS is built over active nodes selected by the area coverage algorithm [GCSS], where a dominant evaluation algorithm proposed by [CS] is used.

As a sensor having energy no more than E_{th} will quit (mentioned in assumption section), every sensor should receive all available neighbors' working status decisions before T2, where $T_2 = \frac{c}{E_{th}} + \Delta t$ (Δt is supposed to be the sum of delays of one decision message transmission and one evaluation processing). An active sensor will start its dominant evaluation when *T2* expires. Node priority required by [CS] will be decided by a key= {residual energy, node *id*}:

1. Node with more residual energy has higher priority.
2. For nodes with same energy, the node with lower node id has higher priority.

According to this configuration, sensors having more residual energy have more chance to be dominant and are burdened with more duty in data aggregation. Each active sensor will send its dominant evaluation result to all active neighbors. All dominants make up of a new UDG *U'*. And one dominant's distance to actor over *U'* is named *UD'*.

We propose that dominants and non-dominants adopt different routing protocols in aggregation tree construction: DHP protocol will be performed by dominants and non-dominants attach to the aggregation tree through their closest dominants. Certainly, dominants having non-dominant children should consider the delay resulting from these nodes in their desired hop progress computation. Since only dominants implement DHP protocol, LMST *LT'* in DHPAC is constructed over *U'*. One dominant's LMST-distance to actor over *LT'* is denoted by *LD'*. When one dominant receives all active neighbors' dominant decisions, it will select LMST neighbor candidates from all neighboring dominants and put them into a list *LNCL'*. Actor will be treated as one special dominant in this process.

When the system needs a data collecting, actor will initiate a request message and send it to all active neighbors. This message includes request id, sender id, delay limit, sender's *UD'*, *LD'*, and *LNCL'*.

If a dominant receives a request message, it will update the sender's request id, *UD'* and *LD'* in its neighbor table and update its own *UD'* if the request id is new to it or the sender's *UD'* is less than its *UD'-1*. In addition, if the request id is new to the dominant and the message sender is dominant's LMST neighbor, the dominant will refresh its request id, delay limit, and *LD'*. After refreshing the information, the dominant generates a new request message according to its own knowledge and sends it to all active neighbors. When all dominants complete their request message broadcasting, a LMST connecting actor and all dominants is built.

When one dominant knows all neighboring dominants' *LD'*, it will pick up parent candidates (neighboring dominants having shorter *LD'*) and child candidates (neighboring dominants having longer *LD'* and all non-dominant neighbors). Once the dominant receives all child candidates' parent decisions, it computes its desired hop progress and then carries out its parent selection as sensors do in DHP protocol. Note that in this process, parents do not distinguish between dominant and non-dominant children; all of them will be treated equally. Its decision will be sent to all parent candidates.

As for non-dominant, it just listens to request messages and parent decisions from its neighbors. The sensor will update its request id when it receives a request message from its closest dominant and broadcast its parent selection decision that the closest dominant is chosen. As non-dominants will never be parents of others, reports from these nodes always reach their parents in a delay of one hop. Therefore, the evaluated report delay in parent decision from one non-dominant must be one hop.

Example

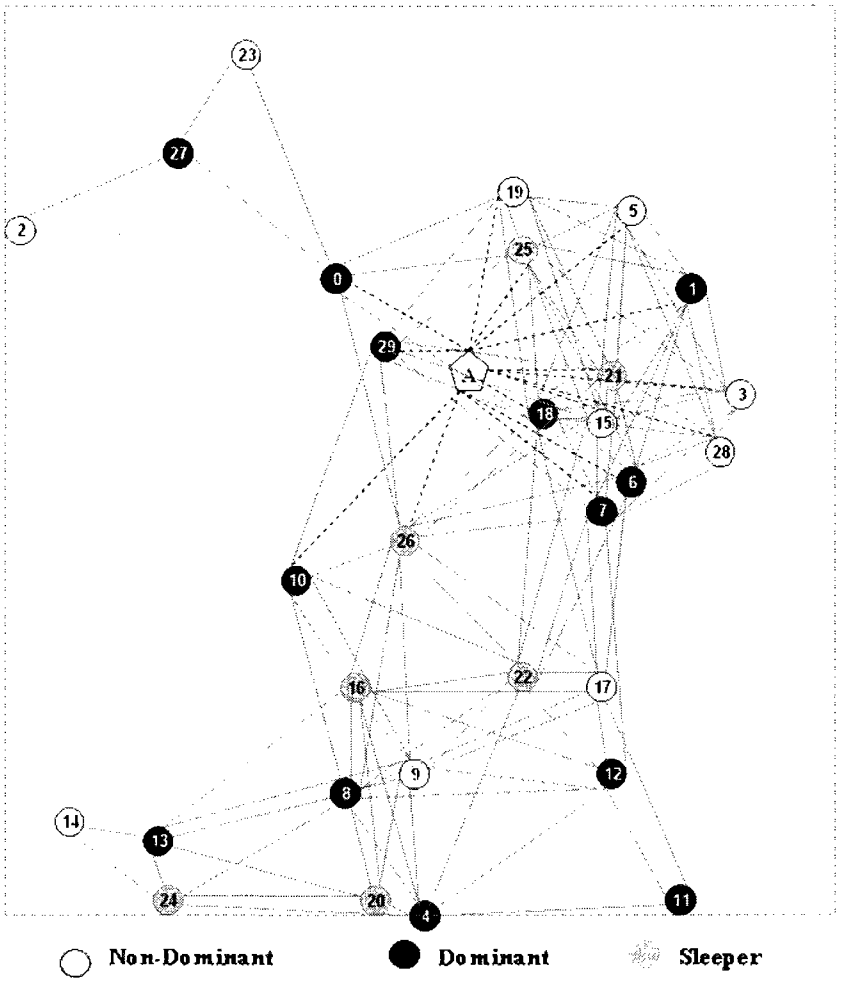


Figure 26. Dominant, Non Dominant, and Sleeper

We assume that one data collecting will be implemented over the network shown in Figure 15. All sensors have same initial energy. As shown in Figure 26, when the network finishes area coverage algorithm and CDS construction, every node in the network will get one of three roles: dominant, non-dominant, and sleeper (inactive node). When a data collecting is triggered, a request message is forwarded along the LMST from actor to all dominants shown in Figure 27. All dominants' *LD*'s (along the LMST) and *UD*'s to actor are also presented in the figure.

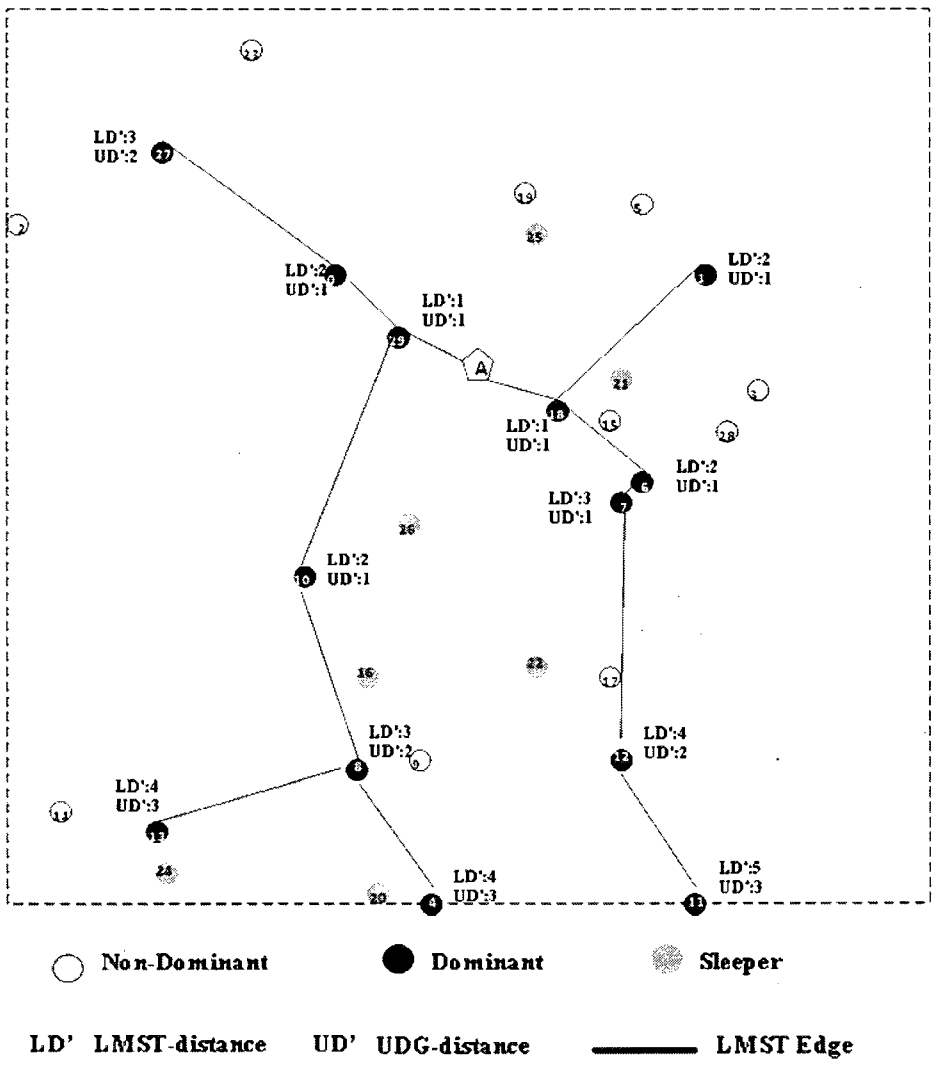


Figure 27. LMST, LMST-distance and UDG-distance over UDG' after CDS Construction

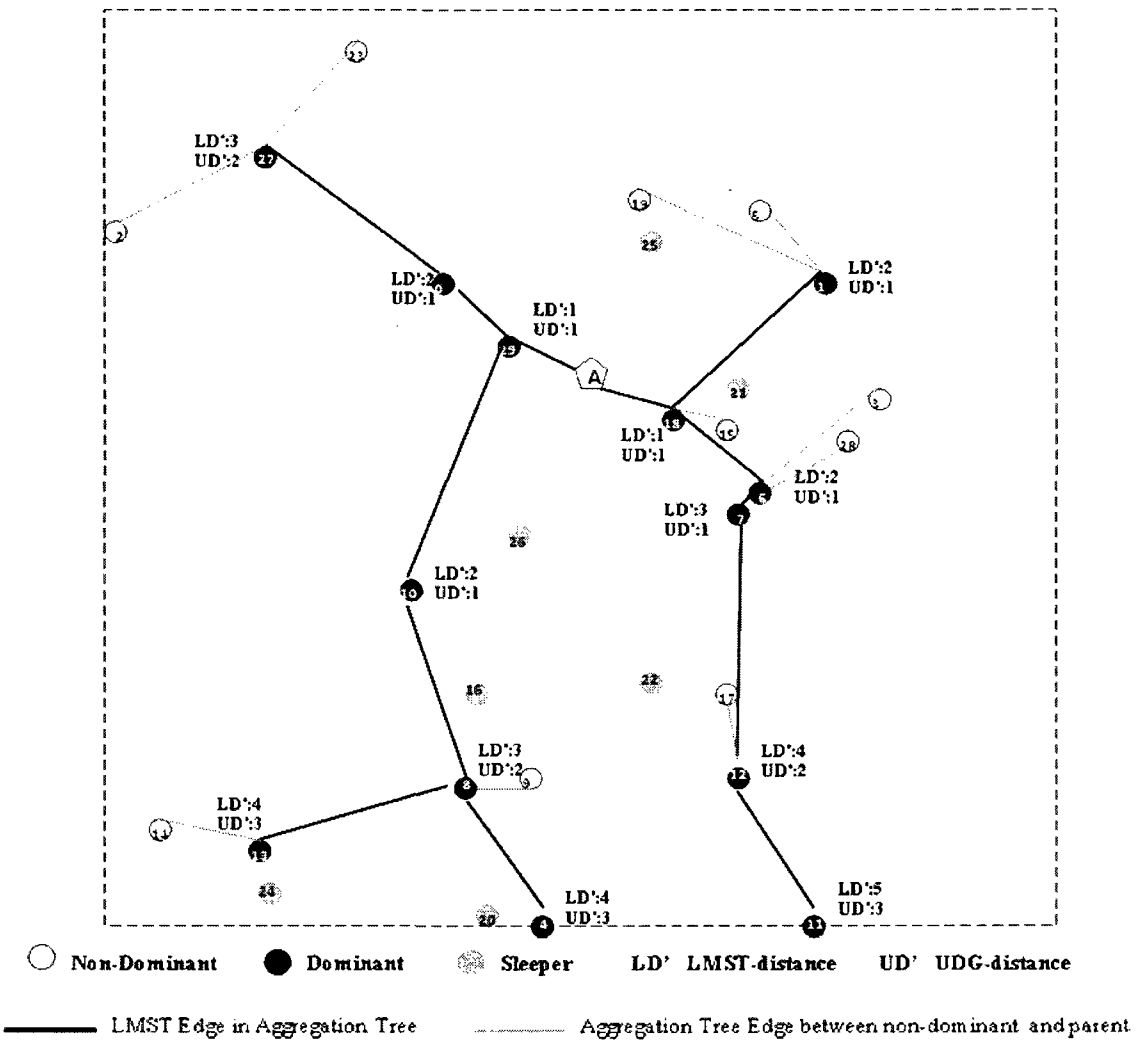


Figure 28 DHPAC Aggregation Tree with delay limit=5 hops

When the data aggregation delay limit is larger than 4 hops, all dominants find their delay along LMST can satisfy the delay requirement. Then, edges in LMST become routes for collecting reports from all dominants. As for non-dominants, their reports will be delivered to their closest dominants in data aggregation. Figure 28 indicates the aggregation tree generated by DHPAC when delay limit is 5 hops.

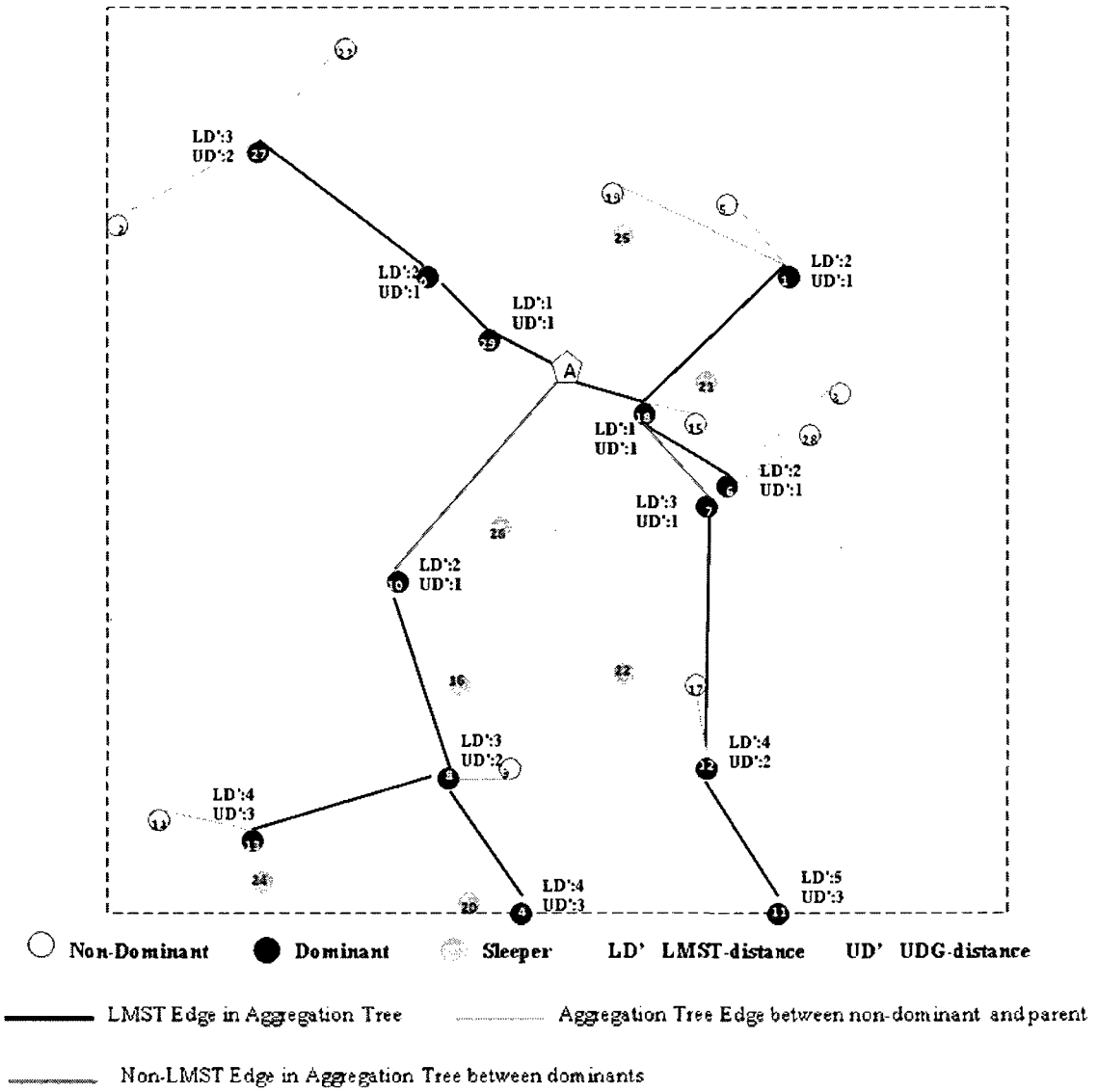


Figure 29 DHPAC Aggregation Tree with delay limit=4 hops

When delay limit is not larger than four hops, some dominants detect that their known paths to actor (along LMST) cannot meet delay requirement any more, and that parents with more progress have to be found so that their reports can reach actor before deadline. On the other hand, despite the change of delay limit, non-dominants still maintain their parent selection. In Figure 29, we show the aggregation tree generated by DHPAC at delay limit=4 hops. In this tree, all non-dominants (nodes 2, 3, 5, 9, 14, 17, 19, 23, and 28) select their closest dominants as parents. Some dominants (nodes 0, 1, 4, 6, 18, 27, and 29) consider their delay along LMST can satisfy delay requirement and do not implement parent switching. And the remaining dominants detect that their LMST parents cannot

offer as many progress as needed. Parents with more progress are required at these nodes. According to the computed desired hop progress, they attach to appropriate parents. For instance, node 13 having one non-dominant child, node 14, is aware that its desired hop progress is $\left\lceil \frac{4}{4-1} \right\rceil = 2$ hops. But in its neighborhood, no dominant neighbor can offer so many progresses, and then it has to select the only parent candidate, node 8, to be its parent. Some other dominants (nodes 8, 11, and 12) face the same problem after their desired hop progress computations, and have to choose their LMST parents to be their parents. Nodes 7 and 10 also desire a hop progress= 2 hops. Fortunately, they can find neighbors satisfying the requirement. Then node 18 and actor are respectively chosen to be parents of node 7 and 10.

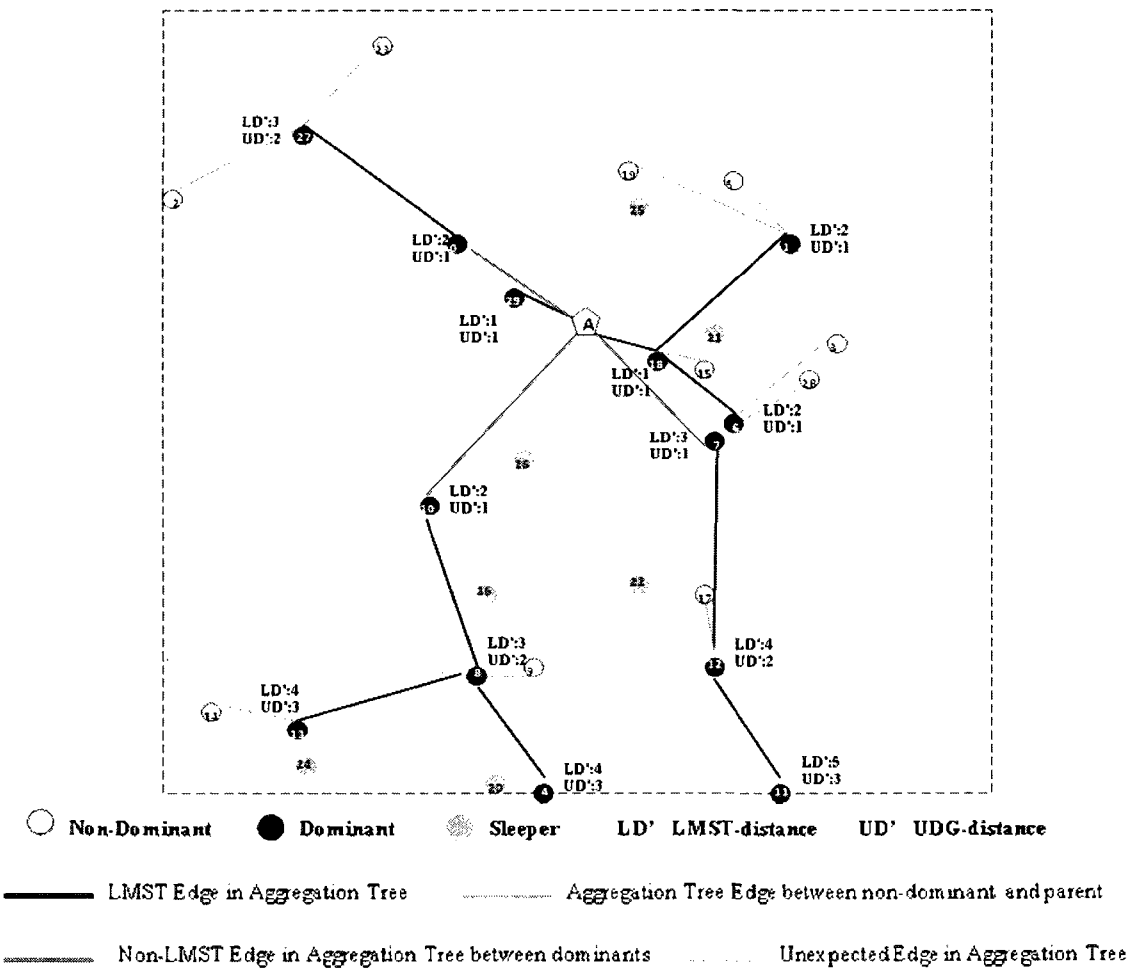


Figure 30. DHPAC Aggregation Tree with delay Limit=3 hops

When delay limit drops to 3 hops, certain nodes will fail to find paths meeting the delay requirements. For example, the *UD'* of node 13 (in Figure 30) is 3 hops equal to delay limit, which can not accept any expected child. Therefore, when it receives node 14's parent declaration, node 13 ensures this child is unexpected. The entire aggregation tree at this delay limit is shown in Figure 30. All non-dominants still attach to their closest dominants. Then dominants (no matter whether they have non-dominant children or not) implement DHP protocol to select their parents.

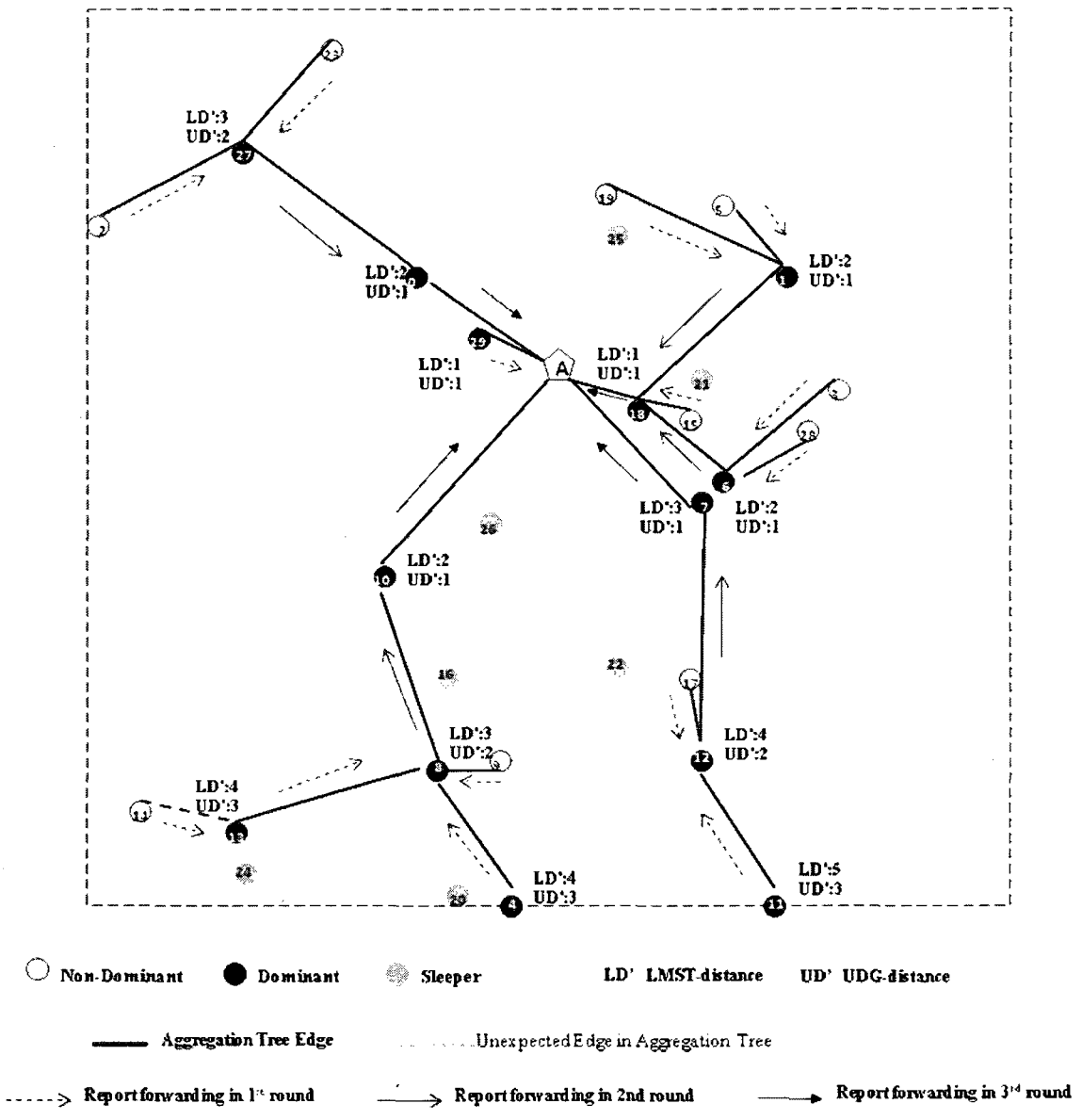


Figure 31. Data Aggregation (delay limit=3 hops) by DHPAC

When nodes know their children and parents, they are going to report their data as DHP and DHPA. First of all, leaf nodes and internal nodes without expected child in the aggregation tree send out their own data without any delay. Internal nodes will wait until all expected reports are received, then perform aggregation and send aggregated reports to parents. Meanwhile, those unexpected reports will be abandoned by their parents. Figure 31 indicates data reporting along the aggregation tree shown in Figure 30. All data from active nodes in the network are aggregated and forwarded to actor in 3 rounds (a total delay of 3 hops). During this process, data report sent by node 14 is neglected by the receiver (node 13), which sends its report simultaneously.

4.4 Algorithm Pseudo Code

Suppose network consists of sensor set V and actor A, and

N (v): neighbor table of node v

AN (v): active neighbor table of node v

DN (v): dominant neighbor table of node v, actor here is treated as special dominant

NDN (v): non dominant neighbor table of node v

Num (N): the number of nodes in node set N

UD': UDG-distance to actor over UDG made up of dominants

LD': LMST-distance to actor over UDG made up of dominants

DL: Delay limit

ED: Evaluated Delay of report

MED: Most Experienced Delay of all expected reports

PC: parent candidate list of node

CC: child candidate list of node

ECL: expected children list

LNCL': sensor's direct neighbors in its neighborhood-wide MST connecting all neighboring dominants.

Sensors by the DHPAC algorithm implement area coverage algorithm and CDS construction before data aggregation. Only active nodes selected by area coverage algorithm will attend to data reporting. Active nodes are classified into dominants and non-dominants after dominant evaluation. In data aggregation, non-dominants send their

data to closest dominants, and dominants will implement aggregation if necessary and then send generated reports to parents selected by the DHP protocol.

Algorithm 4. DHPAC, Actor A

1. Data collecting is required
2. A creates request packet p (request id, node id, A's UD'and LD', DL, and LNCL')
3. A sends p to AN(A)
4. ***for*** each report received by A
5. A saves report
6. ***end for***

Algorithm 4. DHPAC, Sensor S

```
1.  if internal timer notice to start area coverage evaluation
2.      S implements area coverage algorithm
3.  end if
4.  if S is active
5.      S implements Dominant Evaluation Algorithm [CS]
6.      S sends evaluation result to AN(S)
7.      if S is dominant
8.          S creates LNCL'
9.      end if
10.     for each data collecting request received by S
11.         if S is non dominant and request sender is the closest dominant of S
12.             S updates request id
13.             if new DL ≠ old DL
14.                 ParentSelection()
15.             end if
16.         end if
17.         if S is dominant
18.             S updates sender's request id, UD' and LD' in N(S)
19.             If (S request id<sender request id)  or (sender's UD'<S's UD'-1)
```

20. S's UD'=sender's UD'+1
21. ***if*** (S request id<sender request id) and (sender and S are LMST neighbor)
22. S updates request id
23. ***if*** new DL ≠ old DL
24. S sets Aggregation-Tree-Construction-Flag ON
25. S updates LD' (=sender's LD'+1), DL
26. ***end if***
27. S creates a request packet p (request id, node id, S's UD' and LD', DL, and LNCL')
28. S sends p to AN(S)
29. ***end if***
30. ***if*** Aggregation-Tree-Construction Flag ON and S has received all requests from DN(v)
31. **CandidateClassify()**
32. ***if*** CC is empty
33. **ParentSelection()**
34. ***end if***
35. ***end if***
36. ***end if***
37. ***end for***
38. ***for*** each parent declaration received by S
39. /* ***Note:*** only dominant will receive such message*/
40. ***if*** *sender* ∈ *CC*
41. CC=CC-{sender}
42. ***if*** S is selected parent
43. ***if*** (sender's ED+S's UD'< DL)
44. ECL=ECL+{sender}
45. ***if*** sender's ED >MED
46. MED=ED
47. ***end if***

48. ***end if***

49. ***end if***

50. ***if*** CC= Φ

51. **ParentSelection()**

52. ***end if***

53. ***end if***

54. ***end for***

55. ***end if***

The task of procedure CandidateClassify performed by dominants is to separate active neighbors into parent candidates and children candidates.

Procedure: CandidateClassify (), Dominant S

1. PC= Φ
2. ECL= Φ
3. MED=0
4. PC={ n $|n \in DN(S)$ and n having shorter LD than S }
5. CC={n $|n \in DN(S)$ and n having longer LD than S}∪{n $|n \in NDN(S)$ }

The task of procedure ParentSelection is to select parents for active nodes. Non-dominants select their closest dominants as parents, and dominants select parents by the scheme proposed by DHP protocol.

Procedure: ParentSelection (), Sensor S

1. ***if*** S is dominant
2. DHP=$\left\lceil \frac{LD}{DL - MED} \right\rceil$
3. MPS={n | |n's hop progress – DHP| be minimum, $n \in PC$ }
4. ***if*** Num(MPS)=1
5. parent=MPS
6. ***else***

7. SPS= {n | n's LD be minimum, $n \in MPS$ }
8. ***if*** Num(SPS)=1
9. parent=SPS
10. else
11. parent= {n | n's geographic distance to S be minimum, $n \in SPS$ }
12. end if
13. *end if*
14. ED=MED+1
15. S sets Aggregation-Tree-Construction-Flag OFF
16. else
17. Parent is the closest dominant
18. ED=1
19. end if
20. S creates a parent declaration P(node id, parent id, ED)
21. S sends P to DN(S)

Algorithm 1. DHPAC (Data Reporting Part), Sensor S

1. after updated sensor's request id in request broadcasting
2. ***if*** S is non dominant
3. S creates a report p (* report id, node id, data)
4. S sends p to parent
5. else
6. N=|ECL|
7. ***do*** while N≠0
8. ***if*** S receives report from *sender* ∈ *ECL*

a. N=N-1

b. S saves data

9. end if
10. end do
11. S implements aggregation
12. S creates a report p (* report id, node id, data)

13. S sends p to parent

14. end if

Both DHPA and DHPAC adopt area coverage algorithm to improve energy efficiency. As a considerable number of nodes switch to sleep mode, both algorithms are expected to be more energy saving than DHP protocol. Besides, considering energy balance, nodes with more energy are prior to do area coverage evaluation, which have higher chance to be active. In DHPAC, a CDS is built over active nodes selected by area coverage algorithm. Then the CDS will make up the backbone of the aggregation tree, and non-dominants will be leaves of the tree through dominants closest to them. In the generated aggregation tree, non-dominants normally connect to their parents at low energy costs, while dominants have to choose longer links and consume more energy. To reduce energy imbalance, we propose nodes with more energy to have higher priorities in dominant evaluation which have more chance to be dominants. Due to the great energy saving at non-dominants, DHPAC algorithm may be more energy saving than DHPA algorithm in most cases. However, if delay limit is large enough so that most nodes in LMST (by both algorithms) find their routes to actor along LMST can satisfy delay requirement, DHPA will be more energy efficient than DHPAC.

Chapter 5. Simulations

All simulations in this chapter have been done in a home-made simulator by Java. The MAC and physical layers in of this simulator are ideal. The experiments are carried using static random unit disk graphs. 100 nodes are placed in a given square of size l (area=$l \times l$) uniformly and independently. In order to control the average node degree (the average number of neighbors) d, all 4950 (=100 × (100-1)/2) potential edges in the networks are sorted by their lengths in ascending order. The length of the $50d$-th shortest edge is chosen to be communicating radius R. The first $50d$-th shortest edges in the list remain in the graph and others are eliminated. Then, this candidate graph is checked for connectivity and void area. The former is examined by Dijkstra's Shortest Path Algorithm [D]. And the latter is checked by implementing greedy routing [F] at each node. If the graph is disconnected or has void area, the procedure is repeated until an expected graph is generated.

Each node can only directly communicate with nodes locating within a circle of R, the communicating radius. All nodes including actor and sensors have identical communicating radius. The ratio of communicating radius over sensing radius of sensor is 2. All sensor nodes have the same initial energy (1 J). The first order radio model [RH] is selected as energy model for all simulations. The values of parameters required by the formula $\beta d^{\alpha}+c$ are $\beta=100pJ/bit/m^{\alpha}$, α=4 and c (the power consumption of electronic circuits in transmitting and receiving) is a constant larger than zero. In each iteration of data aggregation, sensor consumes energy of 10^{-6} J in idle status, and $2*10^{-7}$ J in sleep status. The energy efficiency of protocols is evaluated in two ways: the average node energy consumption for 5 iterations of data aggregations and network lifetime measured by the number of iterations before the first node fails (runs out of power).

For all protocols the costs to construct aggregation tree isn't included in energy expenditures. In each protocol simulation, five data aggregation tasks have been implemented in sequence (simultaneous data aggregations are not expected).

Besides, we assume data aggregation is fully-aggregated. Therefore, after aggregation, data packets generated by different sensors have the same size (200 bits) no matter how

many source nodes are involved. Energy consumed by aggregation processing is neglected in our simulations.

For fairness, when we compare MS framework with our DHPA and DHPAC protocols, data aggregations by MS have been implemented over active nodes selected by area coverage algorithm [GCSS]. Let's call the new protocol (the combination of MS and area coverage algorithm) MSA.

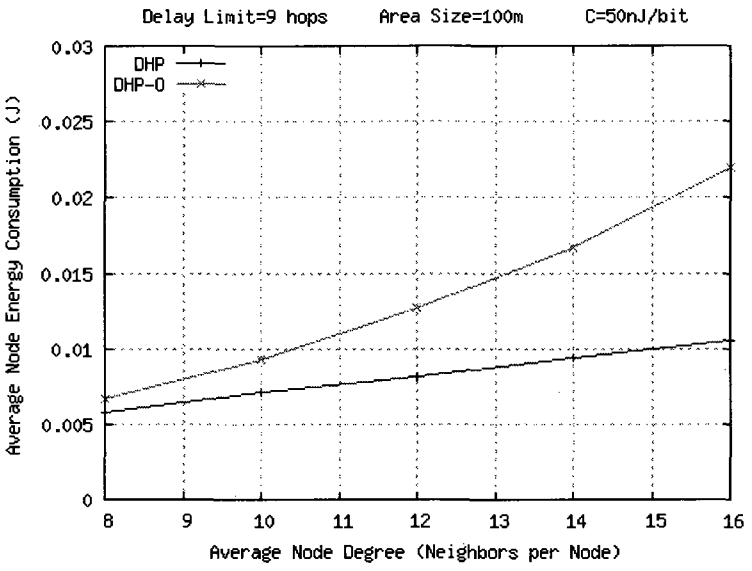
In [MPGA], MS framework uses three routing protocols in three working states respectively. Let's name those protocols after their states, which are start-up protocol, greedy protocol, and aggregation protocol. MS relies on a state switching scheme to control sensor's routing protocol selection. In this switching scheme, there are several parameters needed to be configured:

$r_{th}^{+} = r_{th} + \varepsilon^{+}$: High event reliability threshold

$r_{th}^{-} = r_{th} + \varepsilon^{-}$: Low event reliability threshold

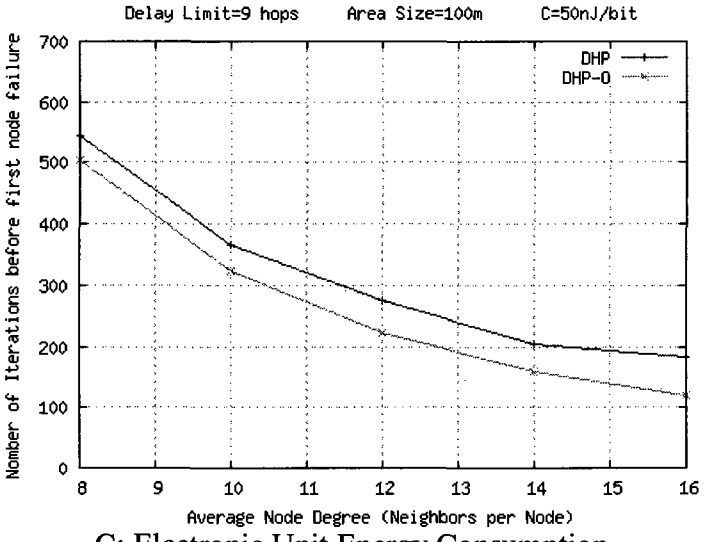
$P_{switching}$: Switching probability when state switching is expected

In our simulations, the values of r_{th} for MS and MSA are the reliable ratios (the number of unexpired reports over the number of all reports) of DHP and DHPA under the same scenario respectively; ε^{+} and ε^{-} are equal to 3%; $P_{switching}$ is 70%. Actor will compute and release feedbacks every aggregation round to inform sensors to do working state decision. As the setting of parameters, data aggregations by MS/MSA may process with or without state transition. In order to examine their performance in two styles, we implemented comparisons under two delay limits: 4 hops (lower than certain nodes' UD) and 9 hops (larger than all nodes' UD). In the former situation, nodes by MS/MSA may switch their working states at different rounds. In the latter situation, nodes in start-up state (by MS/MSA) can complete data aggregations at reliable ratios in the expected range. Therefore, no state transition will take place.



C: Electronic Unit Energy Consumption

Figure 32. Average Node Energy Consumption in Data Aggregations by DHP-O and DHP



C: Electronic Unit Energy Consumption

Figure 33. The Number of Iterations by DHP-O and DHP

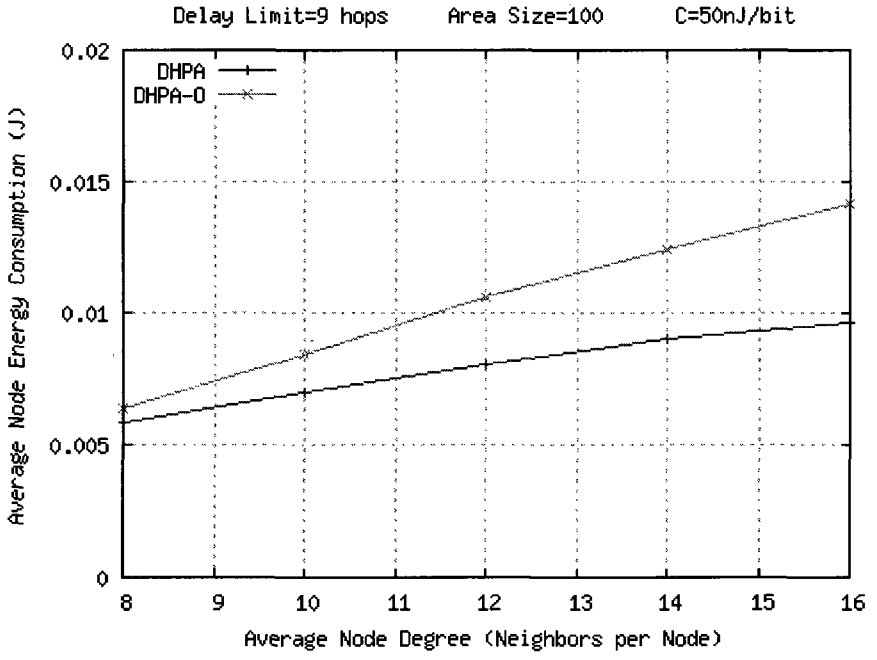
5.1 One-Step Solution and Multi-Step Solution

The main idea of our DHP protocol is to first build a LMST connecting all nodes, and then fix this tree according to the given delay limit. We mentioned that in fixing phase, there are two optional solutions for nodes in selecting new parents: one-step solution and multi-step solution. Our DCPE framework (DHP, DHPA, and DHPAC) adopts multi-step solution in aggregation tree construction. Let us call three algorithms

applying one-step solution DHP-O, DHPA-O, and DHPAC-O. If its LD is larger than its report's lifetime, then desired hop progress for one node following one-step solution is equal to the delay gap and one hop otherwise. Three comparisons have been done between algorithms using one-step solution and multi-step solution. In this section, the delay limit of all data aggregations is 9 hops; nodes are placed in squares of size 100; electronic unit energy consumption is 50 *nJ/bit* and the average node degree of graphs are 8, 10, 12 14 and 16.

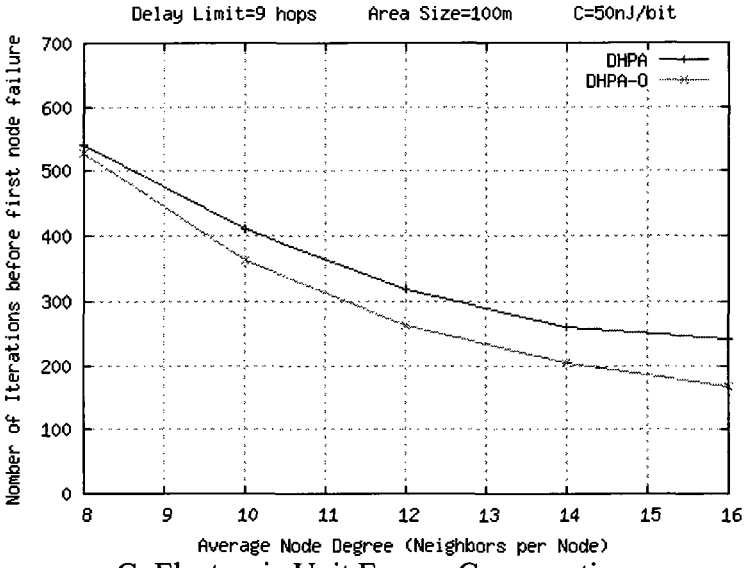
5.1.1 DHP-O VS. DHP

Figure 32 and 33 indicate the average node energy consumption and the network lifetime (the number of data aggregation iterations) by DHP-O and DHP. In general, DHP is more energy saving and better energy balancing than DHP-O. When average node degree rises from 8 to 16, DHP can save energy from 15% to 51% and extends network life 8-55% over DHP-O.



C: Electronic Unit Energy Consumption

Figure 34. Average Node Energy Consumption in Data Aggregations by DHPA-O and DHPA

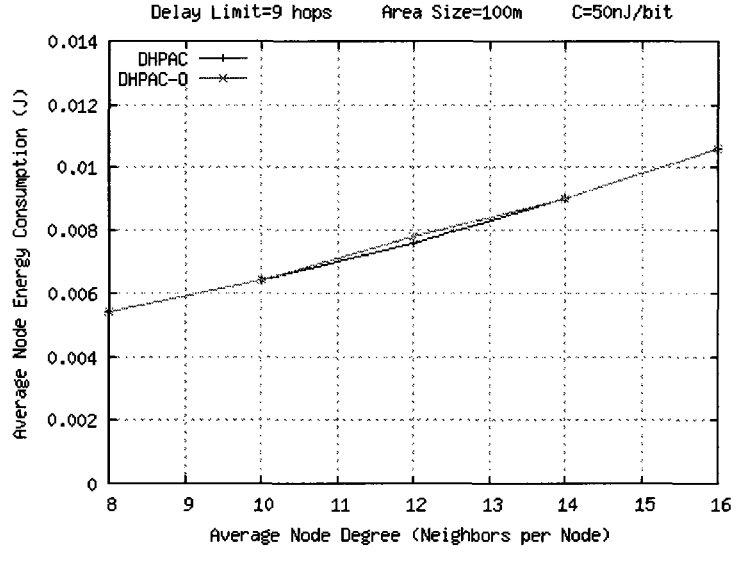


C: Electronic Unit Energy Consumption

Figure 35. The Number of Iterations by DHPA-O and DHPA

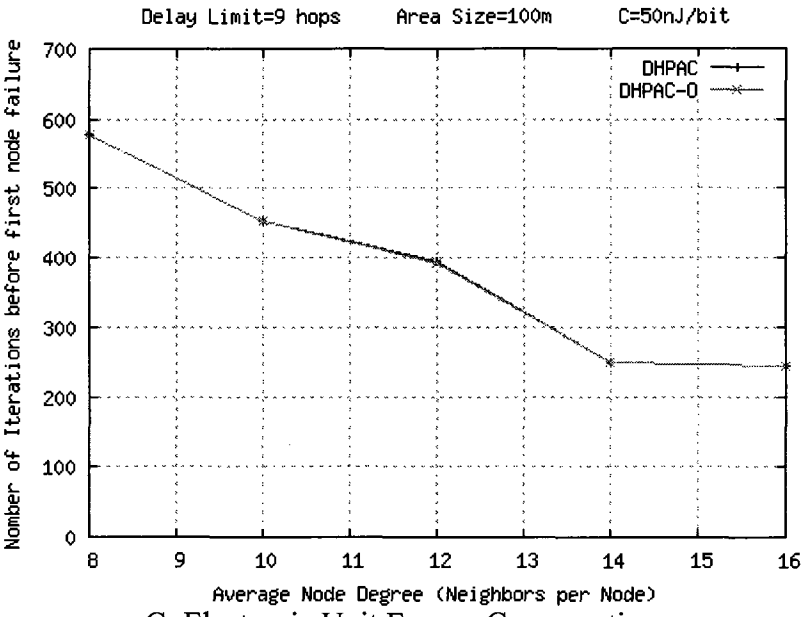
5.1.2 DHPA-O VS. DHPA

Figure 34 indicates DHPA consumes less energy than DHPA-O by 9-33%. Accordingly, in Figure 35 we observe DHPA extends network life 3-44% over DHPA-O. DHPA is still more energy efficient than DHPA-O. But the difference between them is not as significant as that between DHP and DHP-O. It is because smaller LMST leads to less difference of desired hop progress at nodes applying different solution.



C: Electronic Unit Energy Consumption

Figure 36. Average Node Energy Consumption in Data Aggregations by DHPAC-O and DHPAC



C: Electronic Unit Energy Consumption

Figure 37. The Number of Iterations by DHPAC-O and DHPAC

5.1.3 DHPAC-O VS. DHPAC

According to Figure 36 and 37, we observe DHPAC and DHPAC-O present approximately same performance at any network density. CDS is a sparse structure. Thus, there is a considerable chance that one dominant can only find parent candidates with limited hop progress among its neighboring dominants. In this situation, the dominant may select the same parent even if its desired hop progresses computed by two solutions are quite different.

From three comparisons in this section, we can find protocols using multi-step strategy are more energy efficient than those using one-step strategy, especially when no CDS are involved. Therefore, multi-step strategy is more suitable for our power efficient data aggregation protocols.

5.2 Impact of Electric Unit Energy Consumption

In first-order radio energy model, the energy consumption by sender is $\beta d^{\alpha} + C$, where the constant C is energy consumed at electric units (no matter sender and receiver). In this section, we evaluate its impact to protocols, where the value of C is set to 50pJ*/bit, 50nJ/bit, 800nJ/bit, 5 μJ/bit,* and 16μJ*/bit.* The average node degree of graphs is14, and the interest area is a square of size 100m.

Delay Limit=4 hops		Average Node Degree=14			
Area Size=100m	C: Electronic Unit Energy Consumption				
Protocol \ C	***50pJ/bit***	***50nJ/bit***	***800nJ/bit***	***5μJ/bit***	***16μJ/bit***
DHP	0.011	0.011	0.018	0.055	0.15
MS	0.017	0.0178	0.026	0.076	0.203
DHPA	0.009	0.0095	0.015	0.041	0.11
DHPAC	0.0054	0.0056	0.0088	0.027	0.073
MSA	0.013	0.011	0.02	0.051	0.13

Table 2. Average Node Energy Consumption with different Electric Unit Energy Consumption when delay limit=4 hops

Delay Limit=9 hops		Average Node Degree=14			
Area Size=100m	C: Electronic Unit Energy Consumption				
Protocol \ C	***50pJ/bit***	***50nJ/bit***	***800nJ/bit***	***5μJ/bit***	***16μJ/bit***
DHP	0.0045	0.0047	0.0087	0.031	0.089
MS	0.017	0.018	0.026	0.074	0.21
DHPA	0.0043	0.00455	0.00747	0.0234	0.0638
DHPAC	0.0045	0.00452	0.00726	0.0229	0.0629
MSA	0.013	0.0139	0.02	0.0529	0.139

Table 3. Average Node Energy Consumption with different Electric Unit Energy Consumption when delay limit=9 hops

Delay Limit=4 hops		Average Node Degree=14			
Area Size=100m		C: Electronic Unit Energy Consumption			
Protocol \ C	***50pJ/bit***	***50nJ/bit***	***800nJ/bit***	***5μJ/bit***	***16μJ/bit***
DHP	165	165	131	55	21
MS	154	153	117	45	17
DHPA	196	195	162	78	30
DHPAC	253	258	237	129	48
MSA	170	179	135	62	23

Table 4. The Number of Iterations with different Electric Unit Energy Consumption when delay limit=4 hops

Delay Limit=9 hops		Average Node Degree=14			
Area Size=100m		C: Electronic Unit Energy Consumption			
Protocol \ C	***50pJ/bit***	***50nJ/bit***	***800nJ/bit***	***5μJ/bit***	***16μJ/bit***
DHP	205	204	181	96	32
MS	154	148	112	43	16
DHPA	240	259	233	118	49
DHPAC	253	250	232	142	55
MSA	160	154	129	57	22

Table 5. The Number of Iterations with different Electric Unit Energy Consumption when delay limit=9 hops

From Table 2-5, we observe when delay limit is 9 hops our protocols can outperform MS/MSA easily: up to 75% energy conservation and 150% longer network life. When delay limit is 4 hops, our protocols are still more energy efficient than MS/MSA, but the superiority is not as significant as that in cases with delay limit=9 hops. Compared to MS, DHP saves 26-39% energy and extends network life 7-25%. DHPA saves 15-31% energy and extends 15-30% network life over MSA. The advantage of DHPAC is still obvious in comparison to MSA. Its energy conservation can reach 44-58% and network extension is 49-109%.

5.3 Impact of Area Size

The choice of route of our protocols (DHP, DHPA, and DHPAC) and MS framework do not depend on size l of the area containing all the nodes. However, in case of power consumption, the actual distances greatly impact the performance of protocols. In other words, the network energy consumption depends on the actual size of the area. In order to examine the impact of area size to protocols, we implemented data aggregations by protocols in areas of sizes l=50m, 100m, 150m, 200m, 250m, where the average node degree of graphs is 14 and electronic unit energy consumption (c) is 50nJ/bit.

Delay Limit=4 hops		Average Node Degree=14			
Electronic Unit Energy Consumption (C)=50nJ/bit					
Protocol \ Size(m)	***50***	***100***	***150***	***200***	***250***
DHP	0.002	0.011	0.051	0.16	0.445
MS	0.0023	0.0178	0.0835	0.275	0.684
DHPA	0.00165	0.0095	0.043	0.14	0.37
DHPAC	0.00128	0.0056	0.025	0.082	0.2
MSA	0.0018	0.011	0.06	0.204	0.51

Table 6. Average Node Energy Consumption in Different Area Sizes, where delay limit=4 hops

Delay Limit=9 hops		Average Node Degree=14			
Electronic Unit Energy Consumption (C)=50nJ/bit					
Protocol \ Size(m)	***50***	***100***	***150***	***200***	***250***
DHP	0.0013	0.0047	0.022	0.061	0.178
MS	0.0025	0.018	0.088	0.273	0.7
DHPA	0.0012	0.0045	0.019	0.054	0.155
DHPAC	0.0012	0.0045	0.019	0.054	0.15
MSA	0.0019	0.0139	0.0696	0.22	0.56

Table 7. Average Node Energy Consumption in Different Area Sizes, where delay limit=9 hops

Table 6 and 7 indicate the average node energy consumption by protocols in different area sizes where delay limit is 4 and 9 hops respectively. Obviously, when delay limit is 9 hops, our protocols can easily outperform MS/MSA. When area size is 50, DHP consumes less energy than MS by 48%; DHPA and DHPAC can save 37% of energy in comparing to MSA. Once area size is 100 or more, our protocols present more obvious advantages in energy saving. Our protocols only spend about 25% energy needed by MS/MSA. When delay limit is 4 hops, DHP spends 17% less energy than MS when area size=50, and about 40% less when area size is 100 or more. In this delay limit, DHPA and MSA are competing when area size is 50 and 100: only 7% and 13% difference in energy consumption. However, DHPA shows obviously better energy efficiency than MSA in large size (≥150) networks: about 30% energy conservation. DHPAC presents

outstanding performance in cases with low delay limit (4 hops). Its energy consumption varies from 39-71% of that needed by MSA. From both tables, we can find energy consumptions by different protocols have less difference in small size (like 50) of networks than in large size of networks. It is because the energy consumption at electronic units is ignorable in packets transmissions in large networks, while in small size of networks this part of energy consumption is considerable and fixed in each transmission by any protocol.

Delay Limit=4 hops	Average Node Degree=14				
Electronic Unit Energy Consumption (C)=50nJ/bit					
Size(m) / Protocol	***50***	***100***	***150***	***200***	***250***
DHP	1492	165	34	12	5
MS	1463	153	32	11	5
DHPA	1780	195	39	13	6
DHPAC	2451	258	57	21	8
MSA	1666	179	35	12	6

Table 8. The Number of Iteration in Different Area Sizes, where delay limit=4 hops

Delay Limit=9 hops	Average Node Degree=14				
Electronic Unit Energy Consumption (C)=50nJ/bit					
Size(m) / Protocol	***50***	***100***	***150***	***200***	***250***
DHP	2294	204	40	13	6
MS	1324	148	31	10	5
DHPA	2404	259	51	17	8
DHPAC	2457	250	61	21	8
MSA	1530	154	32	11	5

Table 9. The Number of Iterations in Different Area Sizes, where delay limit=9 hops

Table 8 and 9 respectively show the network life in different area sizes when delay limit is 4 and 9 hops. Similarly, when delay limit is 9 hops, our protocols present great superiority in experiments. DHP causes 20-38% longer network life if network area size is 100m and more. If the size is 50m, the extension will reach 73%. DHPA can maintain about 60% longer network life than MSA. DHPAC achieves 60-90% longer network life than MSA. When delay limit is 4 hops, the network life under MS/MSA is close to that

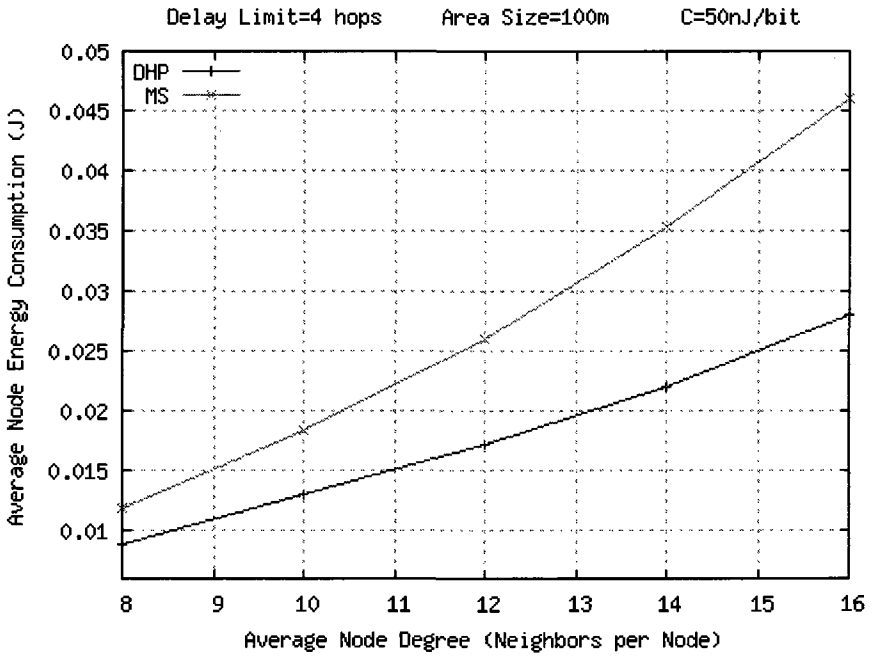
under DHP/DHPA. In this situation, DHPAC still presents its excellent performance, which can extend 33-75% network life over MSA.

5.4 Impact of Network Density

Network density can influence our protocols (DHP, DHPA, and DHPAC) and MS/MSA from many aspects. First of all, if interest area size and node number is fixed, larger network density means longer communicating radius of node. Then nodes may send data report to further neighbors if required. Secondly, in dense networks more nodes will decide to sleep after implementing area coverage algorithm.

In this section, data aggregations by protocols have been implemented over networks placed in squares of size 100m, while the average node degree is 8, 10, 12, 14, and 16. And the electronic unit energy consumption is 50*nJ/bit*.

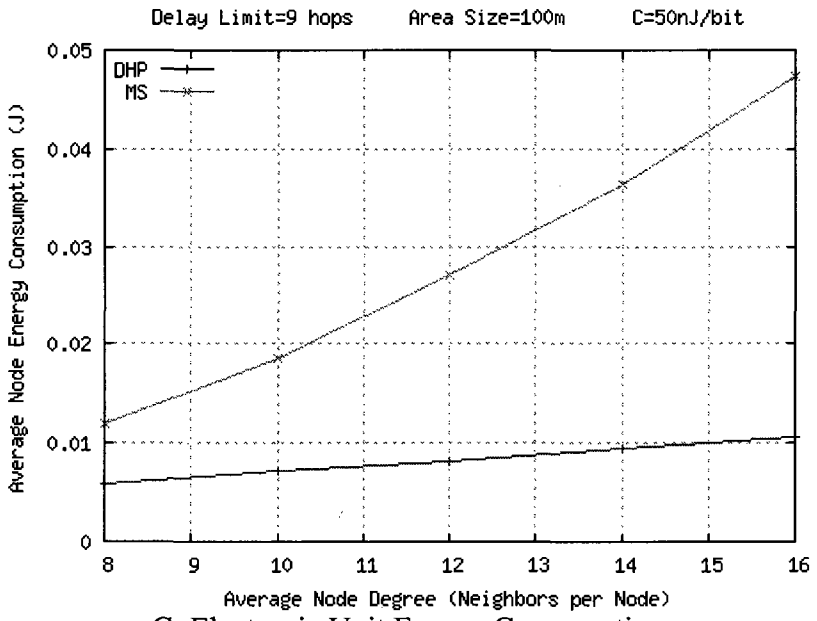
5.4.1 DHP VS. MS



C: Electronic Unit Energy Consumption

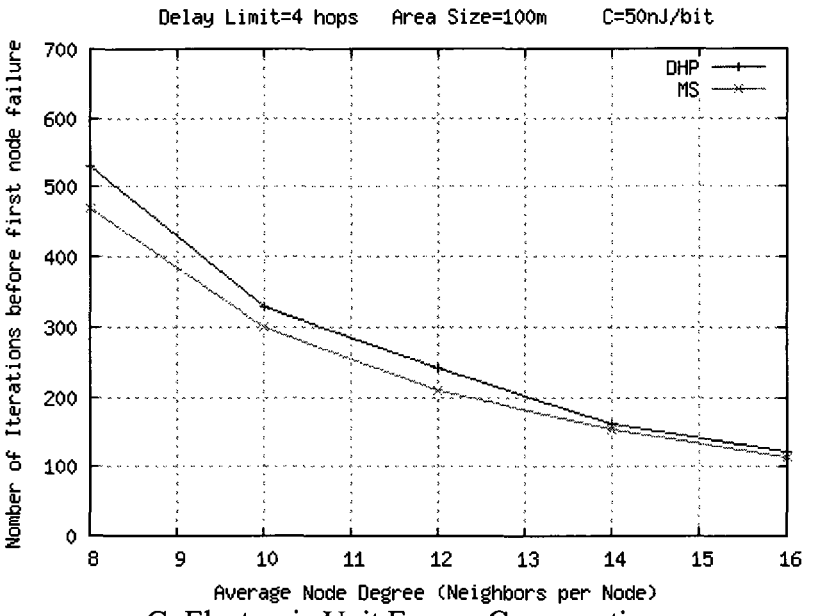
Figure 38. Average Node Energy Consumption by DHP and MS when Delay Limit=4 hops

Figure 38 and 39 indicate node average energy consumption by DHP and MS, where delay limits are 4 and 9 hops. In general, nodes are more energy saving when applying DHP than MS. When delay limit=4 hops (Figure 38), DHP spends less energy than MS by 25-40%. When delay limit=9 hops (Figure 39), the energy conservation by DHP increases to 60-75%. Both figures indicate the gap between two protocols increases with node degree. In denser networks, longer node communicating radius causes nodes can access more and further neighbors. For nodes by MS in start-up and speed-up states, they prefer neighbors with more geographic progress (normally further from nodes) as downstream nodes. Thus in denser networks most nodes in start-up and speed-up states send their reports to further neighbors. For nodes by DHP, they choose neighbors having hop progress closest to desired hop progress as downstream nodes. Far and close nodes have equal chance to be downstream nodes. Therefore, although in denser networks some nodes by DHP may change their parent nodes since they find new neighbors with more suitable hop progress, they are not the absolute majority, especially when delay limit is 9 hops.



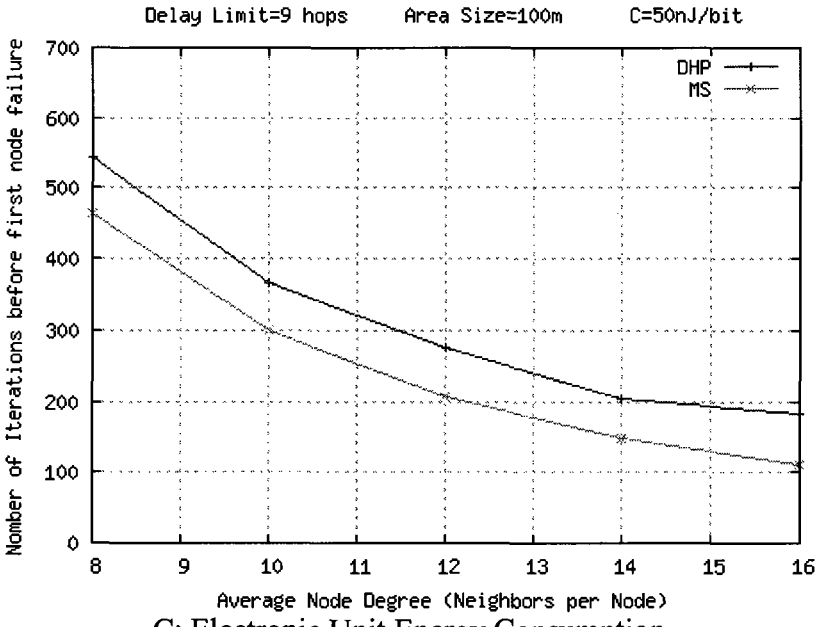
C: Electronic Unit Energy Consumption

Figure 39. Average Node Energy Consumption by DHP and MS when Delay Limit=9 hops



C: Electronic Unit Energy Consumption

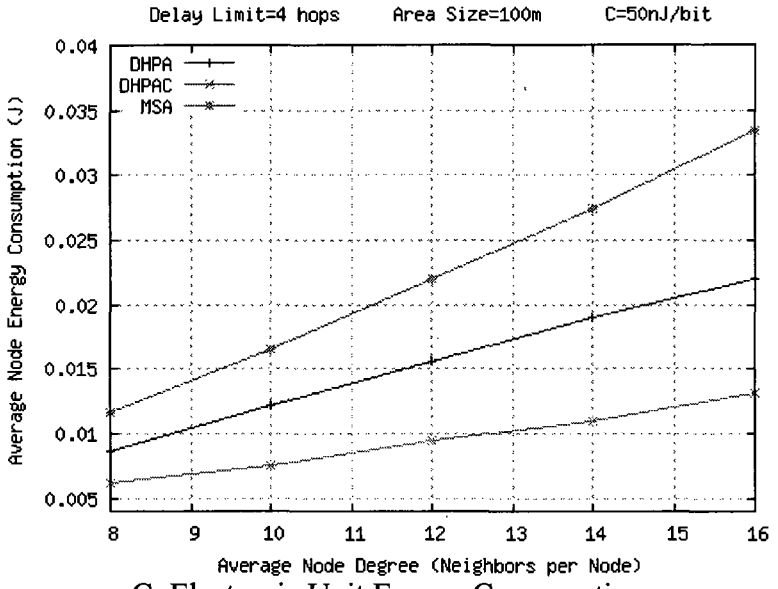
Figure 40. The Number of Iterations by DHP and MS when Delay Limit=4 hops



C: Electronic Unit Energy Consumption

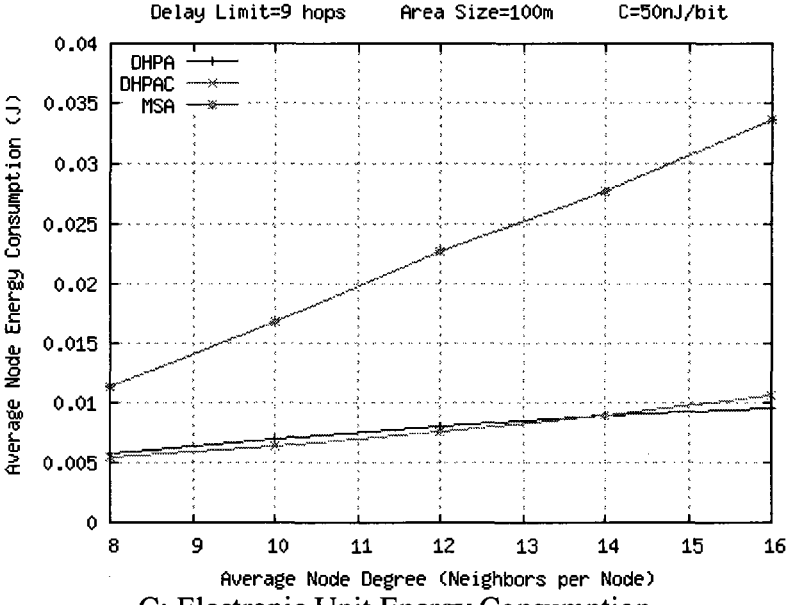
Figure 41. The Number of Iteration by DHP and MS when Delay Limit=9 hops

Figure 41 indicates when delay limit=9 hops, DHP extends 17-66% network life over MS: the denser network, the more network life extension. However, when delay limit=4 hops (Figure 40), DHP only leads to slightly longer (7-15%) network life than MSA.



C: Electronic Unit Energy Consumption

Figure 42. Average Node Energy Consumption by DHPA, DHPAC and MSA when Delay Limit=4 hops



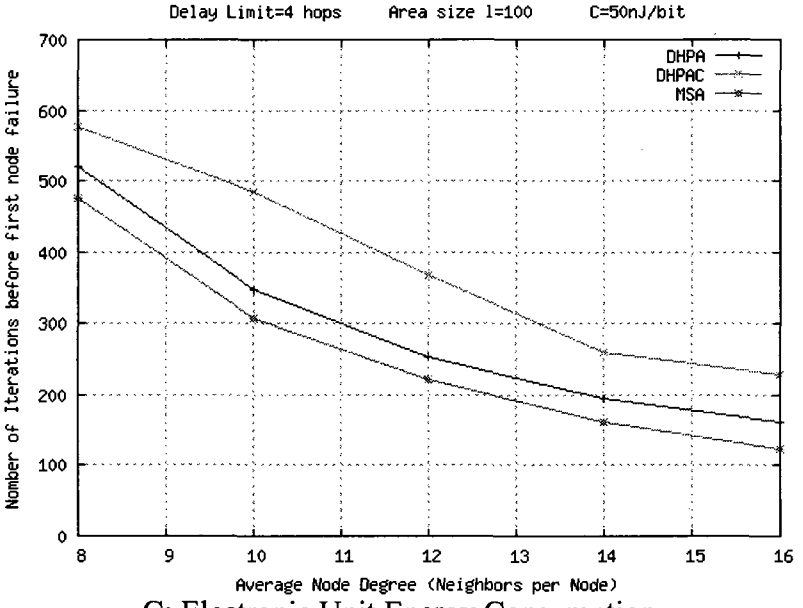
C: Electronic Unit Energy Consumption

Figure 43. Average Node Energy Consumption by DHPA, DHPAC and MSA when Delay Limit=9 hops

5.4.2 DHPA, DHPAC VS. MSA

Like MS, the change of node degree has great impact to MSA. Node energy consumption by MSA is tripled when node degree is doubled. DHPA and DHPAC

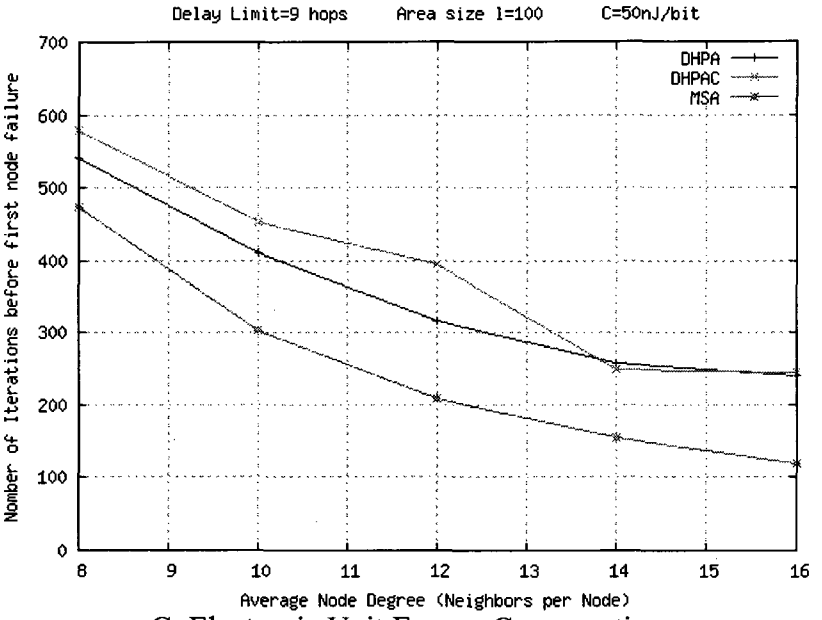
present similar change with the increase of node degree. Their energy consumption gain by 100% (in cases of delay limit=4 hops) and 50% (in cases of delay limit=9 hops) when node degree is doubled. Figure 42 shows the average node energy consumption by DHPA, DHPAC, and MSA when delay limit is 4 hops. In comparison with MSA, DHPA can save energy by 25-35%, and DHPAC can save energy by 45-60%. When delay limit is 9 hops (shown in Figure 43), DHPA and DHPAC have approximately same energy consumption: about 30-50% of that by MSA.



C: Electronic Unit Energy Consumption

Figure 44. The Number of Iteration by DHPA, DHPAC and MSA when Delay Limit=4 hops

From Figure 44 (delay limit=4 hops), we observe the network life caused by DHPA is close to that caused by MSA when node degree≤12: only 7-14% extension. If node degree≥14, the extension will reach 20-30%. As for DHPAC, its network life is 21-84% longer than that of MSA. Figure 45 (delay limit=9 hops) indicates when node degree is 14 and 16, DHPA and DHPAC have similar performance: respectively 65% and 103% network life extension over MSA. When node degree ≤ 12, in comparison to MSA, DHPAC achieves 22-85% longer network life and DHPA causes 14-52% longer network life.



C: Electronic Unit Energy Consumption

Figure 45. The Number of Iterations by DHPA, DHPAC and MSA when Delay Limit=9 hops

From above tables and figures, we can observe, in comparison with MS/MSA, our three data aggregation protocols (DHP, DHPA, and DHPAC) present significant advantages in saving energy and extending network lifetime when delay limit of data aggregation is not strict. If the assigned delay limit (4 hops) is lower than some nodes' UDG-distance to actor, our protocols are able to save considerable energy over MS/MSA, but not as much as that in cases with delay limit=9 hops. Besides, in certain scenarios, MS/MSA can lead to similar network life to DHP/DHPA. As for DHPAC, as the use of CDS, its network life is still obviously longer than that under MSA.

Chapter 6. Conclusions and Future Works

This thesis addressed data aggregation problem with emphasis on delay limit and energy consumption at nodes. We designed a localized delay-constraint power-efficient data aggregation framework to solve the problem. In this framework, we used the first-order radio model to measure energy consumption in transmissions, and counted delay by hop count (a reasonable assumption in low-rate networks).

To balance nodes' energy consumption and reduce overall network energy consumption, we first introduced a localized delay-bounded energy efficient data aggregation protocol, called desired hop progress (DHP) protocol. It first builds an energy optimal aggregation tree like LMST in [TKS], and then fixes it according to desired hop progress computed at nodes so that it does not violate delay limit. Considering the localized area coverage algorithm [GCSS] can save considerable energy while not affecting network monitoring area, we then proposed combining it with DHP protocol to improve the energy efficiency of the algorithm. Finally, to further reduce energy consumption, we suggested constructing a CDS [CS] over active nodes selected by [GCSS]. Then in data aggregation, CDS nodes apply DHP protocol, and the remaining active nodes just send their data to closest CDS nodes, which consider their non-CDS children delay in their DHP protocol implementation.

In comparison with the MS framework [MPGA], our simulation result showed that DHP can save 25-75% energy with up to 123% longer network life. On the other hand, in contrast to MS with area coverage algorithm (MSA), DHPA can complete data aggregations with up to 70% energy conservation and cause 123% longer network life; and DHPAC showed better energy efficiency, which can conserve 30-73% energy and its network life will be 133-250% of that of MSA.

In future works, we are going to extend our solutions to data gathering (data collecting without aggregation). Since data from different source nodes have to be sent separately, the LMST won't be energy optimal collecting structure any more. The algorithms proposed in this thesis will not be as energy efficient as they are in data aggregation (data collecting with aggregation). Hence, in such applications, every node should find energy and delay costs along the path to actor with minimal overall energy costs, and broadcast the information to its neighbors. Then according to this knowledge

of neighbors, each node can compute its desired hop progress in report transmitting and retransmitting.

Moreover, this thesis assumed that MAC and physical layer are ideal. When a realistic physical layer is used, packet reception probability between a pair of nodes has to be taken into account, which may change with transmission distance. In this situation, we should consider expected transmission times at any link in measuring delay and energy consumption.

Finally, when traffic is not low as we have assumed, DHPA and DHPAC can realize their dynamic energy balancing through periodically implementing area coverage algorithm and constructing CDS. As for DHP, it needs extra dynamic energy balancing scheme to maintain network energy balance in this situation. For example, when great differences in residual energy are detected by neighbors, they will enter a transient state to tune this gap, where greedy routing protocol or Greedy-Face-Greedy protocol (in networks with void areas) can be used.

Reference

[BAS] C. Buragohain, D. Agrawal, and S. Suri, Power aware routing for sensor databases, In Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp.1747- 1757, Mar. 2005.

[BCDC] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene, Energy-efficiency of IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives, in Proceedings of Design, Automation and Test in Europe (DATE '05), vol. 1, pp. 196–201, Mar. 2005.

[CBV] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, On Network Correlated Data Aggregation, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, vol.4, pp. 2571- 2582, Mar. 2004.

[CBVW] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer, Network correlated data aggregation with explicit communication: NP-completeness and algorithms, IEEE/ACM Transactions on Networking, vol. 14, pp. 41-54, Apr. 2006.

[CHZ] M. Chu, H. Haussecker, and F. Zhao, Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks, IEEE Journal of High Performance Computing Applications, vol. 16, pp. 293-313, 2002.

[CS] J. Carle and D. Simplot –Ryl, Energy efficient area monitoring by sensor networks, IEEE Computer Magazine, vol. 37, no. 2, pp. 40-46, Feb. 2004.

[D] E. W. Dijkstra: A note on two problems in connexion with graphs. In Numerische Mathematik, vol.1, pp. 269–271, 1959.

[DW] F. Dai and J. Wu, Distributed dominant pruning in ad hoc networks, IEEE International Conference on Communications (ICC'03), vol. 1, pp. 353-357, May 2003.

[F] G. Finn, Routing and addressing problems in large metropolitan-scale internetworks, ISI /RR- 87-180, Tech. Rep., Mar 1987.

[GCSS] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic, Localized Sensor Area Coverage with Low Communication Overhead, IEEE TRANSACTIONS ON MOBILE COMPUTING, vol. 7, no. 5, pp. 661-672, May 2008.

[H] P. Hall, Introduction to the Theory of Coverage Processes, pp. 59 and 181, 1988.

[HB] J. Hightower and G. Borriella, Location systems for ubiquitous computing, IEEE Computer, vol. 34, no. 8, pp. 57-66, Aug. 2001.

[HCB] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, Energy-Efficient Communication Protocol for Wireless Microsensor Networks, In Proceedings of the Hawaii Conference on System Sciences, vol. 2, pp. 4-7, Jan. 2000.

[HKB] W. Heinzelman, J. Kulik, and H. Balakrishnan, Adaptive Protocols for Information Dissemination in Wireless Sensor Networks, In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pp. 174-185, 1999.

[IGE] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, In Proceedings of the 6th annual international conference on Mobile computing and networking, pp. 56-67, Aug. 2000.

[KEW] B. Krishnamachari, D. Estrin, and S. Wicker, The Impact of Data Aggregation in Wireless Sensor Networks, In Proceedings of International Conference on Distributed Computing Systems Workshops, pp. 575-578, Nov. 2002.

[KKK] S.Kwon, J.Kim, and C.Kim, An Efficient Tree Structure for Delay Sensitive Data Aggregation in WSN, In IEEE 22nd International Conference on Advanced Information Networking and Applications, pp. 738-743, Mar. 2008.

[KKN] K. Kalpakis, K. Dasgupta, and P. Namjoshi, Maximum lifetime data aggregation and aggregation in wireless sensor networks, In Proceedings of the 2002 IEEE International Conference on Networking (ICN), pp. 685–696, Aug. 2002.

[LHS] N. Li, J.C. Hou, L. Sha, Design and analysis of an MST-based topology control algorithm, IEEE Transactions on Wireless Communications, vol. 4, pp. 1195-1206, May 2005.

[LR] S. Lindsey and C. S. Raghavendra. PEGASIS: Power Efficient Aggregation in Sensor Information Systems. In Proceedings of IEEE Aerospace Conference, vol. 3, pp. 1125-1130, Mar. 2002.

[LRS] S.Lindsay , C. S. Raghavendra , K.Sivalingam, Data Aggregation in Sensor Networks using the Energy Delay Metric, In Proceedings of the 15th International Parallel & Distributed Processing Symposium, pp.188, Apr. 2001.

[MFHH] S. Madden, M. J. Flanklin, J. M. Hellerastein, and W. Hong, Tag: a tiny aggregation service for ad-hoc sensor networks, In Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI), pp. 131–146, 2002.

[MPGA] T.Melodia, D. Pompili, V.Gungor, and I. Akyildiz, A distributed coordination framework for wireless sensor and actor networks, In Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), pp. 99–110, 2005.

[MSFC] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler, Supporting Aggregate Queries over Ad-Hoc Wireless Sensor Networks, In Proceedings of 4th IEEE Workshop on Mobile Computing and Systems Applications, pp. 49-58, 2002.

[OSGS] F.J Ovalle-Martíneza, I. Stojmenovic, F. García-Nocetti, and J. Solano-González, Finding minimum transmission radii for preserving connectivity and constructing minimal spanning trees in ad hoc and sensor networks, In Proceedings of the 3rd Workshop on Efficient and Experimental Algorithms, May 2004.

[RH] V.Roloplu and T.H. Meng, Minimum Energy Mobile Wireless Networks, IEEE Selected Areas in Communications, vol. 17, pp.1333-1334, Aug. 1999.

[RH2] R. Ramanathan and R. Hain, Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment, In Proceedings of 19th Annual Joint Conference of the IEEE Computer and Communications Societies, vol.2, pp. 404-413, Mar. 2000.

[SBK] B.Sundararaman, U.Buy, and A.Kshemkalyani, Clock Synchronization for wireless sensor networks: a survey, AD HOC Networks (Elsevier), vol. 3, no.3, pp. 281-323, May 2005.

[SL] I.Stojmenovic and X.lin, Power Aware Localized Routing in wireless networks, IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 11, pp. 1122-1133, Nov. 2001.

[SL2] I. Stojmenovic and X. Lin, Loop-free hybrid single-path/flooding routing an algorithms with guaranteed delivery for wireless networks, IEEE Transactions on Parallel and Distributed Systems, vol. 12, pp. 1023–1032, Oct 2001.

[SR] S. Singh and C. S. Raghavendra, PAMAS: Power Aware Multi-Access Protocol with Signaling for Ad Hoc Networks ACM Computer Communications Review, July 1998.

[SR2] R. Shah and J. Rabaey, Energy aware routing for low energy ad hoc sensor networks, In Proceedings of Wireless Communications and Networking Conference, vol. 1, pp.350-355, Mar. 2002.

[SS] I.Stojmenovi'c and M.Seddigh Dominating sets and neighbor elimination-based Broadcasting algorithms in wireless networks, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, 2001

[TKS] HO.Tan, I. Korpeoglu, and I. Stojmenovi, A Distributed and Dynamic Data Aggregation Protocol for Sensor Networks, 21st International Conference on Advanced Networking and Applications(AINA'07), pp. 220-227, May 2007.

[UZ] http://www.ubec.com.tw/product/uz2400.html (UZ2400 Data Sheet)

[WL] J. Wu and H. Li, On calculating connected dominating sets for efficient routing in ad hoc wireless networks, ACM international Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM 1999), pp. 7-14, 1999.

[YHE] W. Ye, J. Heidemann, and D. Estrin, Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks, In IEEE/ACM Transaction on Networking, vol. 12, pp. 493-506, June 2004.

[YKP] Y. Yu, B. Krishnamachari, and V. K. Prasanna, Energy-latency tradeoffs for data aggregation in wireless sensor networks, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 255, Mar. 2004.

[ZMD] http://www.zmd.de/pdf/ZMD44101_DataSheet_03-2005.pdf

[ZPG] Q. Zhu, M. Parsa, and J. Garcia-Luna-Aceves, A source-based algorithm for delay-constrained minimum-cost multicasting, in Proceedings of 14th Annual Joint Conference of the IEEE Computer and Communications Societies, vol.1, pp. 377–385, Apr. 1995.

[ZSS] Y. Zhu, K. Sundaresan, and R. Sivakumar, Practical limits on achievable energy improvements and useable delay tolerance in correlation aware data aggregation in wireless sensor networks, in Proceedings of SECON 2005, Sep. 2005.