Energy-efficient Data Aggregation using Realistic Delay Model in Wireless Sensor Networks

by

Shuo Yan

Thesis submitted to the
Faculty of Graduate and Postgraduate Studies
In partial fulfillment of the requirements
For Masters degree in
Computer Science

School of Electrical Engineering & Computer Science
Faculty of Engineering
University of Ottawa

Acknowledgements

I would like to give my sincerest gratitude to my supervisor Prof. Amiya Nayak, for his continuous guidance and support not only in academic level but also in my personal life. It is an honor for me to work with him and my co-supervisor Prof. Ivan Stojmenovic. I would like to thank Dr. Xu Li for his advices and instructions. This thesis would not have been possible without the great support of my family and friends. It is a pleasure to thank them who made this thesis possible.

Abstract

Data aggregation is an important technique in wireless sensor networks. The data are gathered together by data fusion routines along the routing path, which is called *data-centralized routing*. We propose a localized, *Delay-bounded and Energy-efficient Data Aggregation* framework (DEDA) based on the novel concept of DEsired Progress (DEP). This framework works under request-driven networks with realistic MAC layer protocols. It is based on localized minimal spanning tree (LMST) which is an energy-efficient structure. Besides the energy consideration, delay reliability is also considered by means of the DEP. A node's DEP reflects its desired progress in LMST which should be largely satisfied. Hence, the LMST edges might be replaced by unit disk graph (UDG) edges which can progress further in LMST. The DEP metric is rooted on realistic degree-based delay model so that DEDA increases the delay reliability to a large extent compared to other hop-based algorithms. We also combine our DEDA framework with area coverage and localized connected dominating set algorithms to achieve two more resilient DEDA implementations: A-DEDA and AC-DEDA. The simulation results confirm that our original DEDA and its two enhanced variants save more energy and attain a higher delay reliability ratio than existing protocols.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Sensors are multi-functional devices that are able to respond to physical stimuli and communicate via wireless links. They are small in size and low in cost with limited resources such as CPU power, main memory and battery lifetime. Sensors are usually dropped in an area of interest. Once deployed, they may self-organize into a multi-hop ad hoc network and operate autonomously: sampling the environment, processing the measurements and reporting the results. After sampling and processing, they will communicate their reports to other sensors or actors/sinks directly according to different routing protocols. During the routing process, different reports might amalgamate into a single one at intermediate nodes as per different underlying data aggregation routines. In this way, the number of reports is reduced in order to save energy and bandwidth.

Generally speaking, there are two kinds of data aggregation protocols: centralized and localized. Should each node be provided with global network topology information, and the network is static over time, a centralized solution can be applied for data aggregation which might lead to an optimal aggregation structure.

However, if global knowledge is not available or the network is highly dynamic, centralized solutions will not work as expected because obtaining such knowledge is costly and the recommended solutions will be ephemeral. On the other hand, localized solutions only require the information available from neighbors to make routing decisions. Although this type of protocols yields sub-optimal results, they are more efficient for large-scale networks and thus more suitable for distributed environments.

1.2 Problem Statement

We consider a request-driven, large-scale, static wireless sensor network (WSN). Nodes (sensors and sinks) are aware of their own positions by either attached GPS devices or existing localization algorithms. They share their location information with one-hop neighbors by periodic "hello" messages. Sinks spontaneously flood the network with a message carrying their own location, and each sensor retransmits only when the flooding message comes from the nearest sink it has seen. Therefore, all sensors get to know their closest sink.

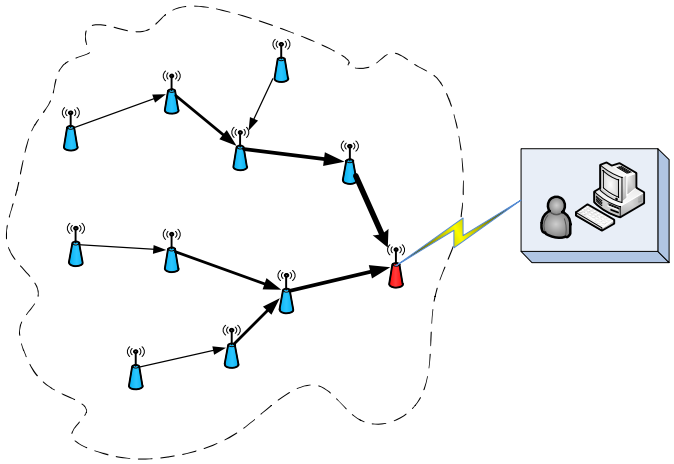


Figure 1.1: Data aggregation with a single sink.

Each sink broadcasts a request message in the entire network to activate only a group of sensors in a subregion. Sensors report only to the closest sink upon receiving request message. By doing so, the network is virtually partitioned into a number of sub-networks, each containing a single sink and sensors reporting to it. This allows us to focus on a single-sink scenario(Fig. 1.1) without loss of generality.

Sensor reports are of the same size (depending on the type of data requested) and not prioritized. They are treated equally for transmission and data packets are all of the same predefined size. So, a single report may have to be split into multiple data packets. In this case, reporting delay is linearly proportional or equivalent to packet delay. For simplicity, we consider that a report is equivalent to a packet and use these terms interchangeably.

Sensors are equipped with an omni-directional antenna and able to communicate directly if their distance is smaller than the maximum transmission radius R. In other words, the network is modeled as a unit disk graph (UDG). Each edge (communication link) is associated with a cost, defined as the minimum transmission power for this edge. It is computed using the first order radio model [1], $e(d) = \beta d^{\alpha} + c$, where $\beta > 0$ is the transmission amplifier, d is the transmission distance (edge length), α is a signal attenuation factor, and $c > 0$ is the energy for running electronic circuit. Sensors can adjust their transmission power so that they send reports along a selected edge using the least possible power, which equals the associated cost of the edge. Moreover, each sensor is aware of its own residual energy. If the remaining energy level falls below a threshold E_{th}, it will transmit a "quit" message to neighbors and switch off for energy saving.

The delay model is based on different MAC models and real-time traffic conditions. Different sorts of delay have been taken into account, such as processing delay, propagation delay, transmission delay and queuing delay [2]. If the traffic

is low, the delay is relatively the same for each node. Thus, the path delay is approximately linearly proportional to the number of hops conveying the packet. But when the traffic is high, the queuing delay varies according to different delay models.

The research goal is to develop, based on the above network model, a localized energy-efficient data aggregation framework that satisfies a given set of delay constraints.

1.3 Motivation

Sensors are normally powered by low-energy batteries. Manual replacement or recharge of sensor batteries is often infeasible due to operational factors such as inaccessibility to the sensory field or tight maintenance budget. It is highly desirable to prolong the lifetime of the network as a whole by minimizing and balancing energy usage among individual sensors. In critical real-time scenarios such as disaster management, emergency rescue and battlefield surveillance, sensor reports are often required to arrive at a particular sink with a bounded delay so that a timely response to detect events can be orchestrated.

Existing data aggregation algorithms usually require centralized control and emphasize mostly on energy efficiency. They seldom consider the delay requirement. The few known localized, delay-bounded, power-aware algorithms have major drawbacks in energy saving as well as in delay modeling and thus have limited effect both on prolonging the network lifetime and on meeting the delay requirement. Motivated by the insufficiency and incompleteness of previous works, we address power optimality in data aggregation with respect to a given delay constraint.

1.4 Objectives

Based on the motivation mentioned above, our framework aims to solve the data aggregation problem with optimal energy consumption and balanced distribution. In addition to minimize and distribute the energy usage, the delay bound should be largely satisfied as well under realistic network conditions.

1.5 Contributions

We design a localized aggregation framework termed *DEDA*, which is based on realistic MAC layer for the above stated data aggregation problem. We model the network as unit disc graph (UDG), where each edge (communication link) is weighed by the minimal transmission cost needed for that edge (subject to its length). A localized minimum spanning tree (LMST) [3] sub-graph is spontaneously constructed. At the beginning of each data aggregation round, a shortest path (thus energy-efficient) tree is built on demand, with no concern about delay, as the initial data aggregation tree over the LMST, with the sink/actor as its root.

The tree will be used without modification if it generates an acceptable reporting delay during data aggregation. Otherwise, its structure is changed by replacing LMST edges with selected UDG edges. The assessment and adjustment are done locally at each node "on the fly", in accordance with a novel concept of DEsired Progress (DEP), which is defined as the ratio of potential delay to remaining lifetime of a report. The final data aggregation tree yields an approximately minimal overall energy consumption and balanced energy usage among sensors, while respecting the given delay constraint.

We further propose to integrate DEDA with a localized area coverage algorithm [4] and a localized connected dominating set (CDS) formation algorithm [5] for en-

ergy efficiency improvement. This gives us two new algorithms, respectively called *A-DEDA* and *AC-DEDA*. Through extensive simulations we comparatively evaluate our DEDA framework (and its two variants) with other competing algorithms. Simulation results indicate that DEDA and its two offspring are dramatically more efficient and achieve a higher delay reliability ratio under different network configurations.

1.6 Organization

This thesis is organized as follows. We give a literature review about previous related work in Chapter 2. Then we discuss different delay models and propose our DEDA framework in Chapter 3. Afterwards, we present the two DEDA variants, i.e., A-DEDA and AC-DEDA, in Chapter 4. Then, we evaluate the proposed algorithms through extensive simulation in Chapter 5 and draw final conclusions in Chapter 6.

Chapter 2

Literature Review

In this chapter, we will review the existent *data aggregation* techniques for wireless sensor networks. Data aggregation is also known as *data fusion*, given that it collects multiple data reports and combines them together. The data are processed before being forwarded to the next hop, so the number of packets are largely reduced and thus a large deal of energy is saved. Data aggregation routines are usually applied based on the user's requirements.

Data aggregation routines are categorized into two major streams, i.e. *lossless* vs. *lossy* ones. For the lossy stream, some straightforward functions such as average, minimum, maximum and median have been proposed in [6, 7] and some redundancy suppression approaches have been put forth in [8, 9]. It is easy and energy-efficient to implement these lossy data aggregation procedures but they can only provide general information to the user. However in some cases, the user needs more details on the data in the monitoring area.

A spatially-decaying aggregation approach is presented in [10]. The authors observe that the data value is usually decayed with its distance to the observing location. In this approach, weights are assigned to the data items. The weight is

based on a decay function which is a non-increasing mapping of the distance.

Not only is the data aggregation routine the metric to classify different categories of data aggregation approaches, but also some other metrics have been considered such as data storage location, aggregation direction and single/cross layer.

In this review section, the most popular metric which is based on data aggregation structure will be used to compare all these data aggregation approaches. There are many data aggregation structures on wireless sensor networks. We classify them into four categories: cluster-based, tree-based, chain-based and grid-based ones. We will go through these approaches in Sec. 2.1.

After reviewing these protocols, we will survey some delay-bounded data aggregation algorithms in Sec. 2.2. Then, area coverage algorithms will be examined in Sec. 2.3 as well as some backbone construction methods in Sec. 2.4.

2.1 Data Aggregation Protocols based on Network Structures

Network structures play an important role in data aggregation. Some data aggregation approaches are based on classic routing protocols and thus a tree-based structure is usually constructed during the aggregation process. In order to locally fuse data to save energy, sometimes the sensors are locally grouped together into a cluster. A node is selected as cluster head to aggregate data from all nodes in its cluster. Then, these amalgamated data reports are delivered to a sink. This process can be locally carried out so that energy consumption is reduced and potential collisions are avoided.

Some other approaches have also been presented such as chain-based and grid-

based structures. We will review each of them in this subsection and describe their advantages and disadvantages.

2.1.1 Cluster-based Data Aggregation

Sensor are devices with limited energy, so their energy will be exhausted quickly if they directly transmit data to the sink. In order to save energy, some intermediate sensors should be selected as data relays. The cluster-based structure picks a cluster head for a group of sensors. The cluster head fuses incoming data from all sensors in its cluster and either transmits directly to the sink or leans upon a relay path made up by other cluster heads. Several cluster-based approaches have been proposed in literature.

2.1.1.1 LEACH - *Low Energy Adaptive Clustering Hierarchy*

LEACH [11, 12, 13] arranges nodes into clusters, each having only one node chosen as the cluster head. All non-cluster head nodes transmit their data to the cluster head, which performs aggregation functions on the data and transmits the final result to the sink.

Therefore, being a cluster head node consumes much more energy than serving as a "regular" node. In order to avoid fast energy depletion and keep energy balance for all nodes, LEACH applies a cluster head selection algorithm which chooses different nodes as cluster heads in turn. In this way, the energy load of being a cluster head is evenly distributed among the nodes.

Two phases can be identified in LEACH: the *setup* phase and the *steady-state* phase.

In the setup phase, the nodes first locally elect the cluster heads guided by a cluster head selection algorithm. Afterwards, the cluster heads broadcast adver-

tisement (ADV) messages to let regular nodes know about their roles. According to the signal strength of the messages received, each regular node will choose the closest cluster head in order to reduce communication energy. After deciding to join a particular cluster head, the regular node will notify its selected cluster head about its decision by a "join request" (Join-REQ) message. Now the cluster head gets to know all nodes in its cluster and also all regular nodes have chosen their cluster heads. Then, the cluster heads individually make up a TDMA schedule for nodes in this cluster to avoid communication collisions and broadcast this schedule to all of their cluster members. The nodes in the cluster will follow this schedule when transmitting data to their cluster head.

The actual data aggregation process starts from the steady-state phase. Time is divided into lots of frames in which every node has its own time slot to transmit data to the cluster head. If one regular node detects that its time slot has not arrived yet, it can switch into idle mode and therefore spend less energy. So, this TDMA schedule ensures that there are no collisions within the clusters while at the same time preserves energy efficiency. After cluster heads have received all the data from the cluster members, they forward that information to the sink using a single direct transmission. The Direct-Sequence Spread Spectrum (DSSS) is used to avoid interference among different clusters.

Even though LEACH can balance and save energy via its cluster head selection algorithm and is able to avoid collisions through its intra-cluster TDMA schedule, it has some limitations and preconditions. LEACH assumes that all sensors have the ability to reach the sink if needed for data transmission. In other words, every sensor has the capability to become a cluster head which performs data aggregation. LEACH also assumes that all nodes always have data to send. These assumptions might not be valid in real situations and one must also bear in mind

that in dynamic environments the network topology often changes. Hence, more control message overhead in the setup phase is required to reconstruct topology knowledge for every node. For example, if one node moves far away from its cluster head, it needs to switch to a new cluster head in order to reduce its communication energy expenditures.

A centralized algorithm entitled LEACH-C [13] is proposed to optimize the cluster head selection scheme. In LEACH-C, the steady-state phase is the same as in LEACH. Only the setup phase is modified. Now, the sink will receive location information from all nodes, select cluster heads and regular nodes and broadcast the result to every node. This centralized algorithm will ensure the nodes' placement and number of clusters.

In [14], the authors propose a similar algorithm to LEACH. Every node becomes the cluster head with probability p and broadcasts the result to nodes within k hops. The nodes that received this message will join the closest cluster head. If some regular nodes receive no messages, they will become cluster heads as well. Then the cluster members send data to their cluster head either directly or indirectly. There is a timeout t during which the data should be aggregated by the cluster head. Because of this timeout, there is no need for a clock synchronization method like in LEACH. At last, the authors analyze the optimal parameters for this algorithm to ensure minimization of the energy consumption.

2.1.1.2 COUGAR

In [15, 16], the authors propose a query layer for wireless sensor networks. Users or clients issue queries to the query layer, each generating an associated query plan. Considering the query "How many empty tables are there in the SEECS

cafeteria?"[1], some tables might organize themselves into a cluster and send an empty/full state notification to their cluster head, which will in turn compute the number of empty tables in the cluster and forward the result to the sink. All these query plans are implemented through *flow blocks*, whose task is to collect information on relative sensors and have some nodes do the computation.

The intra-cluster routing in COUGAR allows multiple hops as opposed to LEACH's TDMA schedule. The AODV [17] routing protocol is extended in order to be applied as an intra-cluster routing method. With the goal of increasing packet merging or partial aggregation, a child prediction approach based on history information is presented. The parent node will expect data from its child in the last round. If this prediction fails, either a *timer* is used to detect such situation or the child will send a *notification* packet to its parent. This bi-directional prediction approach works well in COUGAR.

The disadvantage here is similar to LEACH's, i.e. a dynamic network topology will entail a large overhead for cluster reconstruction and cluster head reelection.

2.1.1.3 HEED - *Hybrid Energy-Efficient Distributed clustering*

Another clustering algorithm is introduced in [18] which is called HEED. It has two built-in parameters: residual energy percentage and average minimum reachability power (AMRP), which means the average minimum power level required by all M nodes in a cluster to reach the cluster head u. Assume $MinPwr_i$ denotes the minimum power level required by a node v_i ($1 \leq \text{i} \leq \text{M}$), then $AMRP = \frac{\sum_{i=1}^{M}(MinPwr_i)}{M}$. During the cluster head election phase, each node computes a probability to be a cluster head equal to $C_{prob} \times \frac{E_{residual}}{E_{max}}$. If this probability is 1, it will become the cluster head; otherwise, it is just a candidate. An undecided node with a cluster

[1]SEECS is the acronym for School of Electrical Engineering and Computer Science.

head in its cluster range will choose the one with smallest cost as a tentative cluster head, thus becoming "covered" by it. After each iteration, any "uncovered" node will double its probability until it reaches 1.

HEED assumes that each node has a fixed number of transmission power levels. However, it achieves a better performance than gen-LEACH in extending the network lifetime based on the simulation study in [18].

2.1.2 Tree-based Data Aggregation

Besides the cluster-based data aggregation structure, the tree-based structure is also very suitable for data aggregation. In a tree, data can be fused at intermediate nodes with the aid of good data aggregation routines, thus saving a great deal of energy. The data aggregation tree is usually rooted at the sink and all the other sensors are arranged as nodes in this tree. So, constructing an energy-efficient data aggregation tree is critical to reduce energy consumption during the data collection process.

2.1.2.1 TAG - *Tiny AGgregation*

TAG [7] is a data aggregation framework for TinyOS. There are two phases for TAG: a *distribution* phase, in which data queries are distributed to nodes in the network, and a *collection* phase, where the data aggregation reports are continually routed up from children to parent nodes.

During the first phase, TAG creates a tree structure by broadcasting messages in the network. The sink broadcasts a packet asking nodes to organize themselves into a routing tree. In each packet there is a field specifying the level (or distance from the root), of the sender node. Any node without an assigned level that hears this message sets its own level to be the level included in the packet plus one. It

also chooses the sender of the message as its parent, through which it will route messages to the root. Each node then rebroadcasts the received packet including its own identifier (ID) and level. This process goes on until all nodes have been assigned an ID and a parent. This tree maintenance process will periodically run (initiated by the sink node) to keep the tree structure updated.

After the construction of the tree, queries are sent along the structure to all nodes in the network. Their format is similar to SQL grammar's SELECT. During the second phase, a parent node must wait for data from all its children before it can report to its own parent. In order to achieve this, a time period called *epoch* is specified to differentiate each round of data aggregation. An epoch is divided into different intervals based on the number of levels in the network or the depth of the aggregation tree. Each node will occupy one of the time intervals to perform aggregation. This interval will be split so that all of its children are able to send up aggregated data to it. So during the specified time, the parent node will listen to data reports from one of its children, and send the amalgamated data to its parent when the specified time by its parent elapses.

There are many advantages for TAG. During one data aggregation round, each node will transmit the same number of messages, thus causing a balanced energy consumption throughout the network and prolonging its lifetime. By dividing time into different epochs and intervals, the nodes can avoid MAC layer collisions and switch into sleeping mode when it is not their turn to perform data aggregation. Furthermore, the periodical reconstruction of the tree layout maintains the topology consistent in the event of node failure.

2.1.2.2 PEDAP - *Power Efficient Data gathering and Aggregation Protocol*

In [19] a near-optimal, minimum-spanning-tree-based routing protocol is proposed. The radio model used is $C_{ij}(k) = 2 * E_{elec} * k + E_{amp} * k * d_{ij}^2$, where $C_{ij}(k)$ is the cost of transmission between node i and node j, E_{elec} is the energy consumption for circuitry, E_{amp} is the transmission amplifier and d_{ij} is the distance between node i and node j. PEDAP computes the minimum spanning tree (MST) of the network through Prim's algorithm and aggregates data along this tree. By doing so, the total energy consumption during one round is minimized and data fusion can be applied in intermediate nodes of the MST.

Aiming at solving the energy balance problem, a PEDAP variant is put forward, which is called PEDAP-PA (power aware). Here, $C_{ij}(k)$ is changed to $(2 * E_{elec} * k + E_{amp} * k * d_{ij}^2)/e_i$, where e_i is the residual energy of node i. The cost of the link is thus inversely proportional to the remaining energy of the sender node. So the Prim's algorithm will derive a MST based on both distance and residual energy.

PEDAP's disadvantage is obvious: the algorithm needs to possess global knowledge of the locations of all nodes. It operates in a centralized manner, with the sink exercising as main controller where the routing paths are calculated. Yet the simulation results demonstrate that it improves the network's lifetime compared to LEACH and PEGASIS. Moreover, PEDAP-PA also achieves the best performance for balancing energy among all nodes.

2.1.2.3 EADAT - *Energy Aware Data gathering and Aggregation Tree*

The authors in [20] propose a new energy-aware protocol (EADAT). The sink broadcasts the message all over the network to construct a data aggregation tree. This control message, denoted by *msg*, includes 5 fields: *ID, parent, power, status,*

hopCnt. The packet holds information about the node's ID, its parent in the aggregation tree, its residual power, it's status in the tree and the number of hops towards the sink. There is also a timer T which is inversely proportional to the residual energy of the node. After monitoring the channel for T idle time units, this node will start to broadcast its *msg*, which includes its ID, selected parent, updated residual power and status and increased hopCnt by one.

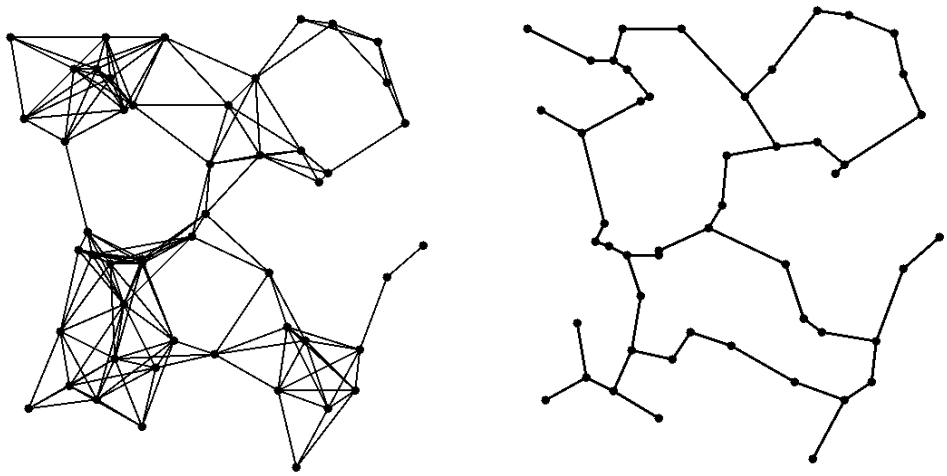
During this span of T time units, if the node receives a control message, it records the parent with higher residual power and less hops to the sink. Based on this scheme, a node with lower energy will have a higher probability to become a non-leaf node in the aggregation tree.

To maintain the tree layout up-to-date, there is a residual power threshold P_{th} associated with each sensor. When the residual power of a sensor falls below P_{th}, it periodically broadcasts help messages for T_d time units and then shuts down its radio. A child node, upon receiving a help message, switches to another candidate parent if it is able to, otherwise it enters into a "danger" state. If a node in the danger state receives a hello message from a sleeping node u with shorter distance to the sink, it invites u to join the tree.

The simulation results show that EADAT maximizes the residual energy of alive nodes and that the network lifetime will linearly rise with the network density.

2.1.2.4 LMST-based Data Gathering and Aggregation Protocol

In [21] a MST-based topology control algorithm, called *Local Minimum Spanning Tree*(LMST), for data aggregation purposes is put forth. MST is a topology connecting all nodes in the network where the link cost can be any metric such as distance between two nodes, bandwidth of the link or its energy consumption. But the construction of MST like in PEDAP needs global topology information,



(a) Original graph (b) LMST graph

Figure 2.1: LMST construction.

which is not suitable in a distributed environment. However, LMST can be locally constructed with only two-hop knowledge as shown below.

Each node constructs a MST in the subgraph including all its one-hop neighbors and itself by Prim's algorithm. Each node u selects an edge (u, v) where v is the direct neighbor in this MST. All such directed edges make up the $LMST^{+}$. If we only include the edge (u, v) when both u and v select this edge out of their MSTs, all such directed edges make up the $LMST^{-}$. We can treat these bi-directed edges as undirected edges. An LMST construction example is shown in Fig. 2.1.

The authors in [22] propose an LMST-based data aggregation tree for wireless sensor networks. Because this protocol is anchored in LMST, all nodes must build the LMST first as described in [21]. Then the sink node floods the LMST with a *ROUTE-DISCOVERY* packet in order to find routes from all nodes toward itself. This packet includes the sequence ID, distance which is to estimate the cost of reaching the sink, where the cost of an edge is expressed as its power consumption, and a neighbor list including all LMST neighbors of the sender.

Upon receiving a new *ROUTE-DISCOVERY* packet, the sensor node compares the neighbor list in the packet with its own LMST neighbor list. This packet will be ignored or not based on different LMST versions ($LMST^+$ or $LMST^-$). An update is necessary if a new route with a smaller cost is detected or a new *ROUTE-DISCOVERY* packet with a larger sequence ID is received. Should a sensor node choose to update its table, it broadcasts a *ROUTE-DISCOVERY* packet to all of its LMST neighbors informing the new distance to the sink. It also includes its own LMST neighbor list in the packet instead of the one it received. Additionally, if the node chooses to update its table and the parent node is changed, the node informs the old and new parent nodes about the changes so that they can update their children lists accordingly. As this process is completed network-wide, every sensor will know its parent node, to which its data packet will be sent and its children nodes, from which data packets will be received. In the resulting data aggregation tree, the energy consumption during data fusion is approximately the one with highest energy-efficiency.

2.1.3 Chain-based Data Aggregation

The chain-based structure implies that the nodes are organized in a linear structure and thus each intermediate node has two neighbors along this chain. Data aggregation can be achieved through the intermediate nodes of the chain. When the energy efficiency and bandwidth usage requirements are more important than the delay requirement, the chain structure is a good solution for fusing data along a linear path ending at a sink.

2.1.3.1 PEGASIS - *Power-Efficient GAthering in Sensor Information Systems*

In [6], the authors propose a power-efficient method called PEGASIS to aggregate data in a wireless sensor network. There is a chain leader among all nodes in the chain which is responsible for collecting data and sending them to the sink. The chain leader is the only node that can transmit to the sink and is reelected for each round. In this way, energy consumption can be balanced among all nodes.

The chain construction can be realized in a centralized manner or in a distributed fashion. In both cases, global knowledge of the network topology is necessary. The nodes will take turns to become chain leader as follows: node i is the leader at round i. The chain construction is started from the farthest node to the sink. Then the closest neighbor of this node is chosen as the next hop and continues this process. Finally, the leader will receive the data from all other nodes and send them to the sink. A data packet is aggregated along this path until it reaches the leader node.

The obvious disadvantage is that each node needs to have a complete view of the network layout and must have the ability to transmit directly to the sink. Therefore, this algorithm is not suitable for a dynamic network topology. Additionally, should any node in the chain fail, all the aggregated data at that point will be lost. Thus the robustness needs to be improved so as to guarantee the delivery of all collected data to the sink.

2.1.3.2 Some Other Chain Construction Algorithms

In [23], a new metric energy*delay is introduced to add delay into consideration. The authors also propose a new structure based on PEGASIS, which is a binary combining scheme using CDMA. It performs parallel communication in

about $\log(n)$ delay time units, where n is the number of nodes, and spends a little more energy than PEGASIS. Each node that received a message will continue to the next level. This structure will reduce the delay time to $\log(n)$.

The authors in [24] also present a multiple-chain structure. This scheme divides the whole sensing region into four subareas centered at the midpoint of the sensing region. In each subarea, there exists a linear chain which is shorter than a single chain in PEGASIS. Finally, the four chains come together at the center node (leader), which is responsible for transmitting to the sink. By this multiple-chain scheme, the distance or delay is largely reduced with negligible increase in energy expenditures.

2.1.4 Grid-based Data Aggregation

For mobile sinks, a grid-based data aggregation scheme, named *Two-Tier Data Dissemination* (TTDD), is brought forward in [25]. Before the data query of sink, a grid construction process takes place in which each source will use greedy geographical forwarding method to build a grid structure with itself as the center. Each crossing point is referred to as *dissemination point* and is associated with the closest node, which is called *dissemination node*.

After the grid construction process is over, there are three phases for data aggregation.

1. Query Forwarding: The sink floods the query all over the local grid to search for the closest dissemination node which has the path to the source node along the crossing points.

2. Data Forwarding: When the source receives a query packet, it will send back data along the path from which the query is forwarded.

3. Trajectory Forwarding: The closest dissemination node to the sink will forward data to the primary agent which is appointed by the sink. Afterwards, this primary agent will forward data to the sink. If the sink is moving during this process, it will choose some intermediate agents to forward data from the primary node back to itself.

It is easy to conclude that there are two tiers in TTDD. The upper tier is shaped by the dissemination points, which stand as the grid's backbone. In the lower tier, a mobile sink sends a query to, and receives data from, its nearest dissemination point through intermediate and primary agents. By doing so, when a sink moves to another grid, it can quickly connect to the grid structure and hence the access delay is largely diminished.

2.2 Delay-bounded Data Aggregation

The protocols discussed above mainly focus on energy efficiency in order to achieve a long network lifetime. With the development of delay-sensitive applications, however, the delay limit requirement as one metric in the Quality of Service (QoS) is becoming more and more important. So in this section, we will review at short length some delay-bounded data aggregation algorithms.

The authors in [26] examined two kinds of routing schemes: address-centric (AC) and data-centric (DC). For the latter, three suboptimal schemes (CNS, SPT and GIT) are further studied. The results confirmed that data-centric protocols are much more energy-efficient than traditional address-centric protocols. The delay effect involved in data aggregation is also studied, but the authors did not propose any delay-bounded data fusion scheme.

In [27], a structure called *delay-bound minimum-degree spanning tree* (DB-

MDST) is presented. MDST is known as the structure achieving the optimal energy balance which is fair to every node. Delay-bounded MDST considers the height of the tree as a delay estimate, which should not exceed the delay limit of application by edge changes. But this protocol requires some global knowledge and is still dependent on the "hop count" delay model, which might not reflect the realistic delay conditions.

In [28], the authors introduced a scheduling method to have sensors switch among different states and analyzed the energy consumption. The authors in [29] studied the delay bound for data aggregation. But TDMA MAC layer assumption is applied in both research studies.

The tradeoff between energy and latency has been studied in [30]. Their approach is to minimize the sum of the energy consumption subject to a latency constraint. Yet before aggregating data, every sensor might need to wait for its children, which introduces some additional delay.

In [31], the authors came up with a distributed multi-state data aggregation framework, referred to as MS, which takes into account a given delay bound and pursues minimum energy consumption. In MS, a power-aware tree is initially built with no concern about delay, and then dynamically changed based on the measured delay in ongoing traffic. Specifically, the actor maintains a reliable ratio, which is defined as the ratio of the number of unexpired reports to the total number of reports, and feeds it back to the sensors. The tree is locally adjusted if the ratio is greater than an upper threshold or less than a lower threshold. In the former case, each sensor takes as parent the closest neighbor that is closer to the actor than itself for minimizing transmission power. In the latter case, it chooses the neighbor geographically closest to the actor so as to minimize path length and reduce delay.

However, MS has its own weakness compared to our DEDA framework: it will

consume more energy and cannot satisfy delay requirement under realistic MAC layer. We will provide detailed comparative analysis in Chapter 5, when presenting our simulation results.

2.3 Area Coverage Algorithms

In this section, we will review some area coverage algorithms that will be combined with our data aggregation framework. The sensors are deployed in an area once. We assume that recharging the batteries is inconvenient/infeasible. So, shutting down some sensors which are not working is a very helpful method to prolong the lifetime of the network. There are several requirements for any area coverage algorithm such as full coverage, energy efficiency and network connectivity. At first, we will review a classic algorithm called PEAS which does not ensure full coverage.

In [32], the authors describe an algorithm entitled *Probing Environment and Adaptive Sleeping* (PEAS) which works as follows. At first, all nodes are sleeping and then wake up after a random time span. A *PROBE* message including a probing range R_p will be broadcast by an active node. All other active nodes within this R_p range will receive the message and send back a *REPLY* message. If there is no *REPLY* message received, this node will stay active; otherwise it will continue to sleep for a random amount of time. The R_p range is provided by the application depending on how much area coverage it needs. A small R_p will bring more robustness into the protocol yet lead to a greater number of active nodes, which entail superior energy expenses. On the other hand, a large value of R_p will save more energy by having more nodes go into sleep but the area might not be fully covered by the active nodes.

Authors in [33] propose an optimal algorithm to solve the area coverage problem. Time synchronization is applied to the network and sensors make decisions on whether to be active or sleeping in each round. At the beginning, a single sensor initiates the decision-making process, which is afterwards propagated to other sensors. Some sensing units will be selected to construct an optimal hexagonal area coverage topology. In [33], there is a theoretical proof that if the communication range is at least twice the sensing range, a complete coverage of a convex area implies connectivity of the active nodes. This analytical result has been applied in many other subsequent research studies.

For the algorithm proposed in [34], each node needs to know the locations of its neighbors. In each round, one node has a timeout interval after which it starts to figure out whether all of its neighbors have covered its own sensing range or not based on their location information. If the answer is positive, it will switch into sleep mode after transmitting a *Status Advertisement Message* (SAM) to notify its neighbors. Otherwise, it will stay active so as to monitor its sensing range.

Many localized area coverage protocols that do not have any limitation on communication and sensing ranges are described in [4]. These algorithms do not need any a priori knowledge of their neighbors yet a time-synchronized network is necessary. The goal of such schemes is to reduce the communication overhead and thus increase the network lifetime. In each round, every node waits for activity messages from its neighbors to check whether its sensing range has been covered by its active neighbors. When its timeout expires, it will decide whether to be active or sleeping. After making the decision, it will also send some messages to notify its neighbors.

Based on different considerations, there are three timeout schemes:

1. Random Time (RT): The simplest way is to select a random timeout.

2. Extensible Random Time (ERT): If a node knows that its sensing area has been almost entirely covered by its neighbors, it can extend its timeout to wait for other neighbors to be active. Therefore, the timeout is proportional to the current coverage ratio of this node's sensing area.

3. Activity-Aware Timeout (AAT): This variant is motivated by the residual energy of previously active nodes. So, this scheme will have passive nodes become active with high priority and active nodes will be turned into passive with high priority.

When the timeout expires, each node evaluates its coverage and makes a decision on whether to be active or sleeping. The authors provide four schemes to inform neighbors with three kinds of messages (activity message, withdrawal message and retreat message):

1. Activity Only (AO): Only active nodes will send a message.

2. Activity and Withdrawal (AW): Every node will notify its neighbors about its decision. An activity message indicates that the node will stay in active status and a withdrawal message means that the node will switch into sleep mode.

3. Activity and Retreat (AR): Similar to AO, but after decision this node will continue to listen to messages from neighbors and re-evaluate its coverage situation when new messages arrive. If they show that its sensing area can be fully covered by the active neighbors, the node will "change its mind" and choose to sleep, not before transmitting a retreat message to its neighbors.

4. Activity, Withdrawal and Retreat (AWR): All nodes send out activity or withdrawal messages when their timeout expires according to their status.

After that, the nodes still listen to messages from their neighbors and compute their sensing area coverage. If their area can be fully covered by their active neighbors, they will transition into sleep mode and broadcast a retreat message.

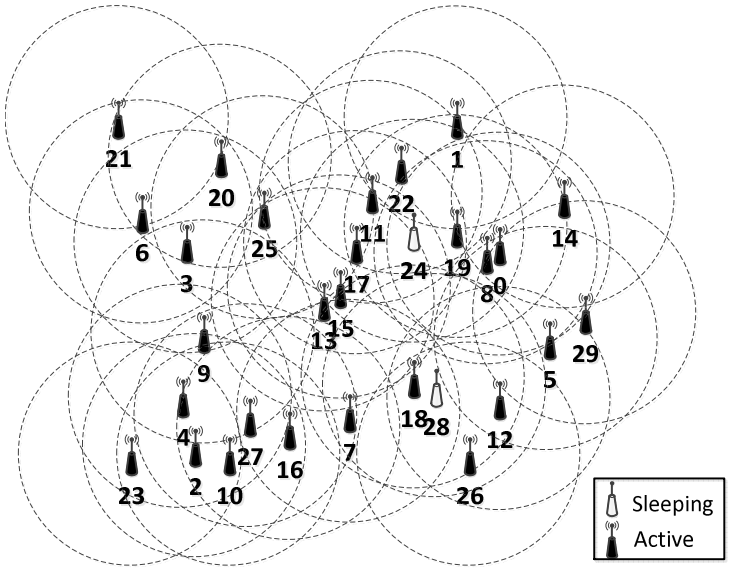


Figure 2.2: Area coverage evaluation.

There are many coverage evaluation criteria proposed in literature. The most popular or general one is intersection-based coverage algorithm in [35] 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.

By this criterion, we can evaluate whether a node's sensing area has been covered by its active neighbors for any ratio of communication range to sensing range. An example about area coverage evaluation is shown in Fig. 2.2.

The area coverage algorithm in [4] will be used in the design of our data aggregation framework to prolong the network's lifetime. Details will be discussed in Chapter 4.

2.4 Backbone Construction Algorithms

The backbone is a subset of the sensor nodes which are responsible for broadcasting messages within the network. In [36], the authors review many backbone construction algorithms.

The clustering scheme in [37] has the node with lowest ID among all its neighbors become the cluster head and issue the corresponding notification message to its neighbors at first. Then, every undecided node that got the cluster head notification message will become a member of its cluster and transmit a cluster member packet to its neighbors. Every undecided node that heard the cluster member packet will check whether its ID is minimal among all its undecided neighbors or not. If so, it will become a cluster head and broadcast its own cluster head message. This iterative process ensures that each node is either turned into a cluster head or attached to a particular cluster.

The authors in [38] propose a grid-partitioning backbone construction algorithm GAF. The nodes are split into different grids. For each grid, one node is elected to communicate with nodes in the adjacent grid. This scheme could jeopardize the network connectivity though. So, the ratio of grid size to communication radius should be carefully worked out so as to avoid that.

Another backbone construction scheme is proposed in [39]. This protocol is called *Multipoint relays* (MPRs) and selects a subset of one-hop neighbors that can cover all two-hop neighbors. How to select a MPR of minimal size is a NP-complete problem, hence a heuristic method is put forward to select the subset of one-hop neighbors as described below:

1. *Check all one-hop neighbors $N(x)$ of node x and select as relay node the one that is the only neighbor of one two-hop neighbor $N^2(x)$ in $N(x)$;*

2. *For each node in $N(x)$ which is not selected as relay node yet, compute the number of nodes it covers among uncovered nodes in $N^2(x)$. Select the one with maximum coverage and add it to the group of relay nodes;*

3. *Repeat (2) until all nodes in $N^2(x)$ are covered.*

The dominating set is one kind of backbone. Each node in the network either belongs to the dominating set (gateway node) or has a one-hop neighbor which belongs to the dominating set (non-gateway node). A connected dominating set (CDS) is a dominating set in which its nodes are connected. The CDS construction of minimum size is known as a NP-complete problem. Therefore, there are many algorithms that compute CDS with low communication overhead and low cardinality.

In [40], Wu et al. present a basic algorithm to construct a CDS with local knowledge. The idea is that each node shares its neighbor list with all of its neighbors. If there is a pair of unconnected neighbors, this node will become the gateway node in the dominating set.

A generalized distributed dominating set construction algorithm is explained in [41]. Each node has a unique key which can be its identity number, residual energy or a combination of both. We say that a node u is covered by the subset $a, b, c, \ldots$ of its one-hop neighbors if

1. key(u) > max(key(a), key(b), key(c),...);

2. $a, b, c \ldots$ are connected;

3. any other one-hop neighbor of u is neighbor of at least one node in $a, b, c, \ldots$.

In [5], a simplified method to implement the algorithm in [41] is brought forth. At the outset, each node checks whether it is an intermediate node for any two

unconnected neighbors. If it does not have unconnected neighbors, it will become a covered node. Then, only intermediate nodes start building a subgraph G of its neighbors with lower keys. If G is empty or disconnected, this node will become a dominant node. If there exists a neighbor of this node that is not a neighbor of any node in G, it will belong to the dominating set. Otherwise, this node becomes a covered node which is not in the dominating set. The Fig. 2.3 shows an example of connected dominating set. In this algorithm, either two-hop knowledge or one-hop knowledge with geographic information is enough. So the implied communication overhead is very low and a small-size CDS can be constructed. The scheme in [5] will be applied to our data aggregation framework whenever we want to build a connected dominating set.

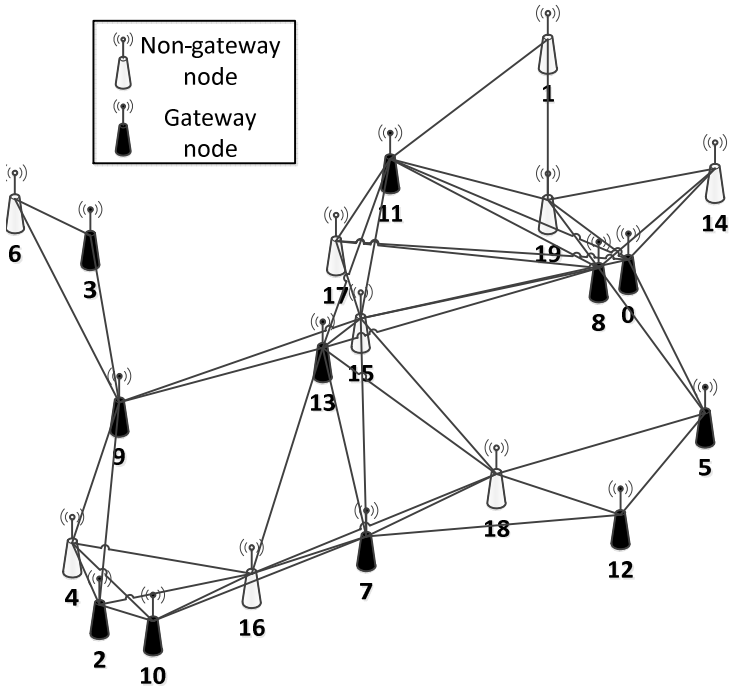


Figure 2.3: Connected dominating set construction.

Chapter 3

Delay-bounded and Energy-efficient Data Aggregation Protocol

The Delay-bounded and Energy-efficient Data Aggregation(DEDA) protocol is our new framework which aims at solving data aggregation problems in wireless sensor networks. It will build a delay-bounded and energy-efficient data aggregation tree based on the novel concept of *DEsired Progress (DEP)*, using edges selected from LMST and UDG. The term "*progress*" means reduction in distance to actor. As usual, we measure the distance between two nodes in a given network graph by the length of the shortest path connecting them. In our framework, the path length is measured by the delay variable which should be proportional to path delay. However, the path delay itself highly depends on the underlying delay model under consideration.

In this chapter we first list all terminologies used afterwards and then study many different delay models in detail. For each delay model, we explain how to

perform data aggregation according to each specific condition. Later on, an early hop-based version of this algorithm called DHP[42] will be reviewed. Based on that, our new framework called DEDA will be thoroughly introduced.

3.1 Terminology

This section acquaints the reader with the terminology used throughout the rest of this chapter. We first enunciate the following definitions for an arbitrary node w.

- Closed neighborhood $N(w)$: a set of nodes composed of w and its one-hop neighbors;
- Family $F(w)$: a subset of $N(w)$ containing the children of w in the data aggregation tree;
- Degree $d(w)$: the size of $N(w)$, i.e., $|N(w)|$;
- Cardinality $a(w)$: the size of $F(w)$, i.e., $|F(w)|$;

A path P_u^v between two nodes, source u and destination v, is a set of forwarding nodes consisting of u and all the intermediate nodes in order. The hop sum $h(P_u^v)$ of P_u^v is the number of nodes in P_u^v, i.e., $h(P_u^v) = |P_u^v|$.

When dealing with a delay-aware problem, we need to be clear about the behavior of packet transmission.

- Path delay $t(P_u^v)$: the delay that a packet experienced when being transmitted along P_u^v; it is subject to the number of times the packet is forwarded, i.e., $h(P_u^v)$, and the delay of each forwarding;

- Forwarding delay $t(w)$: typically modeled as the aggregation of processing delay $t_{proc}(w)$, queuing delay $t_{queue}(w)$, transmission delay $t_{tran}(w)$ and propagation delay $t_{prop}(w)$ at node w. [2]

Compared to the other delay types, $t_{prop}(w)$ is negligible and can be regarded as a small constant, if not ignored. Due to data aggregation, only one report is sent from each sensor in each iteration of data collection. Thus $t_{proc}(w)$ and $t_{tran}(w)$ are approximately constant at different w whereas $t_{queue}(w)$ is dependent on the MAC layer used and can be rather different.

Hence for a single report at each node w we define *delay constant* as $c = t_{proc}(w) + t_{prop}(w) + t_{tran}(w)$. Forwarding delay $t(w)$ and path delay $t(P_u^v)$ are given by

$$t(w) = t_{queue}(w) + c, \tag{3.1}$$

$$t(P_u^v) = \sum_{w \in P_u^v} t(w). \tag{3.2}$$

Let α be a constant *delay factor*. We define the *delay variable* of w and P_u^v respectively as

$$r(w) = \frac{1}{\alpha}\, t_{queue}(w) + \frac{c}{\alpha}, \tag{3.3}$$

$$r(P_u^v) = \sum_{w \in P_u^v} r(w). \tag{3.4}$$

Then Eqn. (3.1) and (3.2) can be rewritten as follows:

$$t(w) = \alpha \times r(w), \tag{3.5}$$

$$t(P_u^v) = \alpha \times r(P_u^v). \tag{3.6}$$

3.2 Delay Models

3.2.1 Ideal Model

We will talk about many delay models in this section. The first one and also the simplest one is called "ideal model" because it relies upon an ideal MAC layer in which outgoing packets are always successfully transmitted immediately after their arrival.

We have $t_{queue}(w) = 0$ at each node w. Let $\alpha = c$. By Eqn. (3.3) and (3.4), we have $r(w) = 1$ and $r(P_u^v) = h(P_u^v)$. Since path delay variable is equivalent to hop sum, we call this delay model "hop-based". It has been adopted by many existing delay-bounded algorithms such as MS [31] and DHP [42].

In reality, however, wireless medium is a common network resource shared by all network nodes. If a node is within the interference range of several transmitting nodes, it will not be able to correctly receive the message of any of them because their radio signals overlap, thus resulting in unreadable noise. In order to ensure successful transmission, nodes have to compete for medium access, introducing non-zero queuing delay at MAC layer.

In light of this fact, we will then propose three other delay models that take MAC competition into consideration, along with discussions on their influence on the DEDA framework. These models differ in the definition of nodal delay variable $r(w)$ in Eqn. (3.3), which is subject to the particular type of MAC layer underpinning the protocol's implementation.

3.2.2 Cardinality-based Model

We assume a semi-ideal MAC layer as follows. Nodes have a sufficiently long MAC queue so that no packet is lost due to queue overflows. Interference range is a

logical range, equal to nodal family. That is, nodes from different families can simultaneously successfully transmit, while family members have to compete for MAC and transmit in turn so that the packets can be successfully delivered. MAC competitors (i.e. members of the same family) have equal opportunity to transmit and their transmissions are somehow coordinated so that no collision (thus no packet loss) may occur.

With this MAC layer, we may reasonably conclude that $t_{queue}(w)$ is approximately a linear increasing function of the cardinality of node w, i.e., $t_{queue}(w) = \alpha \times a(w)$, where α is the delay factor. Then the delay variable of w is defined as

$$r(w) = a(w) + \frac{c}{\alpha}.$$

Notice that if the delay constant c is negligible compared to $\alpha \times a(w)$, $t(P_u^v)$ will be directly proportional to the aggregated cardinality of nodes in P_u^v. When $a(w) \approx a(w')$ for any $w \neq w'$, this model converges to the hop-based model.

Hop sum (thus delay variable defined on it) is a stable factor (in the absence of node failure) and regardless of the data aggregation structure. The hop-based DHP framework reviewed in Sec. 3.3 is therefore effective with the hop-based delay model. Nevertheless, node cardinality is a variable factor which is sensitive to the data aggregation structure. When a node changes parent, the previously established cardinality distance of the nodes in the subtree of its old and new parents become invalid and need to be recalculated. When the cardinality-based delay model is used, local behavior is no longer sufficient to build the desired global data aggregation tree. Since this model is neither realistic nor widely used, we list it here only for completeness and will not provide a solution to it.

Nonetheless, we notice that each node has roughly two tree neighbors, one

parent and one child, in the shortest path tree of LMST. Under this circumstance, the cardinality-based model is roughly the same as the hop-based model. If the tree does not differ much from the ideal cardinality-based data aggregation tree, DHP in Sec. 3.3 will provide a good approximation to the solution for data aggregation with the cardinality-based model.

3.2.3 Degree-based Model

We assume a semi-realistic MAC layer. It uses the same set of assumptions such as sufficiently long message queue, equal transmission opportunity and zero collision as the above semi-ideal MAC layer; yet it bears a different, but practical, interference range definition: *interference range is of the same size as communication range*, meaning that all one-hop neighbors are MAC competitors. This difference makes it relatively more realistic than the semi-ideal MAC layer.

With this MAC layer, the delay variable of node w is subject to its degree, instead of cardinality, as follows

$$r(w) = d(w) + \frac{c}{\alpha}.$$

Again, if delay constant c is negligible compared to $\alpha \times d(w)$, $t(P_u^v)$ will be directly proportional to the degree sum of nodes in P_u^v. When $d(w) \approx d(w')$ for any $w \neq w'$, this model converges to the hop-based model.

As in the hop-based model, node degree is also a stable factor. But all the hop-based algorithms (including DHP) can not be applied here for degree-based delay model. Hence our DEDA framework is proposed to solve this data aggregation problem under the umbrella of the degree-based delay model.

3.2.4 Two-Stage Model

We consider a realistic MAC layer, specifically IEEE 802.11 CSMA/CA, with short packet queue and potential collision. IEEE 802.11 CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) defines two medium access mechanisms, including Distributed Coordination Function (DCF) and Point Coordination Function (PCF). The latter is a centralized scheme and thus off our interest, considering the distributed nature of the WSN. Below we briefly introduce the DCF scheme.

At the MAC layer, a node with a packet to transmit monitors the medium status. As soon as the medium is continuously idle for a Distributed InterFrame Space (DIFS) in time, it prepares for the transmission. It sets a backoff timer randomly selected in a range called *contention window* and transmits after the timer times out. The purpose of having the backoff interval is to reduce collision and ensure fair medium access. A backoff interval may be broken into pieces due to channel activities. That is, the backoff timer is frozen when the medium is sensed busy, and reactivated after an idle DIFS.

Packet transmission obeys a basic access mechanism as follows. Immediately at the end of receiving a packet, the intended receiver waits for a time period equal to a Short InterFrame Space (SIFS) and signals the sender by an ACK packet. If, within a specified timeout after the packet transmission, the sender does not receive the ACK or senses the transmission of a different packet, it considers the transmission unsuccessful and reschedules it following the same algorithm. Note that the size of the contention window is at the first transmission attempt set to a default minimum value CW_{min}, and then doubled after each unsuccessful attempt up to a maximum value CW_{max}.

Optionally, a RTS/CTS access mechanism can be used. Instead of sending the packet immediately after the backoff interval, the sender transmits a short

frame called Request To Send (RTS). When the receiver detects the RTS frame, it responds, after a SIFS, by a Clear To Send (CTS) frame. The sender then transmits the packet only if the CTS frame is correctly received. Both RTS and CTS frames carry the length of the packet to be transmitted. Hidden nodes overhearing any of the two frames can delay further transmission and avoid collision. This mechanism improves the system's performance because it reduces the length of the frames involved in the contention process.

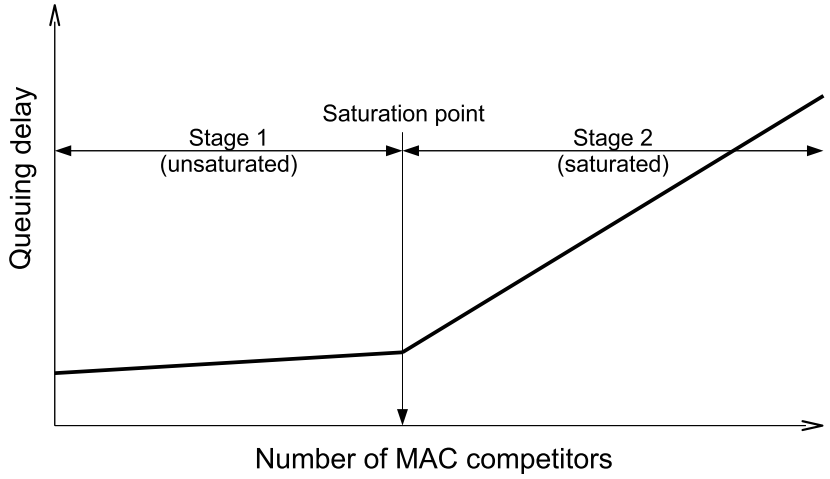


Figure 3.1: Average queuing delay t_{queue} in relation with the number of MAC competitors.

Whichever access mechanism is used, we easily have the following conclusion. The more MAC competitors, the higher the probability that the medium is busy, and the more often a backoff interval is interrupted, resulting in more frequent waiting, each time for an idle DIFS, and finally larger delay. The relation of MAC delay to number of MAC competitors has been studied under both saturation conditions [43, 44] and non-saturation conditions [45, 46]. Mathematical models were presented. Despite the underlying model difference and complexity, we however observe that MAC delay is approximately a two-stage increasing linear function of the number of MAC competitors. See Fig. 3.1 for reference. The figure is ab-

stracted from [45, 46]. Saturation point (d_{sat}) is the point after which there is at least one packet in the MAC queue whenever a new outgoing packet arrives.

For each transmission there must be at least one backoff stage, whose length is $CW_{min}/2$ on average (because of the uniform random selection from the default contention window CW_{min}). When the communication channel is saturated, the queuing delay is approximately a linear increasing function of the node degree with a large slope, say α. When the communication channel is unsaturated, it is also nearly a linear increasing function of the node degree, but with a very small slope such that the curve is almost flat. In this case, we may ignore the slightly increasing trend and regard queuing delay as constant and equal to $CW_{min}/2$. Summarizing, $t_{queue}(w) \approx \alpha \times x(w) + CW_{min}/2$, where $x(w)$ is equal to 0 if $d(w) \leq d_{sat}$ and $d(w) - d_{sat}$ otherwise. Let $c' = c + CW_{min}/2$ be the new delay constant. Then the delay variable of node w will be

$$r(w) \approx \begin{cases} \frac{c'}{\alpha} & \text{for } d(w) \leq d_{sat}; \\ d(w) - d_{sat} + \frac{c'}{\alpha} & \text{for } d(w) \geq d_{sat}. \end{cases}$$

Notice that when $d(w) \leq d_{sat}$ for any w or when $d(w) \approx d(w')$ for any $w \neq w'$, this model converges to the hop-based model. This justifies the rationality of using the hop-based delay model (equivalently, ideal MAC layer) to facilitate protocol design, particularly in sparse or uniform networks.

The two-stage model is in essence degree-based which requires our degree-based DEDA framework. It also requires to set up a maximum waiting time for data aggregation at each node. This additional change is necessary because reports may be dropped due to queue overflow and repeated collision at the MAC layer. The maximum waiting time is defined as $t_{queue}(w)(1 - \frac{1}{d(w)})$ so as to tolerate the

reporting delay due to node w itself. After the waiting time, each node w reports according to the currently available information.

3.3 Hop-based DHP

In this section, we will review a hop-based DHP (*Desired Hop Progress*) framework for data aggregation in wireless sensor networks. As we know, in low-traffic networks, there is almost no queuing delay for data reports because the channel is not saturated. So the delay is almost proportional to the hop sum along the path from the sender sensor to the actor and therefore the delay limit and distance are measured by hop sum. At first, the initial data aggregation tree based on LMST is constructed. After that, taking into account different delay limits designated by the actor, DHP is utilized to build the data aggregation tree dynamically in order to satisfy different delay limit requirements. The sample sensor network in Fig. 3.2 will be used to explain the DHP framework.

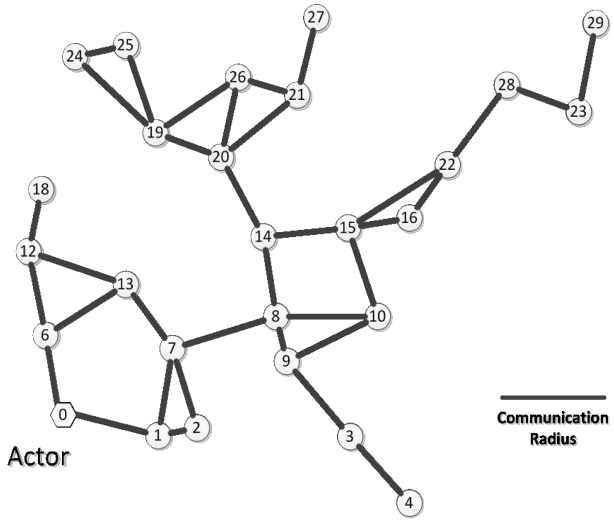


Figure 3.2: Sample sensor network.

3.3.1 Building the Initial Data Aggregation Tree

The initial data aggregation tree is usually constructed when the network is first configured and oftentimes reconstructed when the network topology changes, e.g. an actor receives some node failure messages. In other words, the initial data aggregation tree is reconstructed only when the original tree is changed. In this way, a lot of messages involved in the creation of such tree-based structure are saved while fault detection ability is guaranteed.

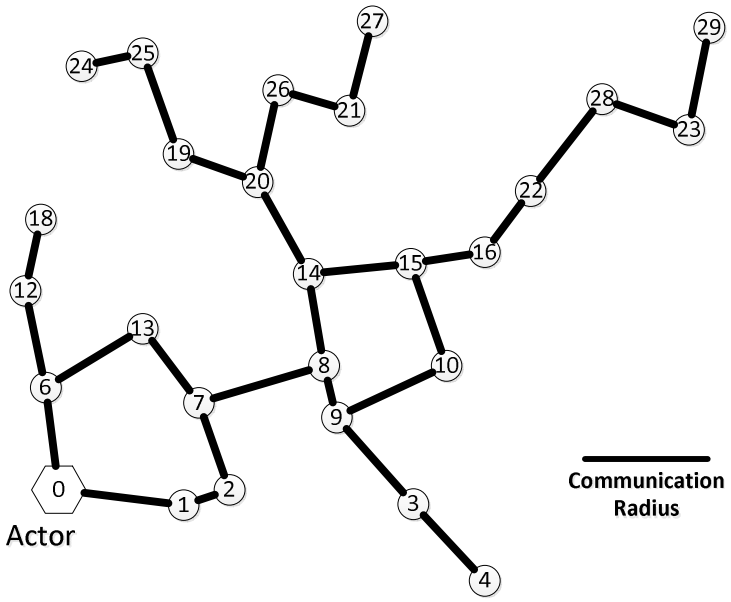


Figure 3.3: LMST in sample sensor network.

Whenever the initial data aggregation tree is to be assembled, the actor floods the LMST (Fig. 3.3) with a request message having an *id* equal to 0, which means this message is intended to construct the initial data aggregation tree as opposed to collecting any information. The message includes the sender's distance (UD) to the actor in UDG and the distance (LD) to the actor in LMST. For the actor, its UD and LD both equal to 0. When a sensor node S receives a new request, it sets its own UD to the received $UD + \mathrm{r}(S)$ and will set its own LD to the received

LD + r(S) if the sender is an LMST neighbor. For hop-based DHP, r(S) = 1. In the case that it receives the same fresh request (whose id is 0) again, it updates the two distances only if their current values are larger. S retransmits the received request message if and only if the sender is an LMST neighbor (to ensure flooding along LMST) and the message has caused the reset or update of its LD. Before retransmission, it updates the message with its own UD and LD.

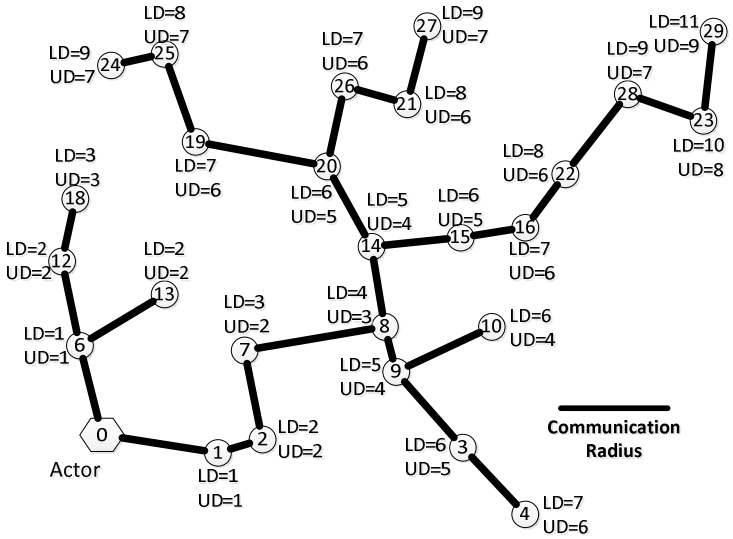


Figure 3.4: LMST-based initial data aggregation tree.

After this flooding process, an LMST-based shortest path tree is established as displayed in Fig. 3.4. This tree is rooted at the actor. Every sensor knows its parent (identified by stored sender id) as well as its UDG distance (UD) and LMST distance (LD). It also learns about all parent (child) candidates, which are the neighbors with equal or smaller (resp., larger) LMST distance. The tree is an initial data aggregation tree, along which sensors aggregate and send reports upward to the actor. Because of the LMST properties, the energy consumption is largely reduced along LMST edges. But this structure may be subject to change during data aggregation when the delay limit is taken into consideration.

3.3.2 Taking Delay Limit into Consideration

Definition 1 *Consider a sensor that is at k hops from the actor in UDG. Suppose that the sensor receives a report which already experienced m hops delay. The report is expected if* $m + k \leq DL$ *and unexpected otherwise.*

Definition 2 *The Desired Hop Progress (DHP) of a sensor is defined as the rounded-up ratio of the sensor's LMST hops to the actor to the number of remaining hops allowed for its aggregated report. That is,* $DHP = \lceil \frac{LD}{DL-MED} \rceil$ *where MED is the most experienced delay of all expected reports (MED = 0, if no expected report.)*

The initial data aggregation tree is based on the LMST structure, so the total energy consumption in the network is almost optimal. But usually under this initial tree, the delay time will be very large for every sensor. Given the different delay limits required by the end user, we should take delay limit into consideration. Indeed, a report can be delivered along the tree to the actor by the specified deadline if it progresses at least DHP *LMST distance* per hop toward the actor on average. As we know each UDG distance progress of a message along the tree is fixed as 1, sensors must seek shortcuts in LMST in the case of DHP > 1. This involves the use of non-LMST edges and will change the tree structure, as will be discussed below.

3.3.3 Selecting Shortcut for Fitting Delay Limit

Whenever the actor wants to collect information in the deployment area, it floods a request message along LMST, which includes a delay limit designated by the end user. If the initial data aggregation tree does not satisfy the delay limit, each node will find some DHP-based shortcut to advance toward the actor. The selected

shortcut only provides enough progress instead of advancing too much to keep the data aggregation tree balanced.

If sensor S has an empty child candidate set, it starts the aggregation process immediately after receiving the request from the actor. The sensor computes its DHP value and selects as parent the node (among all parent candidates) with LMST progress equal to this value. In the presence of multiple such candidates, the one with shortest Euclidean distance is chosen so as to save transmission power for reporting. If DHP can not be satisfied exactly, the parent candidate whose progress is closest to it is picked, with preference given to the one with greater progress in case of tie. Finally, any remaining tie can be broken by node ID. Once S decides on its parent, it notifies about such decision at maximum transmission power. The decision message includes the sender's ID, the parent node's ID and the evaluated reporting delay $ERD = 1$ at the parent node.

If sensor S has a non-empty child candidate set, it will wait for parent declarations from all of its child candidates. For each received decision message, S deletes the sender from the candidate list and adds it to the child list if S is its selected parent. Then, S considers the evaluated reporting delay ERD in the declaration as experienced delay of reports from this child during 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 unexpected accordingly. If the child is expected and the ERD is larger than the local MED, S will update MED to ERD. When the child candidate list becomes empty, S starts the parent selection process and broadcasts the result via a decision message, where the ERD is set to $MED + 1$.

The final data aggregation tree is constructed as in Fig. 3.5. Take sensor **22** for example, after it receives the parent decision message from node **28** with $ERD=1$.

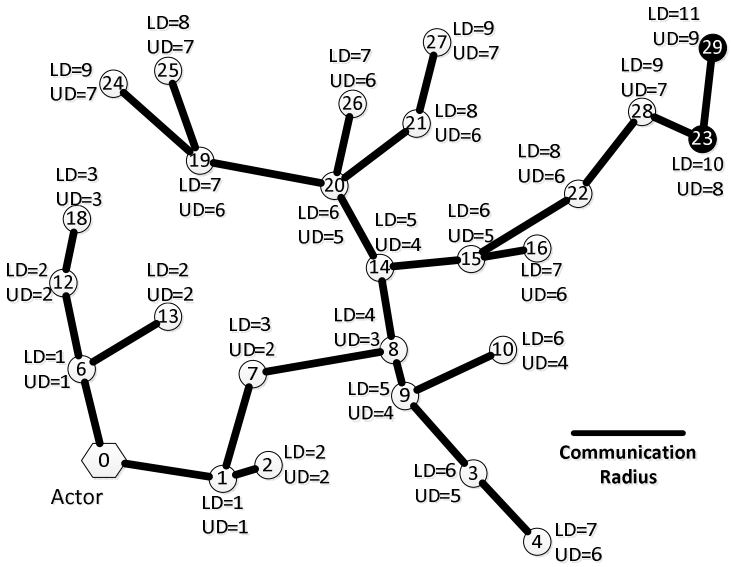


Figure 3.5: Final data aggregation tree (DL = 7).

Firstly, it decides whether this child is expected or not based on Def. 1. Because $ERD + UD = DL$, the child is expected and its report will be relayed by **22**. Secondly, this node computes DHP based on Def. 2. Because it knows its own *UD,LD and MED*, DHP= $\lceil \frac{8}{7-1} \rceil = 2$. Finally, the node selects **15** as its parent because it has the closest progress ($LD(22) - LD(15) = 2$) to DHP.

You can see here that the node **22** selects a shortcut to satisfy the delay limit requirement. But for node **10**, it gets DHP= $\lceil \frac{6}{7-0} \rceil = 1$. So, the parent candidate **9** with 1 LMST distance progress is chosen as parent instead of candidate node **8** which progresses 2 in terms of LMST distance. These two examples show that the hop-based DHP framework not only tries to satisfy the delay limit but also saves energy by choosing LMST edges under a loose delay limit.

3.3.4 Pseudocode

In this section, we offer the algorithmic pseudocode in detail. Actually, we have implemented this hop-based DHP framework in the simulator. Alg. 1 describes the behavior of the algorithm when sensor S receives a request message. Alg. 2 explains how to handle the parent decision message. Finally, the data report aggregation procedure is introduced in Alg. 3.

Algorithm 1 Hop-based DHP (at sensor node S)

```
On receiving request message REQT
if REQT.SN > SN then
  SN := REQT.SN
  DL := REQT.DL
  UD := REQT.UD + 1
  if REQT.SENDER is LMST neighbor then
    LD := REQT.LD + 1
    update and retransmit REQT message
  end if
end if
if REQT.SN = SN then
  if UD > REQT.UD + 1 then
    UD:= REQT.UD + 1
  end if
  if REQT.SENDER is LMST neighbor && LD > REQT.LD + 1 then
    LD := REQT.LD + 1
    update and retransmit REQT message
  end if
end if
if having received REQT with latest SN from all LMST neighbors then
  select parent and child candidates
  if no child candidates then
    compute DHP value by Definition 2
    select PARENT with LMST-distance progress closest to DHP
    send to all parent candidates a PARED message, with ERD=1
  end if
end if
```

Algorithm 2 Hop-based DHP (at sensor node S)

On receiving parent decision message PARED
if $S = PARED.PARENT$ && $PARED.ERD \leq DL - UD$ **then**
  CHILD := CHILD ∪ PARED.SENDER
  if $PARED.ERD > MED$ **then**
    MED := PARED.ERD
  end if
end if
if *having received* $PARED$ *from all child candidates* **then**
  compute DHP value by Definition 2
  select PARENT with LMST-distance progress closest to DHP
  send a PARED message with ERD set to MED+1 to all parent candidates
end if

Algorithm 3 Hop-based DHP (at sensor node S)

On receiving a report
if *the child is expected* **then**
  buffer the report locally
end if
if *all* $buffered$ *reports are ready* **then**
  aggregate the reports into a single report
  send the aggregated report to PARENT
end if

3.4 Degree-based DEDA

In this section, we will talk about the degree-based DEDA framework. As mentioned before, degree-based DEDA is used to solve the data aggregation problem based on the degree-based delay model or the two-stage delay model. Like the discussion in hop-based DHP, we will firstly construct the initial data aggregation tree and then based on different delay limits, we use DEP to rebuild the data aggregation tree dynamically in order to satisfy different delay limit requirements. The sample sensor network in Fig. 3.2 will also be used to illustrate our DEDA framework.

3.4.1 Building the Initial Data Aggregation Tree

As we mentioned in Sec. 3.3.1, the initial data aggregation tree is constructed when the network is first set up and reconstructed whenever its topology changes due to node failures. This dynamic construction strategy helps save a great deal of energy and also preserves the ability to recover from node failures.

In the degree-based framework, the actor floods the UDG (see Fig. 3.2) instead of LMST (as in Fig. 3.3) because we cannot get correct UDG distance (UD) with LMST flooding. All the remaining steps in the flooding process are similar to those in hop-based DHP. The flooding message includes the sender's UD and LD. When a sensor node S receives a new flooding message, it sets its own UD to the received $UD + \mathrm{r}(S)$ and its own LD to the received $LD + \mathrm{r}(S)$ if the sender is an LMST neighbor. For the degree-based DEDA, $\mathrm{r}(S)$ equals $d(S)$. In the case that it receives the same fresh flooding message again, it updates the two distances only if their current values are larger than the sum of the received value and $d(S)$. Sensor node S retransmits the received message only if UD or LD is updated. Before

retransmission, it updates the message with its own *UD* and *LD*.

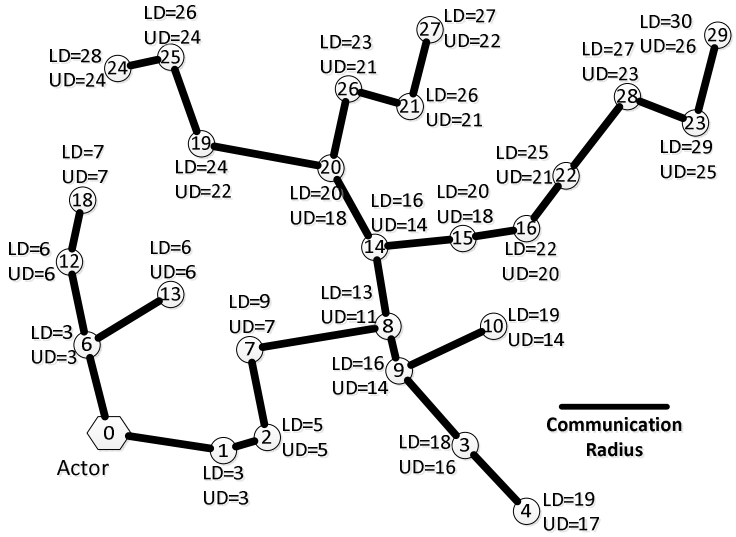


Figure 3.6: LMST-based initial data aggregation tree.

After this flooding process, an LMST-based shortest path tree is established, as depicted in Fig. 3.6. This LMST-based tree will collect data almost in an optimal energy consumption fashion. Every sensor not only knows its own *UD* and *LD* but also gets to know the *LD*s of its neighbors. Hence, it can decide which ones are parent/child candidates. When the delay limit is taken into consideration, this initial tree may be changed as discussed below in Sec. 3.4.2.

3.4.2 Selecting Shortcut for Fitting Delay Limit

At first, we redefine the expected report in Def. 3 and introduce a new concept which is DEsired Progress (DEP) in Def. 4. Because we will use degree as the distance metric rather than hop count, the new definitions will be more precise and suitable for a realistic delay model.

Definition 3 *Consider a sensor that is at UDG distance k from the actor, measured by degree sum. Suppose that the sensor receives a report with already experi-*

enced delay m. The report is expected if $m + k \leq DL$ *and unexpected otherwise.*

Definition 4 *The DEsired Progress DEP of a sensor is defined as the rounded-up ratio of the sensor's LMST distance to the actor to the remaining lifetime of its aggregated report. That is,* $DEP = \lceil \frac{LD}{DL-MED} \rceil$ *where MED is the most experienced delay of all expected reports (MED = 0, if no expected report).*

As in Sec. 3.3.2, we will consider the delay limit by the *DEP* value which equals to $\lceil \frac{LD}{DL-MED} \rceil$. That means that, for each time unit in the future, we need to progress *DEP LMST-distance(LD)* on average in order to satisfy the delay limit. Let us shed light on how to select a reporting shortcut in the degree-based DEDA.

We use the request-driven model in data aggregation. The request message including the delay limit value will be broadcast along the sensor network. If the DEP at some node is greater than 1, which means that the LMST edge does not satisfy the delay limit, that node needs to find a DEP-based shortcut to advance toward the actor. Under the shortcut selection method we use, the chosen UDG edge just provides enough progress instead of advancing too much and thus jeopardize the network-wide energy consumption balance.

During the request message broadcast process, the sensors without any child candidates will start to select parents at first. The sensor S will compute the DEP, which means the desired average progress in terms of LMST distance per delay time unit. Because the delay time unit consumed for next hop at the current node will be *d(S)*, we multiply DEP by *d(S)* and then get the new value of $\mathcal{DEP}$ as stated in Def. 5. $\mathcal{DEP}$ means the desired LMST distance progress for the next hop and is the basis for parent selection like in hop-based DHP (cf. Sec. 3.3.3).

Definition 5 *The* $\mathcal{DEP}$ *of a sensor* w *is defined as* $d(w) \times DEP$ *and stands for the desired progress in LMST distance for the next hop.*

The sensor S will select the neighbor (among all parent candidates) with LMST distance progress equal to $\mathcal{DEP}$ as its parent. The one with shorter Euclidean distance will be preferred in case of multiple candidates. If there does not exist such neighbor with $\mathcal{DEP}$ LMST distance progress, the parent candidate whose progress is closest to $\mathcal{DEP}$ will be chosen and the one with greater progress will be selected in case of tie. Once sensor S has selected its parent, it broadcasts this decision to all of its neighbors. The decision message includes its own ID, the parent's ID and the evaluated reporting delay $ERD = d(S)$.

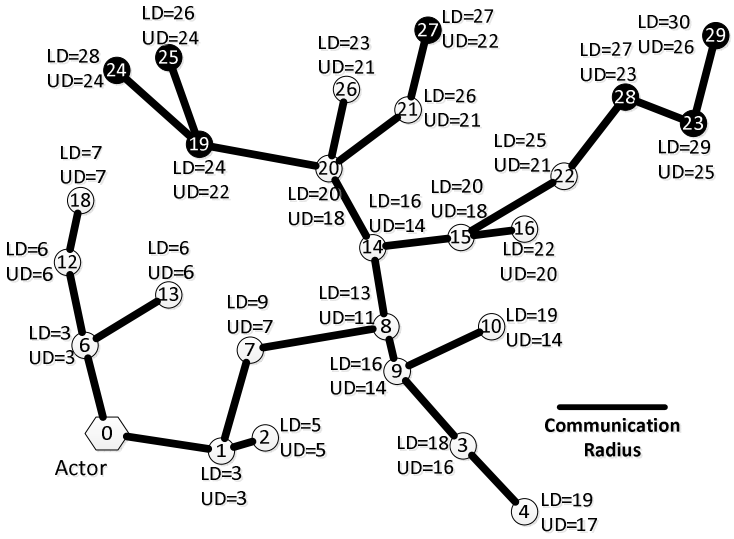


Figure 3.7: Final data aggregation tree (DL = 21).

The sensors with child candidates will wait for parent decision messages from all of them. For each received decision message, sensor S removes the sender out of the child candidate set and attaches this neighbor to the child set if it is selected as the child's parent. Sensor S receives the ERD in the message as experienced delay of reports from this child in data aggregation. Based on ERD, S evaluates whether a data report from this child is expected or not according to Def. 3. If this is an expected child, the report will be delivered to the next hop, else the report will be discarded. When the received ERD from an expected child is larger than

the current *MED*, node S will update *MED* with *ERD*. After receiving all parent decisions from child candidates, S begins to compute $\mathcal{DEP}$ based on the updated *MED* and broadcasts the parent decision message in which $ERD = MED + d(S)$.

The final data aggregation tree is constructed as in Fig. 3.7. Take sensor **22** as an example. Upon it receiving a parent decision message from node **28** with *ERD=2*, it first decides whether this child is expected as per Def. 3. Because $ERD + UD > DL$, the child is not expected and the report will be consequently discarded by **22**. Afterwards, this node computes $\mathcal{DEP}$ according to Def. 4 and 5. Because it knows its own *UD, LD* and *MED*, $DEP = \lceil \frac{25}{21-0} \rceil = 2$ and $\mathcal{DEP} = 2 * d(22) = 6$. Finally, **22** selects **15** as its parent because it bears the closest progress ($LD(22) - LD(15) = 5$) to $\mathcal{DEP}$. You can see here that node **22** selects a shortcut to satisfy the delay limit. As to node **10**, it gets $DEP = \lceil \frac{19}{21-0} \rceil = 1$ and $\mathcal{DEP} = 1 * d(10) = 3$. So, the parent candidate **9** with LMST distance progress of 3 is chosen as parent rather than candidate node **8** which progresses 6 in LMST distance. These two examples show that our degree-based DEDA framework not only saves substantial energy using LMST edges but also strives to satisfy the delay limit requirement by choosing shortcut edges in UDG.

3.4.3 Pseudocode

In this section, the pseudocode for the intended degree-based DEDA framework will be discussed. It is similar to that of hop-based DEDA. Alg. 4 is responsible for handling the request message whereas Alg. 5 takes care of processing parent decision messages and Alg. 6 is concerned with performing data aggregation. Yet at this time the request message will be broadcast along UDG. After we get DEP, $\mathcal{DEP}$ will be subsequently computed. The parent selection mechanism is guided by the idea of choosing, among all parent candidates, the node with LMST distance

progress closest to $\mathcal{DEP}$.

Algorithm 4 Degree-based DEDA (at sensor node S)

```
On receiving request message REQT
if REQT.SN > SN then
  SN := REQT.SN
  DL := REQT.DL
  UD := REQT.UD + d(S)
  if REQT.SENDER is LMST neighbor then
    LD := REQT.LD + d(S)
  end if
  update and retransmit REQT message
end if
if REQT.SN = SN then
  if UD > REQT.UD + d(S) then
    UD:= REQT.UD + d(S)
  end if
  if REQT.SENDER is LMST neighbor && LD > REQT.LD + d(S) then
    LD := REQT.LD + d(S)
  end if
  if UD or LD is updated then
    update and retransmit REQT message
  end if
end if
if having received REQT with latest SN from all LMST neighbors then
  select parent and child candidates
  if no child candidates then
    compute DEP value by Definition 4
    compute DEP value by Definition 5
    select PARENT with LMST-distance progress closest to DEP
    send a PARED message with ERD = d(S) to all parent candidates
  end if
end if
```

Algorithm 5 Degree-based DEDA (at sensor node S)

On receiving parent decision message PARED
if $S = PARED.PARENT$ && $PARED.ERD \leq DL - UD$ **then**
  CHILD := CHILD $\cup$ PARED.SENDER
  if $PARED.ERD > MED$ **then**
    MED := PARED.ERD
  end if
end if
if *having received* $PARED$ *from all child candidates* **then**
  compute DEP value by Definition 4
  compute $\mathcal{DEP}$ value by Definition 5
  select PARENT with LMST-distance progress closest to $\mathcal{DEP}$
  send a PARED message with ERD=MED+d(S) to all parent candidates
end if

Algorithm 6 Degree-based DEDA (at sensor node S)

On receiving a report
if *the child is expected* **then**
  buffer the report locally
end if
if *all* $buffered$ *reports are ready* **then**
  aggregate the reports into a single report
  send the aggregated report to PARENT
end if

Chapter 4

Two Variants of DEDA Framework

Two versions of DHP, namely DHPA (DHP with area coverage algorithm) and DHPAC (DHPA with connected dominating set) have been published in [42]. Similarly, in this chapter we will also present two DEDA variants, A-DEDA and AC-DEDA. They will require less sensors to report and broadcast, thus achieving a superior performance compared to the canonical DEDA implementation.

For A-DEDA, we first switch some sensors into sleeping mode as a way of sparing energy. All the remaining nodes will stay active and run the DEDA framework to construct the data aggregation tree. This wake-up scheduling selection is based on an area coverage algorithm. The network topology is afterwards maintained to monitor the whole area that has not changed. In other words, we can save energy by turning off some sensors yet still realize the same work as before when all sensors were active. A large number of sensors will be commanded to go into sleep mode, especially in dense networks. Such nodes will switch back into active mode whenever the currently active ones crash down or run out of battery. Therefore, the

A-DEDA protocol exhibits an enhanced fault recovery ability and energy balance. In Sec. 4.1, we will elaborate on the A-DEDA protocol through visual examples.

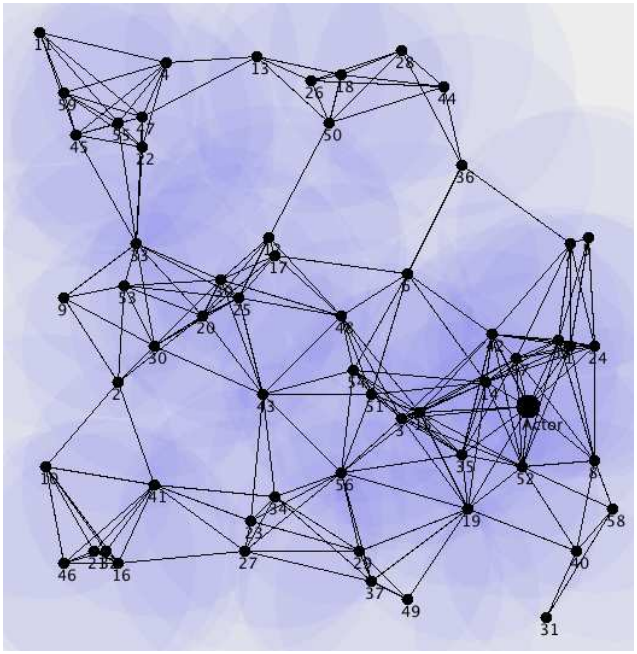


Figure 4.1: Sample network for data aggregation.

For AC-DEDA, we switch some sensors into sleeping mode like in A-DEDA and then assemble a connected dominating set on the active nodes. After connected dominating set construction, we categorize the sensing units into dominating and non-dominating. The former are responsible for broadcasting messages in a flooding manner and locally running the DEDA framework. The latter will not broadcast to neighbors after receiving messages yet they send data reports to the closest dominating unit rather than carrying out the DEDA data aggregation process. The details and examples are shown later on in Sec. 4.2.

The sample network in Fig. 4.1 will be used to illustrate the data aggregation process conducted by A-DEDA and AC-DEDA.

4.1 DEDA with Area Coverage Algorithm (A-DEDA)

The first improvement to the framework outlined in Chapter 3 is A-DEDA which adopts a localized area coverage algorithm [4] to select an active sensor set. If one sensor timeouts, it has to decide whether to stay active or transit into sleep mode based on localized area coverage algorithm. If its monitoring area has been covered by other active neighbors, it will switch into sleeping mode and do not broadcast any "hello" message. Otherwise, it will stay active and broadcast "hello" messages. Active sensors monitor the environment and generate reports whereas redundant sensors switch to a low-activity mode for energy saving. DEDA is therefore run only on active nodes. Considering that sensors with shorter timeout t will have a higher opportunity to stay active, t is inversely proportional to the remaining nodal energy level. This definition favors sensors with more residual energy.

Fig. 4.2 – Fig. 4.5 indicate the aggregation process on the sample network performed by A-DEDA. For illustrative purposes, we assume that there is more residual energy in nodes with lower IDs. By the area coverage algorithm, three sensors (circles) with higher IDs are scheduled to sleep. The rest of the sensors (solid) stay active and locally run the DEDA framework. Fig. 4.2 portrays the network status after the execution of the area coverage scheme. An initial data aggregation tree shown in Fig. 4.4 over these active nodes is built on LMST and shown in Fig. 4.3. Taking sensor **53** as an example, we notice that its monitoring area is covered by its neighbors **2, 9, 20, 25, 30, 33, 39**. When sensor **53** timeouts and decides whether to stay active or not, it has received hello messages from all its neighbors which have smaller ids. So, **53** applies the logic behind the localized area coverage algorithm and finally gets to know that it does not need to stay

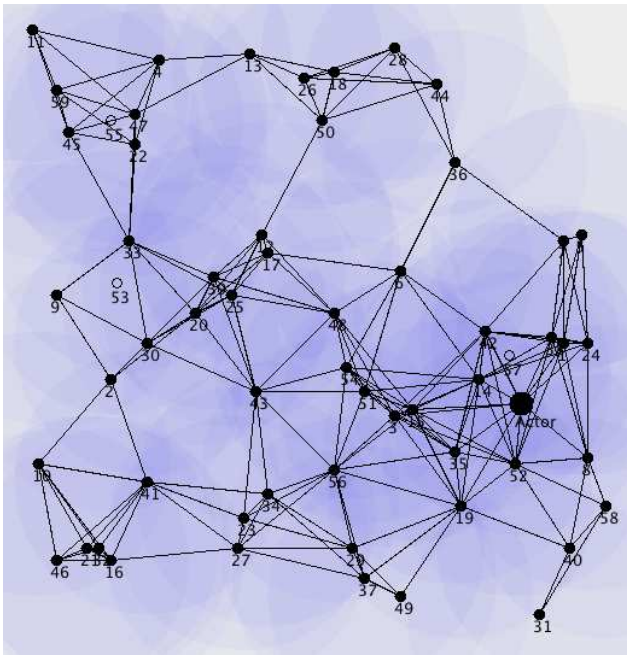


Figure 4.2: Node statuses after execution of the area coverage algorithm.

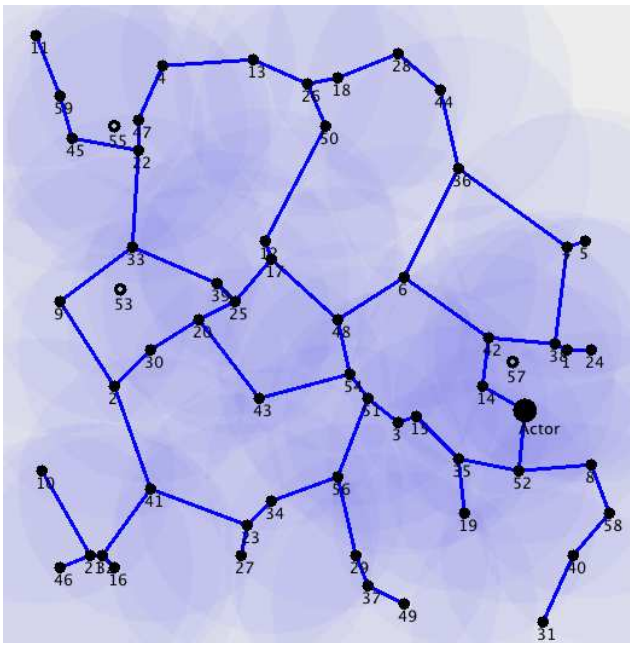


Figure 4.3: LMST built on active nodes.

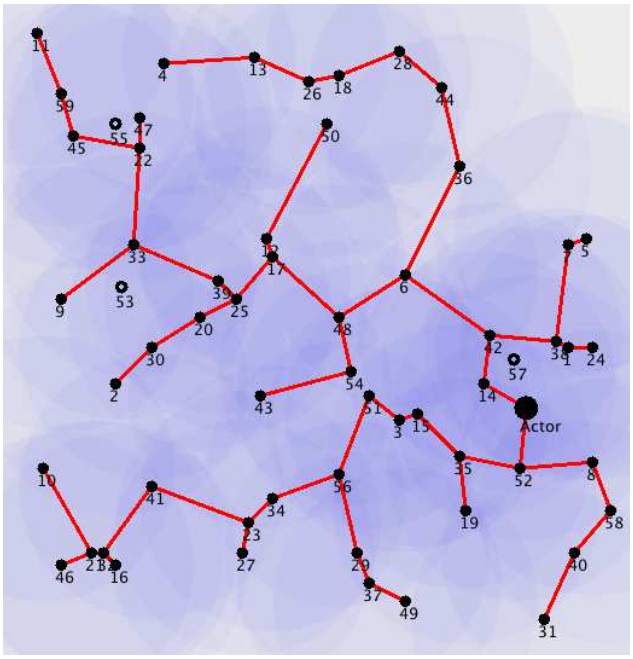


Figure 4.4: Initial data aggregation tree.

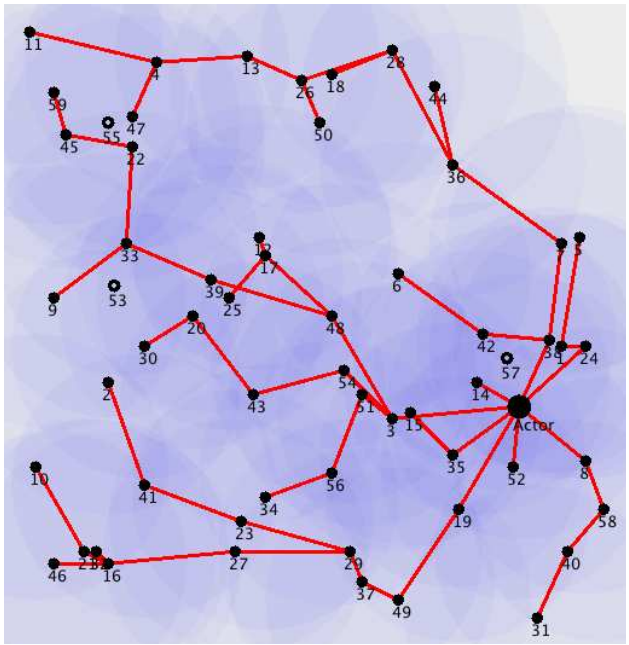


Figure 4.5: Final data aggregation tree (DL = 9 * 6).

active because its monitoring area has been fully covered by its active neighbors.

All active sensors apply DEDA algorithm to build the final data aggregation tree. They adjust their parent selections according to the DEP values locally calculated and the delay limit required by the application, which is (9*6) in this case. The final data aggregation tree is displayed in Fig. 4.5. Because the sleeping sensors save a large amount of energy, they will be switched on when their neighbors are close to depletion. In this way, the entire area can still be monitored by enough sensors. In order to avoid the energy imbalance, the status of each sensor will be reevaluated based on the new energy conditions that will periodically arise.

4.2 DEDA with Area Coverage Algorithm and Connected Dominating Set (AC-DEDA)

The second variant, AC-DEDA, is a combination of A-DEDA and the CDS construction algorithm in [5]. In this implementation, the CDS localized formation protocol is run on the active sensors determined by the area coverage algorithm [4]. Each active sensor either belongs to the CDS or has a direct neighbor which is part of the CDS. Non-CDS sensors report to the closest CDS neighbor while CDS units run DEDA for data aggregation.

Fig. 4.6 – Fig. 4.9 illustrate the data aggregation process conducted by AC-DEDA in the sample network of Fig. 4.1. In the CDS construction algorithm, sensors with smaller IDs have a higher priority to be in the CDS so that the nodes with more energy will become CDS members with higher likelihood.

Before data aggregation, each node could be one of: active node in the CDS (solid point), active node not in the CDS (square), or sleeping node (circle). The nodes' states are displayed in Fig. 4.6. For instance, sensor **44** has five neighbors.

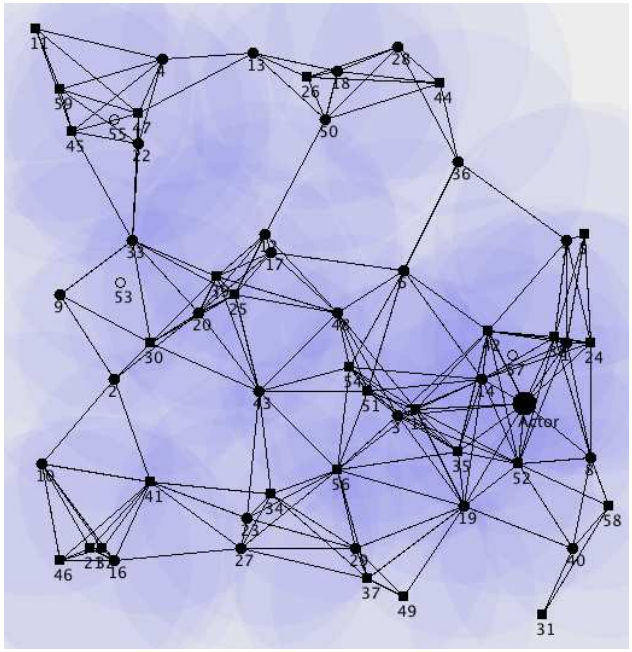


Figure 4.6: Connected dominating set built on the active nodes.

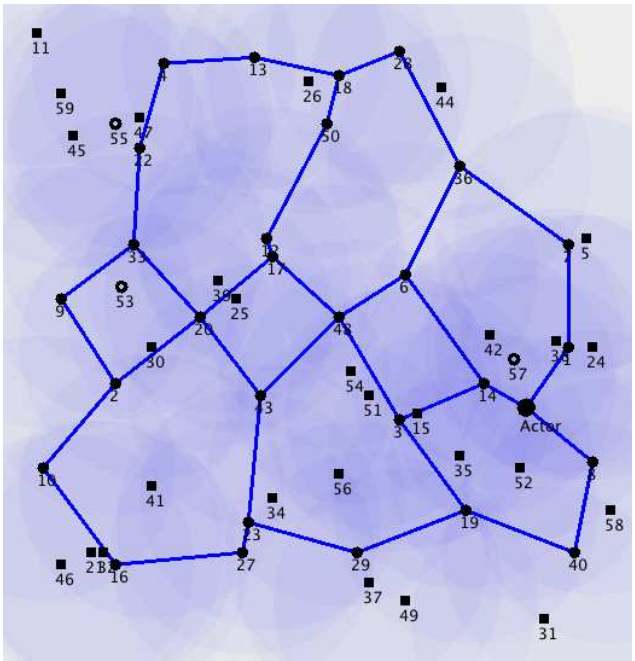


Figure 4.7: LMST built on the CDS nodes.

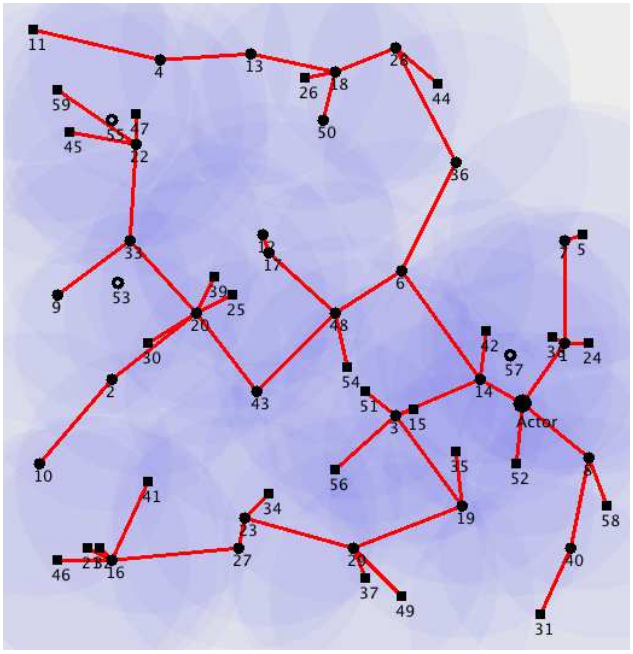


Figure 4.8: Initial data aggregation tree.

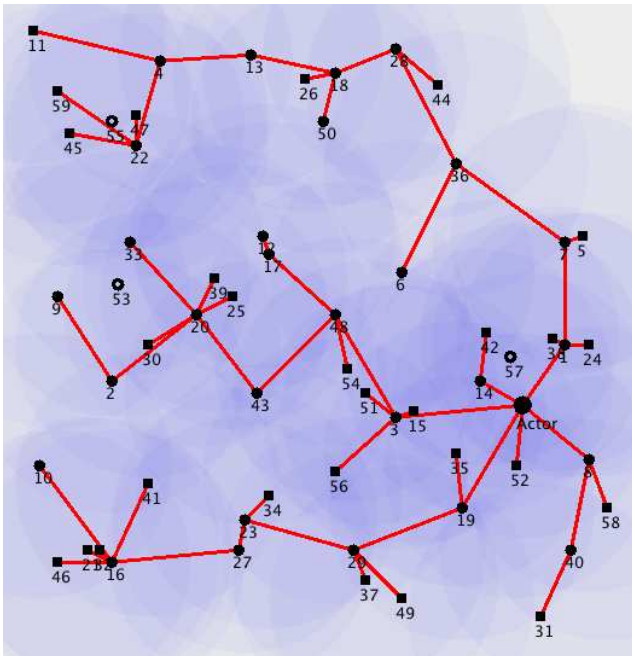


Figure 4.9: Final data aggregation tree (DL = 9 * 6).

Because neighboring nodes **18, 26, 28** with smaller IDs are connected to each other, and the other two neighbors **36** and **50** are connected to neighbor **28**, then **44** is covered by neighbors **18, 26, 28**. Therefore, sensor **44** becomes a non-CDS node.

Whenever data aggregation is required, a request message is broadcast by the actor in the LMST made up of CDS nodes (Fig. 4.7). After the flooding phase, an initial data aggregation tree is built as in Fig. 4.8. Non-CDS nodes attach to the tree via their closest CDS neighbors while CDS nodes implement the DEDA framework to determine their progress in the aggregation tree. You may notice that the initial tree is built upon CDS nodes solely. The difference with A-DEDA is that the request message is broadcast only along the CDS members. In this way, the message overhead is largely reduced and network congestion is thus prevented.

The final data aggregation tree formed by AC-DEDA is unveiled in Fig. 4.9. AC-DEDA combines both localized area coverage and connected dominating set algorithms. The former allows some sensors (with low remaining energy) to switch into sleeping mode which saves a significant share of the energy. The latter divides the active sensors into two groups: CDS nodes, which are responsible for shaping the backbone of the network and deal with broadcasting request messages and non-CDS nodes, which deliver data reports to their closest CDS neighbor. As a result, non-CDS units save more energy because they are not involved in any broadcasting task. As we have mentioned before, a node's ID is inversely proportional to its remaining battery level. So, if one node possesses little energy left, it will have a larger node ID which makes it transition into sleeping mode with a higher probability. If this is not the case, the node will become a non-CDS member with higher priority because of its larger ID. In a nutshell, AC-DEDA provides more room for low-energy nodes to spare power while perfectly satisfying the delay limit requirement as in the original DEDA framework.

Chapter 5

Simulations and Performance Evaluations

This chapter studies, through simulations, the performance (i.e. energy efficiency and delay reliability) of the DHP and DEDA frameworks under a realistic MAC layer with two stages (unsaturated and saturated). As visualized in Fig. 3.1, the relationship between queuing delay and the number of MAC competitors varies under different traffic conditions. For low-traffic networks or unsaturated channels, the delay is almost the same for different number of competitors. So, the total delay time is almost proportional to the hop sum along the path from the source node to the actor. Conversely, for high-traffic networks or saturated channels, the delay is proportional to the number of MAC competitors. In this case, the total delay time is almost proportional to the degree sum along the path from the source unit to the actor.

Another hop-based, delay-bounded protocol, MS [31], will also be evaluated for comparison purposes. We use average node energy consumption and delay reliability ratio as the performance metrics. Without loss of fairness, we omitted the

energy consumption in aggregation processing and idle state, which are identical for all tested protocols.

5.1 Network Setup

All simulations are implemented in a static wireless sensor network of *60* nodes including a single actor, using the Glomosim network simulator with IEEE 802.11 CSMA/CA protocol at the MAC layer. The ratio of communication radius to sensing radius is *2*. All sensor nodes have the same initial energy amount. The first-order radio model $e = \beta d^{\alpha} + c$ is adopted as the energy model for transmission. We set $\beta = 100\ pJ/bit/m^{\alpha}$ and $\alpha = 4$. In the experiments, we varied the electronic unit energy consumption c, the size of the region of interest (ROI) l^2, the average node degree d and the delay limit DL. These parameters have proved to exert a direct impact on the protocols' performance. In order to control d, we use different signal strength values to change communication range because we can obtain d from the formula $d = (N-1)\pi R^2/A$, where R is the communication radius and A is the area of the deployment region.

As we have already mentioned, the relation between delay time and number of MAC competitors fluctuates under different traffic conditions. Aided by our simulation environment, we take a note at the empirically observed results and figure out the functional form of the relation. In Fig. 5.1, when the number of MAC competitors is below about *8*, the average delay time is roughly *6500* μs but when there are more than eight entries competing for the medium, the average delay time is proportional to the number of entries and the function takes approximately the form of $T = 1000 * n$.

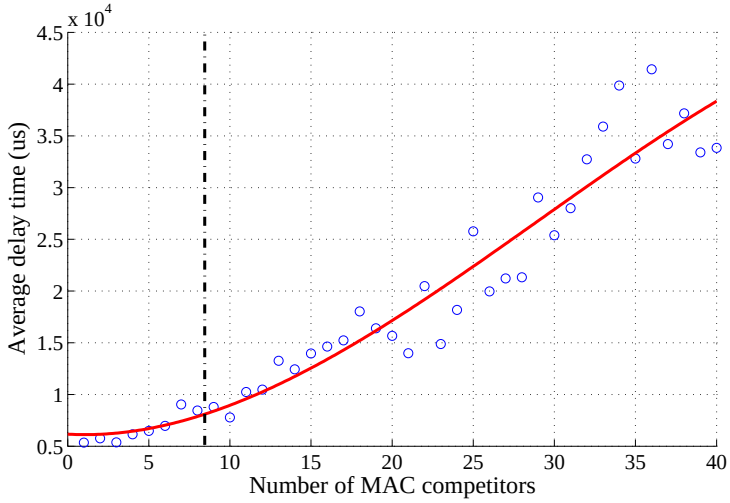


Figure 5.1: Relation between average delay time and number of MAC competitors.

5.2 Model Analysis

As we have discussed, the realistic model is composed of two stages. Before the saturation point or in a low-traffic channel, the delay time is approximately proportional to the hop sum along the path. So we can use the hop-based DHP to perform data aggregation in an energy-efficient and delay-bounded manner. This means that, if we know the delay limit in time units T, we can express the delay limit in terms of hop sum as $DL = T/T_1$ (T_1 is about *6500* μs in our simulation environment). Now we can use DL to generate the data aggregation tree by means of the hop-based DHP scheme.

But in fact, when data are being collected, the traffic is always very high and the waiting queue is seldom empty. As the channel gets saturated, the delay time would be proportional to the degree sum along the path from the source node to the actor. Hence, our degree-based DEDA framework is suitable for data aggregation purposes under this circumstance. In a similar way as with hop-based DHP, this algorithm will do its best to satisfy the delay limit (degree sum) and meanwhile

use more LMST edges to save energy. Based on the delay limit T specified by the user in time units, we can compute the delay limit in terms of degree. The relation is dependent on a realistic network setup and we can derive it out of several tests. Fig. 5.1 shows our test result: $T_1 = 1000 * n$. So we can calculate the entire path delay time $T = 1000 * (degree\ sum)$ and then work out the value for DL (degree sum) from T, which is used to generate the data aggregation tree by our degree-based DEDA protocol.

5.3 Simulation Setup

We implemented hop-based DHP and its two variants (DHPA and DHPAC) under the above set of network assumptions. The initial data aggregation tree is created as a preliminary step, so that every sensor becomes cognizant of the minimum distance to the actor in the UDG and LMST graphs. Whenever the actor would like to collect information from the network, it broadcasts a *REQUEST* message all over the initial data aggregation tree. After receiving this message, every sensor selects its parent and children in order to satisfy the delay limit and save as much energy as possible. The result will be conveyed as a *PARENT DECISION* message so that every sensor knows its own parent and all its children. Because of this, data fusion can be used while data are still being collected. If any sensor suddenly shuts down, its neighbors will realize that this child failed by *BEACON* messages and notify the actor through an *ERROR* message, which prompts the actor to reconstruct the data aggregation tree.

The degree-based DEDA and its two variants (A-DEDA and AC-DEDA) have also been implemented on the basis of the aforementioned network setup. The experimental configuration is very similar to that of hop-based DHP except that

the distance is measured in degree sum instead of in hop sum.

The MS algorithm uses three routing subprocedures in three working states respectively. These subprocedures are named after their states as follows: start-up procedure, greedy procedure and aggregation procedure. MS relies on a state-switching scheme to control the sensor's routing subprocedure selection. In the switching scheme, there are several parameters that deserve a watchful configuration. We apply a similar rationale to the one in [42]:

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

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

$P_{switching}$: State switching probability.

In our simulations, r_{th} is set to the reliable ratio of DHP under the same scenario; $\varepsilon = 3\%$, and $P_{switching} = 70\%$. The actor computes these parameters and releases feedbacks after receiving every data packet to inform sensors that they can make a working state decision.

The MS protocol is a kind of greedy algorithm based on the hop sum criterion. So, for the degree-based DEDA method, MS is not suitable for comparison anymore. In other words, MS cannot adjust its behavior to operate in a realistic network with a high traffic pattern because its delay consideration is rooted on hop sum.

5.4 Energy Consumption Evaluation

5.4.1 Energy Consumption for Hop-based DHP

Delay limit DL is measured by the number of hops. The selection of an appropriate DL value has an obvious bearing on the reliability of the data aggregation mechanisms enforced by the algorithms under discussion. Considering that state transitions in MS are determined by data aggregation reliability ratios and r_{th}, we choose two delay limit settings: DL = 4 and DL = 9. In the former case, frequent state transitions are implemented to reach an expected reliability ratio in data aggregation by MS. In the latter case, reliability ratios by DHP and the start-up protocol can reach *100%* in all graphs. Then, feedbacks from the actor recommend nodes in MS to stay in start-up mode since their reliability ratios can match r_{th} (*100%*) well, i.e. no state transition is necessary. Finally, for each simulation setting we conducted *50* independent runs with randomly generated network graphs to compute average results. The length of the different types of messages is given as:

Length (REQUEST message) = *80 bits*
Length (PARENT DECISION message) = *50 bits*
Length (ERROR message) = *10 bits*
Length (DATA) = *200 bits*

5.4.1.1 Impact of Electric Unit Energy Consumption

We first study the impact of constant c in the energy model over the protocols' performance. We set the value of c to *50 pJ/bit, 50 nJ/bit, 800 nJ/bit, 5 μJ/bit* and *16* μJ/bit, respectively. The chosen value for the average node degree was *6* and a ROI of size *100 m*$\times$*100 m* was envisioned. The simulation results are listed in Fig.

5.2. Observe that as c grows, both DHP and MS exhibit a degrading performance. This is because the energy consumption per hop rises with the augment of a first-order energy function. However, DHP shows better performance than MS all along. Whether DL = 4 or DL = 9, DHP saves about *40%* of the energy over MS. As the average node degree is only *6*, few nodes can switch to sleeping mode and therefore DHPA only saves about *10%* of energy with respect to DHP. Because the number of messages broadcast in the connected dominating set decreases, DHPAC saves roughly *25%* of the energy compared to DHP.

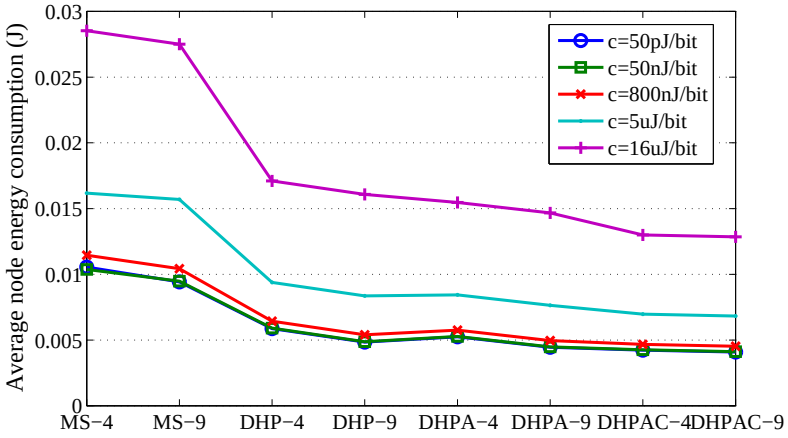


Figure 5.2: Average nodal energy consumption for different values of c and DL.

5.4.1.2 Impact of Area Size

Hop selections in DHP and MS do not depend on the size $l \times l$ of the ROI. However, in terms of energy efficiency, the actual hop distance greatly affects the performance of the protocols. In order to examine the impact of the ROI size on the algorithmic solutions, we run simulations with l equal to *50 m, 100 m, 150 m, 200 m* and *250 m*. The average node degree is fixed to *6* and c was set to *50 pJ/bit*. In this case and with a random node distribution, the larger the ROI, the longer the communication

link on average and therefore the more energy is consumed. Empirical results in Fig. 5.3 point out to this phenomenon clearly.

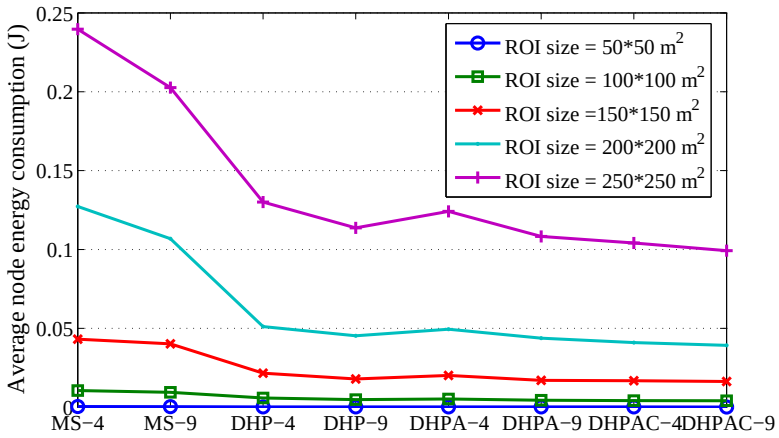


Figure 5.3: Average node energy consumption for different l and DL.

From Fig. 5.3 it is observed that the DHP family exhibits a significantly better performance than MS in areas with varying sizes. As the ROI becomes larger, more transmission energy will be spent with a larger communication range. Whether DL = 4 or DL = 9, the energy consumption of DHP is *55%* of that of MS. Yet DHPA only saves *5%* - *10%* of the energy in DHP whereas DHPAC spares *10%* - *20%* of DHP's power because of the low average node degree.

5.4.1.3 Impact of Network Density

For this experiment, we set the average node degree d to *6, 8, 10, 12, 14* respectively and fix the electronic unit energy consumption constant c to *50 pJ/bit* and the size of the ROI to *100 m* $\times$ *100 m*. When the network size is static, the larger the average node degree (d), the more compact the network turns out to be. Compactness here implies network diameter. Recall that, in our simulations, a desired average node degree is obtained by simply adjusting the nodal maximum

transmission power. This means that more energy will be expended for transmission of the request messages at each node. Hence, as d goes up, the two protocols show an increasing trend in energy consumption. As reported in Fig. 5.4, energy expenditures grow for any protocol as the average node degree goes up.

Fig. 5.4 plots the average energy consumption per node under different average node degrees. In general, nodes save *30%* - *50%* of the energy in DHP when contrasted to MS. For DHPA and DHPAC, sensors can save more energy when their average degree rises because more nodes can switch into sleep mode. DHPA saves *2%* more with respect to DHP when the degree equals *6*, while it economizes *50%* when the degree is *14*. For DHPAC, it saves about *60%* - *70%* of the energy compared to DHP. It can also spare more than that when the average node degree gets bigger because the CDS can substantially lower the number of broadcast messages that circulate in dense networks.

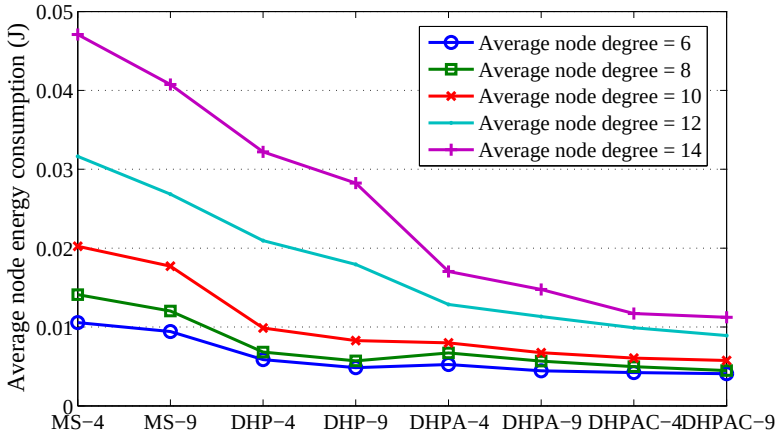


Figure 5.4: Average node energy consumption for different average node degrees.

5.4.2 Energy Consumption for Degree-based DEDA

Delay limit is measured by the degree sum, which can be computed from the delay limit time as we have discussed earlier. In order to homogenize the simulation setups for DHP and DEDA, we just assign *DL = 4*(average node degree) and DL = 9*(average node degree)*, which is similar to the hop-based DHP simulation in the above section.

5.4.2.1 Impact of Electric Unit Energy Consumption

As in the hop-based DHP empirical study, we will at first shed light on the impact of constant c in the energy model over the DEDA protocol's performance. We also set the value of c to *50 pJ/bit, 50 nJ/bit, 800 nJ/bit, 5 μJ/bit* and *16 μJ/bit* and use average node degree *14* and a ROI of size *100 m* $\times$ *100 m*. The simulation results are listed in Fig. 5.5. As c increases, the energy consumption rises up as well for DEDA and its two variants. For the same constant c, A-DEDA and AC-DEDA exhibit a better performance than DEDA regardless of whether DL = 4 or DL = 9. A-DEDA readily saves about *50%* of the total energy over DEDA while AC-DEDA yields *75%* - *80%* energy saving rates over DEDA. The performance improvement ratio is better than that of hop-based DHP because the average node degree is larger, so more nodes can switch into sleep mode and more messages can be eliminated in the connected dominating set.

5.4.2.2 Impact of Area Size

In the hop-based DHP performance evaluation section, we took the impact of the area size into account. In the same way, we will use five different ROI sizes in the simulation of the degree-based DEDA. The average node degree is fixed to *14* and c to *50 pJ/bit*. As we have found out before, the larger the ROI, the longer the

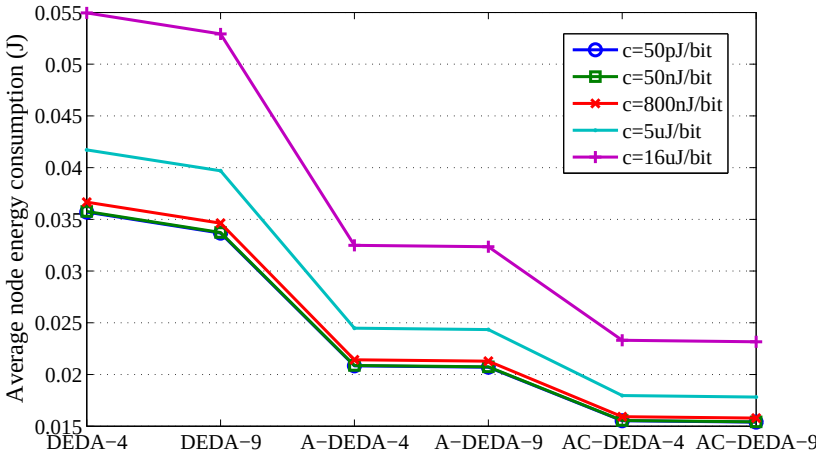


Figure 5.5: Average node energy consumption for different c and DL.

communication link is on average and therefore the more energy is consumed. In Fig. 5.6, we schematically depict the area size vs. energy consumption relationship.

When DL = 4 or DL = 9, A-DEDA spends *60%* of the energy required by DEDA and AC-DEDA consumes *75%* compared to DEDA. The performance improvement ratio is also superior than that of the hop-based DHP algorithm because of the more compact network and the existence of the connected dominating set.

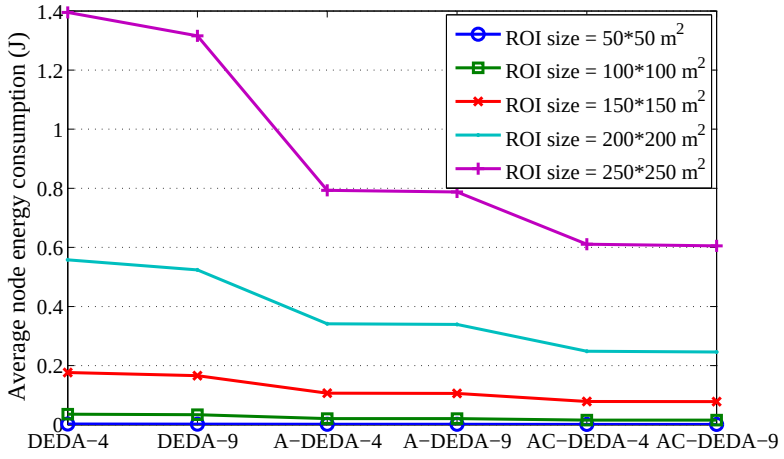


Figure 5.6: Average node energy consumption for different l and DL.

5.4.2.3 Impact of Network Density

As done analogously for hop-based DHP, the average node degree d has been tested with the following values: *6, 8, 10, 12, 14* and the electronic unit energy consumption constant c has been fixed to *50 pJ/bit* while the size of the ROI was defined as *100 m* $\times$ *100 m*. When the network size is fixed, the larger the average node degree (d), the more compact the network becomes.

An increase in compactness entails more broadcast messages flooding the network and hence a higher energy consumption rate per node. Fig. 5.7 discloses the nature of the network density vs. energy consumption relationship.

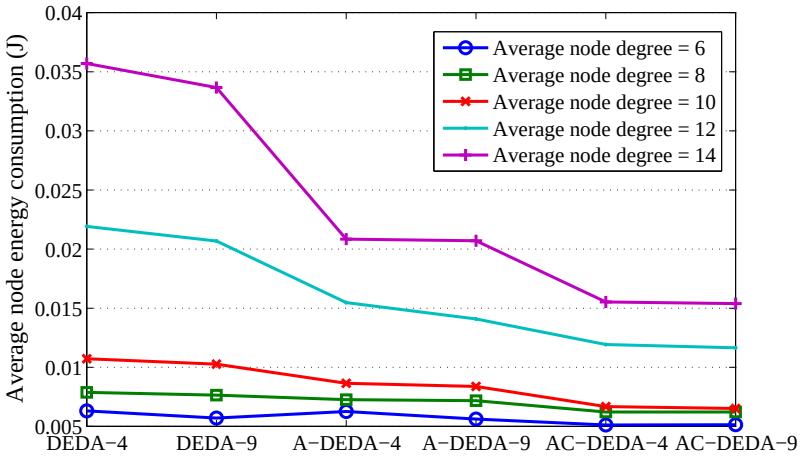


Figure 5.7: Average node energy consumption for different average node degrees.

One may notice that with A-DEDA and AC-DEDA the nodes can save more energy as their average degree climbs up. This is due to the fact that more nodes can switch to sleep mode. So in this figure, the spent energy grows faster for DEDA than for A-DEDA or AC-DEDA as the network density augments. Moreover, the gap between DEDA and its two variants is enlarged with the network density. AC-DEDA is able to spare about *10% - 30%* of the energy compared to A-DEDA. Concerning A-DEDA, it saves about *10% - 45%* of the energy with respect to

DEDA.

As for the comparison with hop-based DHP, the degree-based DEDA framework consumes a little bit more energy (approximately *10%*) when the average node degree is large. The reason is that degree-based DEDA reflects the real delay patterns of a high-traffic network, thus it may need to select some energy-consuming links to satisfy the delay requirement. As a result, the energy consumption rate is a tad higher than in hop-based DHP.

Since we have mentioned the delay issue across the two protocols, we will contrast as well the delay reliability ratios in the following section.

5.5 Delay Reliability Evaluation

5.5.1 Delay Reliability Evaluation for Hop-based DHP

5.5.1.1 Impact of Delay Limit Requirement

In order to explore the impact of different delay limits on the delay reliability ratio, five different delay limits have been chosen, viz. (*25000 μs, 35000 μs, 45000 μs, 55000 μs and 65000 μs*) to test our protocol. We know that the number of hops required to convey the information from the source to the actor can be figured out as per the function discussed above. Therefore, five corresponding hop counts, namely (*4, 5, 7, 8* and *10*) have been derived for use in the hop-based DHP algorithm. Because hop-based DHP is applicable to low-traffic networks alone, the average node degree has been correspondingly anchored to the value of *6*. The delay reliability ratio chart is given in Fig. 5.8.

For DHP, the reliability ratio grows with a larger delay limit time. When the delay limit time is *25000 μs*, the delay reliability ratio is about *35%*; however if

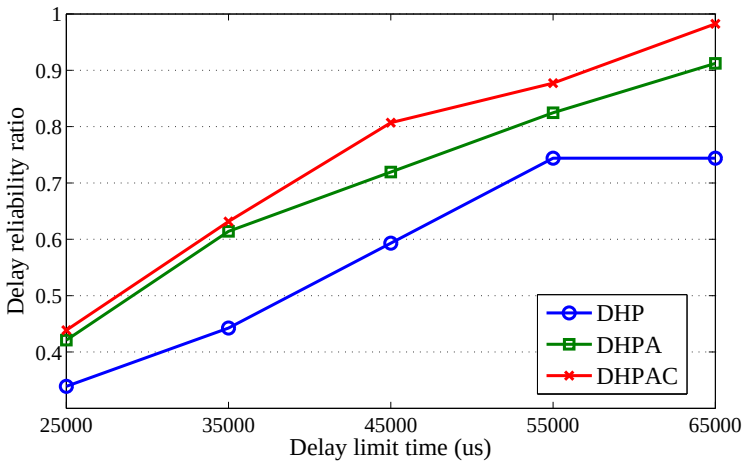


Figure 5.8: Delay reliability ratio for different values of delay limit time.

the delay limit time is extended to *65000* μs, the corresponding ratio is doubled to *75%*. For its two variants, DHPA and DHPAC, the reliability ratio undergoes a more noticeable surge from *40%* to *95%*.

5.5.1.2 Impact of Average Node Degree

The advice to deploy more nodes in a region is not always good. Network density is a serious factor that affects the delay reliability ratio because more messages will likely involved as there are more nodes in the network. So, we choose different average node degrees (from *4* to *20*) and set the delay limit time to *65000* μs to test our protocols. The results are unwound in Fig. 5.9.

As discussed above, the hop-based DHP method is adaptive to low-traffic networks. So, in Fig. 5.9, the delay reliability ratio decreases as the average node degree climbs up for DHP. For instance, the ratio is about *85%* when the average nodal degree is *4* yet it falls to about *35%* as soon as the average degree reaches *20*. Fortunately, the problem can be solved with area coverage and connected dominating set algorithms like DHPA and DHPAC because they shrink the traffic

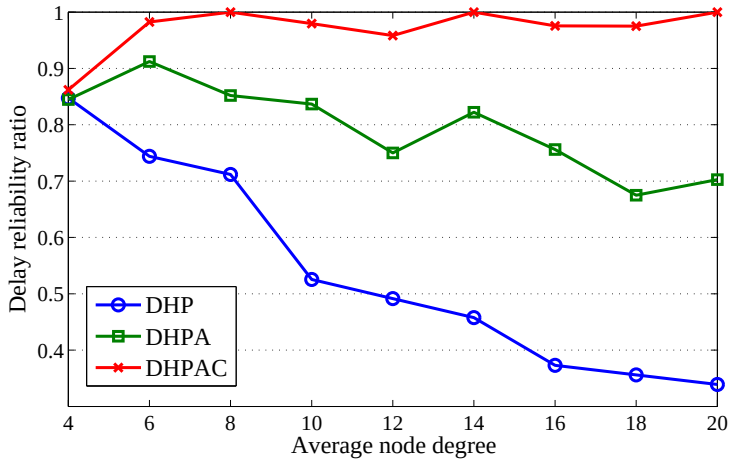


Figure 5.9: Delay reliability ratio for different average node degrees.

congestion to a large extent. Observe how DHPA can still achieve about *70%* in delay reliability ratio when the average node degree is *20* and DHPAC even gets *100%* for the same value of nodal degree.

5.5.2 Delay Reliability Evaluation for Degree-based DEDA

5.5.2.1 Impact of Delay Limit Requirement

We will test our degree-based DEDA under different delay limit times, ranging from *25000* μs to *65000* μs. The average node degree is settled to *14*. The obtained results are analogous to those of the hop-based DHP scheme. With the increase of delay limit time, the delay reliability ratio is also stretched. Details are shown in Fig. 5.10.

When the delay limit time is *25000* μs, DEDA accomplishes *40%* of the delay reliability ratio but A-DEDA attains *60%* and AC-DEDA reaches *70%*. If the delay limit time is increased to *65000* μs, all the algorithms soar above *90%* of delay reliability ratio. It is crystal clear to see that it is easier to satisfy a larger delay limit time. This is especially true for AC-DEDA, which is able to achieve

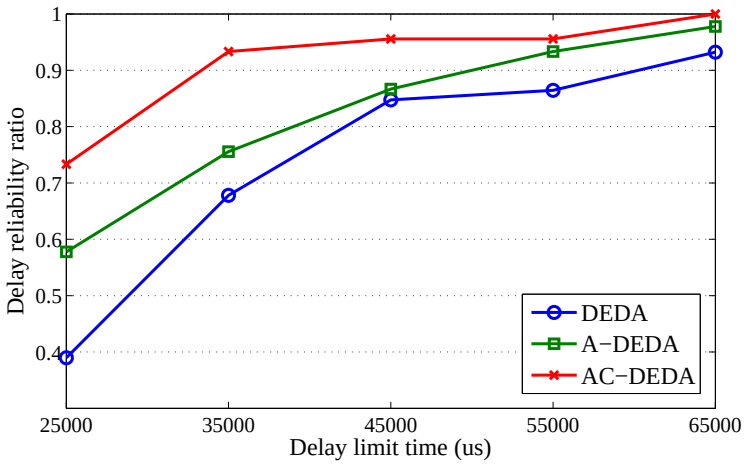


Figure 5.10: Delay reliability ratio for different delay limit times.

more than *90%* reliability ratio since the delay limit time is larger than *35000* μs.

5.5.2.2 Impact of Average Node Degree

The degree-based DEDA was evaluated under fixed delay limit time *65000* μs and a varying average node degree sweeping from *4* to *20*. As one can behold in Fig. 5.11, the delay reliability ratio is almost *100%* when the average node degree is beyond *8*. On the other hand, when the average node degree is low, its performance is almost as good as hop-based DHP. Generally speaking, AC-DEDA outperforms A-DEDA, which in turn excels plain DEDA.

5.5.3 Delay Reliability Comparison between Hop-based DHP and Degree-based DEDA

In this section, we will underscore the performance differences between hop-based DHP and degree-based DEDA under different traffic conditions and delay limit times.

Fig. 5.12 illustrates the delay reliability ratio when the delay limit time is

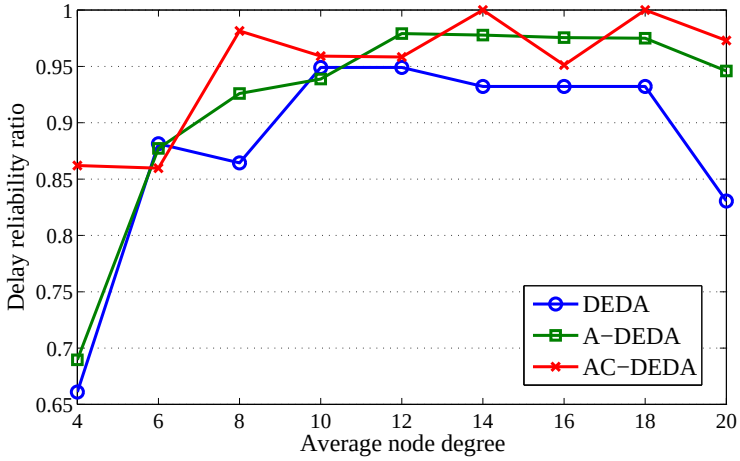


Figure 5.11: Delay reliability ratio for different average node degrees.

25000 μs. Because the delay limit is so strict, the ratio is relatively low for all protocols. When the average node degree is below *10*, hop-based DHP and degree-based DEDA are almost the same at *40%* - *50%*. But when the average node degree goes beyond *10*, then hop-based DHP only yields a *20%* ratio whereas degree-based DEDA attains *45%*. That is the reason why degree-based DEDA is suitable for high-traffic networks but hop-based DHP is not. As for their other variants, degree-based DEDA is also better than hop-based DHP by *25%*.

In Fig. 5.13 and Fig. 5.14, the delay limit time is changed to *35000 μs* and *45000 μs*. When the traffic is low, the delay reliability ratios are the same for hop-based DHP and degree-based DEDA. Yet when the traffic is high, the delay reliability ratio rises for the DEDA family but declines for the DHP family. For instance, if the average node degree is *20*, the ratio is greater than *60%* for degree-based DEDA and about *25%* for hop-based DHP.

In Fig. 5.15 and Fig. 5.16, the delay limit time is enlarged to *55000 μs* and *65000 μs*. You may realize that the delay reliability ratios of all protocols are above *70%* when the average node degree is below *10*. If it goes up, the ratio is brought

down below *70%* for hop-based DHP/DHPA. The other protocols all get a high delay reliability ratio (above *80%*), especially for the one combined with connected dominating set, which can reach about *100%*.

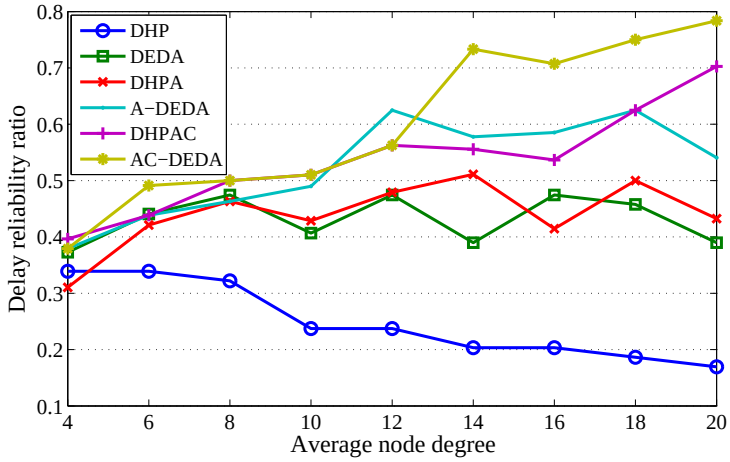


Figure 5.12: Delay reliability ratio under different average node degrees when delay limit time is 25000 μs.

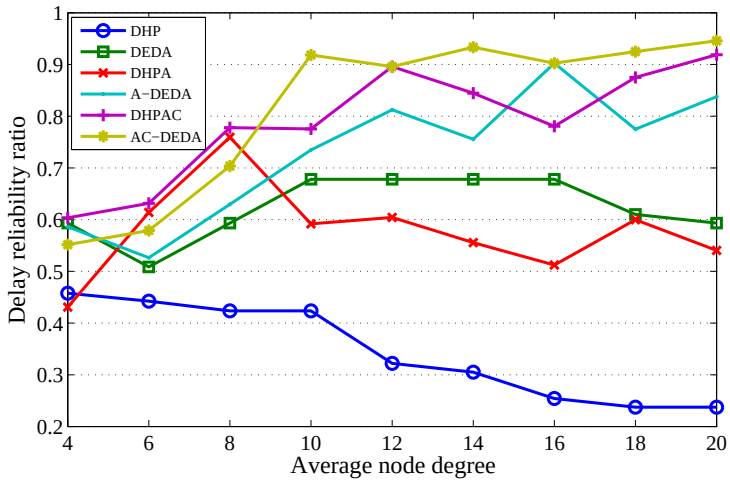


Figure 5.13: Delay reliability ratio under different average node degrees when delay limit time is 35000 μs.

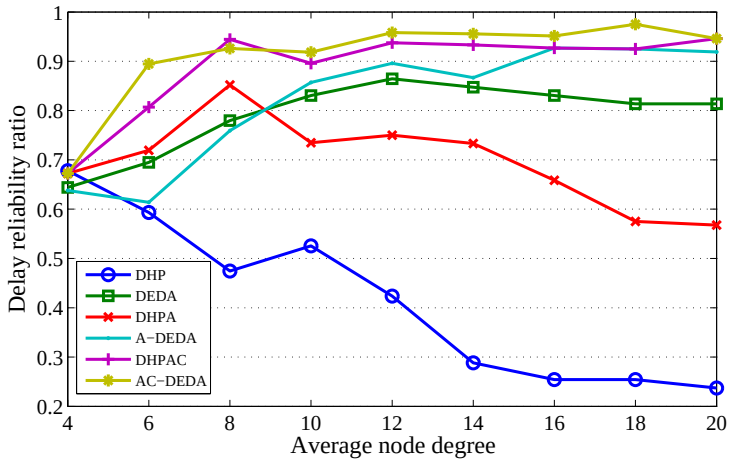


Figure 5.14: Delay reliability ratio under different average node degrees when delay limit time is 45000 μs.

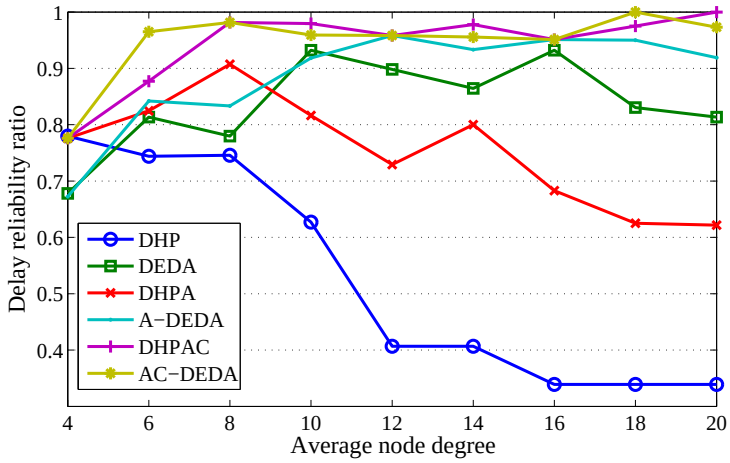


Figure 5.15: Delay reliability ratio under different average node degrees when delay limit time is 55000 μs.

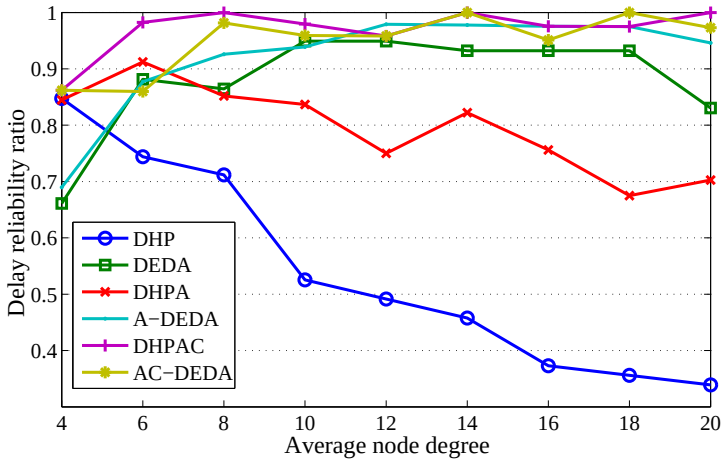


Figure 5.16: Delay reliability ratio under different average node degrees when delay limit time is 65000 μs.

Chapter 6

Conclusions and Future Works

In this thesis, we introduced a localized delay-bounded and energy-efficient data aggregation framework called DEDA, which solves the delay-bounded aggregation problem with high energy efficiency using realistic delay model.

This framework initially creates a power-efficient aggregation tree and subsequently refines it according to the DEsired Progress (DEP) value computed at each node in order to satisfy the different delay limit requirements.

We also presented two DEDA variants, A-DEDA and AC-DEDA, by expanding the canonical DEDA formulation with an area coverage algorithm [4] and a localized connected dominating set formation approach [5]. Compared to DEDA, the two improved versions save more energy and better balance the energy consumption during the data aggregation process because of backbone construction and activity scheduling.

Simulation results demonstrated that DEDA (and its two offspring) achieve almost the same energy efficiency as DHP [42] framework, which is much more energy-efficient in overall consumption and distribution than MS [31]. A more sophisticated delay model applied in DEDA endows the algorithm with superior

reliability on delay limit when contrasted to other existing hop-based approaches such as DHP and MS, which do not reflect well the realistic delay modeling.

Our DEDA framework is based on a realistic MAC layer modeled after a two-stage delay conception. When the traffic is low (unsaturated channel), the total delay along the path is approximately proportional to the hop count. When the traffic is high (saturated channel), it is nearly proportional to the degree sum. As a result, our DEDA framework can be applied on an IEEE 802.11 CSMA/CA MAC layer.

Finally, we evaluated the performance of the algorithms in terms of energy consumption and delay reliability ratio under different conditions and verified that the proposed framework attains high improvement rates.

The potential problem is that this data structure will not be suitable when only part of sensors need to send data back to sink. In this case, LMST does not optimize energy consumption and thus produces more energy overhead, and degree based delay model does not reflect the realistic delay condition because some nodes stay silent and thus will not jam the traffic. Therefore, designing a new data structure seems necessary to solve this problem.

DEDA can be extended to deal with mobile sensors or sinks. Even though beacon message can be used to update new locations of mobile nodes, the risk of package loss and relative reconstruction overhead of highly dynamic topology need to be further investigated in future works.

References

[1] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344, 1999.

[2] Y. Zhu, K. Sundaresan, and R. Sivakumar. Practical limits on achievable energy improvements and usable delay tolerance in correlation aware data aggregation in wireless sensor networks. *IEEE SECON*, 2005.

[3] N. Li, J.C. Hou, and L. Sha. Design and analysis of an MST-based topology control algorithm. *IEEE Transactions on Wireless Communications*, 4(3):1195–1206, 2005.

[4] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic. Localized sensor area coverage with low communication overhead. *IEEE PerCom*, 2006.

[5] J. Carle and D. Simplot-Ryl. Energy efficient area monitoring by sensor networks. *IEEE Computer Magazine*, 37(2):40–46, 2004.

[6] S. Lindsey, C. S. Raghavendra, and K. M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel Distributed System*, 13(9):924–935, 2002.

[7] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *5th symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[8] J. Chou, D. Petrovic, and K. Ramchandran. A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks. *IEEE INFOCOM*, 2003.

[9] A. Scaglione and S. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. *ACM MoBiCom*, 2002.

[10] E. Cohen and H. Kaplan. Spatially-decaying aggregation over a network: Model and algorithms. *ACM SIGMOD*, 2004.

[11] W. R. Heinzelman. *Application-specific protocol architectures for wireless networks.* PhD thesis, Massachusetts Institute of Technology, 2000.

[12] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. *33rd Hawaii International Conference on Systems Science (HICSS)*, 2000.

[13] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, 2002.

[14] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. *IEEE INFOCOM*, 2003.

[15] Y. Yao and J. Gehrke. Query processing in sensor networks. *ACM CIDR*, 2003.

[16] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 31(3):9–18, 2002.

[17] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. *IETF. RFC 3561*, 2003.

[18] O. Younis and S. Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, 2004.

[19] H. O. Tan and I. Korpeoglu. Power efficient data gathering and aggregation in wireless sensor networks. *ACM SIGMOD Record*, 32(4):66–71, 2003.

[20] M. Ding, X. Cheng, and G. Xue. Aggregation tree construction in sensor networks. *IEEE 58th Vehicular Technology Conference*, 4(4):2168–2172, 2003.

[21] N. Li, J. C. Hou, and L. Sha. Design and analysis of an mst-based topology control algorithm. *IEEE INFOCOM*, 2003.

[22] H. Tan, I. Korpeoglu, and I. Stojmenovic. A distributed and dynamic data gathering protocol for sensor networks. *IEEE AINA*, 2007.

[23] S. Lindsey, C. S. Raghavendra, and K. M. Sivalingam. Data gathering in sensor networks using the *energy*delay* metric. *IEEE IPDPS*, 2001.

[24] K. Du, J. Wu, and D. Zhou. Chain-based protocols for data broadcasting and gathering in sensor networks. *IEEE IPDPS*, 2003.

[25] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. TTDD: Two-tier data dissemination in large-scale wireless sensor networks. *Wireless Networks*, 11(1-2):161–175, 2005.

[26] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. *IEEE ICDCS*, 2002.

[27] S. Kwon, J. Kim, and C. Kim. An efficient tree structure for delay sensitive data aggregation in wsn. *IEEE AINA*, 2008.

[28] Y. Wu, X. Li, Y. Liu, and W. Lou. Energy-efficient wake-up scheduling for data collection and aggregation. *IEEE Transactions on Parallel Distributed System*, 21(2):275–287, 2010.

[29] X. Xu, X. Li, X. Mao, S. Tang, and S. Wang. A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(1):163–175, 2011.

[30] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. *IEEE INFOCOM*, 2004.

[31] T. Melodia, D. Pompili, V. Gungor, and I. Akyildiz. A distributed coordination framework for wireless sensor and actor networks. *ACM MobiHoc*, 2005.

[32] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. *IEEE ICDCS*, 2003.

[33] H. Zhang and J. C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc & Sensor Wireless Networks*, 1(1-2):89–124, 2005.

[34] D. Tian and N. D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. *ACM WSNA*, 2002.

[35] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. D. Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks*, 1(1):36–72, 2005.

[36] I. Stojmenovic and J. Wu. *Broadcasting and activity scheduling in ad hoc networks*. IEEE Press, 2004.

[37] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, 1997.

[38] Y. Xu, J. S. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. *ACM MoBiCom*, 2001.

[39] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. *35rd Hawaii International Conference on Systems Science (HICSS)*, 2002.

[40] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. *ACM DIALM*, 1999.

[41] F. Dai and J. Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Transactions on Parallel Distributed System*, 15(10):908–920, 2004.

[42] C. Xu. Delay-constrained power-efficient data aggregation framework in sensor-actor networks. Master's thesis, University of Ottawa, 2009.

[43] G. Bianchi. Performance analysis of IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[44] A. Banchs and L. Vollero. A delay model for IEEE 802.11e EDCA. *IEEE Communications Letters*, 9(6):508–510, 2005.

[45] J. Liu and Z. Niu. Delay analysis of IEEE 802.11e EDCA under unsaturated conditions. *IEEE WCNC*, 2007.

[46] D. Malone, K. Duffy, and D. Leith. Modeling the 802.11 distributed coordination function in non-saturated heterogeneous conditions. *ACM/IEEE Transactions on Network*, 15(1):159–172, 2007.