Semi-Asynchronous and Distributed Weighted Connected Dominating Set Algorithms for Wireless Sensor Networks

Orhan Dagdeviren[a], Kayhan Erciyes[b], Savio Tse[c]

[a]*Ege University, International Computer Institute, Bornova, Izmir 35100, Turkey*
[b]*Izmir University, Computer Engineering Dept., Uckuyular, Izmir 35350, Turkey*
[c]*Istanbul University, Computer Engineering Dept., Avcilar, Istanbul 34320, Turkey*

Abstract

Energy-efficient backbone construction is one of the most important objective in a wireless sensor network (WSN) and to construct a more robust backbone, weighted connected dominating sets can be used where the energy of the nodes are directly related to their weights. In this study, we propose algorithms for this purpose and classify our algorithms as weighted dominating set algorithms and weighted Steiner tree algorithms where these algorithms are used together to construct a weighted connected dominating set (WCDS). We provide fully distributed algorithms with semi-asynchronous versions. We show the design of the algorithms, analyze their proof of correctness, time, message and space complexities and provide the simulation results in ns2 environment. We show that the approximation ratio of our algorithms is $3\ln(S)$ where S is the total weight of optimum solution. To the best of our knowledge, our algorithms are the first fully distributed and semi-asynchronous WCDS algorithms with $3\ln(S)$ approximation ratio. We compare our proposed algorithms with the related work and show that our algorithms select backbone with lower cost and less number of nodes.

Keywords: wireless sensor networks, weighted connected dominating set, distributed approximation algorithm, backbone formation

1. Introduction

WSNs do not have any fixed infrastructure and consist of sensor nodes that perform sensing and communicating tasks. Habitat monitoring, health care, environmental control, military surveillance and target tracking are example application areas of WSNs [1, 2, 3]. Backbone formation to construct a robust communication structure is a significant research area in WSNs. Backbones are provided in WSNs in order to decrease the number of messages and total time spent for routing the sensed data to the sink. By clustering the network and backbone formation, an energy-efficient topology is constructed which makes routing and data aggregation tasks easier. In clustering schemes, each node is classified as either cluster head or cluster member. Cluster members are ordinary nodes whereas cluster heads perform various task on behalf of the members of the clusters.

A WSN can be modeled as a graph $G(V, E)$ where V is the set of vertices (nodes of WSN) and E is the set of edges (communication links between the nodes). A connected dominating set is a subset S of a graph G such that S forms a dominating set and is connected. CDSs have many advantages in network applications such as ease of broadcasting and constructing virtual backbones [4]. Due to this fact, CDSs have been extensively studied by researchers [5, 6, 7, 4, 8, 9, 10]. Existing CDS algorithms generally aim at minimizing the number of backbone nodes without considering other issues, on the other hand energy-efficient cluster head selection is very important for sensor networks. In the weighted connected dominating set (WCDS) construction

Email addresses: orhan.dagdeviren@ege.edu.tr,orhandagdeviren@gmail.com (Orhan Dagdeviren), kayhan.erciyes@izmir.edu.tr (Kayhan Erciyes), sshtse@istanbul.edu.tr (Savio Tse)

problem, the total weight of the set is aimed to be minimized. When the node weights are related to their energy then WCDS becomes a robust backbone architecture. To the best of our knowledge, although central WCDS is studied by researchers [11, 12, 13, 14], there are few work on distributed WCDS construction [15, 16, 17, 18].

In this study, we propose WCDS algorithms for energy-efficient backbone formation for sensor networks. For WCDS construction, we firstly design a weighted dominating set algorithm then we provide a weighted Steiner tree algorithm to connect dominators. Our proposed algorithms are semi-asynchronous and fully distributed in nature making them suitable for large scale applications in sensor networks. We use β synchronizer independent from the timer events.

The rest of this paper is organized as follows. In Section 2, the network model, backbone formation problem and distributed algorithm model with the synchronizers are described and the related work is surveyed in Section 3. The proposed algorithms are described and analyzed in Section 4 and Section 5. The results of performance tests of the proposed algorithms are presented in Section 6. The performance evaluation of a sensor network application is analyzed in Section 7 and conclusions are given in Section 8.

2. Background

2.1. Network Model

The following assumptions are made about the network [19]:

- Each node has distinct *node_id*.
- The nodes are stationary.
- Links between nodes are symmetric. Thus if there is a link from u to v, there exists a reverse link from v to u.
- Nodes do not know their positions and they are not equipped with a position tracker like a GPS receiver.
- Each node knows its neighbors and its own energy.

Based on these assumptions, the network may be modeled as a node weighted undirected graph $G_w(V_w, E)$ where V_w is the set of vertices (nodes) with weights (costs), E is the set of the edges. A node's weight (w) is set as $1/e$ where e is its energy. An example weighted undirected graph model is depicted in Fig. 1 where ids are written inside nodes, weights and energies (in Joules) are placed near to nodes.

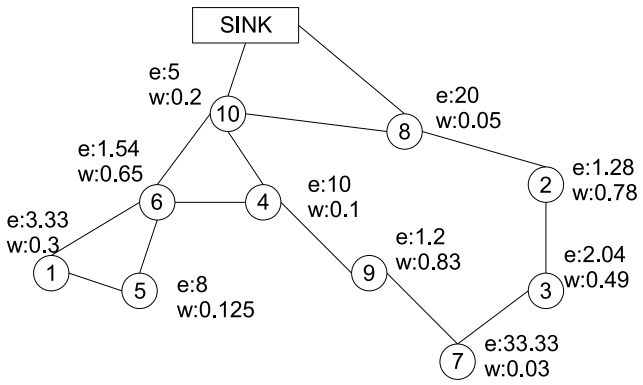

Figure 1: Network Model.

2.2. Backbone Formation Problem

Clustering is a basic method to group similar objects from a whole set of objects. In networks, clustering is performed to simply partition the whole network into subnetworks to ease communication tasks. Backbone formation is the construction of the virtual path of cluster heads to provide the relaying of sensed data to the sink. Backbone formation objectives for sensor networks can be listed as follows:

1. Nodes may initiate the backbone formation operation at any time locally. Distributed time synchronization can be a costly operation for battery powered sensor nodes. Hence, these operations should be distributed and asynchronous.
2. The cluster heads are the servers of their cluster members. They collect sensed data from their members to process, aggregate, filter and route this data to the sink. A cluster head may consume its energy very fast, thus selection of cluster heads with high energy is crucial.
3. The backbone formation algorithms should be independent from the underlying protocols as much as possible to interface to various MAC and physical layer standards such as in [20],[21],[22],[23],[24].
4. The algorithms should be efficient in terms of time, message and space complexity to provide low energy consumptions of sensor networks.

2.3. Distributed Algorithm Models and Synchronizers

In a distributed algorithm each device starts in an initial state $S_i \in S$, and changes its state from S_i to S_j and may output an $O_n \in O$ after receiving an input $I_k \in I$ according to defined state transition procedures. An input can be an interval event such as a timer interrupt; it can be an external event such as a neighbor's failure or message receiving. The operation is divided into rounds in synchronous communication where each operation step is executed in a round. To be independent from the timer events, synchronizers may be used to design semi-asynchronous algorithms for sensor networks [25]. One of the most practical synchronizers that is suitable for sensor network algorithm design is the β synchronizer. To maintain a β synchronizer, firstly, a rooted tree should be constructed. Each node should know its children and parent in this tree. When a node completes its operation and receives OK from children, it sends OK message to its parent. Since a sensor network has a natural root node, the sink, the implementation of this synchronizer for sensor networks is straightforward [26].

3. Related Work

A two-phased CDS algorithm is proposed by Wu [8], in which initially each vertex marks itself as dominator due to some predefined rules by exchanging neighbor lists. Dai [9] and Cokuslu [10] added extra heuristics to Wu's algorithm to reduce the size of the connected dominating set. Besides, there are many studies on CDS construction [27, 5, 6, 7, 28, 29, 4, 8, 9, 10] where the $O(\log(\Delta))$ approximation ratio (Δ is the maximum node degree) can be achieved in $O(\log^2(\Delta))$ rounds [28]. All of these algorithms do not use node weights, they focus on the minimization of the number of dominators. Thus they are out of our concern.

Chvatal proposed a central weighted set cover based dominating set algorithm (CENTSET) with $\ln S$ approximation ratio where S is the minimum weight of the dominating set [30]. In each round, the dominator with the minimum weight ratio is chosen and it is covered with its neighbors. The algorithm stops when all nodes are covered. The weight ratio of the node n with cost c_n is calculated as c_n/Γ_u where Γ_u is the weight of uncovered neighbor nodes. In [31, 5], the authors proposed the distributed synchronous weighted dominating set algorithm (SSET) which is the distributed version of the CENTSET. In SSET, each node finds the weight of its two hop neighbors (2-hop span), then a node enters dominating set if it has the smallest weight ratio among its 2-hop span. Chatterjee proposed a weighted maximal independent set based dominating set algorithm for ad hoc networks [15]. In this algorithm, a node enters the dominating set if it has the smallest weight ratio among dominatee neighbors. After this state change, the dominator node informs its neighbors about its state where the dominatee neighbors

check their neighbor's weight until all nodes are covered. Bao proposed an algorithm in which a node becomes a dominator if it has the smallest weight ratio among its one-hop neighbors or it has the smallest weight ratio among one of its two-hop neighbors excluding one-hop neighbors [16]. Chatterjee and Bao's algorithms may produce weighted dominating sets with very high costs as shown in [17]. To further decrease the weight of the dominating set, Wang proposed a two phased weighted CDS algorithm where the first phase constructs dominating set [17]. In this phase, nodes first construct a MIS similar to [15], then each node runs the CENTSET on its 2-hop span and becomes dominatee if a node with its neighbors are covered by a smaller cost than its cost. The algorithm has an approximation ratio of $\min(18\log(\Delta),4\delta+1)$ for unit disk graphs where δ is the maximum ratio of costs of two adjacent wireless nodes. This approximation ratio can be high for general graphs. Our proposed weighted dominating set algorithm is a fully distributed, improved and semi-asynchronous version of SSET which aims to solve backbone formation problem in sensor networks as explained in Section 4.

In the second phase of the Wang's algorithm, node weighted minimum spanning tree (MST) algorithm is executed to connect the dominators. The approximation ratio of the second phase is 10 for the unit disk graphs but may be high for general weighted graphs as shown in Section 4.2 since weight ratio of a connector is static during the execution of the algorithm. Our algorithms update the weight ratio of the candidate connectors in each round to further decrease the total weight of the selected connectors by simplifying the rules in Klein's central node weighted Steiner tree algorithm which has an approximation ratio of $2 \ln S$ [32]. In Klein's algorithm the connector with the smallest weight ratio is chosen as the connector. The weight ratio of a connector is calculated as: ((cost of the node plus sum of distances to the trees)/(number of trees)) where a tree is a connected component of the dominators. When a connector is selected, the neighbor trees are merged with this connector. The algorithm ends when all dominators are in a single tree. Since our focus is the distributed and deterministic node weighted Steiner tree construction, we omit the unweighted algorithms like the one in [33], probabilistic algorithms in [18] and other central algorithms in [12, 13, 14].

Guha proposed that a WCDS can be constructed in two phases centrally [34]. In first phase, a node weighted dominating set algorithm is executed. In second phase, in order to connect dominators produced in the first phase, a node weighted Steiner tree algorithm is applied. To achieve this, Guha proposed a central WCDS algorithm by simply applying CENTSET and Klein's algorithm sequentially on an undirected node weighted graph. The approximation ratio of this algorithm is $3 \ln S$ where S is the total weight of the optimum solution.

4. Proposed Algorithms

In this study, we propose a semi-asynchronous and fully distributed version of Guha's algorithm. In order to achieve this we should design distributed approximation algorithms for both CENTSET and Klein's algorithm. We relax some constraints of Guha's algorithm in order to decrease the time, message and space complexities. At the same time, we show that the approximation ratios of our algorithms are still $3\ln(S)$. To the best of our knowledge, our algorithms are the first fully distributed and semi-asynchronous WCDS algorithms with $3\ln(S)$ approximation ratio.

4.1. Weighted Dominating Set Algorithm

Our motivations to design the weighted dominating set algorithm are followings:

1. Approximation ratio of Wang's algorithm is high for networks modeled with undirected weighted graphs. Example cases are given in Fig. 2.a and Fig. 2.b where weights are written with small letters and placed near to nodes with their ids that are written with capital letters. Also w, ϵ, $\alpha \in \mathbb{R}^+$, $\epsilon \leq 0.0001w$ and $\alpha \geq 2$ in Fig. 2.a and Fig. 2.b. Ideal algorithm and SSET select $\{B_1,B_2,...,B_n\}$ as dominators because weight ratio of each node in this set equals to $(w+\epsilon)/(w(4+2\alpha)+4\epsilon)$ is smaller than their neighbors' weight ratios. This selection results n dominators with $n(w+\epsilon)$ total weight. Wang's algorithm first

selects $\{A_1,A_2,...,A_n\}$ as dominators since their weights are smaller than their neighbors' weight. After that $\{C_1,C_2,...,C_n\}$ is selected as dominators because $\{B_1,B_2,...,B_n\}$ become dominatees. Lastly, $\{C_1,C_2,...,C_n\}$ quit to become dominators since $\{B_1,B_2,...,B_n\}$ cover their neighbors with lower cost. Wang's algorithm selects $\{A_1,A_2,...,A_n\}$ and $\{B_1,B_2,...,B_n\}$ as dominators, thus produces $2n$ dominators with $n(w+\epsilon)+nw$ total weight approximately 2 times more than those of SSET in Fig. 2.a. A worse scenario is given in Fig. 2.b. Ideal algorithm and SSET select only B_1 as dominator since its weight ratio equals to $(w+\epsilon)/(w(n+1)+\epsilon)$ and it has the smallest weight ratio among all nodes. This selection results $(w+\epsilon)$ total weight. On the other side, Wang's algorithm selects $\{A_1,A_2,...,A_n\}$ as dominators since their weight is smaller than B_1's weight and produces n dominators with nw total weight approximately n times more than those of SSET in Fig. 2.b.

2. In SSET algorithm, the nodes are assumed to be time synchronized. To maintain this situation, the nodes should implement a time synchronization protocol [35] which can consume energy of sensor nodes.
3. A flooding based approach can be designed to disseminate weight ratios of nodes like the one in [36]. But in this case, for a network with N nodes, message complexity can be as large as $\Theta(N^2)$. For battery constraint sensor nodes, this theoretical upper bound may not be suitable.

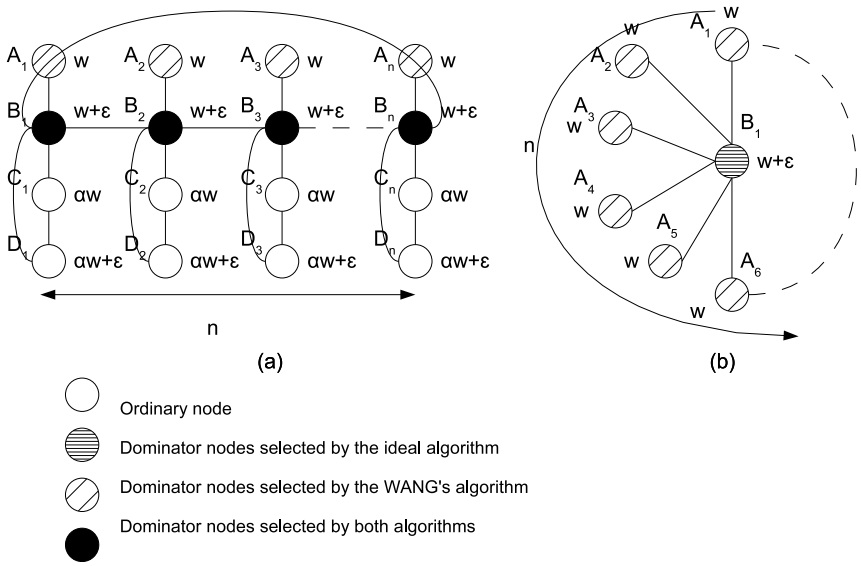

Figure 2: Example cases in which MST Approach Fails

We propose a semi-asynchronous weighted dominating set algorithm (ASYNSET) which is an improved, semi-asynchronous, weighted and sensor network adopted version of the SSET algorithm. In ASYNSET, rounds are triggered by the sink node which is the root of a β synchronizer. In this way, the nodes do not need to run a time synchronization protocol as in SSET. The β synchronizer tree can be constructed by a distributed tree algorithm like the one in [26]. The message complexity of the algorithm is decreased by using the idea in SSET [31, 5] that is given in Observation 1.

Observation 1. *If node n_v's weight ratio ($w(v)$) is smaller than its 2-hop neighbors, it is selected as a dominator by the CENTSET algorithm.*

The steps of the ASYNSET algorithm is given in Alg. 1. When a node awakens, it firstly sends its energy by MY_ENERGY message to its neighbors. A node must collect the energy values of its neighbors to calculate its weight ratio which is then sent as in MY_WRATIO message to the neighbors. After a node receives all MY_WRATIO messages from its neighbors,

Algorithm 1 ASYNSET Algorithm for $node_m$

initially: e:energy; wr:weight ratio; swr:smallest wr; Γ:neighbors
ϕ:children in β tree; $synchronized \leftarrow$ FALSE
upon waking up: multicast $MY\_ENERGY(e)$ to Γ
upon receiving $MY\_ENERGY(e)$ **from** Γ:
  calculate wr; multicast $MY\_WRATIO(wr)$ to Γ
upon receiving $MY\_WRATIO(wr)$ **from** Γ:
  find swr; multicast $2HOP\_WRATIO(swr)$ to Γ
upon receiving $2HOP\_WRATIO(swr)$ **from** Γ **with state**$\neq DTOR$:
  if swr=wr **then** multicast $CONNECT(m)$ to Γ; $state_m$=$DTOR$; call synch($state_m$)
upon receiving $CONNECT(i)$: $state_i \leftarrow DTOR$
  if $state_m$=$IDLE$ **then** $state_m \leftarrow IDLE$; multicast $CONNECTED(m)$ to Γ
upon receiving $CONNECTED(i)$: $state_i \leftarrow DTEE$; call updateWratio()
upon receiving $START$: multicast $START$ to Γ once for each round
  if $state_m$=$DTOR$ or ($state_m$=$DTEE \wedge IDLE \notin states_\Gamma$) **then** call synch($state_m$)
  else multicast $NOT\_CONNECTED(m)$ to Γ; call updateWratio()
  $synchronized \leftarrow$FALSE
upon receiving $NOT\_CONNECTED(i)$: call updateWratio()
upon receiving $OK(b)$:
  if OK received from $\phi \wedge synchronized$=TRUE **then**
    if m=sink **then if** any OK(TRUE) received **then** multicast $START$ to Γ
      else multicast END to Γ
    else if any OK(TRUE) received **then** multicast $START$ to Γ
      else multicast OK(FALSE) to Γ
upon receiving END: multicast END to Γ; terminate
procedure synch(state s):
  if OK received from ϕ **then**
    if any OK received or s=$DTOR$ **then** send OK(TRUE) to p_s
    else send OK(FALSE) to p_s
  $synchronized \leftarrow$TRUE
procedure updateWratio():
  if $NOT\_CONNECTED$ received from Γ with state=$IDLE$ **then**
    recalculate wr; multicast $MY\_WRATIO(wr)$ to Γ

it sends a $2HOP\_WRATIO$ message including the id of the node having the smallest weight ratio. When all of these messages are transmitted successfully, the nodes learn the weight ratio of their two-hop neighbors, if a node has the smallest weight ratio, it sends a $CONNECT$ message and becomes a dominator node. If a non-dominator node receives a $CONNECT$ message for the first time, it sends a $CONNECTED$ message to its neighbors indicating that it is connected to a dominator node. After a node finishes its operation in a round, it executes the synchronize procedure which is given in Alg. 1 with the other procedures. A node will send OK message to its parent in β synchronizer tree if it receives OK from its children and it executes the synchronize procedure. The OK messages are convergecasted to the sink node in this way and if a dominator is selected in a round, the sink node receives OK(TRUE) message. When the sink node receives OK message with TRUE parameter, it multicasts $START$ message to neighbors for starting the new round, otherwise END message is multicasted to terminate the execution of the algorithm.

4.2. Weighted Steiner Tree Algorithm

Our motivations for designing the distributed semi-asynchronous weighted Steiner tree algorithm (ASYNTREE) are given in Fig. 3.a and Fig. 3.b where the performance of the MST approach is far from the optimum. We simplify the connector selection rule in Klein's algorithm in order to design the distributed versions which aim to select a small number of low cost nodes as connectors. We assume the cost of each edge as zero. The rules for the connector selection of our algorithms are given below:

- If a candidate node connects more than one tree, its weight ratio is calculated as (cost of the node)/(number of trees to be connected).
- If a candidate node is a neighbor of only one tree and has a candidate neighbor which is a neighbor of at least one of the other trees, its weight ratio is calculated as (sum of cost of the node and its candidate)/(number of trees to be connected).

- The node with the smallest weight ratio is selected in each round.

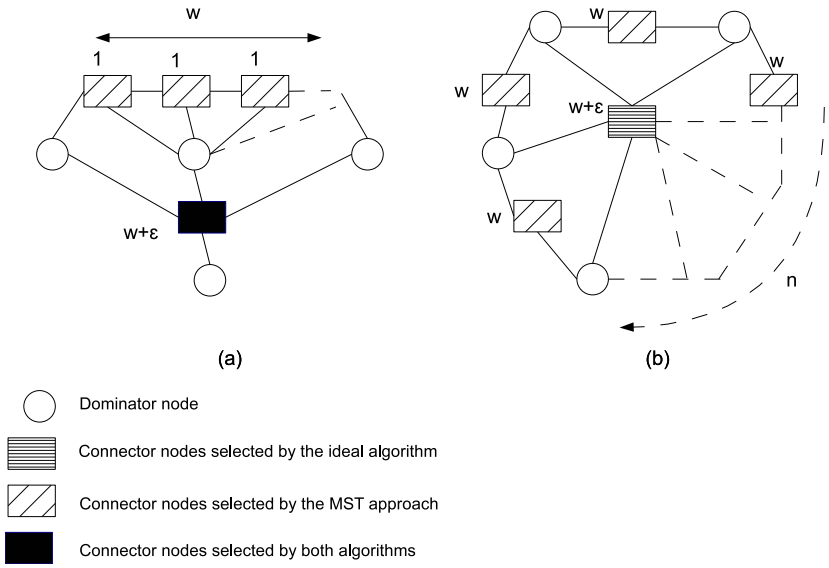

Figure 3: Example cases in which MST Approach Fails

By applying these rules, our algorithms choose the same connectors with the ideal algorithm as shown in Fig. 3.a and Fig. 3.b. Ideal algorithm and ASYNTREE select 1 connector with $w+\epsilon$ total weight whereas MST algorithm selects $w+1$ connectors with $2w+\epsilon$ total weight in Fig. 3.a. A worse scenario is given in Fig. 3.b. Ideal algorithm and ASYNTREE select 1 dominator with $w+\epsilon$ total weight whereas WANG's algorithm selects n dominators with nw total weight in Fig. 2.b. The reason of this increase in the performance compared to the MST approach due to fact that the weight ratio of each candidate connector is calculated as a function of its cost and number of neighbor trees and in each iteration it is updated as opposed to the static weight ratios in MST approach.

We use rooted tree structures consisting of the dominators which are in the same connected component in order to decrease the message complexity. Each dominator should know its children and parent in its dominator tree. To provide asynchronous operation we modify and adopt the GHS algorithm [37] and we use a β synchronizer.

The steps of ASYNTREE algorithm are given in Alg. 2. When a dominator node awakens it firstly runs the modified GHS algorithm in order to calculate its tree id and the total node count in its tree. The modifications are listed below:

- We added an extra field to the $REPORT$ message which is used by the core nodes to calculate the total number of nodes in tree. An example operation is given in Fig. 4. The black nodes in this example are the core nodes. The directions of the edges show the flow of the $REPORT$ message. A leaf node in this tree sends $REPORT$ message where it only knows itself. After a node receives all $REPORT$ messages from its children, it sends $REPORT(total\ count+1)$

- After the core nodes construct the minimum spanning tree, one of them is chosen as the leader of the tree which then sends an END((tree id, node count of the tree) message to its children. This message is forwarded to all nodes in the tree to inform the tree id and the node count of the tree.

- When a node receives an END message, it makes a synchronization operation on the β synchronizer tree.

Algorithm 2 ASYNTREE Algorithm for $node_m$

initially: my tree id(mt_id)←1; my tree count(mt_cnt)←1
T_d:children in dominator tree; p_d:parent in dominator tree; L_c:trees for connection
upon waking up: **if** $state_m$=$DTOR$ **then** execute modified GHS algorithm
upon receiving $START$ **once for each round**: **if** $state_m$=$DTOR$ **then** multicast $TREE\_INFO(mt\_id, my\_cnt)$ to Γ
upon receiving $TREE\_INFO(t\_id, t\_cnt)$ **from** Γ **with state**=$DTOR$:
  if $state_m \neq DTOR$ **then** multicast $DTEE\_INFO(received\ t\_ids, received\ t\_counts)$ to Γ
upon receiving $DTEE\_INFO(list\ t\_ids, list\ t\_cnts)$ **from** Γ **with state**$\neq DTOR$:
  calculate wr; multicast $MY\_WRATIO(wr)$ to Γ
upon receiving $MY\_WRATIO(wr)$ **from** Γ **with state**$\neq DTOR \wedge T_d$:
  if $state_m$=$DTOR$ **then** find swr
    if p_d=$\emptyset$ **then if** swr=∞ **then** multicast $NOT\_CONNECT$ to Γ
      else multicast $CONNECT(id\ having\ swr)$ to Γ
    call synch(FALSE)
  else send $MY\_WRATIO(swr)$ to p_d
upon receiving $NOT\_CONNECT$:
  if $state_m \neq DTOR$ **then** call synch(FALSE)
  else multicast $NOT\_CONNECT$ to Γ once for this round
upon receiving $CONNECT(i)$ **from** Γ **with state**=$DTOR$:
  if $state_m \neq DTOR$ **then if** all i=m **then** call synch(TRUE)
    else call synch(FALSE)
  else multicast $CONNECT(id\ having\ swr)$ to Γ once for this round
procedure synch(b): $synchronized$←TRUE
  if ϕ=$\emptyset$ **then** multicast $CONNECT\_REQ(b)$ to Γ
  else call conReq()
procedure conReq():
  if $CONNECT\_REQ$ received from ϕ **then**
    if m=sink **then if** swr=∞ multicast END to Γ
      else send $TREE\_MRG(L_c)$ to p_s
  else send $CONNECT\_REQ(swr, L_c)$ to p_s
upon receiving $CONNECT\_REQ(swr, L_c)$: call conReq()
upon receiving END: multicast END to Γ; terminate
upon receiving $TREE\_MRG(L_c)$: update tree information according to L_c
  multicast $TREE\_MRG(L_c)$ to Γ once for each round
  update mt_id and mt_cnt
  if $state_m$=$DTOR$ **then** multicast $TREE\_INFO(mt\_id, mt\_cnt)$ to Γ

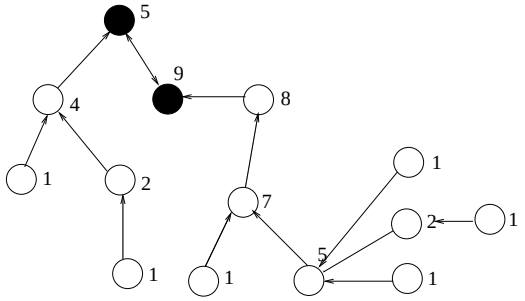

Figure 4: An Example REPORT Message Transfer in Modified GHS.

After the dominator nodes finish the execution of the modified GHS algorithm, the sink node starts the execution of the next step by sending a $START$ message. When dominator nodes receive this message, they send $TREE\_INFO$ messages to inform candidate connectors which are dominatees at that time. When a dominatee receives $TREE\_INFO$ from all dominator neighbors it sends $DTEE\_INFO$ to its dominatee neighbors and wait for the same message from them. After $TREE\_INFO$ and $DTEE\_INFO$ messages are exchanged, a dominatee node calculates its weight ratio by using our heuristic and sends MY_WRATIO message. When a dominator node receives all MY_WRATIO messages from its neighbors and children in dom-

inator tree, it sends this message to the parent in this dominator tree. After a dominator tree leader receives all MY_WRATIO messages, it sends a $CONNECT$ message including the id of the candidate having the smallest weight ratio or it sends a $NOT\_CONNECT$ message if the smallest weight ratio is ∞. When a dominator node receives a $CONNECT$ or $NOT\_CONNECT$ message it forwards this message by sending to its neighbors. When a dominatee node receives a $CONNECT$ message which includes the id of another dominatee node or a $NOT\_CONNECT$ message, it finishes the execution of this round and calls synchronize procedure. Otherwise it becomes a real candidate for the next connector. The nodes aggregate the weight ratios to find the smallest by sending $CONNECT\_REQ$ on the β synchronizer tree. When the sink node finds the smallest weight ratio, it sends $TREE\_MRG$ to its children in β synchronizer tree to merge the trees with the newly selected connector. If all of the dominators are connected on the same tree, the sink node ends the algorithm by sending an END message.

5. Analysis

In this section we give proof of correctness, approximation ratio, message complexity, time complexity and space complexity analysis.

5.1. Proof of Correctness

Observation 2. *ASYNSET algorithm can be divided into 6 steps.*

1. *Each node learns energies of its neighbors to calculate its weight ratio at the beginning of the algorithm.*
2. *Each node learns the weight ratio of its 2 hop neighbors to find the node with the smallest weight ratio.*
3. *Nodes having the smallest weight ratio among its 2 hop neighbors enter dominating set.*
4. *After a new dominator is selected, its 2 hop neighbors update their weight ratios.*
5. *Nodes are synchronized after a round is finished.*
6. *Algorithm execution is terminated at each node after dominating set is constructed.*

Theorem 1. *ASYNSET produces the same dominating set with the CENTSET algorithm and the execution is terminated at each node.*

Proof. Assume the contrary. In this situation, execution of at least one of the step of ASYNSET given in Observation 2 should fail. At first round all nodes exchange MY_ENERGY messages to acquire the energy information of their neighbors, so Step 1 is achieved. Nodes learn the weight ratios of their 2 hop neighbors by firstly transmitting MY_WRATIO messages then $2HOP\_WRATIO$ messages which provides Step 2. Then a node having the smallest weight ratio among its 2 hop neighbors enters dominating set as in Observation 1 and sends $CONNECT$ message to its neighbors, so Step 3 is achieved. When a node receives $CONNECT$ message it marks the source node as covered and sends a $CONNECTED$ message to its neighbors. The recipients of the $CONNECTED$ messages marks the source nodes as covered so the weight ratio is updated successfully that provides Step 4. Before finishing the execution of a round, each node calls the synchronize procedure and sends a OK message to its parent in β synchronizer tree when it receives all OK messages from its children, so that synchronization process is completed successfully and Step 5 is provided. If a node receives the OK message with TRUE field or it is selected as a dominator in a round, it sends the OK message with TRUE parameter. The rounds are triggered by the sink node with the dissemination of the $START$, after it receives OK messages with TRUE parameter from its children in β synchronizer tree. At the beginning of a round, if a node is not covered it sends a $NOT\_CONNECTED$ message to its neighbors thus all nodes can distinguish the covered and uncovered nodes. Sink node terminates the algorithm by sending an END message if a new dominator is not selected in previous round. By applying these operations, Step 6 is provided. All steps are achieved in ASYNSET, thus we contradict with our assumption, ASYNSET produces the same dominating set with CENTSET.

Observation 3. *ASYNTREE algorithm can be divided into 7 steps.*

1. *Connected dominators construct trees at the beginning of the algorithm.*
2. *Each dominatee calculates its weight ratio.*
3. *Each leader dominator learns weight ratios of candidate dominatees.*
4. *Dominatee with the smallest weight ratio is chosen as the new dominator (connector).*
5. *Weight ratios of dominatees are updated after the selection of the new dominator.*
6. *Nodes are synchronized after a round is finished.*
7. *Algorithm execution is terminated after a connected dominating set is constructed.*

Theorem 2. *ASYNTREE connects the elements of a dominating set and the execution is terminated at each node.*

Proof. Assume the contrary in which execution of at least one of the steps of ASYNTREE given in Observation 3 should fail. At the beginning of the algorithm each dominator learns its tree information by using modified GHS, this provides Step 1. In the other rounds, each dominatee calculates its weight ratio as in proposed heuristic by receiving $TREE\_INFO$ and sending $DTEE\_INFO$ messages, so Step 2 is provided. A leader dominator can learn all weight ratios of the neighbor dominatees by receiving aggregated MY_WRATIO messages where Step 3 is provided. A dominatee becomes a real candidate after receiving $CONNECT$ messages from all neighboring trees when all nodes synchronize with the sink by sending the smallest weight ratio in $CONNECT\_REQ$ messages which they have received, sink finds the connector with the smallest weight ratio. Sink node connects the trees by sending $TREE\_MRG$ message so tree_ids are updated. By applying these operations, Step 4, Step 5 and Step 6 are achieved. The sink node terminates the algorithm if the smallest weight ratio is ∞ by sending an END message where Step 7 is provided. We contradict with our assumption since all steps are achieved in ASYNTREE.

5.2. Approximation Ratio

Theorem 3. *The total approximation ratio of ASYNSET and ASYNTREE to the optimum WCDS is 3*$\ln(S)$.

Proof. From Theorem 1, ASYNSET produces the same dominating set with CENTSET. Thus the approximation ratio of ASYNSET is $\ln S$.

In ASYNTREE, we apply Klein's steiner-tree algorithm in a distributed manner. Klein's algorithm minimizes the total cost of nodes and edges in G', and our problem aims at minimizing the total cost of nodes in S only, and therefore, we can assume the cost of each edge zero. As Klein's algorithm does not forbid zero-cost edges, their performance bound ratio $2 \ln S$ is still valid. The output of Klein's algorithm is used as the output of our problem.

For any instant of input, let c_1 be the total cost of nodes in S in the output, and c_1^{opt} be the cost from the optimal solution to the problem; and with the assumption that the edges are of zero cost, define c_2^{opt} to be the minimum cost of the problem, which minimizes the total cost of nodes and edges.

For convenience, we call the problem we are tackling *problem 1*, and the other problem, which counts the edge-costs, *problem 2*. From the optimal solution of problem 1, counting the costs of tree edges, we have $c_2^{opt} \leq c_1^{opt}$. On the other hand, from the optimal solution of problem 2, taking away the tree edges, we have $c_1^{opt} \leq c_2^{opt}$. Therefore, $c_2^{opt} = c_1^{opt}$. By Klein's result, we have $c_1 \leq (2 \ln N) c_2^{opt} = (2 \ln N) c_1^{opt}$, since the output of their algorithm is used directly. Consequently, $3\ln(S)$ approximation ratio is valid for ASYNSET and ASYNTREE.

5.3. Message Complexity

Definition 1. *λ set: is the set of dominators of a connected component in G' graph in which dominators at most 2 hops away from each other produced by the ASYNSET algorithm running on G graph are assumed to be connected in G' graph. The λ_m is the set with the maximum cardinality among λ sets for a graph G.*

Lemma 1. *ASYNSET terminates in $|\lambda_m|$ rounds.*

Proof. Assume that there are multiple λ sets where λ_i=$\{v_{i1}$,v_{i2},v_{i3},...$\}$. A v_{ij} and v_{kl} can be selected as dominators by ASYNSET algorithm concurrently in a round when $i \neq k$ since they are at least 3 hops away from each other as given in Definition 1. Algorithm selects dominators in this way until all of the elements of the $|\lambda_m|$ are marked as dominator.

Lemma 2. *The message complexity of a round in ASYNSET is O(N).*

Proof. At the worst case a node sends at most 6 messages in a round: $START$, MY_ENERGY, MY_WRATIO, $2HOP\_WRATIO$, $CONNECT$ and OK. Thus the message complexity of a round is O(N).

Theorem 4. *The message complexity of the ASYNSET algorithm is $\Theta(|\lambda_m|N)$, at the worst case is $O(N^2)$, at the best case is $\Omega(N)$.*

Proof. The algorithm terminates in λ_m rounds as given in Lemma 1 where the message complexity of each round is O(N) as proven in Lemma 2. So that the message complexity of the algorithm is $\Theta(|\lambda_m|N)$. At the worst case for $|\lambda_m|$=N, the message complexity is O(N^2). The best case occurs when $|\lambda_m|$=1, the message complexity in this case is $\Omega(N)$.

Lemma 3. *The message complexity of the modified GHS algorithm is O(Nlog(N)).*

Proof. The message complexity of the GHS algorithm is O(Nlog(N)+M)[37] where M is the edge count. It can be implemented with O(Nlog(N)) radio multicast messages. In the modified GHS algorithm, additionally each node sends a END and OK message with an overhead of O(N) message complexity, thus message complexity is still O(Nlog(N)).

Theorem 5. *The message complexity of the ASYNTREE algorithm is $\Theta(N(|C|+log(N)))$, at the worst case is $O(N^2)$, at the best case is $\Omega(Nlog(N))$ where C is the connector count.*

Proof. A dominator node may send $TREE\_INFO$, MY_WRATIO, $CONNECT$ or $NOT\_CONNECTED$, $CONNECT\_REQ$ and $TREE\_MRG$ messages concurrently a dominatee node may send $DTEE\_INFO$, MY_WRATIO, $CONNECT\_REQ$ and $TREE\_MRG$ messages in a round. Thus the message complexity of a round in ASYNTREE is O(N). The total message complexity of the ASYNTREE algorithm is $\Theta(N(|C|+\log(N))$ from Lemma 3. At the worst case for $|C| \in$ O(N), the message complexity is O(N^2). At the best case for $|C|$=1, the message complexity is $\Omega(N\log(N))$.

5.4. Time Complexity

Lemma 4. *Assuming that diameter is D and degree is Δ, the time complexity of a round in ASYNSET is O(D+Δ).*

Proof. β synchronizer tree can be constructed in O(D) time [26]. A round is started by the sink node with the dissemination of the $START$ message in O(D) time. Then idle nodes send $NOT\_CONNECTED$ message to their neighbors where this operation may take O(Δ) time. After that the nodes exchange their weight ratios by sending MY_WRATIO messages which takes same time as the previous operation. This operation is followed by the transmission of $2HOP\_WRATIO$, $CONNECT$ messages where both of these operations may accomplish in O(Δ) time individually. The recipient of the $CONNECT$ messages send $CONNECTED$ message that takes O(1) time since each of this recipient is at most 2 hops away from each other. At last, the nodes end the execution of the round and terminate by sending OK messages in O(D) time. The time complexity of a round is: O(2D+4Δ+1) $\in$ O(D+Δ).

Theorem 6. *The time complexity of the ASYNSET algorithm is $\Theta((D+\Delta)|\lambda_m|)$, at the worst case is $O(N(D+\Delta))$ and at the best case is $\Omega(D+\Delta)$.*

Proof. The time complexity of the ASYNSET algorithm is $\Theta((D+\Delta)|\lambda_m|)$ which can be derived from Lemma 1 and Lemma 4. At the worst case for $|\lambda_m|=N$, the time complexity is $O(N(D+\Delta))$. At the best case for $|\lambda_m|=1$, time complexity is $\Omega(D+\Delta)$.

Lemma 5. *The time complexity of the modified GHS algorithm is $O(N_d^2+D)$ where N_d is the maximum number of the elements of the dominators in the same connected component at the beginning of the algorithm.*

Proof. The time complexity of the GHS algorithm is $O(N^2)$ [37], since trees of connected components are constructed concurrently, the operation will be completed with the construction of the tree with the maximum node count (N_d), the time complexity of the modified GHS algorithm is $O(N_d^2)$. Additionally, in the modified GHS algorithm, synchronization operation takes $O(D)$ time at the worst case. Thus the total time complexity is $O(N_d^2+D)$.

Lemma 6. *The time complexity of a round in ASYNTREE algorithm is $O(N_d+D+\Delta)$.*

Proof. The round is started by the sink node either with sending $START$ or $TREE\_MRG$ message where this operation takes $O(D)$ time. The dominatee nodes collect $TREE\_INFO$ messages from dominator nodes, then send $DTEE\_INFO$ messages in $O(\Delta)$ time. These operations are followed by the transmission of $CONNECT$ or $NOT\_CONNECTED$ messages in $O(N_d)$ time. Lastly, nodes are synchronized in $O(D)$ time. Consequently, the time complexity is $O(N_d+D+\Delta)$.

Lemma 7. *The time complexity of the ASYNTREE algorithm is $\Theta((C+N_d)N_d+C(D+\Delta))$ at the worst case is $O(N^2)$.*

Proof. From Lemma 5 and Lemma 6, the time complexity of the algorithm is $\Theta((C+N_d)N_d+C(D+\Delta))$. At the worst case for $N_d=C=N$, the time complexity is $O(N^2)$.

5.5. Space Complexity

Theorem 7. *The space complexity of the ASYNSET and ASYNTREE algorithm per node are $\Theta(\Delta)$.*

Proof. In ASYNSET and ASYNTREE, for storing the states of its neighbors, $\Theta(\Delta)$ space is required per node. Other space consumption is constant.

Theorem 8. *The greatest message sizes in ASYNSET and ASYNTREE are $2log(N)$ bits and $Nlog(N)$ bits respectively.*

Proof. In ASYNSET, the messages having the greatest number of fields are MY_ENERGY and MY_WRATIO with 2 fields, this requires $2\log(N)$ bits. In ASYNTREE, $TREE\_MRG$ message may have N fields for merging thus $N\ \log(N)$ bits may be required.

6. Performance Evaluation of the Proposed Algorithms

We implemented ASYNSET and ASYNTREE algorithms in *ns2* simulator version 2.31 to test their performance with extensive simulations. To compare ASYNSET with the existing work, we implemented Chatterje's MIS based algorithm (MIS), Bao's algorithm (BAO), SSET algorithm, the first phase of the Wang's algorithm (WANG) and a flooding based weighted dominating set algorithm (FLOODSET) with the same rules in ASYNSET. To compare our weighted Steiner tree algorithms, we implemented second phase of the WANG's algorithm (WANG2nd) and a flooding based weighted Steiner tree algorithm (FLOODTREE) with the same rules in ASYNTREE. Although our proposed algorithms are asynchronous and do not need time synchronization, we assume that nodes are time synchronized in order to implement other algorithms.

We generated randomly connected networks with 100 to 400 nodes that are uniformly distributed. IEEE 802.11 radio and MAC standards readily available in *ns2* simulator were chosen

for lower layer protocols. Our algorithms only require *send*, *receive* primitives which are important services required from MAC and physical layer [20, 21, 22, 23, 24], thus various types of lower layer standards can be used. Two ray ground was used as the propagation model. The transmission power is 0.660 mW, the receive power is 0.395 mW, and the communication range of a sensor node is 250 m. We measured the performance of the algorithms for average node degrees varying between 4,5 and 6. To vary degrees, different size of flat surface areas were chosen. We also varied the initial energy of the nodes by randomly generating from 0-50J, 0-100J, 0-150J intervals. We measured the total weight of the dominators and connectors, count of dominators and connectors, energy consumption, wallclock times. The surface areas are given in Table 1.

Table 1: Size of Surface Areas (X × Y (m)).

Node Number/Degree	4	5	6
100	2700×1200	2520 ×1200	2340×1040
200	5100×1200	4760 ×1120	4420×1040
300	7800×1200	7280 ×1120	6760×1040
400	10200×1200	9520 ×1120	8840×1040

6.1. Weight of Dominators

One of the most important target of the weighted dominating set algorithms is to select a low weighted backbone where the selected nodes will have high energy when the weights of the nodes are directly related to their energies. So that, we firstly measured the total weight of the dominators produced by the algorithm. Unless otherwise stated, the default node degree is 5, node count is 200 and node energies are varied between 0-100J.

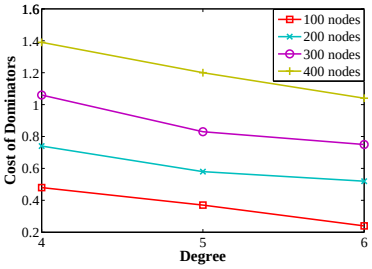

Figure 5: Weight of dominators against node degree

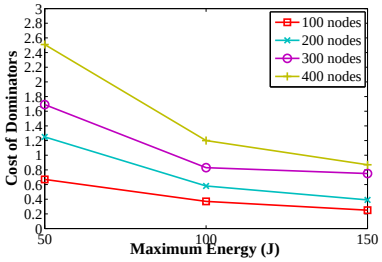

Figure 6: Weight of dominators against energy interval

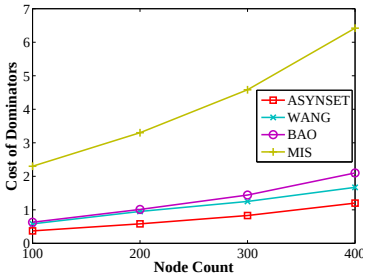

Figure 7: Weight of dominators against node count

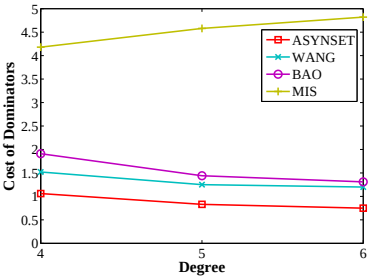

Figure 8: Weight of dominators against node degree

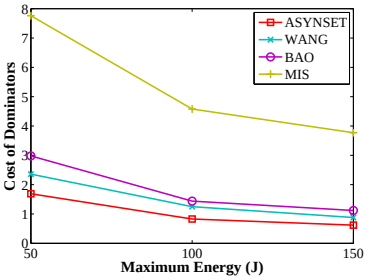

Figure 9: Weight of the dominators produced by algorithms against energy interval

In Fig. 5, the weight of the dominators produced by the ASYNSET against the node degree is shown. When the node degree is increased, ASYNSET selects higher energy nodes as dominators.

This shows that the algorithm responds well with the increase of the network connectivity. The weight of the dominators produced by the ASYNSET against the energy interval is shown In Fig. 6. The measurements show that when the energy interval is increased, ASYNSET selects dominators with higher energy thus the total weight of the backbone decreases.

A comparison of the algorithms against the node count is given in Fig. 7. The total cost of the dominators produced by the ASYNSET is the smallest among the other algorithms. WANG and BAO performs well and approximate to the ASYNSET where MIS performs the worst. Other comparisons are shown in Fig. 8 and Fig. 9 against the varying node degree and energy interval. In all of three comparisons, the performance order of the algorithms are same.

6.2. Dominator Count

Weighted dominating set algorithms aim to choose a small subset of nodes with low weight. Thus, minimizing the dominator count is important. In Fig. 10 and Fig. 11, the dominator count against the node degree and energy interval are given. At the worst case, the dominator count is one fourth of the total node count. The dominator count is stable against the varying node degree and energy interval.

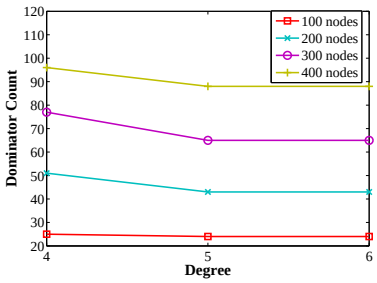

Figure 10: Dominator count against node degree

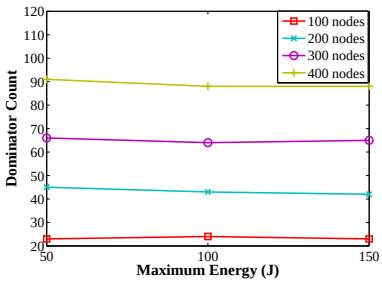

Figure 11: Dominator count against energy interval

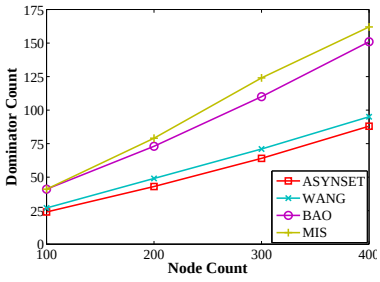

Figure 12: Dominator count against node count

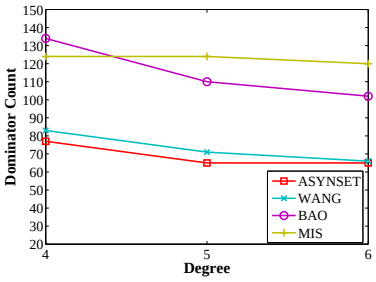

Figure 13: Dominator count against node degree

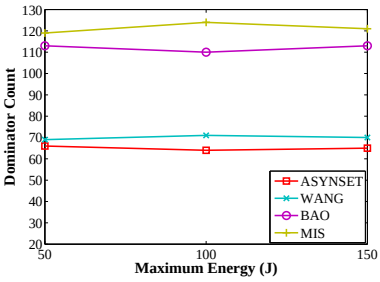

Figure 14: Dominator count produced by algorithms against energy interval

The comparisons of the MIS, BAO, WANG and ASYNSET are shown in Fig. 12, Fig. 13 and Fig. 14. ASYNSET produces the smallest dominating set compared to the other algorithms against the node count as shown in Fig. 12. ASYNSET's performance is the best, the performance of the WANG is close to the ASYNSET and MIS performs worst with the BAO. This performance order does not change when the node degree and the energy interval is varied as shown in Fig. 13 and Fig. 14. In all algorithms except the MIS, the size of dominating set reduces when the node degree is increased as shown in Fig. 13.

6.3. Weight of Connectors

Minimizing the total weight of connectors is an important objective of the node weighted Steiner tree algorithms. Thus, we measured the total weight of the connectors produced by the

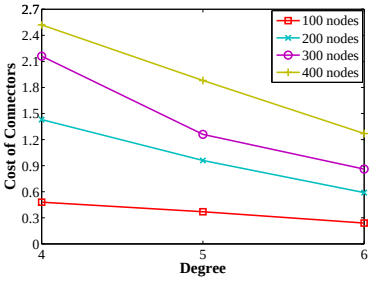

Figure 15: Weight of the connectors against node degree

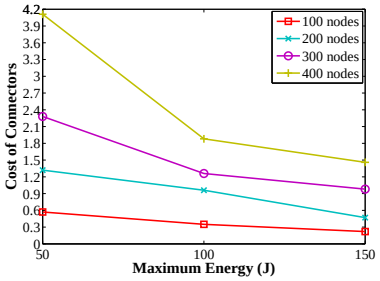

Figure 16: Weight of the connectors against energy interval

algorithms. Before running the Steiner tree algorithms, the dominating set algorithms are executed. The ASYNSET and ASYNTREE are executed together and to compare the performance of these algorithms, WANG and WANG (2nd) are executed in the same way. The total weight of the connectors produced by the ASYNTREE algorithm against the varying node degree and energy interval are shown in Fig. 15 and Fig. 16. The total weight of the connectors decrease when the node degree and energy interval are increased. As the node degree increases, the connectivity between low weighted nodes and high weighted nodes increases where ASYNTREE benefits from this fact. Also ASYNTREE runs efficiently and chooses nodes with higher energy as the energy interval increases.

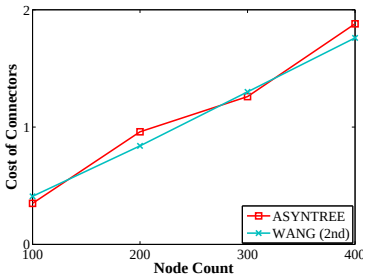

Figure 17: Weight of the connectors against node count

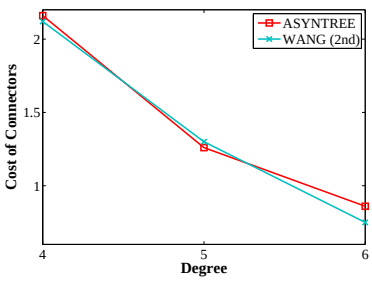

Figure 18: Weight of the connectors against node degree

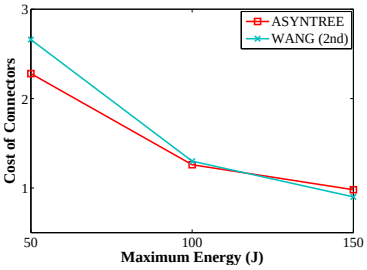

Figure 19: Weight of the connectors produced by algorithms against energy interval

Comparisons of the selected connector weight produced by the ASYNTREE and WANG (2nd) are given in Fig. 17, Fig. 18 and Fig. 19. The total weight of the connectors of both algorithms against the total node count, node degree are approximate. Although at first look it may be stated from these results that the performance of the algorithms are similar, it should be noted that the energy of the nodes in the dominating set produced by the ASYNSET is higher than those of WANG which means that the set of candidate of connectors for ASYNTREE consists of lower energy than those of WANG (2nd). Although ASYNTREE has this drawback at the beginning of the execution, the performance of the ASYNTREE and WANG (2nd) are approximate in the end. In the next section, we show the reason for this situation.

6.4. Connector Count

The other important target of the weighted Steiner tree algorithms is to minimize the connector count. In Fig. 20 and Fig. 21, the connector count of the ASYNTREE algorithm against the varying node degree and energy interval is shown. The connector count decreases when the node degree is decreased. The reason of this situation is when the network connectivity increases, a connector may connect more trees. The performance of the ASYNTREE algorithm is stable against the varying energy interval as shown in the previous section where the total weight of the connectors decreases. The performance comparisons of the ASYNTREE and WANG (2nd)

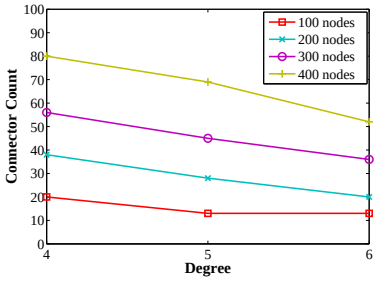

Figure 20: Connector count against node degree

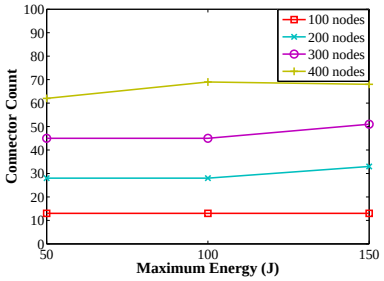

Figure 21: Connector count energy interval

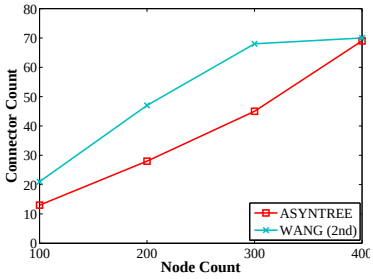

Figure 22: Connector count produced by algorithms against node count

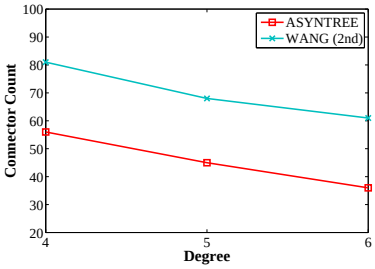

Figure 23: Connector count against node degree

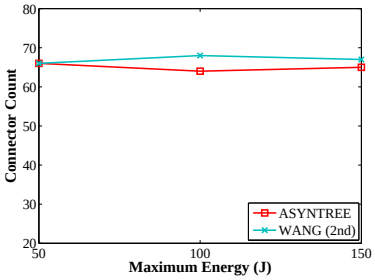

Figure 24: Connector count produced by algorithms against energy interval

are given in Fig. 22, Fig. 23 and Fig. 24. ASYNTREE produces less number of connectors than WANG (2nd) against the varying node count, node degree and energy interval. Although ASYNTREE and WANG (2nd) produces approximate weight of connectors as shown in the previous section, ASYNTREE produces a smaller connector set than WANG (2nd).

6.5. Energy Consumption

The energy consumption of the ASYNSET algorithm against the varying node degree is given in Fig. 25. As shown in this figure, the energy consumption increases linearly as the node count increases and it behaves stable as the node degree increases. The energy consumptions of the algorithms are shown in Fig. 26. The performance order of the other algorithms is: WANG, SSET, ASYNSET and FLOODSET. The energy consumption of the ASYNSET is approximately 10 times greater than the consumption of the WANG. ASYNSET consumes slightly more than SSET because of the synchronization operations. The performance order of the algorithms shows that when the algorithms get complicated by adding extra messages and extra rules in order to reduce the size and the cost of the dominating set, then the energy consumption of the algorithms increase.

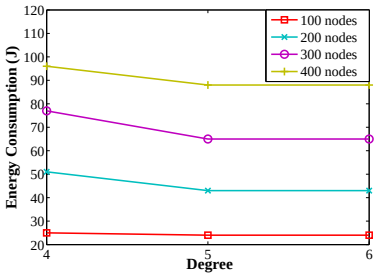

Figure 25: Energy consumption of the ASYNSET against node degree

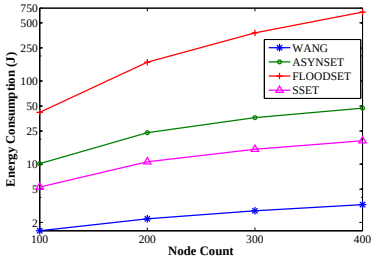

Figure 26: Energy consumptions of the dominating set algorithms against node count

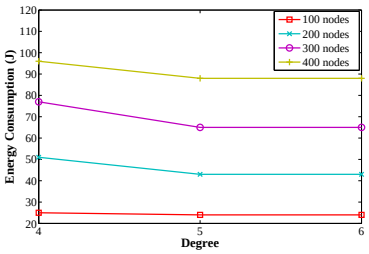

Figure 27: Energy consumption of the ASYNTREE against node degree

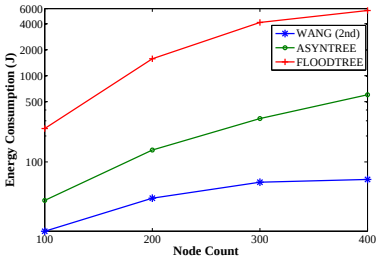

Figure 28: Energy consumptions of the Steiner tree algorithms against node count

In Fig. 27, the energy consumption of the ASYNTREE algorithm against the varying node degree and node count is given. When the node degree is increased, the number of connectors decrease in ASYNTREE thus the number of rounds and energy consumption decreases. The comparison of the energy consumption of the weighted Steiner tree algorithms are given in Fig. 28. The lowest energy is consumed by the WANG (2nd), whereas the highest energy is consumed by FLOODTREE.

As a result of our tests, it can be concluded that although our algorithms construct robust and low cost backbones for sensor networks, they consume more energy than the related work. Since energy efficiency is a very important objective for sensor networks, one may claim that our algorithms are not suitable for sensor networks. To disprove this claim, we will show an implementation of an example sensor network application on top of different backbones and measure their energy consumptions in Section 7. We will give a break-even analysis and we will show that the energy consumption of a sensor application may increase significantly if the quality of the backbone is reduced which may result that the energy consumption of the backbone formation becomes trivial compared to the energy consumption of the application.

6.6. Wallclock times

In this section, we give the wallclock time measurements of the algorithms. In Fig. 29, the wallclock time measurements of the weighted dominating set algorithms are given. The performance order of the weighted dominating set algorithms is: WANG, SSET-ASYNSET and FLOODSET. The wallclock time measurements of SSET and ASYNSET are much the same where the nodes are idle in SSET when they finish the execution of a round, on the other hand, during this idle time, the synchronization operation is executed by the nodes in ASYNSET. The time consumption of the FLOODSET is much more than the other weighted dominating set algorithms. In Fig. 30, the wallclock time measurements of the node weighted Steiner tree algorithms are shown. ASYNTREE performs the best among other approaches since it is a semi-asynchronous algorithm in which there is no idle time.

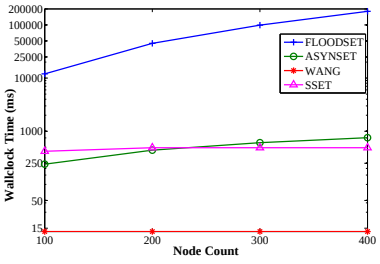

Figure 29: Wallclock times of the dominating set algorithms against node count

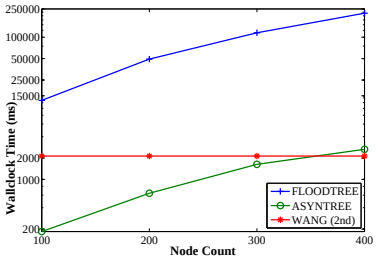

Figure 30: Wallclock times of the Steiner tree algorithms against node count

7. Performance Evaluation of an Example Application

In this section, we show the implementation of an example sensor network application and analyze the energy consumption of the application against two backbone architectures which are listed below:

1. The backbone constructed by the semi-asynchronous algorithms: We executed ASYNSET and ASYNTREE algorithm to form the backbone (ASYNSET+ASYNTREE).
2. The backbone constructed by the WANG's algorithms: To form the backbone, we executed the first and second phases of the WANG's algorithm (WANGALL).

7.1. Application Setup

The sensor network applications need two important services from the underlying network architecture: data aggregation and data routing. The data aggregation can be accomplished by clustering the network where each cluster head may aggregate the sensed data of cluster members. The data routing can be divided into two tasks as intra-cluster routing and inter-cluster routing. The intra-cluster routing operation involves the transmission of a sensed data message from an ordinary node to its cluster head, the inter-cluster operation is the relaying of the aggregated data from a cluster head to the sink along the way of the backbone.

Algorithm 3 Rooted Tree Formation and Clustering Algorithm for $node_m$

1: **initially:** C_m: set of cluster member of $node_m$
2: **upon receiving** $CONSTRUCT\_PATH(id)$:
3: **if** this message is not received before **then**
4: **if** $state_m$=$DTEE$ **then** $my\_cluster\_head \leftarrow id$; send $IN\_CLUSTER(m)$ to id
5: **else** $my\_parent \leftarrow id$; multicast $CONSTRUCT\_PATH(m)$ to Γ
6: **upon receiving** $IN\_CLUSTER(id)$: $C_m \leftarrow C_m \cup id$

Algorithm 4 The Application for $node_m$

1: **upon the application started or the timer expired**:
2: **if** $state_m$=$DTEE$ **then** send $SENSED\_DATA(data)$ to $my\_cluster\_head$, start the timer
3: **upon receiving** $SENSED\_DATA(data)$: $aggregated\_data \leftarrow$ aggregate the $data$
4: **if** $SENSED\_DATA$ from all C_m **then**
5: send $AGG\_SENSED\_DATA(aggregated\_data)$ to my_parent
6: **upon receiving** $AGG\_SENSED\_DATA(data)$:
7: send $AGG\_SENSED\_DATA(data)$ to my_parent

To accomplish the data aggregation and data routing operations, we first formed the backbone, we then constructed a tree structure rooted at the sink and finally we clustered the network. To construct rooted tree structure and clusters, sink floods $CONSTRUCT\_PATH$ message to the network where the other nodes execute the Alg. 3. If the receiver and sender of this message is a dominator, the receiver sets its parent to the message source and forwards this message. If the receiver of this message is a dominatee and the sender is a dominator, the receiver sets its cluster head to the message source and sends $IN\_CLUSTER$ message to acknowledge the source dominator that it belongs to the cluster of the source dominator.

By using the above described network topology, each node runs the application given in Alg. 4 where periodically each cluster member node senses, records and then sends this recorded data in a $SENSED\_DATA$ message to its cluster head. The cluster heads aggregate these messages and send them in a AGG_SENSED_DATA message over the backbone.

7.2. Energy Consumption

We measure the energy consumption of the application against the varying node count, node degree and energy interval. To evaluate the performance of the backbone structures, we do a break-even analysis. In Fig. 31, Fig. 32 and 33, the total energy consumptions against the node count are given. The "B" letters in figures show the break-even point. The energy

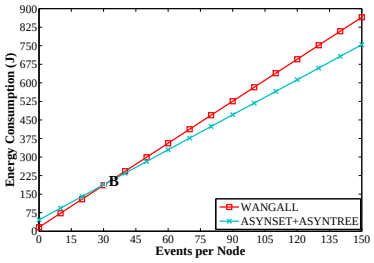

Figure 31: Energy consumptions of the network with 100 nodes, 5 node degree and 0-100 energy interval

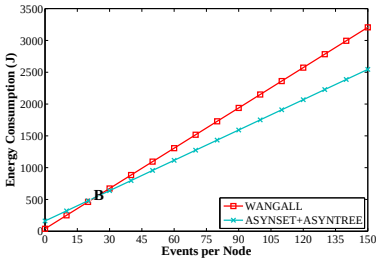

Figure 32: Energy consumptions of the network with 200 nodes, 5 node degree and 0-100 energy interval

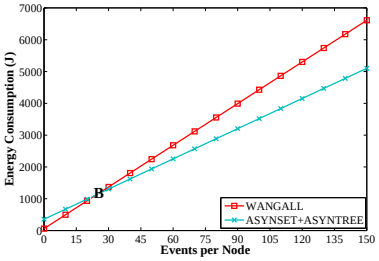

Figure 33: Energy consumptions of the network with 300 nodes, 5 node degree and 0-100 energy interval

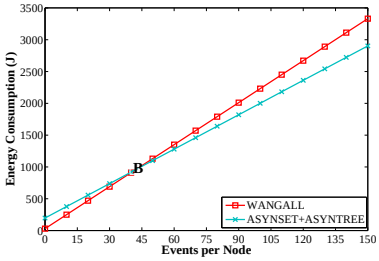

Figure 34: Energy consumptions of the network with 200 nodes, 4 node degree and 0-100 energy interval

consumption of the network with 100 nodes is shown in Fig. 31. In this figure, the energy consumption of WANGALL exceeds ASYNSET+ASYNTREE after the 40th event. In Fig. 32 and Fig. 33 the energy consumptions of the network with 200 nodes and 300 nodes are given respectively, where WANGALL exceeds ASYNSET+ASYNTREE after the 30th event. From these measurements we may say that our proposed backbone algorithms become more energy-efficient than the WANGALL when the node count is increased.

The total energy consumptions against the varying node degree are given in Fig. 32, Fig. 34 and Fig. 35. The energy consumption of the network with 200 nodes and 4 degree is shown in Fig. 34. In this figure, the energy consumption of WANGALL exceeds ASYNSET+ASYNTREE after the 50th event. When the node degree is increased, our backbone algorithms are more energy-efficient as shown in Fig. 32 and 34. Lastly, we investigate the energy consumptions against the varying energy interval. As shown in Fig. 32, Fig. 36 and Fig. 37, when the energy interval is increased, since our algorithms produce lower cost backbones with less number of nodes, they become more energy-efficient than WANGALL.

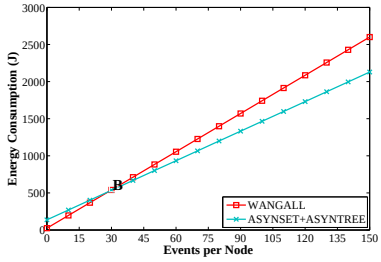

Figure 35: Energy consumptions of the network with 200 nodes, 6 node degree and 0-100 energy interval

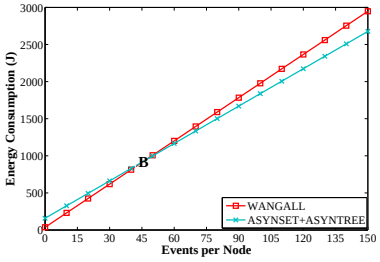

Figure 36: Energy consumptions of the network with 200 nodes, 5 node degree and 0-50 energy interval

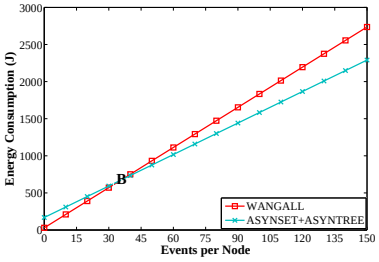

Figure 37: Energy consumptions of the network with 200 nodes, 5 node degree and 0-150 energy interval

Consequently, although our algorithms consume more energy than WANGALL during the backbone construction phase, the total energy consumed by the given sensor network application with the backbone construction of the WANGALL exceeds the total consumption of our algorithms after the 50th event at the worst case from our measurements. Besides, when the node count, node degree or energy interval is increased our proposed backbone algorithms become more energy-efficient since the quality of the produced backbone increases.

8. Conclusions

We proposed weighted connected dominating set algorithms for energy-efficient backbone formation in WSNs. We first design ASYNSET algorithm in order to produce a small weighted connected dominating set. ASYNSET is the semi-asynchronous and fully distributed version of CENTSET. After producing the weighted dominating set, we connect the dominators by designing ASYNTREE algorithm. This fully distributed algorithm use a heuristic derived from the Klein's algorithm. We gave the analysis for the proof of correctness, approximation ratio, time complexity, message complexity and space complexity of the algorithms. All algorithms are implemented in ns2 simulation environment and the results obtained show that our proposed algorithms select lower cost backbone than the other approaches. Although our proposed algorithms consume more energy than the related work during the backbone construction phase, the backbone construction is performed initially and possibly only after a topology change in the network. On the other side, communication via the backbone is performed regularly and the energy consumption of a sensor network application is decreased when our backbone is applied instead of the others as a result of our break-even analysis. So for the energy-efficient and long-lived sensor network applications, our algorithms are favorable.

References

[1] A. ur Rehman, A. Z. Abbasi, N. Islam, Z. A. Shaikh, A review of wireless sensors and networks' applications in agriculture, Computer Standards and Interfaces 36 (2) (2014) 263 – 270. doi:http://dx.doi.org/10.1016/j.csi.2011.03.004.
URL http://www.sciencedirect.com/science/article/pii/S0920548911000353

[2] Z. Li, N. Wang, A. Franzen, P. Taher, C. Godsey, H. Zhang, X. Li, Practical deployment of an in-field soil property wireless sensor network, Computer Standards and Interfaces 36 (2) (2014) 278 – 287. doi:http://dx.doi.org/10.1016/j.csi.2011.05.003.
URL http://www.sciencedirect.com/science/article/pii/S0920548911000651

[3] C. Alcaraz, J. Lopez, Diagnosis mechanism for accurate monitoring in critical infrastructure protection, Computer Standards and Interfaces 36 (3) (2014) 501 – 512. doi:http://dx.doi.org/10.1016/j.csi.2013.10.002.
URL http://www.sciencedirect.com/science/article/pii/S0920548913001281

[4] I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks, IEEE Trans. Parallel Distrib. Syst. 13 (1) (2002) 14–25. doi:http://dx.doi.org/10.1109/71.980024.

[5] L. Jia, R. Rajaraman, T. Suel, An efficient distributed algorithm for constructing small dominating sets, Distrib. Comput. 15 (4) (2002) 193–205. doi:http://dx.doi.org/10.1017/s00446-002-0078-0.

[6] F. Kuhn, R. Wattenhofer, Constant-time distributed dominating set approximation, Distrib. Comput. 17 (4) (2005) 303–310.

[7] F. Kuhn, T. Moscibroda, R. Wattenhofer, The price of being near-sighted, in: SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, ACM, New York, NY, USA, 2006, pp. 980–989. doi:http://doi.acm.org/10.1145/1109557.1109666.

[8] J. Wu, H. Li, A dominating-set-based routing scheme in ad hoc wireless networks, Telecommunication Systems Journal 3 (1999) 63–84.

[9] F. Dai, J. Wu, An extended localized algorithm for connected dominating set formation in ad hoc wireless networks, IEEE Trans. Parallel Distrib. Syst. 15 (10) (2004) 908–920. doi:http://dx.doi.org/10.1109/TPDS.2004.48.

[10] D. Cokuslu, K. Erciyes, O. Dagdeviren, A dominating set based clustering algorithm for mobile ad hoc networks, in: Proc. ICCS2006, LNCS 3991, Springer-Verlag, 2006, pp. 571–578.

[11] T. Erlebach, A. Shahnaz, Approximating node-weighted multicast trees in wireless ad-hoc networks, in: IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing, ACM, New York, NY, USA, 2009, pp. 639–643. doi:http://doi.acm.org/10.1145/1582379.1582518.

[12] A. Srinivas, G. Zussman, E. Modiano, Construction and maintenance of wireless mobile backbone networks, IEEE/ACM Trans. Netw. 17 (1) (2009) 239–252. doi:http://dx.doi.org/10.1109/TNET.2009.2012474.

[13] F. Zou, Y. Wang, X.-H. Xu, X. Li, H. Du, P. Wan, W. Wu, New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs, Theor. Comput. Sci. 412 (3) (2011) 198–208. doi:10.1016/j.tcs.2009.06.022. URL http://dx.doi.org/10.1016/j.tcs.2009.06.022

[14] J. A. Torkestani, M. R. Meybodi, Finding minimum weight connected dominating set in stochastic graph based on learning automata, Information Sciences 200 (0) (2012) 57 – 77. doi:http://dx.doi.org/10.1016/j.ins.2012.02.057.
URL http://www.sciencedirect.com/science/article/pii/S0020025512001776

[15] M. Chatterjee, S. K. Das, D. Turgut, Wca: A weighted clustering algorithm for mobile ad hoc networks, Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks 5 (2001) 193–204.

[16] L. Bao, J. J. Garcia-Luna-Aceves, Topology management in ad hoc networks, in: MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing, ACM, New York, NY, USA, 2003, pp. 129–140. doi:http://doi.acm.org/10.1145/778415.778432.

[17] Y. Wang, W. Wang, X.-Y. Li, Efficient distributed low-cost backbone formation for wireless networks, IEEE Trans. Parallel Distrib. Syst. 17 (7) (2006) 681–693. doi:http://dx.doi.org/10.1109/TPDS.2006.86.

[18] M. Ghaffari, Near-optimal distributed approximation of minimum-weight connected dominating set, in: J. Esparza, P. Fraigniaud, T. Husfeldt, E. Koutsoupias (Eds.), Automata, Languages, and Programming, Vol. 8573 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2014, pp. 483–494.

[19] O. Dagdeviren, K. Erciyes, Graph matching-based distributed clustering and backbone formation algorithms for sensor networks, Comput. J. 53 (10) (2010) 1553–1575. doi:10.1093/comjnl/bxq004.

[20] IEEE Standard for Information technology-Telecommuncations and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), 2003.

[21] Draft Standard for Information Technology - Telecommunications and Information Exchange Between Systems - LAN/MAN Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2007.

[22] W. Ye, J. Heidemann, D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, IEEE/ACM Transactions on Networking 12 (2004) 493–506.

[23] L. Choi, S. H. Lee, H. Choi, M-mac: Mobility-based link management protocol for mobile sensor networks, in: STFSSD '09: Proceedings of the 2009 Software Technologies for Future Dependable Distributed Systems, IEEE Computer Society, Washington, DC, USA, 2009, pp. 210–214. doi:http://dx.doi.org/10.1109/STFSSD.2009.47.

[24] J. Polastre, J. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in: SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, ACM, New York, NY, USA, 2004, pp. 95–107. doi:http://doi.acm.org/10.1145/1031495.1031508.

[25] S. Lai, B. Ravindran, On distributed time-dependent shortest paths over duty-cycled wireless sensor networks, in: INFOCOM'10: Proceedings of the 29th conference on Information communications, IEEE Press, Piscataway, NJ, USA, 2010, pp. 1685–1693.

[26] K. Erciyes, D. Ozsoyeller, O. Dagdeviren, Distributed algorithms to form cluster based spanning trees in wireless sensor networks, in: ICCS '08: Proceedings of the 8th international conference on Computational Science, Part I, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 519–528.

[27] M. Luby, A simple parallel algorithm for the maximal independent set problem, in: STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing, ACM, New York, NY, USA, 1985, pp. 1–10. doi:http://doi.acm.org/10.1145/22145.22146.

[28] F. Kuhn, T. Nieberg, T. Moscibroda, R. Wattenhofer, Local approximation schemes for ad hoc and sensor networks, in: DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing, ACM, New York, NY, USA, 2005, pp. 97–103. doi:http://doi.acm.org/10.1145/1080810.1080827.

[29] Y. P. Chen, A. L. Liestman, Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks, in: MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, ACM, New York, NY, USA, 2002, pp. 165–172. doi:http://doi.acm.org/10.1145/513800.513821.

[30] V. Chvatal, A greedy heuristic for the set-covering problem, Mathematics of Operations Research 4 (3) (1979) 233–235. doi:10.2307/3689577.

[31] B. Liang, Z. J. Haas, Virtual backbone generation and maintenance in ad hoc network mobility management, in: Proc. IEEE INFOCOM, 2000, pp. 1293–1302.

[32] P. Klein, R. Ravi, A nearly best-possible approximation algorithm for node-weighted steiner trees, J. Algorithms 19 (1) (1995) 104–115. doi:http://dx.doi.org/10.1006/jagm.1995.1029.

[33] R. Muhammad, Distributed steiner tree algorithm and its application in ad hoc wireless networks, in: ICWN, 2006, pp. 173–178.

[34] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (1996) 374–387.

[35] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync protocol for sensor networks, in: SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems, ACM, New York, NY, USA, 2003, pp. 138–149. doi:http://doi.acm.org/10.1145/958491.958508.

[36] B. Das, V. Bharghavan, Routing in ad-hoc networks using minimum connected dominating sets, in: Proceedings of IEEE ICC, 1997, pp. 376–380.

[37] R. G. Gallager, P. A. Humblet, P. M. Spira, A distributed algorithm for minimum-weight spanning trees, ACM Trans. Program. Lang. Syst. 5 (1) (1983) 66–77.