

An Enhanced Top-Down Cluster and Cluster Tree Formation Algorithm for Wireless Sensor Networks

H. M. N. Dilum Bandara¹ and Anura P. Jayasumana²

^{1,2}Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA.
dilumb@engr.colostate.edu¹, Anura.Jayasumana@Colostate.edu²

Abstract – Clustering is a key technique to simplify network management while enabling power conservation and reduced channel contention in large scale Wireless Sensor Networks. A hierarchy of clusters in the form of a cluster tree can further enhance upper layer functions such as routing, broadcasting and query delivery. We propose a generic top-down cluster and cluster tree formation algorithm that does not depend on neighborhood information, location awareness, time synchronization and network topology. It also scales well into large networks. By varying parameters in the algorithm, cluster trees with desirable properties such as controlled breadth and depth, uniform cluster size and improved circularity can be achieved. Different characteristics of clusters and cluster trees are evaluated using simulation based results.

I INTRODUCTION

Recent advances in wireless sensor network (WSN) technology allows the sensing of the physical world around us at a far greater temporal and spatial granularity than has been hitherto possible, leading to novel applications in areas such as habitat monitoring, disaster response, eldercare and battlefield intelligence.

Energy efficient operation, meeting response time requirements, management and maintenance of such networks are complex and critical issues that have to be addressed with large WSN deployments. Many solutions and algorithms for overcoming these problems depend on decomposing the network into number of administrative entities called *clusters* [1, 3-6, 9-12]. The structure imposed by clustering makes it somewhat easier to manage the problems introduced by the complexity of large-scale WSNs. In general, the nearby nodes in a network are grouped into a set of clusters, with each cluster managed by a *Cluster Head* (CH). In many solutions, the nodes within a cluster communicate only with their CH. Communication among CHs can be via either single or multi-hops. CHs are responsible for coordinating both inter-cluster and intra-cluster communication. Clustering is particularly useful for logically separating multiple sensor applications that perform different tasks and that are deployed in the same physical area [8].

Clustering based solutions can reduce the power consumption of a WSN, thus increasing its lifetime [6, 12]

and enable the spatial reuse of communication channel thus reducing collisions. The number of messages that flow through the network can be further reduced by aggregating data within clusters [6]. Upper layer functions such as routing, broadcasting and query delivery can be further enhanced by forming a hierarchy of clusters known as the *cluster tree*. Hierarchical routing allows better selection of routing paths and requires only keeping track of individual CHs rather than each and every node in the network.

However forming and maintaining clusters is a complex task and the associated communication messages may add a considerable overhead. The rotation of role of becoming a CH (e.g., to balance the workload) and handling node dynamic such as new, moving or deteriorating nodes are among other issues that need to be addressed.

Clusters can be formed using either centralized [6] or distributed [3-4, 6, 11-12] approaches. The lowest ID clustering [9], DCA [4] and Max-Min d-clustering [1] are solutions that are relatively simple to implement, yet not directly applicable to WSNs since they are not energy aware. LEACH [6] achieves longer network lifetime by selecting CHs based on residual energy of a node and aggregation. It does not guarantee good CH distribution and some of these problems are addressed in [11, 12].

In a bottom-up hierarchical approach, [3, 10] for example, the individual clusters are formed independently and later combined together to form a higher-level structure such as a cluster tree. This approach, although conceptually appears to be relatively simple, involves considerable communication overhead while building the tree and provides very little or no control on depth and breadth of the tree formed.

In Top-Down Clustering (TDC), a distributed technique for building cluster trees, a designated *root node* forms its own cluster first. Then it selects some of its neighbors to become CHs and form their own clusters. This process continues until the entire sensor field is covered. The cluster tree is formed automatically by keeping track of parent and child CH relationships. The top-down approach provides more control while forming clusters and the cluster tree. However as we explain in [2] uncontrolled TDC approaches such as the basic scheme given in the IEEE 802.15.4 cluster tree [7], referred to hereafter as Simple Hierarchical Clustering (SHC), result in undesirable cluster and tree characteristics such as large variations in cluster size and distance to leaf nodes. The IEEE 802.15.4 standard

This work was supported in part by the grant from Environmental Sciences Division, Army Research Office (AMSRD-ARL-RO-EV).

however is quite flexible and does not prevent one from deploying alternative clustering approaches.

In SHC, the Personal Area Network (PAN) coordinator forms the first cluster and broadcasts beacon frames with its PAN ID to neighboring nodes. A node receiving a beacon may join the cluster. The PAN coordinator adds the new node as a child node in its *neighbor list* and the newly joined node adds the PAN coordinator as its parent. Then it keeps forwarding beacon frames from the PAN coordinator and other nodes may then join the cluster at this node. Once a predefined application or a network requirement such as number of nodes in a cluster or maximum hop count is met, the PAN coordinator may instruct a node to become a coordinator of a new cluster. This process continues until all the nodes are covered. It is unlikely that any of the clusters will be circular in shape geographically, except perhaps the first cluster. Although this approach is simple to implement it produces non-optimal clusters and trees with a high depth [2]. ZigBee proposes a different implementation for clustering in 802.15.4 networks [10]. It first forms individual clusters and later combine them to form a cluster tree. ACP [5], another TDC solution, depends on device location information. ACP does not form a cluster tree and produces a large number of overlapping clusters.

In this paper, we propose GTC, a Generic Top-down Cluster and cluster tree formation algorithm that does not depend on neighborhood information, location awareness, time synchronization and network topology. The algorithm scales well into large networks, and is configurable. The SHC scheme of IEEE 802.15.4 is a special case of GTC. Another special case, Hierarchical Hop-ahead Clustering (HHC), is presented that result in significantly better clustering solutions. By varying parameters of the GTC algorithm, cluster trees with desirable properties such as controlled breadth and depth, more uniform cluster size and circular clusters can be achieved.

Section II explains the GTC algorithm. Control of cluster and tree characteristics are covered in Section III. Sections IV and V discuss the simulator and simulation results respectively. Conclusions are in Section VI.

II TOP-DOWN CLUSTERING ALGORITHM

The Generalized Top-down Clustering (GTC) algorithm is shown in Fig. 1. The root node initiates the cluster formation by executing the *Form_cluster* function. It sends a broadcast (*Broadcast_cluster*) indicating its Node ID (*NID*), Cluster ID (*CID*), maximum distance to a child node from the CH (*MaxHops*) and no of hops to forward the broadcast (*TTL*). Nodes execute the *Join_cluster* function and waits for a cluster formation broadcast (*Broadcast_cluster*). A node hearing this broadcast will join the cluster if it is not already a member of another cluster (*MyCID=0*) and within *MaxHops*. An acknowledgement is sent to the CH indicating its node ID (*CNID*), its distance to the CH (*hops*) and properties of the node (P_1, P_2) such as the residual energy and node degree. The CH receiving this acknowledgement adds the node to its *ack_list*. Then the

```

Form_cluster(NID, CID, T, N, MaxHops, TTL) {
    Wait(T)
    Broadcast_cluster(NID, CID, MaxHops, TTL)
    ack_list ← Receive_ack(CNID, hops, timeout, P1, P2)
    For i = 1 to N {
        CCHi ← Select_candidate_CH( TTL, ack_list, P1, P2)
        CIDi ← Select_next_CID()
        Ti ← Select_delay()
        Request_form_cluster(CCHi, CIDi, Ti, N, MaxHops, TTL)
    }
}

Join_cluster() {
    Listen_broadcast_cluster(NID, CID, MaxHops, TTL)
    If(hops ≤ MaxHops & MyCID = 0)
        MyCID ← CID, MyCH ← NID
    Send_ack(CNID, Hops)
    TTL ← TTL - 1
    If(TTL > 0)
        Forward_broadcast_cluster(NID, CID, MaxHops, TTL)
    Else {
        Listen_form_cluster(CCH, CID, T, N, MaxHops, TTL, timeout)
        Form_cluster(CCH, CID, T, N, MaxHops, TTL)
    }
}

```

Fig. 1. Cluster and tree formation algorithm

child node forwards the cluster formation broadcast (*Forward_broadcast_cluster*) if the *TTL* has not expired (i.e. $TTL > 0$). If the *TTL* is expired ($TTL = 0$) the node is capable of being selected as a candidate to become a CH of a new cluster. At this stage the node is either at the edge of the cluster (if $TTL = MaxHops$) or outside the cluster (if $TTL > MaxHops$). Therefore the node waits for a cluster formation request (*Listen_cluster_formation*) from the CH. Algorithm terminates if such a message does not arrive before the function time-outs.

The CH keeps receiving acknowledgements until the *Receive_ack* function time-outs. Then *N* candidate CHs (*CCH_i*) are selected from the *ack_list* by the *Select_candidate_CHs* function such that $hops = TTL$ (i.e. node that are either at the edge or outside the cluster). Then the *Request_form_cluster* function sends a message to those selected Candidate CHs (CCH) asking them to form their own clusters. It also sends their new cluster ID (*CID_i*) and a time delay (*T_i*) to wait before forming their own clusters. Upon receiving the message CCHs form their own clusters by executing the same *Form_cluster* function. This process continues until all the nodes are covered. If a CCH is not able to attract any child nodes it will join the parent CH (only if it is at the edge of the parent cluster) as a child node. Cluster tree is rooted at the root node and is formed by each CH keeping track of its parent and child CHs.

III CONTROL OF CLUSTER AND TREE CHARACTERISTICS

The solution generated by the algorithm depends on implementation of functions such as *Select_next_CID*, *Select_delay* and parameters such as *MaxHops*, *TTL*, *T_i* and *N*. By controlling these parameters, a wide range of solutions could be obtained.

By controlling *TTL* distance between parent and child CHs can be controlled. If $TTL = MaxHops$, CCHs can be selected

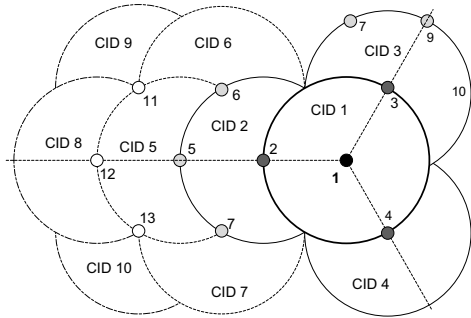


Fig. 2. Physical shape of ideal SHC clusters. $MaxHops = TTL = 1$

from any node that are $MaxHops$ away. Fig. 2 illustrates conceptually how the sensor field can be optimally covered by selecting 3 CCHs at each level, that are separated physically as widely as possible. In the optimum case those CCHs should be selected from nodes that are at the edge of the parent cluster. In order to make the diagram simple, only CCHs are indicated and one branch is expanded into several levels. Node 1 acts as the root node and forms cluster 1 by joining all the one-hop neighbors. Then it asks 3 of its neighbors (2-4) which are at the edge of the cluster ($TTL=1$) to form their own clusters. It is sufficient to expand 3 CCHs if those are in different directions in the sensor field. However effectiveness of this decision depends on availability of location information. Then node 2 asks 3 of its neighbors (5-7) to form clusters 5, 6 and 7. Even in this conceptual case, the shapes of clusters are not circular except for the 1st cluster. Notice that clusters 6 and 7 are smaller than cluster 5. Clusters 9 and 10 are even smaller. Therefore as the depth of the tree increases average cluster size becomes smaller and variance gets higher. Large clusters can be generated by increasing $MaxHops$ and communication range of nodes. This approach is similar to the SHC scheme of IEEE 802.15.4 standard.

In Hierarchical Hop-ahead Clustering (HHC) TTL is selected such that; $TTL = 2 \times MaxHops + 1$. This allows nodes that are several hops away from the edge of the parent cluster to be selected as CCHs. Only nodes within $MaxHops$ join the cluster and other nodes keep forwarding the cluster formation broadcast if $TTL > 0$. Fig. 3 illustrates conceptually how the sensor field can be optimally covered with HHC. Node 1 acts as the root node and sends a cluster formation broadcast with $MaxHops=1$ and $TTL=3$. All the one-hop neighbors join cluster 1 and the broadcast is forwarded to the next set of neighbors since $TTL > 0$. This process continues until $TTL=0$. For an example, after hearing the broadcast from node 1, node 2 joins the cluster. Then forward the broadcast to node 3 and from node 3 to 4. Finally node 4 is 3 hops away from the CH and is a candidate to be selected as a new CH. After hearing from all the nodes within 3-hop range CH select 6 CCHs (nodes 4-9) and request them to form their own clusters. It is sufficient to expand 6 CCHs, if those are in different directions in the sensor field. After level 1 only 3 CCHs are need to be selected by each parent CH (e.g.

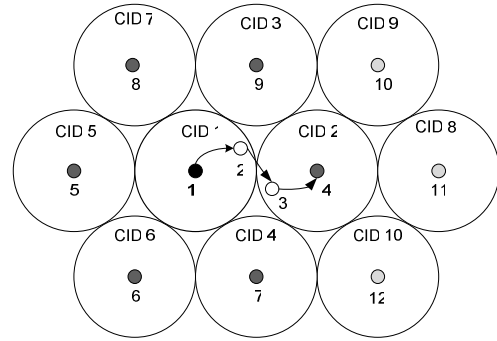


Fig. 3. Physical shape of ideal HHC clusters. $MaxHops=1, TTL=3$

clusters 8-9 are formed by cluster 2). Key advantages of this approach are the large clusters, more circular clusters and better distribution of CHs. Though there are uncovered areas between clusters in Fig. 3 in a practical scenario those will be covered where cluster shapes are not ideal.

The *Select_next_CID* function assigns cluster IDs to the newly formed clusters. These IDs can be assigned either by the root node or based on the node ID of the CCH. The root node can assign IDs sequentially or hierarchically. In the hierarchical approach, cluster IDs can be assigned based on the branch of the cluster tree that a cluster belongs to. Such information can be useful in hierarchical routing and while optimizing the cluster tree in order to achieve lower depth. This is also useful in enabling communication across branches without routing data through the root node. Such a hierarchical numbering scheme will also enable distributed cluster formation where individual cluster trees are later combined together and forms a *cluster graph*. However this approach generates more messages, since for each CCH, a new CID is needed to be requested from the root node. If the CCH is unable to form a new cluster, this also needs to be informed back to the root node. When cluster ID is derived from the NID, the CCH needs to inform the root node only if it creates a cluster.

The *Select_candidate_CH* function selects number of nodes (N) as candidates to become CHs of a new cluster. Nodes having the highest hop count ($hops=TTL$) in the *ack_list* are selected as candidates. An optimal set of CCHs can be selected if node location information is available. However if such information is not available random selection can be used. There can be large number of nodes to be elected as CCHs if the communication range or $MaxHops$ is higher. When CCHs are randomly selected from such a large number of nodes there is a higher chance of selecting physically nearby nodes. Because of this, the tree may not span into part of the network and this can reduce the connectivity. As we explain in [2] the maximum number of branches off a CH, and hence the overall tree depth distribution depends on N . Therefore it is important to select suitable number of CCHs as the communication range or $MaxHops$ increases. Parameters (P_1, P_2) such as node's residual energy and node degree that are reported while acknowledging cluster formation broadcast can also be used to select the appropriate CCHs. Better load

balancing can be achieved by selecting CCH based on residual energy while dense clusters can be built by selecting nodes with higher node degree [12].

When multiple CCHs start sending the cluster formation broadcast at the same time it increases potential for collision. Because of this some nodes may not even hear from any of the CHs. This may also affect the lifetime of the network and node connectivity. The problem can be overcome by assigning some delay that each CCH has to wait before sending the cluster formation broadcast. This is achieved by the *Wait* function. Further more, the shape of the cluster tree can also be controlled by varying the delay (T). By assigning appropriate time delays the algorithm above can be used for breadth-first, depth-first or some scheme in between [2]. For breadth-first tree formation, delay should ensure that the current level in the tree finishes expanding before going to the next level ($T_L(i) < T_L(i+1)$), where $T_L(i)$ defines time to expand level i . For depth-first tree formation delay should be set such that a branch is allowed to complete its expansion before another branch starts expanding ($T_B(i) < T_B(i+1)$), where $T_B(i)$ defines time to expand branch i . The *Select_delay* function decides on a suitable delay based on the desired shape of the cluster tree. For each CCH the delay (T_i) is calculated and indicated in the *Request_form_cluster* message. In HHC, if a CCH hears a cluster formation broadcast from another CH while it is delaying its cluster forming it will join that CH if it is within the *MaxHops*.

Another approach would be to use a *Wait_delay* function rather than a *Select_delay* function. Rather than sending the *Request_form_cluster* message with T_i the parent CH can wait the required delay and then ask CCH to form its own cluster. This approach has much more control over the previous one, where the parent CH can dynamically decide on suitable CCHs based on the information on clusters that are already formed. When a CCH forms a cluster it can inform the parent CH about the newly acquired child nodes. Then the parent CH can avoid selecting those nodes as CCHs. This prevents the issue of CCH being overtaken by another cluster in HHC. Time to form clusters depends on the delay assignment. Since 2 clusters are not allowed to form at the same time the worst case time complexity for breadth-first tree formation depends on the number of nodes (D) in the network $O(D)$. It would be even worse for depth-first case since we may not have an idea about the maximum depth when the algorithm starts. Time to form clusters can be reduced by allowing CHs on different branches of the tree to expand at the same time. However this does not guarantee optimal tree formation. Also note that this is just a delay, not an exact time, therefore the algorithm does not require clock synchronization among sensor nodes. Delay for the root node can be set to zero. For 1-hop clusters, both SHC and HHC have a worst case message complexity of $O(D)$.

To measure how circular a given clusters is, Maximum Achievable Circularity (MAC) is defined. It is the ratio

between the actual number of nodes that are in a cluster and total number of nodes that are in the range of the CH.

$$MAC_i = \frac{\text{No of nodes in cluster } i}{\text{Total no of nodes in the range of } CH_i} \times 100$$

If a cluster can attract all the neighbors in a one-hop or multi-hop neighborhood, MAC is 100%. In the multi-hop case if there is no node to forward a message from hop n to hop $n+1$, nodes belong hop $n+1$ are not considered to be in the range of the CH. Circular clusters cannot be formed at the border of the sensor field. However these clusters have attracted the maximum number of nodes that they can attract. MAC takes this into account and assigns 100% to those clusters, allowing us to discard the border effects.

IV SIMULATOR

A discrete event simulator was developed using C. Nodes were randomly placed on a 100x100 square grid with a given probability (e.g., 1, 0.5 and 0.25). Distance between adjacent grid points in X and Y directions were set to 6 units. For each node density 100 random networks were generated. CCHs are randomly selected and breadth-first tree formation approach was used to build the cluster tree. Parameters such as communication range (R), number of nodes in the network (D), number of CCHs at each level (N), maximum number of hops to a child node from the CH (*MaxHops*) and location of the root node were varied. Average of 100 sample runs based on the pre-generated networks was taken for calculating each variable of interest. Except where noted, the simulation results presented use 5000 nodes (i.e., node probability of 0.5) with the root node in the middle of the sensor field. N was selected as 3 and 6 for SHC and HHC respectively based on Fig. 2 and 3. Multi-hop communication within a cluster and single-hop communication from cluster-head to cluster-head is assumed. The circular communication model is used for signal propagation. The nodes were homogeneous, stationary, and had a fixed transmission range.

V RESULTS

Fig. 4 show the physical shape of the clusters for sample runs of the two variants of the algorithm. For SHC (Fig. 4(a)), it can be seen that only the first cluster has approximately circular shape while the shape of area covered by other clusters vary widely. It illustrates the fact that the practical results differ widely from the conceptual scenario. Also note that in Fig. 4(a) CHs for clusters 2 and 4 are inside cluster 1. This cannot be prevented unless information about node locations are available, since any node within 1-hop (*MaxHops*=1) is a candidate to become a CH. CH of cluster 3 is curved into cluster 1. This happens because CCHs are selected at the edge of the cluster. Also note that clusters G and F (indicated by the arrow) do not have any child nodes. It is not because they initially did not have any child nodes, but because those child nodes were later selected to become CCHs of clusters W and 'c'. On the other hand HHC clusters in Fig. 4(b) are much larger and more circular. However we cannot guarantee that a CH

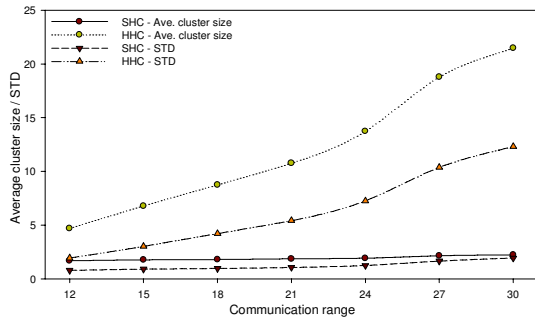


Fig. 7. Number of clusters/CHs produced by each solution. MaxHops=1

minimize the effect of selecting physically near by nodes as CCHs which ensure maximum expansion of branches in the cluster tree at each level. However as N increases average cluster size slightly reduces and more clusters are formed. If the root node happens to be at an edge of the sensor field both maximum depth and average node depth increase. Similar behaviors were observed for a less dense network of 2500 nodes.

VI CONCLUSIONS AND FUTURE WORK

We illustrated a generic top-down cluster and cluster tree formation algorithm that does not depend on neighborhood information, location awareness, time synchronization and network topology. By selecting appropriate parameters, clusters and cluster trees with desirable properties such as controlled breadth and depth, uniform cluster size and more circular clusters can be achieved. Based on the observations, it is clear that HHC outperforms the SHC. An optimization phase after generating the clusters and cluster tree may further improve the solution. Such a phase may also be able to maintain clusters in spite of deterioration of node power by re-assigning role of CH to a different node without reclustering the entire network. In future we also expect to analyze the impact of variables N , T , TTL , etc. Currently we are working on such an optimization phase.

REFERENCES

- [1] A.D. Amis, R. Prakash, Thai H P Vuong and Dung T Huynh, "Max-Min d-cluster formation in wireless ad-hoc networks", In Proc. IEEE INFOCOM'2000, Tel Aviv, March 2000.
- [2] H. M. N. D. Bandara and A. P. Jayasumana, "Challenges in cluster tree formation with top-down approach for wireless sensor networks", Unpublished
- [3] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks", In Proc. INFOCOM 2003, San Francisco, Apr. 2003.

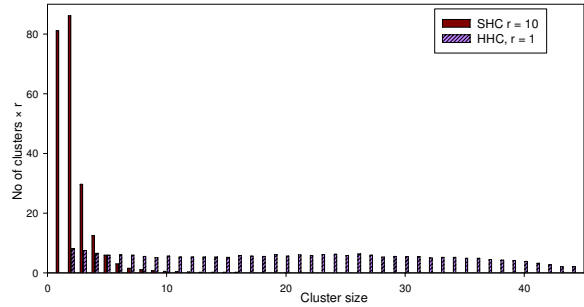


Fig. 8. Distribution of cluster size. R=30, MaxHops=1

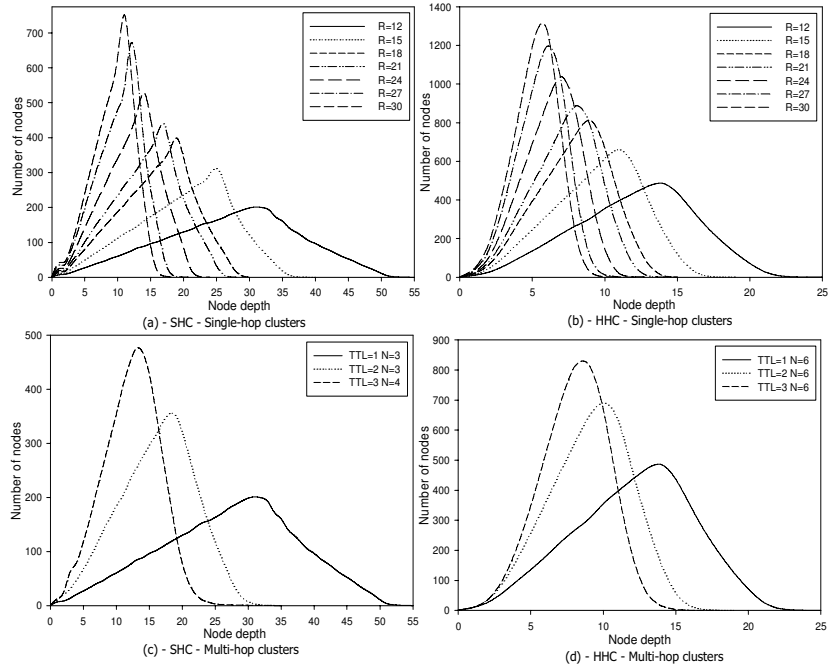


Fig. 9. Number of nodes at different levels of the cluster tree for breadth first tree formation.

- [4] S. Basagni, "Distributed clustering for ad-hoc networks", In Proc. I-SPAN'99, Australia, June 23-25, 1999, pp.310-315.
- [5] A. Durrresi and V. Paruchuri, "Adaptive clustering protocol for sensor networks", IEEE Aerospace Conf. 2005, March 2005, pp. 1-8.
- [6] W. B. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks", IEEE Trans. Wireless Commun., vol. 1, no. 4, Oct. 2002. pp. 660-670.
- [7] IEEE Computer Society, "IEEE.802.15.4: Wireless medium access control and physical layer specifications for low-rate wireless personal area networks", September 2006.
- [8] A. P. Jayasumana, Q. Han and T. Illangasekare, "Virtual sensor networks - a resource efficient approach for concurrent applications", In Proc. ITNG 2007, Las Vegas, April 2007.
- [9] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks", IEEE Journal on Selected Areas in Communications, vol. 15, no. 7, Sept. 1997, pp. 1265-1275.
- [10] M. Maeda and Ed Callaway, "Cluster tree protocol (ver. 0.6)", Apr. 2001, available at: http://www.ieee802.org/15/pub/2001/May01/01189r0P802-15_TG4-Cluster-Tree-Network.pdf
- [11] L. Ying and Y. Haibin, "Energy adaptive cluster-head selection for wireless sensor networks", Proc. PDCAT 05, China, Dec 2005, pp. 634 - 638.
- [12] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach", Proc. IEEE INFOCOM 2004, Hong Kong, Mar. 2004