On Tree-Based Convergecasting in Wireless Sensor Networks

V. Annamalai, S. K. S. Gupta
Dept. of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287
sandeep.gupta@asu.edu

L. Schwiebert
Computer Science Department
Wayne State University
Detroit, MI 48202
loren@cs.wayne.edu

Abstract— **A Wireless Sensor Network (WSN) consists of sensors implanted in an environment for collecting and transmitting data regarding changes in the environment based on the requests from a controlling device (called base station) using wireless communication. WSNs are being used in medical, military, and environment monitoring applications. Broadcast (dissemination of information from a central node) and Convergecast (gathering of information towards a central node) are important communication paradigms across all application domains. Most sensor applications involve both convergecasting and broadcasting. The time taken to complete either of them has to be kept minimal. This can be accomplished by constructing an efficient tree for both broadcasting as well as convergecasting and allocating wireless communication channels to ensure collision-free communication. There exist several works on broadcasting in multi-hop radio networks (a.k.a. ad hoc networks) which can also be used for broadcasting in WSNs. These algorithms construct a broadcast tree and compute a schedule for transmitting and receiving for each node to achieve collision-free broadcasting.**

In this paper we show that we need a new algorithm for applications which involve both convergecasting and broadcasting since the broadcast tree may not be efficient for convergecasting. So we propose a heuristic algorithm (Convergecasting Tree Construction and Channel Allocation Algorithm (CTCCAA)) which constructs a tree with schedules assigned to nodes for collision free convergecasting. The algorithm is capable of code allocation (Direct Sequence Spread Spectrum (DSSS)/ Frequency Hopping Spread Spectrum (FHSS)), in case multiple codes are available, to minimize the total duration required for convergecasting. We also show that the same tree can be used for broadcasting and is as efficient as a tree exclusively constructed for broadcasting.

I. Introduction

Advances in the micro-electronics industry and wireless technology have led to the development of sensors that can be used for accurate monitoring of inaccessible environments. A sensor is a device that can be controlled and queried by an external device to detect, record, and transmit information regarding a physiological change or the presence of various chemical or biological materials in the environment [4]. A collection of such sensors form a network that can be used for efficient monitoring of health, environment, military etc. [5].

A wireless sensor consists of a small processor, memory, power, sensing and transceiver units. Additionally, a sensor can have location finding system, mobilizer and a power generator which are application dependent sub-units [5]. The size and weight of a sensor limits the processing capability, amount of memory and the amount of power that it can store. A Major part of power consumed by a sensor is used to run the transceiver circuitry. As, the transmission range of a sensor increases the power consumed by the transceiver also increases. Since they have limited transmission range sensors together form a multi-hop radio network to accomplish communication amongst themselves.

Many sensor applications require broadcasting and/or convergecasting. Broadcasting is the process of information dissemination from a node in the network to all other nodes in the network. Retinal prosthesis of [1] is one such application. Convergecast is the aggregation of data collected at each node in the network towards a central node in the network. A convergecast is usually preceded by a broadcast, or both can occur in an interleaved manner. An example of this communication pattern is the environment monitoring application in which sensors embedded near the area of interest collect and transmit data to an external monitoring station in response to a query (which is broadcast) by the monitoring station.

Collisions occur in a wireless network when multiple nodes simultaneously transmit to the same node over the same channel or a receiver is in the transmission range of another communication taking place over the same channel. Such collisions waste resources (e.g. bandwidth and energy) as well as increases data latency and hence they are undesirable. For broadcast and convergecast to work in a collision-free manner, we have to construct a tree and allocate a schedule that specifies for each node in the network the time-slots in which it will receive data from other nodes and the time-slot(s) in which it will send data to other node(s). This schedule is assigned for a particular duration of time and it gets repeated for each such time duration. This form of allocating time-slots and avoiding collision is called TDMA channel allocation. Instead of all the nodes contending for the channel, the nodes use

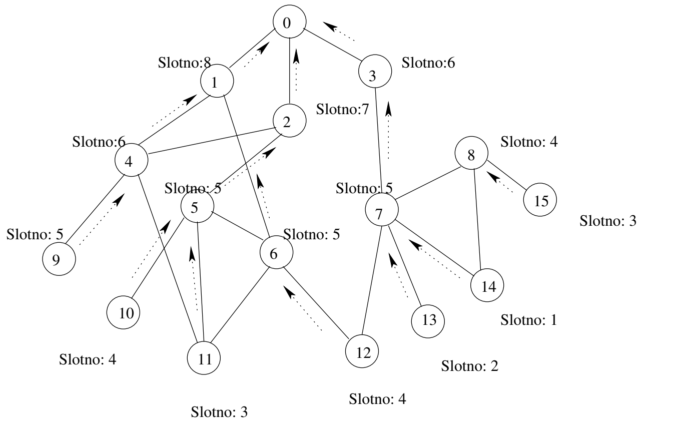


Fig. 1. **Sample network from [3] with schedule assigned for convergecasting.**

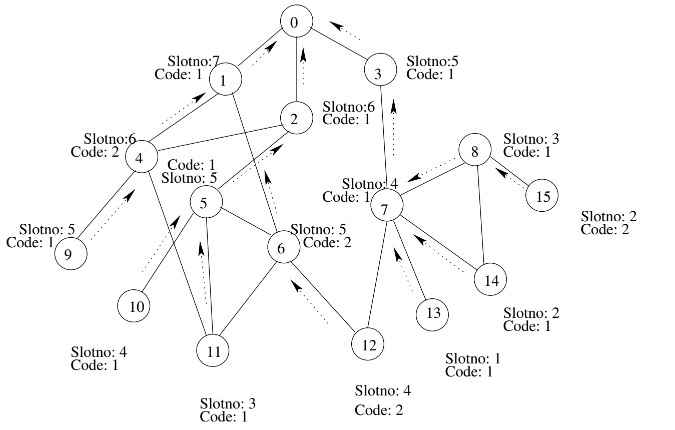


Fig. 2. **Sample network from [3] with schedule assigned for convergecasting using multiple code.**

the channel in a synchronized manner thereby increasing reliability. This is desirable for real-time applications. Each application will have an upper limit on the time within which it has to aggregate and/or disseminate the data. The number of time-slots we can allocate and the duration of time-slots is constrained by this time limit which in turn depends on the tree constructed. An example of convergecast tree is given in Figure 1 with the schedule assigned to each node. Data aggregation starts at the leaf nodes (nodes with numbers 9,10,11,12,13,14 and 15) and propagates up through the intermediate nodes towards the root node (node 0).

Most sensor applications make use of the ISM band which necessitates them to use spread-spectrum multiaccess techniques such as Direct Sequence Spread-Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS). These multiaccess techniques employ orthogonal codes or hopping sequences, which permits neighboring nodes to receive without any interference if they use different codes for communication. This can be used to further reduce the completion time (latency) of one broadcast or convergecast operation by assigning different time-slot and code pair to interfering communications. Figure 2 illustrates a sample network with schedule assigned for convergecasting using multiple codes.

In [6] the authors have proposed a directed diffusion paradigm for communication in sensor networks. This paradigm can be used for both broadcasting and for convergecasting at the network layer. Reliability is achieved by using a contention based MAC layer. Contention is undesirable because it leads to resource wastage and increases latency. Therefore this approach cannot be used. In [2] and [3] the authors have proposed TDMA channel allocation schemes for broadcasting in wireless multi-hop radio networks. In [2] it is shown that the problem of determining optimal channel allocation for broadcasting is NP-hard. Similarly, it can be shown that the problem of optimal channel allocation for convergecasting is also NP-hard. One simple way to achieve collision-free convergecasting is to compute a new schedule for convergecasting over a broadcast tree constructed by the broadcast approach of [2]. As illustrated by the simulation results presented later in the paper, this tree is not efficient for convergecasting. Hence, in this paper we propose a centralized algorithm which constructs a tree and generates the schedule for convergecasting instead of reusing any existing broadcast tree. In Section IV we compare the total duration for aggregation of data by doing slot allocation on the trees constructed by both approaches and show that the slot duration is high if the broadcast tree is used instead of constructing a new convergecast tree. We also show that the total time taken for broadcasting on a tree constructed for convergecasting is as efficient as over a tree constructed exclusively for broadcasting.

II. SYSTEM MODEL AND PROBLEM DESCRIPTION

The system consists of n sensors placed randomly in the environment that is being monitored. These sensors are controlled by the base station (node number 0) placed away from the environment of interest. Each sensor has an omni-directional antenna. The sensor nodes, including the base station in the network have equal transmission range (say d). We also assume that the base station has complete information about the location of the nodes and the nodes in the network use synchronized clocks and there is no drift between them. The data aggregation starts at the leaf nodes and propagates towards the base station. There is no mobility in the network. All nodes generate equal amount of data. The communication channel is divided over both time and code.

Formally, the network is modeled as a graph $G = \{V, E\}$. V is the set of nodes $\{0, \ldots, n\}$ in the network and E is set of edges between nodes. We define the edges as follows $\forall x and \forall y \in V$, $\exists (x, y) \in E \iff dist(x, y) \leq d$, where d is the transmission range of sensor nodes in the network and $dist(x, y)$ is the Euclidean distance between nodes x and y. Nodes x and y are said to be neighbors of each other. The set of neighbors of a node x is denoted as $neighbors(x)$. For a tree T constructed for convergecasting from a graph G, we call a set $X \subseteq V$, a **group** iff $X = \{y\}$

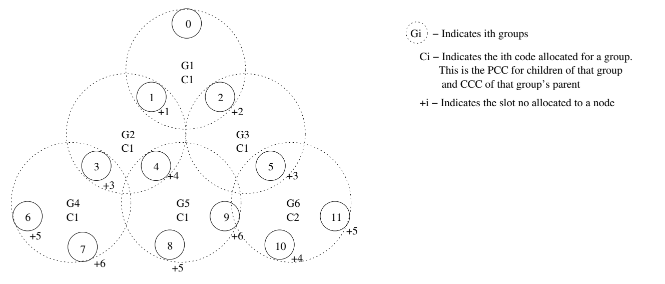

0
G1
C1
1
+1
2
+2
G2
C1
G3
C1
3
+3
4
+4
5
+3
G4
C1
6
+5
7
+6
G5
C1
8
+5
9
+6
G6
C2
10
+4
11
+5
Gi – Indicates ith groups
Ci – Indicates the ith code allocated for a group. This is the PCC for children of that group and CCC of that group's parent
+i – Indicates the slot no allocated to a node

Fig. 3. **Example illustrating the constraints.**

$\cup$ Z where y is the parent of nodes of Z in the tree T. Proximity of one group's parent p to children of another group leads to interference at p.

Our primary aim is to construct a tree for a given graph G, which will reduce the total time required for convergecasting for a given set of orthogonal spread spectrum codes (C). We call this problem as *Convergecasting Tree Construction and Channel Allocation Problem* (CTCCAP). A solution to CTCCAP is to construct a tree and to assign to each node i a code called $PCC(i)$ and a slot called $PCS(i)$ to communicate with its parent node. All nodes in a group except the parent node share the same PCC. The parent node of the group uses the PCC of its children while it receives data from its children. This it calls it CCC. The following are the constraints that an algorithm should consider in order to compute a valid solution for this problem:

- If the children of one group are in the transmission range of another group's parent then the two groups' children must have different $PCC(i)$ for communicating with their parent if available or they should use different set of $\{PCS(i)\}$, where i is the node number of each of the child nodes, for communicating with their parents. In Figure 3, node 9 is in the transmission range of group 6. Since there are two codes available group 6 makes use of the code c2 and this allows it to reuse slot 5 used by node 9. If only one code were available then group 6 cannot use slot 5. This is called the **co-channel criterion**.
- If $dist(nodes1, B) < dist(node2, B)$ then $PCS(node1) > PCS(node2)$ else $PCS(node1) < PCS(node2)$. In each of the groups in Figure 3 the node that is closest to node 0 is the parent. The child nodes of each group are allocated a slot that is greater than the slot used by their parent.
- No two children of a group share the same slot for communication with their parent. But all children of a group share the same $PCC(i)$
- Children for each node are selected based on **proximity criterion** i.e. a node is assigned as a child to the closest possible parent node.

III. Convergecasting Tree Construction and Channel Allocation Algorithm (CTCCAA)

The CTCCAA algorithm presented in this section utilizes a greedy approach for tree construction and channel allocation. The formation of the tree takes place one level at a time and simultaneously allocates channel for communication. The algorithm starts channel allocation from the root node (basestation) and proceeds in a breadth first manner. The root node runs the algorithm and after termination the root node transmits a broadcast packet with the schedules allocated for each node in the network. The algorithm takes the neighbor list of each node at the current level and starts assigning a channel for each node in the list such that it adhers to the constraints and these nodes become part of the next layer of the tree. The algorithm is provided with the list of valid codes that can be assigned, a set of nodes and the position of the nodes.

The algorithm maintains a list called "currentlist" and a list called "nextlist". Initially the currentlist consists of all nodes in level 1 (i.e. the root node) and nextlist will be empty. For each node (P) in level 1 it chooses the children and allocates a channel for the each of the child nodes to communicate with P. It uses the proximity criterion while choosing children for each node. All these child nodes are inserted into the nextlist. Once allocation for all nodes in the currentlist is done, all nodes in the nextlist are copied onto the currentlist. Now the algorithm chooses each node from the currentlist (i.e. all nodes in current level (level 2)) and chooses children for them and also allocates channel for the child nodes. For convergecasting the slot allocated to a parent node should be greater than the slot allocated to the child node. But in our case after allocation, the time slot for a parent node will be lesser compared to the slot allocated to a child node. We reverse the order in which the slots were allocated to the nodes in the network by finding the slot with the highest number and from that we subtract the slot number allocated to each node to get the actual slot number.

While choosing code and slot for child nodes, the algorithm has to keep in mind the co-channel criterion. We can find a set of interfering neighbors (IL_k) for a group which has node k as its parent based on co-channel criterion. IL_k can be defined as

$$IL_k = \{s \in V : neighbor(s) \cap neighbor(k) \neq \emptyset\}.$$

Each parent chooses a $PCC(i)$ for its children by choosing the code least used by its interfering neighbors. This is carried out in the choose-code function of the algorithm. Once the parent chooses the $PCC(i)$ for its children it has to choose a slot that can be allocated to its children. This slot has to be greater than the $PSC(i)$ of the parent node. This is carried out with the help of choose-slot function. This way we find the allocation of codes that minimizes the total number of slots required for the network. Valid codes and slots that the algorithm can allocate start from 1.

The following are the descriptions of the functions and variables used in algorithm in Figure 6.

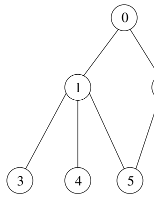


Fig. 4. **Sample network**

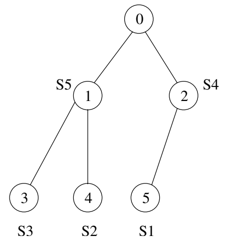


Fig. 5. **Final tree for convergecasting**

Neighborlist(N, d): Returns neighbor list information for each node in the network. It takes information about all the nodes in the network along with the fixed transmission range d.
Choose-slot(S): Returns least value of slot which will not interfere with the communication of nodes in S.
currentList: Array for storing the nodes in the current level.
InterU(a.nl,b.nl): Does the union of arrays only if a.nl contains a node which is also part of b.nl.
Choose-code(orthCode, parent): This function aggregates the neighbors of the parent node and the neighbors of all child nodes and chooses a code that is least used by this aggregated list of nodes.
currentlevel: Variable which holds the current level of the tree for which assignment is being done.
orthCode: List of orthogonal codes (given as input).
nextList: Array for storing the nodes in the next level.
Union(a, b): Performs $a = a \cup b$. This is used to aggregate all child nodes to form the parent list for the next level.
Sort(currentlist): The sort function takes care of sorting the currentlist in ascending order of number of children each node has.
d: Transmission range (given as input).
N: Set of nodes and their coordinates (given as input).

Each node maintains the following information:
pcc: Code used for communication with its parent
psc: Slot used for communication with its parent
ccc: Code used for receiving information from children
csc: Set of slots used to receive information from children
nl: A set that contains information regarding neighbors.

Using the sample network of Figure 4 we illustrate the working of CTCCAA algorithm. At level 1 only node 0 is present. The children for node 0 are chosen based on proximity criterion. Since there are no other nodes in current level except node 0, all neighbors of node 0 are chosen as node 0's children. Therefore node 1 and 2 become the children of node 0 and the channel is allocated for nodes 1 and 2. Now since no other node is present in the current level we move on to level 2 which contains nodes 1 and 2. We select a node from the current list which has the maximum number of neighbors. In this case it is node 1. Now node 1 uses proximity criterion to choose its children. Let us assume that the $dist(1, 5) > dist(2, 5)$. Since $dist(1, 5) > dist(2, 5)$ the algorithm assigns nodes 3 and 4 as the children for node 1. The channel is allocated for nodes 3 and 4 and then we start the allocation of child nodes for node 2. Since node 5 is the only neighbor of node 2 and $dist(1, 5) > dist(2, 5)$ we make node 5 as the child node for node 2. Since all nodes in the network have a channel to communicate with their parents we then do the slot reversal to obtain the network in Figure 5.

IV. Comparison of time taken for convergecasting over trees constructed by Broadcast approach of [2] and CTCCAA

For comparison purpose we assume that the channel is divided only over time and each node generates 1 $unit$ of data. Consider the sample network of Figure 4 for which we construct a tree using broadcast approach of [2] and do slot allocation for convergecasting and obtain the network in Figure 7. In the resultant network, data from nodes 3,4 and 5 are sent to node 1 and node 1 aggregates its own data to the data it received and then sends it to node 0. Since node 2 does not have any children it sends only the data, it had generated, to node 0. In a TDMA schedule the slots are of equal size. The slot duration must be long enough to accommodate the maximum amount of data aggregated at any one node in the network. In this case it is node 1 which has to transmit 4 units of data. Therefore we allocate time-slot large enough to hold 4 units of data to each node. So the total time taken for one cycle of convergecasting would be total number of slots used by the network times the slot duration i.e. 20 time units. The power in node 1 drains faster than the other nodes in the network because node 1 receives more data and has to transmit more data. Other nodes waste a small amount of energy in idle state after they transmit the data that they had generated for the remainder of the slot duration that was allocated to them.

If the tree is constructed using CTCCAA we get the tree in Figure 8. As stated earlier CTCCAA makes use of proximity criterion which aids in distribution of load experienced during data aggregation and avoids depletion of power at a select few nodes in the network. Node 1 aggregates data from its children (i.e. nodes 3 and 4) and node 2 aggregates data from its child (i.e. node 5). The total data transmitted by node 1 to node 0 is thrice the amount of data collected at each node and the total data transmitted by node 2 is twice the data collected at each node. Therefore we allocate time-slot long enough to hold 3 units of data to each node. And

```
Algorithm: CTCCAA(N,d,type)
begin
    Neighborlist(N, d)
    currentList = base-station

    while (true)
        if (currentList does not have unvisited nodes)
            currentList = nextList
            if (currentList == ∅)
                break
            nextList = ∅
            Sort(currentList)
        end if
            Select the node (nnum) from currentList and mark it visited.

        ccode = Choose-code(orthCode, nnum.nl)
        nnum.ccc = ccode
        Union(nextList, nnum.nl)

        for (each n ∈ nnum.nl) // here we find all interfering nodes
                interlist = InterU(n.nl, nnum.nl) // with nnum's children
        end for

        hslot = Choose-slot(interlist) // Choose the next available slot

        for (each n ∈ nnum.nl)
            if (n.psc != 0 and closeness(n, nnum))
                n.pcc = ccode // Code assignment for child
                n.psc = hslot // Slot assignment for child
                append hslot to nnum.csc array // nnum adds this to
                                        //its receiving slot
                hslot++
            end if
        end for
    end while

    referenceslot = Highestslot(N) + 1 // Here we do the reversal of slots
    for (each n ∈ nnum.nl) // to find the actual slot
        n.psc = referenceslot - n.psc
    end for
end
```

Fig. 6. **Convergecast tree construction and channel allocation algorithm.**

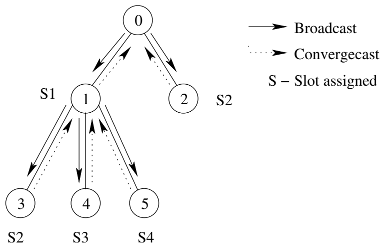


Fig. 7. **Slots assigned for convergecasting on a tree generated for broadcasting**

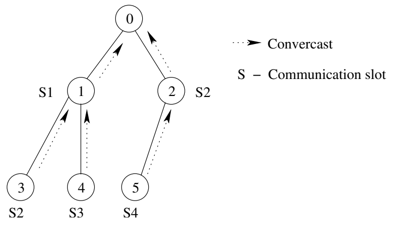


Fig. 8. **Slots assigned for convergecasting by CTCCAA**

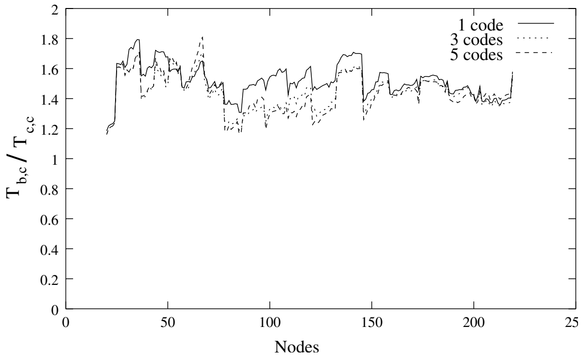
the total time taken for one cycle of data would be total number of slots used by the network times the slot duration i.e. 15 time units. This is less than the slot duration for a tree constructed by broadcast approach and the load from node 1 is now distributed onto node 2. The total power expended by the system also decreases because the power expended by nodes in idle state is less.

Simulations were carried out over randomly generated graphs with node densities varying from 0.1 $node/unit^2$ to 0.8 $nodes/unit^2$. The total duration required for convergecasting was measured for the trees constructed by both approaches. Figures 9 (a) to (c) show the results obtained from our simulations for convergecasting. It shows the ratio between total duration for convergecasting over trees generated by broadcast approach of [2] ($T_{c,b}$) and total duration for convergecasting using trees generated by CTCCAA ($T_{c,c}$) . We can see that if a tree constructed for broadcasting is used for convergecasting, then it requires 10 to 80 percent more time for networks with node density 0.3 $nodes/unit^2$, 0 to 130 percent more time for networks with node density of 0.5 $nodes/unit^2$, 0 to 150 percent more time for networks with node density 0.7 $nodes/unit^2$ than using a tree constructed by CTCCAA for convergecasting.

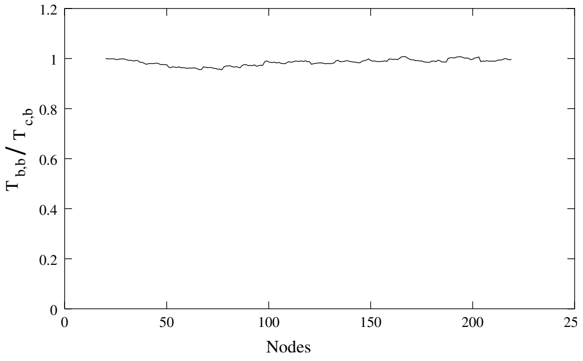
Then we generated schedules for broadcasting over a tree generated by CTCCAA. We compared the total time required for broadcasting over this tree ($T_{b,c}$) with the total time required for broadcasting over a tree constructed by the broadcast algorithm of [2] ($T_{b,b}$). Figure 9 (d) to (f) show the results obtained from some of our simulations for broadcasting. From the results we can observe that the tree constructed by CTCCAA works equally well for broadcasting too.

V. Conclusion

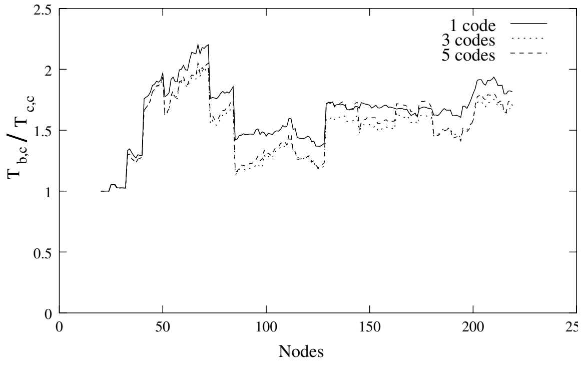
In this paper we have proposed an algorithm that constructs a tree and assigns schedule for convergecasting. We have also shown the need for a new tree convergecasting and the disadvantage of using a broadcast tree. Our algorithm constructs the tree such that it reduces the total time taken for convergecasting and this will be useful in reducing the wastage of power. We have also shown that the same tree can be used even for broadcasting and performs as efficient as a tree explicitly constructed for broadcasting.



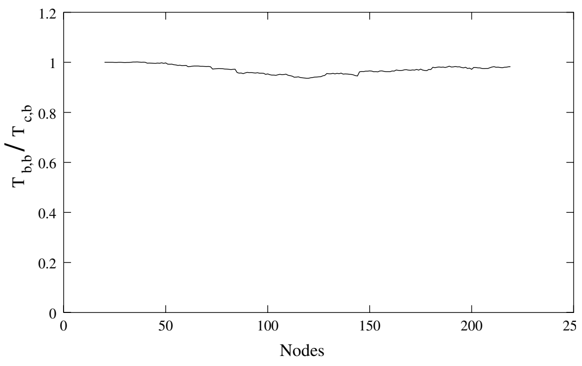
(a) Node density 0.3 $nodes/unit^2$



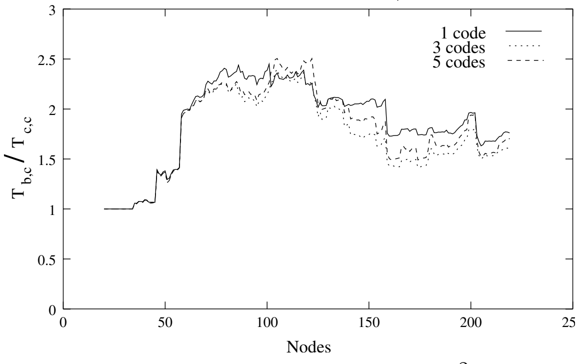
(d) Node density 0.3 $nodes/unit^2$



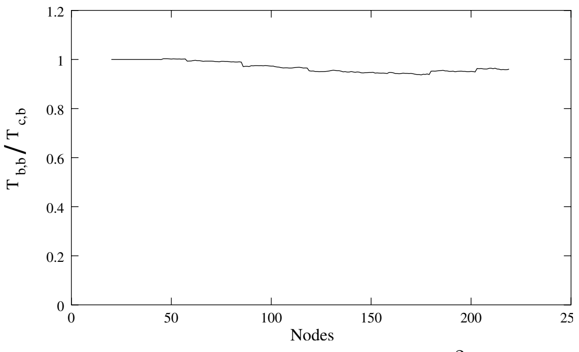
(b) Node density 0.5 $nodes/unit^2$



(e) Node density 0.5 $nodes/unit^2$



(c) Node density 0.7 $nodes/unit^2$



(f) Node density 0.7 $nodes/unit^2$

Fig. 9. **(a) to (c) shows the ratio of total duration for convergecasting over trees generated by broadcast approach of [2] and by CTCCAA algorithm respectively, for 1,3 and 5 codes.(d) to (f) shows the ratio between the total duration for broadcasting over trees generated by CTCCAA and broadcast approach of [2]**

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers comments and suggestions. We would also like to thank NSF for funding this work in part (NSF grants ANI-00196156, ANI-0086020, ANI-0123980, and DGE-9870720).

REFERENCES

[1] L. Schwiebert, S. K. S. Gupta, and J. Weinmann *et al.* Research Challenges in Wireless Networks of Biomedical Sensors. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '01)*, July 2001.

[2] Imrich Chlamtac and Shay Kutten. Tree-based Broadcasting in Multihop Radio Networks. In *IEEE Transactions on Computers, Volume C-36, No. 10, October 1987.*

[3] Imrich Chlamtac and Orly Weinstein. The Wave Expansion Approach to Broadcasting in Multihop Radio Networks. *IEEE Transactions on Communications, VOL. 39, NO. 3, March 1991.*

[4] K. Bruce Jacobson. Biosensors and Other Medical and Environmental Probes. In http://www.ornl.gov/ORNLReview/rev29_3/text/biosens.htm

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. Wireless sensor networks: a survey. In Computer Networks Journal, VOL 38 page(s) 393-422

[6] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00), August 2000*