A Fair Multiple-Slot Assignment Protocol for TDMA Scheduling in Wireless Sensor Networks

Kunal Banerjee[1], Priyanko Basuchaudhuri[2], Debesh Sadhukhan[3], and Nabanita Das[4]

***Abstract*—This paper proposes a distributed protocol for multiple slot assignment in a sensor network for collision-free media access. In the first phase, the nodes reserve multiple slots of a fixed sized frame in a fair and distributed way, so that each node gets a fair share of the available collision-free slots. In the next phase, a centralized greedy algorithm is applied to reduce the frame length that guarantees at least one slot per node making the frame length as small as possible. Simulation results show significant improvements in terms of fairness of slot distribution and reduction in frame length as compared to the slot assignment algorithm for multi-hop radio networks presented earlier. It ensures that, this protocol will be efficient in uniform-traffic sensor networks with large number of nodes resulting enhanced throughput.**

***Index Terms*—Sensor networks, TDMA Scheduling, Interference, Collision, Fair Scheduling**

I. INTRODUCTION

THE sensor nodes are usually scattered in a sensor field for collection of specific data. Each of these scattered sensor nodes has capabilities to sense data and to communicate with other nodes within its range and to route gathered data to the *sink* through a multi-hop path via intermediate nodes. Finally, the *sink* may communicate with task manager via satellite or internet [1]. The sensor nodes share a common channel over which they communicate to their neighbors within its range by broadcasting. This makes the transmissions prone to *interference* and *collisions*. Depending on the signaling mechanism, transmissions may collide in two ways:

- *Primary interference* — It occurs if a node (C) receives from more than one node, say from A and B simultaneously, as shown in fig. 1.
- *Secondary interference* — It occurs when a receiver C to receive a packet from A, is within the range of another transmitter B transmitting for another node D, as shown in fig. 1.

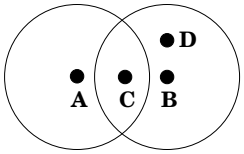

Fig. 1. Primary and secondary interference

Whatever be the case, two nodes u and v are said to be in *collision* if and only if simultaneous transmissions from u and v cause data interference at some node. In the broadcast scheduling mode, assuming that the, interference range is same as the transmission range, collision can happen among the nodes within at most two-hop distance [2].

To ensure collision-free broadcast, one popular technique is TDMA (Time-Division Multiple Access). The multi-hop nature of sensor networks makes spatial reuse possible in the sharing of channels [3], i.e., same channel can be reused at nodes which are non-interfering, i.e. at more than 2-hop distance. This is typically done by constructing a *schedule*, i.e. a sequence of fixed-length time slots where each possible transmission is assigned a time slot in such a way that transmissions never collide. It is assumed that the nodes are synchronized and time is slotted. A minimal sequence of slots is a *frame* within which each node has reserved at least one time slot to transmit its data. Frames are repeated with time and nodes transmit in their slots if they have data. Length of this frame in terms of number of slots in it, is an important parameter as it determines the minimum time to be spent before a node can get its turn in the worst case. Given the fact that long frames lead to high latency and synchronization problems, the frame length of the TDMA schedule should be kept as low as possible [4]. Therefore it also plays a key role in deciding the throughput of the network.

Since the sensor networks are distributed in nature, the nodes should create a schedule of its own on the basis of its local information i.e., at least information about its neighbors within its 2-hop distance. Therefore, the problem is to find an efficient schedule for the network so that each node gets at least one slot for collision-free broadcasting as well as keeps the frame length minimum. Earlier works indicated that this problem can be viewed as a graph-coloring problem where any node cannot have the same color as that of another node within its 2-hop distance [5]. It can be further proved that the problem is NP-complete [6], [5].

Extensive research has been done so far on TDMA scheduling for radio networks [6], [1], [3], [5], [2], [4], [7], [8]. Earlier works [3], [5] presented centralized algorithms where the information of whole network topology is known. Later, many distributed algorithms based on local information only have been proposed [6], [2], [4], [7], [8]. However, in most of these distributed protocols, a significant volume of message exchange is required due to the collisions among neighboring nodes. In sensor networks, collision is not only detrimental in terms of time delay, it also drains the nodes of their fixed energy packs, and reduces the node lifetime.

In [6], authors present a simple algorithm where for N nodes, the frame length is N; each node reserves a slot matching its unique id, that ensures collision-free message exchanges from the very beginning. After that phase, in the second round, according to the node priority each node reserves all the available slots. But this algorithm has two major shortcomings: (i) the priority (determined by node-id) results unfair distribution of reserved slots among nodes, and (ii) it is always using N number of slots, which may be quite large, in general, for sensor networks. Also, many slots are in fact, redundant and can be reduced further.

[1,2,3]Heritage Institute of Technology, Kolkata
[4]Indian Statistical Institute, Kolkata

By *fair distribution* of slots it is meant that each node reserves equal or comparable number of slots in their own schedule. This paper proposes a fair distributed scheduling based on the work presented in [6]. In the second phase, it applies a centralized greedy algorithm at the sink to remove the redundant slots to reduce the frame length. Simulation study shows the proposed algorithm significantly improves the fairness of slot allocation and the frame length as compared to the results of the algorithm presented in [6]. Being fair, the proposed algorithm will perform better when the traffic is uniform for all nodes. It will also help to enhance the throughput of the network by reducing the frame length when the number of nodes is large.

This paper is organized in the following way: section 2 presents the system model, section 3 describes the protocols, section 4 outlines the proposed algorithm, section 5 explains the simulation results and section 6 concludes the paper.

II. System Model

Given a sensor network with N nodes distributed over a region, the network can be represented by a graph $G(V, E)$, where V is the set of nodes, and two nodes $u, v \in V$ are adjacent if and only if the two nodes are within each others range, or in other words u can listen when v broadcasts and vice versa. fig. 2 shows a graph $G(V, E)$ representing a sensor network comprising of 12 nodes.

Definition 1: A node v is a 2-hop neighbor of u, if in $G(V, E)$ the shortest distance $d(u, v)$ between u and v (in terms of hops) is less than or equal to 2.

It is evident from fig. 2 that when the node 10 is broadcasting all 2-hop neighbors of 10 i.e. nodes $\{2, 4, 5, 6, 7, 8, 9, 11, 12\}$ should remain silent for collision-free transmission.

For a sensor network with N nodes, represented as a graph $G(V, E)$, the system model considered in this paper assumes that:

- The sensor network consists of N static nodes.
- Each node is assigned a unique $id i, 1 \leq i \leq N$.
- Each node knows N, the total number of nodes in the network.
- A node can only be in one state at a time: broadcasting or receiving.
- When a node u broadcasts all its adjacent nodes v, $(u, v) \in E$ can listen to it.
- All the links are bi-directional.
- All the nodes are synchronized.

With this model of the sensor network, the following section describes the proposed protocol.

III. Scheduling Protocol

Given a sensor network, represented by a graph $G(V, E)$, the TDMA scheduling attempts to reserve time slots per node such that all broadcasts be collision-free, i.e., no *2-hop neighbors* get the same slot, keeping the length of the frame minimum. As it has been mentioned in Section 1, the problem is NP-Complete [3].

The proposed protocol is based on the distributed broadcast scheduling algorithm for radio networks presented in [6]. Here, initially the algorithm assumes N slots per frame where N is the total number of nodes in the network. Before presenting the algorithms formally, the steps of the algorithm are explained with the help of the network shown in fig. 2.

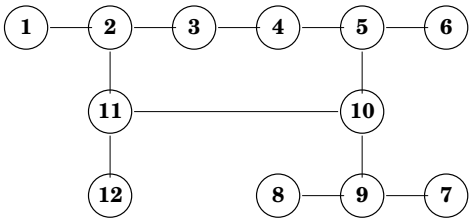

Fig. 2. A sensor network with 12 nodes

A. Initial Reservation

Here, it is assumed that each node-i decides to transmit in slot-i of the frame with N slots in all. Node-i broadcasts a *HELLO* message in slot-i. For the remaining slots it listens and receives *HELLO* packets from its adjacent neighbors, and includes the node-ids in its neighbor list. Therefore, after the first frame, each node will have the ids of its immediate neighbors, i.e., neighbors within 1-hop. Similarly, during next frame node-i broadcasts its *neighbor list* information in slot-i. For the remaining slots it listens and collects neighborhood information from its adjacent neighbors. After this phase each node will have the complete *2-hop neighbors* information. Next, each node constructs its own schedule based on the knowledge of its *2-hop neighbors*. It is to be noted that all message exchanges are collision-free.

Definition 2: For a sensor network with N nodes, an N x N matrix, defined as the schedule matrix (M) is constructed, where element $m_{i,j}$ indicates the status of slot-i for node-j. This status may take one of three values: reserved(R), blocked ($-$), and available or undecided (blank). Initially, each $m_{i,i} = R$, i.e., slot-i is reserved for node-i.

Definition 3: Each node-j knows the id-s of all of its *2-hop neighbors*. Hence each node-j can construct the column $M_j = \{m_{1,j}, m_{2,j}, \ldots, m_{N,j}\}^T$ of the *schedule matrix* (M). M_j is termed as the $schedule - column$ for node-j.

Each node-i broadcasts its schedule column M_i to its *2-hop neighbors*. After this phase each node-i gathers schedule columns from all of its *2-hop neighbors*. It again will require two more frames.

After this step, the initial schedule matrix M for the sensor network in fig. 2 is shown in table I (slots reserved are marked by R, blocked by $-$, undecided by blank). It should be noted that the *initial-schedule-matrix* (M) is not available at any node, but each node knows only the *initial-schedule-columns* of its own and its *2-hop neighbors*.

B. Fair Reservation of Undecided Slots

In the next phase, each node-i executes the reservation procedure of undecided slots when it finds that all its *2-hop*

neighbors-j, $j < i$, have completed their undecided-slot reservation (the node-id is used as the priority).

Definition 4: In order to decide which of the available slots node-i should reserve, it has to calculate the *contention* (C_i) for each available slot S_i, i.e., total number of its *2-hop neighbors* to which the slot S_i is available.

TABLE I
THE *initial-schedule-matrix* M

S↓	1	2	3	4	5	6	7	8	9	10	11	12
1	R	-	-								-	
2	-	R	-	-						-	-	-
3	-	-	R	-	-						-	
4		-	-	R	-	-				-		
5			-	-	R	-			-	-	-	
6				-	-	R				-		
7							R	-	-	-		
8							-	R	-	-		
9					-		-	-	R	-	-	
10		-		-	-	-	-	-	-	R	-	-
11	-	-	-		-				-	-	R	-
12		-								-	-	R

The matrix C in table II shows the contentions for all the slots in the network of fig. 2 (X denotes that the slot is either already reserved or blocked). Again, each node-i just computes and stores the column-i of C.

TABLE II
THE CONTENTION MATRIX C

S↓	1	2	3	4	5	6	7	8	9	10	11	12
1	X	X	X	4	5	4	4	4	5	8	X	2
2	X	X	X	X	3	2	3	3	4	X	X	X
3	X	X	X	X	X	2	4	4	4	6	X	2
4	2	X	X	X	X	X	3	3	4	X	4	2
5	2	3	X	X	X	X	2	2	X	X	X	2
6	4	5	4	X	X	X	3	3	4	X	6	3
7	4	6	6	5	5	3	X	X	X	X	6	3
8	4	6	6	5	5	3	X	X	X	X	6	3
9	3	5	4	4	X	2	X	X	X	X	X	2
10	2	X	2	X	X	X	X	X	X	X	X	X
11	X	X	X	2	X	2	2	2	X	X	X	X
12	2	X	4	4	5	3	3	3	4	X	X	X

Next, node-i finds out N_{low} the number of its *2-hop neighbors* which are yet to start the fair reservation phase, and decides to reserve only T slots, $T = \lceil n_i/N_{low} \rceil$, where n_i is the total number of available slots for node-i. Node-i selects T slots from its available list with least *contention*.

As for example, for the network in fig. 2, node 1 has eight available slots, as shown in table I, and three lower priority nodes (2, 3 and 11) in its broadcasting zone. Therefore, for node 1, $T = 3$. So, it reserves slots 4, 5 and 10, which have the lowest contentions. Node 1 broadcasts its new schedule within its *2-hop neighbors*, which in turn updates their own schedule columns.

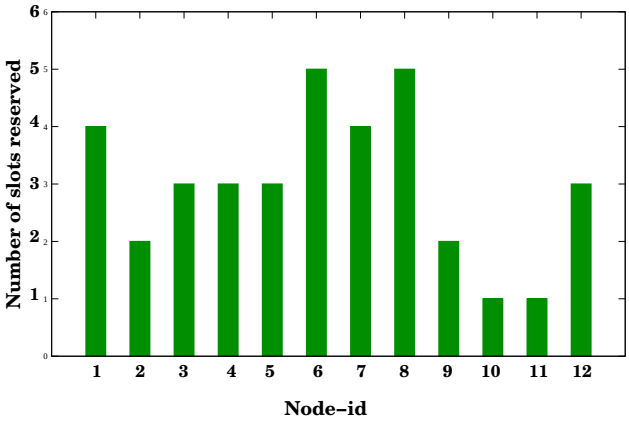

Fig. 3. Distribution of reserved slots among the nodes after fair-reservation

This procedure is repeated for all the nodes giving rise to the *complete-schedule-matrix* shown in table III. Fig. 3 shows the distribution of reserved slots among the nodes after reservation of undecided slots.

TABLE III
THE *complete-schedule-matrix* AFTER FAIR-RESERVATION

S↓	1	2	3	4	5	6	7	8	9	10	11	12
1	R	-	-	R	-	-	-	R	-	-	-	-
2	-	R	-	-	R	-	R	-	-	-	-	-
3	-	-	R	-	-	R	-	-	R	-	-	-
4	R	-	-	R	-	-	-	R	-	-	-	-
5	R	-	-	-	R	-	R	-	-	-	-	R
6	-	R	-	-	-	R	-	R	-	-	-	-
7	-	-	-	-	R	-	R	-	-	-	-	R
8	-	-	-	-	-	R	-	R	-	-	-	-
9	-	-	R	-	-	R	-	-	R	-	-	-
10	R	-	-	-	-	-	-	-	-	R	-	-
11	-	-	-	R	-	-	R	-	-	-	R	-
12	-	-	R	-	-	R	-	R	-	-	-	R

C. Compaction of Frame

From the above example it is evident that not all N slots are always essential to ensure that each node gets at least one slot for transmission in a time frame. Hence there is a scope of reducing the frame length. However, so far each node-i reserves slots in a distributed fashion based on the knowledge of its *2-hop neighbors*, storing and manipulating the column-i of the schedule matrix only. But for compaction of the frame, the whole schedule matrix must be known for computing.

A greedy centralized algorithm is proposed here for compaction of the frame, ensuring at least one slot per node. It is assumed that after reserving the undecided slots each node broadcasts its *schedule-column* to the *sink* node. The *sink* node, according to the node-priority checks for each slot in the *schedule matrix*, and removes the redundant slots, if possible, and

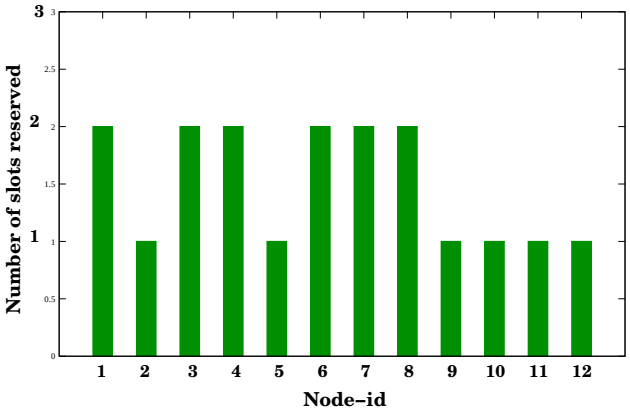

Fig. 4. Distribution of reserved slots among the nodes after compaction

finally broadcasts the *compact schedule matrix* to all nodes.

The algorithm *Algorithm Compact_Schedule* is described formally in the next section.

For example, the execution of *Algorithm Compact_Schedule* on the schedule matrix of table III for the network shown in fig. 2 results the compact schedule shown in table IV. It reveals the fact that only 6 slots, instead of 12, per frame is sufficient for ensuring collision-free slots for each node, i.e., the proposed algorithm results about 50% savings in terms of frame length.

TABLE IV
THE *compact-schedule-matrix*

S↓	1	2	3	4	5	6	7	8	9	10	11	12
1	R	-	-	R	-	-	-	R	-	-	-	-
2	-	R	-	-	R	-	R	-	-	-	-	-
3	-	-	R	-	-	R	-	-	R	-	-	-
4	R	-	-	-	-	-	-	-	-	R	-	-
5	-	-	-	R	-	-	R	-	-	-	R	-
6	-	-	R	-	-	R	-	R	-	-	-	R

Fig. 4 shows the distribution of reserved slots among the nodes after compaction.

It is evident from fig.s 3 and 4 that compaction helps to make the distribution of reserved slots among the nodes more uniform.

For the network given in fig. 2, the algorithm in [6], to be referred here as *Algo[6]*, results the schedule matrix shown in table V.

Fig. 5 shows the distribution of reserved slots among the nodes obtained by *Algo[6]*. Comparison with fig. 4 clearly shows that the distribution of reserved slots among the nodes is much more even for the algorithm proposed in this paper than by *Algo[6]*.

IV. MULTI-SLOT SCHEDULING ALGORITHM (MSA)

This section presents the formal description of the algorithm MSA along with the steps of the protocol.

TABLE V
THE SCHEDULE MATRIX FROM *Algo[6]*

S↓	1	2	3	4	5	6	7	8	9	10	11	12
1	R	-	-	R	-	-	R	-	-	-	-	R
2	-	R	-	-	R	-	R	-	-	-	-	-
3	-	-	R	-	-	R	R	-	-	-	-	R
4	R	-	-	R	-	-	R	-	-	-	-	R
5	R	-	-	-	R	-	R	-	-	-	-	R
6	R	-	-	-	-	R	R	-	-	-	-	R
7	R	-	-	R	-	-	R	-	-	-	-	R
8	R	-	-	R	-	-	-	R	-	-	-	R
9	R	-	-	R	-	-	-	-	R	-	-	R
10	R	-	-	-	-	-	-	-	-	R	-	-
11	-	-	-	R	-	-	R	-	-	-	R	-
12	R	-	-	R	-	-	R	-	-	-	-	R

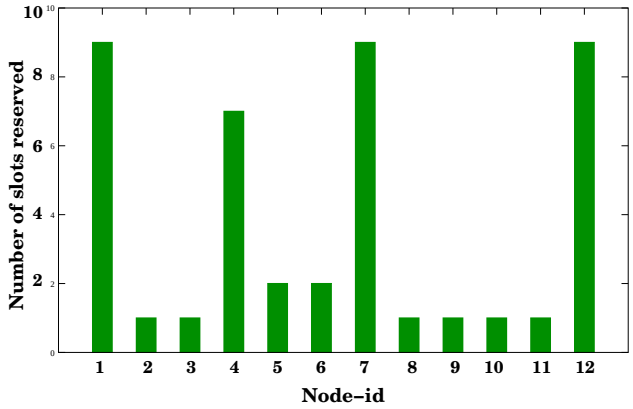

Fig. 5. Distribution of reserved slots among the nodes according to [6]

A. Initial condition

Initially each node only has a unique id assigned to it at the time of deployment. The nodes have no information about the network topology but it knows N, the total number of nodes in the network.

B. Two hop neighborhood information

As discussed in section 3.1, it is assumed that by some lower level protocol (exchange of *HELLO* packets) each node-i gathers its *2-hop neighbor-ids*. Using these data they create their own *initial-schedule-column*, which in fact is column-i of the *initial-schedule-matrix* shown in table I. Each node broadcasts its *initial-schedule-column* among its *2-hop neighbors*. In turn a node receives the *initial- schedule-columns* from all of its *2-hop neighbors*.

C. Reserving the undecided slots

In this phase, each node-i receives the set of *complete-schedule-columns* from all its *2-hop neighbors-j* with higher priority, i.e. $j < i$. Based on these, it computes its own *complete-schedule-column* and broadcasts the *complete-schedule message* containing its *complete-schedule-column* to all its *2-hop neighbors*.

It is to be noted that all the phases A, B and C are distributed in nature.

Algorithm_Fair_Schedule

For each node-i
Input: *Initial-schedule-columns* of node-i and all its *2-hop neighbors*.
Output: *Complete-schedule-column* of node-i.
If node-i receives *complete-schedule* message from its *2-hop neighbor-j* for $j < i$, updates its *initial-schedule-column*.
After node-i receives *complete-schedule* message from all its *2-hop neighbor-j* for $j < i$, it

- calculates *contention* (C_i) for each undecided slot S_i \\ number of its *2-hop neighbors* to which the slot S_i is available;
- calculates $T = \lceil n_i/N_{low} \rceil$, \\ n_i : total number of available slots, N_{low} : total number of *2-hop neighbors* with priorities lower than i

Reserves T least contentious available slots, update *initial-schedule-column*
complete-schedule-column $\leftarrow$ *initial-schedule-column* and broadcasts *complete-schedule-message*

D. Compaction of schedule matrix

This phase is executed in a centralized way at the *sink* node. In this phase the input is the *complete-schedule-matrix* (M_F) of the network. The output is the *compact-schedule-matrix*, where redundant slots are removed.
After the completion of *Algorithm_Fair_Schedule*, each node broadcasts its schedule to the *sink* node. On gathering the whole schedule matrix, the *sink* node computes the compact schedule by the following algorithm:

Algorithm_Compact_Schedule

For $k = 1$ to N
For $j = k + 1$ to N
If $M_F(k, j) = R$ delete row j of M_F
And compact M_F \\by moving all the rows-p $p > k$ to $(p - 1)$
Broadcast M_F to all nodes

V. Simulation Results

For simulation study, the protocol has been implemented and tested on random graphs with number of nodes (N) varying from 50 to 250. For each N, 20 graphs were generated and tested.

For comparison with *Algo[6]*, the parameters frame length (L), standard deviation of the number of slots reserved by individual nodes ($s.d.$), and the average transmission rate (Tr) have been compared. The results are shown in fig.s 6, 7, and 8 respectively.

The parameter L gives a measure of the worst case waiting time for a node before a transmission.

Fig. 6 shows that for *Algo[6]* the frame length is always equal to the number of nodes N but for the proposed algorithm *MSA* it is always significantly less than N. For example with $N = 200$,

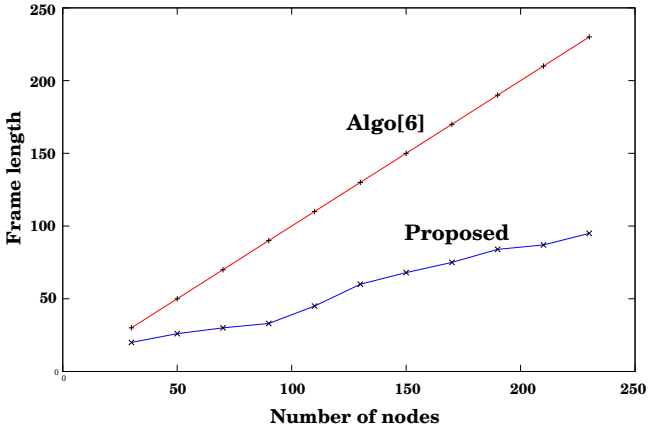

Fig. 6. Comparison based on frame length (L)

MSA results $L = 70$ only, thereby reducing L by 65%. More interestingly this saving goes on increasing with N, showing the efficiency of *MSA* in case of networks with large number of nodes.

Regarding fairness in slot assignment, in *Algo[6]*, a node may reserve all the slots available to it while its *2-hop neighbors* of lower priority may not get that opportunity. But the algorithm discussed here incorporates an even distribution of reserved slots. For comparison of fairness of scheduling, the parameter $s.d.$ is considered, $s.d. = \frac{\sqrt{\sum (x_i - x_{mean})^2}}{N}$, where x_i is the number of slots reserved by node i and x_{mean} is the average number of slots reserved by the nodes.

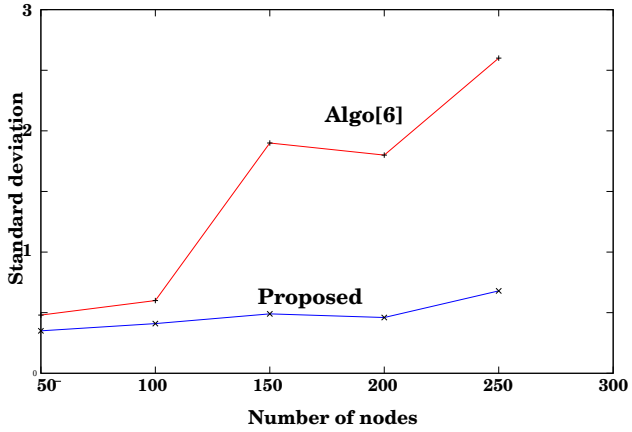

Fig. 7. Comparison based on standard deviation (s.d.) of number of slots assigned to individual nodes

From the above results it is apparent that the standard deviation of the slots per node is much greater for *Algo[6]*. So it is evident that algorithm *MSA* achieves a fair distribution of reserved slots among the nodes. For example, with $N = 200$, *Algo[6]* results $s.d. = 1.8$ compared to 0.5 resulted by *MSA*. This shows that in case of networks with uniform traffic, *MSA* will outperform *Algo[6]* in terms of fairness.

To have an idea about the throughput of the network, assuming uniform traffic, the parameter transmission rate (Tr) is in-

troduced, where Tr = average number of slots reserved per node / frame length. Fig. 8 shows that *MSA* performs better than *Algo[6]* in respect of Tr as well. It should be noted that for small number of nodes the proposed algorithm is not much effective as it is for a network with higher number of nodes.

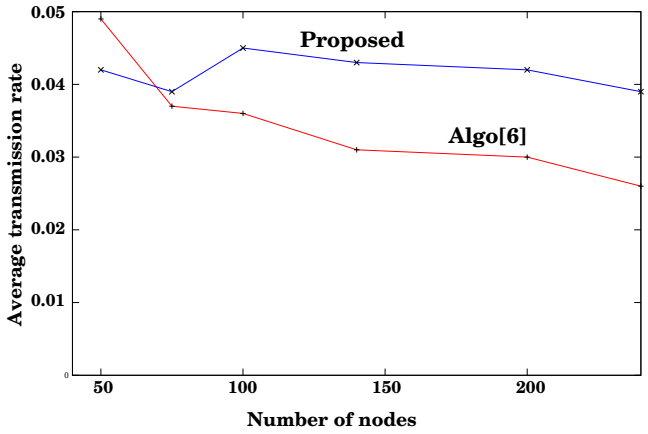

Fig. 8. Comparison based on average transmission rates

VI. Conclusion

This paper considers the problem of assigning interference-free broadcasting schedules for multi-hop wireless sensor networks. A distributed algorithm based on the algorithm given in [6] is followed here for initial reservation of the slots on a frame of length N, where N is the total number of nodes. In the next phase the nodes reserve additional slots in a distributed way that ensures fair scheduling. Finally, the frame length has been reduced by a centralized greedy algorithm. Comparison with algorithm in [6] by simulation shows the proposed algorithm outperforms in terms of frame length, fairness and throughput. It reveals the fact that the proposed algorithm will be efficient for large networks with uniform traffic per node, which is the case for sensor networks, in general.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] Injong Rhee, Ajit Warrier, Jeongki Min, and Lisong Xu, "Drand: distributed randomized tdma scheduling for wireless ad-hoc networks," in *MobiHoc*, 2006, pp. 190–201.

[3] Subramanian Ramanathan and Errol L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 166–177, April 1993.

[4] Y. Wang and I. Henning, "A deterministic distributed tdma scheduling algorithm for wireless sensor networks," in *Proceedings of International Conference on Wireless Communication, Networking and Mobile Computing (WiCOM)*, 2007, pp. 2759–2762.

[5] S. Ramanathan, "A unified framework and algorithm for channel assignment in wireless networks," *Wireless Networks*, vol. 5, pp. 81–94, March 1999.

[6] A. Ephremides and T.V. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Transactions on Communications*, vol. 38, no. 4, pp. 483–495, April 1990.

[7] Shashidhar Gandham, Milind Dawande, and Ravi Prakash, "Link scheduling in sensor networks: distributed edge coloring revisited," in *INFOCO'05*, 2005, pp. 2492–2501.

[8] Subhasis Bhattacharjee and Nabanita Das, "Distributed time slot assignment in wireless ad hoc networks for stdma," in *ICDCIT*, 2005, pp. 93–104.