An Energy-Efficient, Scalable and Collision-Free MAC layer Protocol for Wireless Sensor Networks

Gaurav Jolly and **Mohamed Younis**
Dept. of Computer Science and Elec. Eng.
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
jolly1@umbc.edu and younis@cs.umbc.edu

Abstract: Wide range of applications such as disaster management, military and security have fueled the interest in sensor networks during the past few years. Sensors are typically capable of wireless communication and are significantly constrained in the amount of available resources such as energy, storage and computation. Such constraints make the design and operation of sensor networks considerably different from contemporary wireless networks, and necessitate the development of resource conscious protocols and management techniques. In this paper we present an energy efficient, scalable and collision free MAC layer protocol for sensor networks. The approach promotes time-based arbitration of medium access to limit signal interference among the transmission of sensors. Transmission and reception time slots are prescheduled to allow sensors to turn their radio circuitry off when not engaged. In addition, energy consumption due to active to sleep mode transitions is minimized through the assignment of contiguous transmission/reception slots to each sensor. Scalability of the approach is supported through grouping of sensors into clusters. We describe an optimization algorithm for energy conscious scheduling of time slots that prevents intra-cluster collisions and eliminates packet drop due to buffer size limitations. In addition, we also propose an arbitration scheme that prevents collisions among the transmission of sensors in different clusters. The impact of our approach on the network performance is qualified through simulation.

Keywords: Wireless sensor network, energy aware communication, MAC layer protocols, TDMA slots scheduling

1 Introduction

Recent advances in miniaturization and low-power design have led to active research in large-scale, highly distributed systems of small-size, wireless unattended sensors 122456. A sensor network consists of minute devices that are capable of probing the environment and reporting the collected data, typically using a radio, to the command center. Sensor networks can serve many civil and military applications such as disaster management, combat field surveillance and security. In such applications, the sensors are usually powered using small batteries and replacing sensor's battery is not possible or not practical. Such energy constraints limit sensors' lifetime and thus make efficient design and management of sensor networks a real challenge. Therefore, a lot of the research related to sensor networks has focused on energy-awareness and minimization 178. In this paper we concentrate on the minimization of energy consumption at the MAC layer through time-based arbitration of the sensor's medium access.

Medium access is a major consumer of sensor energy, especially for long-range transmission and when the radio receiver is kept on all the time. Energy consumed for radio transmission is directly proportional to distance squared and can significantly magnify in a noisy environment. Energy-aware routing typically pursues multi-hop paths in order to optimize the transmission energy 910. On the other hand, time-based medium access control (MAC) saves transmission energy by limiting the potential for collisions and minimizes the energy consumed in the receiver by turning the radio off when it is idle 1112. Generally, an efficient MAC layer protocol for sensor networks should have the following attributes:

- The protocol should be scalable since most applications of sensor networks involve a large set of sensor nodes.
- Collisions among the transmissions of various nodes should be avoided. Collisions lead to packet drop and thus reduce throughput and cause energy wastage.
- Energy consumed by the radio circuit in idle mode is almost equal to that consumed in active state. Consequently, idle mode of operation and transmission overhearing among sensors should be minimized.
- To limit energy consumption during idle time, the sensors are typically switched to a sleep mode when not in use. However, active to sleep transitions and vice-versa consume considerable amount of energy. Therefore, an efficient protocol should minimize such transitions 8.
- The protocol should not be contention-based. Control packets overhead and active sensing of the medium, typically performed by contention-based protocols, are inefficient in terms of energy consumption.
- Packet drop due to limited buffer capacity should be prevented.
- The protocol should adapt to changes in the network topology and all sensors should have a fair chance of transmitting.

Unlike contention-based protocols, a Time-Division-Multiple-Access (TDMA) based MAC allows communication traffic to flow according to a preset schedule. Time-based MAC can minimize the energy consumption since the nodes can turn off their transmitters or receivers, unless they are expecting to receive or transmit a packet. It has been shown that turning off the radio receiver significantly reduces energy consumption and extends the life of wireless sensor networks 1213. In addition, collision among nodes can be avoided since each node has its own assigned time slots. However these advantages of time-based MAC are due to the deterministic operation, which requires communication time slots to be scheduled for both transmitting and receiving.

Scheduling time slots can be NP-hard especially when considering the fact that there can be large number of possible ways of scheduling the order of transmission of sensors for a particular flow of packets. Message flow constraints and sensor's capabilities limitations such as buffer size further complicates the problem. In addition, implementation of a TDMA scheme requires that the nodes be synchronized with each other. Since majority of nodes are in sleep mode, many nodes will have to be switched on in order to receive a synchronization message. The energy efficiency of TDMA schemes will diminish if nodes require to be synchronized very frequently. Moreover, a static TDMA scheme cannot be used in sensor networks since in most sensor applications topology changes are very frequent.

In this paper, we present an energy efficient and scalable TDMA-based MAC layer protocol for sensor networks. The proposed scheme handles dynamic changes in the network topology and limits the control packet overhead. In the backend, a slot' scheduling heuristic is proposed for assigning optimized time slots to communicating sensors. The search heuristic minimizes the sensor's transition between idle and active modes while meeting message flow and buffering constraints. The proposed protocol limits the need for frequent clock synchronization messages by including a reference time in topology management related control traffic. Protocols developed for contemporary wireless devices do not address most of the issues stated above and hence cannot be applied to sensor networks. In addition, as discussed in section 1.2, MAC protocols for sensor networks presented in the literature address only a subset of these issues**.** We are not aware of any other approach that comprehensively tackles all the listed efficiency attributes.

In the balance of this section we describe our system model and discuss related work. Section 2 describes our proposed energy-aware MAC layer protocol for sensor networks. Detailed algorithms for scheduling time slots are described in section 3. Description of the simulation environment and analysis of the experimental results can be found in section 4. Finally section 5 concludes the paper.

1.1 System Model

The system architecture for the sensor network is depicted in Fig. 1. In the architecture sensor nodes are grouped into clusters controlled by a single command node. Every cluster has a gateway node that manages sensors in the cluster. Clustering the sensor network can be either performed by the command node or collaboratively among the gateways and is beyond the scope of this paper 1415**.** Sensors are only capable of radio-based short-haul communication and are responsible for probing the environment to detect a target/event. In this paper, we assume that sensor and gateway nodes are stationary and all sensors in a cluster are within the communication range of the gateway of that cluster. The on-board clocks of the gateway nodes are assumed to be synchronized, e.g. via the use of GPS.

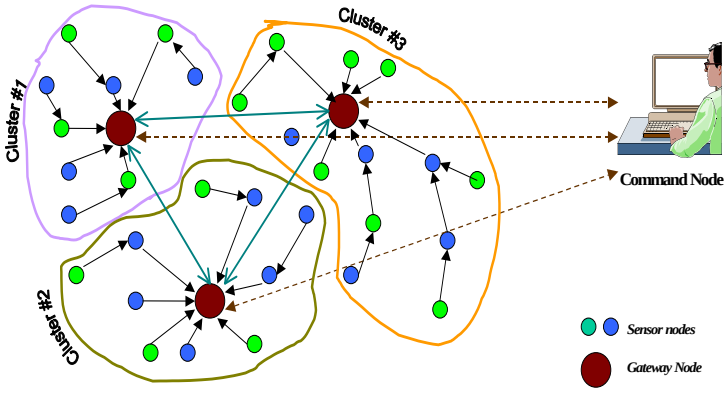


Fig. 1: Architecture of unattended sensor network

The gateway node interfaces the command node with the sensor network via long-haul communication links. Sensors receive commands from and send readings to their gateway node, which processes these readings and transmits the fused information to the command node. The command node performs system-level fusion of collected reports for overall situation awareness. Unlike sensors the gateways are significantly less energy constrained. Hence the gateway is assigned the responsibility of organizing the sensors and routing generated data. Sensor organization refers to activating a subset of available sensors in the cluster to probe the environment based on the application and the sensor's capabilities. The gateway sets multi-hop routes based upon the current state of the network and sends route updates to the sensors. Route formation will designate some sensors to act as relays**.** The sensors then adjust their transmit power based upon their next hop neighbor.

Radios are assumed to have the ability to operate in four distinct modes transmit, receive, idle and sleep. The energy consumed in idle mode is almost equivalent to that in receive mode 16. The energy consumed by the radio is:

$$E_{radio} = N_{tx} [P_{tx} (T_{on\text{-}tx}+T_{st}) + P_{out}T_{on\text{-}tx}] + N_{rx} [P_{rx} (T_{on\text{-}rx}+T_{st})] \quad \ldots\ldots (1)$$

Where $N_{tx/rx}$ is the average number of times per second, the transmitter/receiver is used. T_{st} is the transition time from sleep to active mode. $T_{on\text{-}tx/rx}$ is the on time of the transmitter/receiver. P_{out} is the output transmission power. $P_{tx/rx}$ is the power consumed by the transmitter/receiver 817. It is worth noting that most of these capabilities are available on some of the advanced sensors, e.g. the SenTech Acoustic Ballistic Module 18.

1.2 Related Work

Contemporary MAC layer protocols designed for wireless devices such as MACAW 19 and IEEE 802.11 20 are not suitable for sensor networks. These schemes will consume considerable amount of energy since they require the sensors to continuously probe the medium. In addition, these schemes require nodes to transmit control packets in order to avoid collisions. The control packet sizes will be comparable to the size of data packets, which are small in most sensor applications. On the other hand, Bluetooth 21 uses a TDMA based scheme but assumes that all slave nodes are within transmission range of the master node. This is in contrast to energy-efficient multi-hop mode of transmission employed in sensor networks.

Power management of the radio has gained significant importance in sensor networks since the radio is a major consumer of sensor's energy. It has been shown that the energy consumed in transmitting one bit is several thousand times more than the energy consumed in executing one instruction 22. Several methods have been suggested to reduce the energy consumption of the RF circuitry. One such technique is to power off the sensor when it is idle, by transitioning from active to sleep mode 22. However time taken to make a transition from sleep to active mode consumes a considerable amount of energy. With small packet sizes the energy consumed due to transitions becomes even more prominent and dominates the active mode's energy consumption 23. A circuit-level approach to reduce such startup time in the radio circuitry was suggested in 24.

A number of MAC layer protocols have been proposed for wireless sensor networks in recent years. A contention based MAC protocol that provides node level fairness while minimizing energy is proposed in 7. However, this protocol does not address the issue of turning off the radio when the sensor is not operational and it does not eliminate collisions. Another contention-based protocol has been proposed in 25. In this protocol energy conservation is achieved by utilizing the sleep mode operation of the radio. However, this approach sacrifices per hop fairness and latency. Moreover, if there are multiple sensors that want to transmit to the same node they contend for the medium using a RTS/CTS mechanism. This will consume considerable energy compared to TDMA protocols, where nodes are assigned slots independent of each other and do not have to contend for the medium.

PAMAS 12 is a CSMA based protocol in which the nodes that are not actively transmitting or receiving should power themselves off. The presented approach results in energy savings of up to 70%. However the protocol requires the nodes to sense the medium to transmit and does not eliminate collisions completely. In addition the protocol requires the nodes to have two separate channels (control and data), which will require two radios at each node increasing the cost, size and complexity of the sensor design. Other than PAMAS, the bulk of CSMA based protocols found in the literature have not exploited the potential of energy conservation through selective activation of the radio circuitry.

Energy saving through the use of time-based MAC in wireless devices has been explored in 13. The idea is to schedule when to activate the radio receiver so that it can be turned off while not expecting a message. Nodes that have data to transmit make a reservation request to a base station, which responds with a traffic control message indicating medium access schedule. Nodes that are not included in the traffic control message can turn off their radio receivers. The nodes that have been assigned slots transmit in the order scheduled by the base station. A non-reservation based approach that considers the routing paths has been pursued in 26. However, no attention has been paid to the effect of the ordering of transmission slots on the performance and lifetime of the sensor network. Our approach makes a comprehensive consideration of the different energy conservation opportunities at the MAC layer.

To capture the advantages of a TDMA based scheme nodes should be synchronized. Typically, in sensor networks only the sensors that are performing some function are active at any point of time. Thus, in order to receive a synchronization message many sensors will have to switch on their radios. This will consume considerable amount of energy. Moreover, if the frequency of synchronization messages is large, the energy resources of the sensors will further diminish. In 27, it has been argued that conventional synchronization schemes like NTP will consume a lot of energy in passive listening and hence are not suitable for sensor networks. It has been further concluded that no single scheme is suitable for sensor networks and the

networks should adapt the synchronization scheme based upon the application. Though they have described the characteristics of what would be a good synchronization scheme for sensor networks, they have not explicitly proposed one.

2 Energy Efficient MAC Protocol

Based on the current application mission, the gateway selects a set of sensors to probe the environment. The selected sensors gather data and forward their readings to the gateway for data fusion. The gateway sets the routes for data generated by every probing sensor. In order to save energy multi-hop routing is pursued. The gateway designates some sensors to act as relays. Relay sensors store and forward messages from a source node that is a sensor probing the environment, to the next hop or finally the gateway. Some sensors can be both probing the environment and also relaying data from other sensors. Unselected sensors can be set to low-energy sleep state. There are many energy aware approaches that the gateway can use for route setup, e.g. 9. Contingent upon these routes and the buffer size of the sensors, the gateway then calculates the order in which transmission time slots are assigned to active sensors, both probing and relaying. In order to conserve energy, active sensors should shut down their radio when they are not transmitting or receiving.

Every gateway takes charge of assigning transmission and reception slots to sensors in its cluster subject to flow constraints. We pursue a slot' scheduling mechanism that is inspired by the Tabu-search optimization methodology. The main goal of such scheduling mechanism is to minimize the sensor transition between active and sleep mode while meeting flow constraints and avoiding buffer overflow at all relay nodes. To avoid the potential of inter-cluster interference, the gateways collaborate on the elimination of simultaneous transmission of sensors in neighboring clusters. The gateways designate one of them to check that transmission slots within each cluster will not interfere with the neighboring clusters. Upon forming collision free schedule of time slots, each gateway broadcasts the slot assignment to all sensors in its cluster. The protocol uses predetermined set of operational phases in order to limit the control message traffic. Network clustering ensures scalability for larger number of sensor nodes.

As stated, medium access arbitration follows a sequence of phases that are executed periodically during the lifetime of the network. In this section we describe the different phases of the protocol and the clock synchronization scheme employed by the MAC layer protocol. We will describe the detailed slot scheduling heuristic in the next section.

2.1 Protocol Phases

The proposed MAC protocol for sensor networks consists of distinct phases that are periodically executed in the sequence shown in Fig. 2. This subsection summarizes the different phases of the protocol. In the remainder of this paper, we will focus only on medium access arbitration and the clock synchronization parts of the protocol. More elaborate discussion of the other phases can be found in 26.

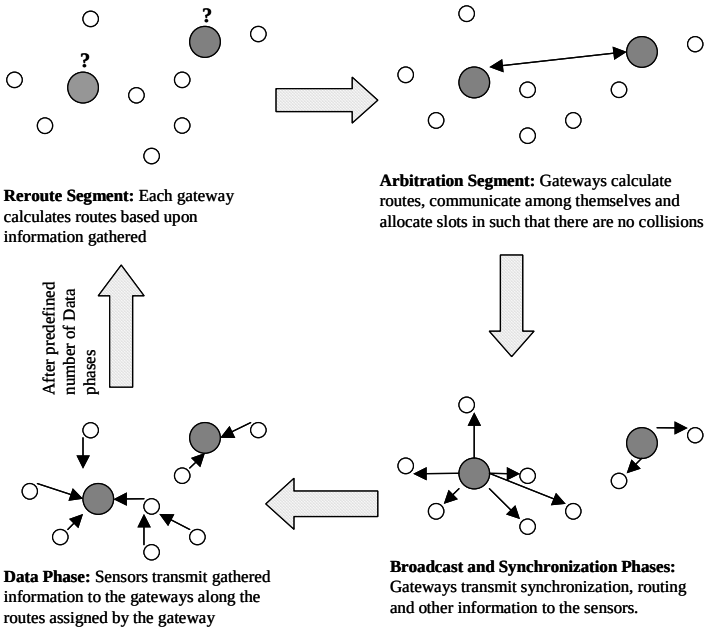


Fig. 2: Description of protocol control and data transmission phases

- *Data Phase:* In this phase the nodes transmit the collected sensor readings to the gateway through their next hop neighbors.
- *Reroute and Arbitration Phase*: This phase consists of two segments the reroute segment and the arbitration segment. In the reroute segment each gateway calculates new routes for the sensors in its cluster based on mission objective, energy depletion, etc. In the arbitration segment the gateways assign time slots to active sensors in their clusters and arbitrate among themselves to ensure collision free transmission over the new routes.
- *Broadcast Phase:* This phase is periodically executed by the gateways to inform the sensors of the new routes, assigned slots and other instructions for the sensor that are applicable until next rerouting cycle.
- *Synchronization Phase:* In this phase the gateways synchronize the clock of sensor nodes by broadcasting synchronization messages.

2.2 Synchronization Mechanism

Time-based access to the communication medium requires the clocks of communicating nodes to be synchronized. Clock drifts can cause some nodes to unexpectedly compete for the medium increasing the potential for collisions among the transmission of the nodes in the network. Maintaining synchrony among the nodes requires periodic enforcement in order to readjust the nodes clocks to a reference value. The frequency of such adjustment depends on many factors such as the resolution of the time schedule and the clock drift rate. While high-resolution TDMA slots enable good utilization of the medium enhancing network throughput and message response time, the implementation of such a very precise schedule requires either using expensive clock crystals in the design of sensor nodes or performing very frequent resynchronization. Since sensors used in many applications are typically unattended and disposable, the incorporation of high quality and expensive clock oscillators is not desirable. On the other hand, frequent resynchronization can become a performance burden as we explain below. Therefore the handling of clock synchronization of sensor networks has to be subject to a system-level trade-off.

In our model, we assume that the gateways are equipped with GPS receivers and will thus maintain synchronized clocks. The gateways will utilize their high precision clocks to synchronize the sensors in their cluster. Such sensor synchronization takes the form of broadcast messages containing the gateway's reference clock reading. Upon receiving a synchronization message each sensor in the cluster will reset its own value to the gateway clock. For highly precise time-based network operation, the gateway has to account for message propagation delay in the clock adjustment. While in typical distributed systems synchronization messages are sent periodically, they can be energy inefficient for sensors networks. Since many of the sensors in a cluster are in sleep mode at any particular instant of time, these sensors will have to make a transition to active mode to get the clock adjustment. Active mode transition requires the sensor to be on for duration equals to twice the startup transient time plus the time required for listening to the synchronization message. With, typical startup time of approximately 470 μsec 8, the energy consumed in transitions can relatively be a significant portion and will often diminish the energy saving achieved by the sleep mode.

Our approach takes advantage of the flow of routing traffic from the gateway and includes the synchronization information in the update messages of the broadcast phase. Since all the sensors have to switch on to receive route and schedule update, the overhead that the sensors would have incurred in switching on to listen to synchronization messages will be eliminated. However, the frequency of route updates for some networks may be insufficient to maintain the desired level of clock synchronization. Therefore, we exploit the trade-off between the guard-time and the frequency of resynchronization. The guard time is a precautionary measure used to tolerate the difference in clock readings of communicating nodes. Increasing the guard time will require a sensor node to be activated earlier than its reception slot in order to tolerate a clock drift. We propose two procedures to handle the variations in synchronization requirements.

In the first procedure, the guard time is increased to account for a low frequency reroute phase. The size of the guard time depends on the clock drift rate and how often the rerouting is performed. With a typical clock drift rate for the sensor clocks of 1 μsec every second 2225, this approach can be very effective. The energy consumption due to incrementing the guard time by a few μ seconds will be negligible compared to that for a state transition time (470*2 μsec), unless many sensors are active all time. The other factor for consideration is the accuracy required by the sensor application since an accumulative clock drift can make the timing of events inaccurate. However, we believe that this is not an issue in many popular applications. For example, the timing accuracy requirements for beam forming applications are about 100 μsec 27 and thus this procedure can be beneficial.

The second proposed procedure, which we call Optimized SYNC, tackles applications with stringent synchronization requirements. In a scenario where the time lapse between two reroute phases can magnify the clock jitter to an intolerable level, synchronization messages are unavoidable. In the proposed optimized SYNC procedure, only the active nodes are periodically synchronized, while the inactive nodes are synchronized only in the reroute phase. Inactive sensors can apply a reroute phase specific guard time turning on a little bit earlier than the next scheduled rerouting phase in order to accommodate the worst-case clock jitter. We argue that this mechanism will yield significant energy savings compared to periodically synchronizing all the nodes, especially when a small fraction of the deployed sensors are engaged at a given time. We will compare the performance of the proposed procedures through simulation in section 4.

3 Time Slot Scheduling

The slot-scheduling mechanism presented in this section is part of the reroute and arbitration phase. The mechanism consists of two parts comprising of intra-cluster and inter-cluster analysis and scheduling. Each gateway executes an intra-cluster algorithm to calculate the transmission schedules for the sensors in its cluster. The goal of the intra-cluster algorithm is to minimize the sensor transition between active and sleep mode while meeting flow constraints and avoiding buffer overflow at

all relay nodes. The inter-cluster analysis ensures that transmission slots within each cluster will not interfere with the simultaneous use of slots at neighboring clusters. The partitioning of slot assignment into two parts allows scalability of the approach to large networks. In this section we first address the intra-cluster slot assignment to sensors and discuss collision avoidance among clusters in subsection 3.3.

Slot scheduling within a cluster for multi-hop traffic is a typical network flow problem. Therefore, depth first search (DFS) and breadth first search (BFS) graph parsing techniques can be applied to assign transmission slots 26. However as shown in the balance of this section, DFS results in excessive wastage of energy due to a large number of transitions while BFS leads to packet drop because of sensors' buffer size limitation. To overcome the limitations of these contemporary approaches, we propose an intra-cluster slot allocation algorithm that is inspired by the Tabu-search optimization technique. The main objective of the optimization is to minimize the energy consumed by the radio circuit in idle mode and due to unnecessary transition between active and sleep modes.

Tabu-search was introduced in 3031 and subsequently has been successfully applied to many applications 323334. Tabu-search has become an accepted technique that in some cases surpassed conventional optimization techniques. Analogous to other optimization techniques Tabu search employs an iterative procedure in order to find a better solution in the neighborhood of the initial (current) solution. The local search procedure can be gradient-based, a random walk, etc. The search process is concluded when a terminating condition, such as maximum numbers of iterations or limited enhancements in the solution, is met. Unlike other techniques Tabu-search employs an evolving memory to prevent getting trapped in local optima. This memory stores recently employed moves, and guides the search process by forbidding recently visited solutions. In the balance of this section we formulate the intra-cluster slot assignment problem and then present our approach for optimal slot assignment to communicating nodes in subsection 3.2.

3.1 Problem Formulation

The sensor network can be modeled as a set G = (V, E), where V is the set of sensors with |V| = *N* and E is the set of links. A link is a set of two sensors. The two primary sources of energy wastage in TDMA based MAC are; the excessive idle time of the sensors when the receiver is unnecessarily left on, and the time taken to make a transition from active (during transmit or receive) to powered-down sleep state and vice versa. Therefore, we set our optimization objective to minimizing the summation of energy consumed in idle mode and state transition across the sensor network. The following notation is used in our formulation:

S : The set of sensing nodes with $S \subseteq V$
R : The set of relay nodes with $R \subseteq V$
P : The set of links that are used in the routes with $P \subseteq E$
C_1 : Energy consumed in activating or deactivating a relay nodes
C_2 : Energy consumed for being in idle state during one time slot
B_{ij} : Number of packets stored in the buffer of sensor *i* during slot *j*
B_{max} : Maximum number of packets a sensor can buffer
TSlots : Total number of slots in a periodic frame
L_{kl} : Link connecting sensor *k* to sensor *l*

Let the set Active = S ∪R. Since in each frame some of the nodes in the network will be in sleep state, it is usual to have |Active| < N. We further define the following decision matrices:

$T_= (T_{ij})$ be a TSlots × |P| indication (schedule) matrix with $T_{ij} = \begin{cases} 1, & \textit{if slot } i \textit{ is used by edge } j, \\ 0, & \textit{otherwise.} \end{cases}$

Idle = $(Idle_{ij})$ be an |Active| × TSlots indication matrix with $Idle_{ij} = \begin{cases} 1, & \textit{if node } i \textit{ is idle,} \\ 0, & \textit{otherwise.} \end{cases}$

A = (A_{ij}) be an |Active| × TSlots indication matrix with $A_{ij} = \begin{cases} 1, & \textit{if node } i \textit{ is activated before slot } j, \\ 0, & \textit{otherwise.} \end{cases}$

D = (D_{ij}) be an |Active| × TSlots indication matrix with $D_{ij} = \begin{cases} 1, & \textit{if node } i \textit{ is deactivated after slot } j, \\ 0, & \textit{otherwise.} \end{cases}$

The algorithm calculates schedule T so as to:

Minimize:

$$\sum_{\forall i \in Active} \sum_{j=1}^{j=TSLots} \left[C_1 (A_{ij} + D_{ij}) + C_2 Idle_{ij} \right] \quad \ldots\ldots (2)$$

Subject to:

$$T_{ij} \in \{0,1\}, \qquad i \le \text{TSlots},\ j \le |\text{P}| \qquad \ldots\ldots (3)$$

$$A_{ij} \in \{0,1\}, \qquad i \in \text{Active},\ j \le \text{TSlots} \qquad \ldots\ldots (4)$$

$$D_{ij} \in \{0,1\}, \qquad i \in \text{Active},\ j \le \text{TSlots} \qquad \ldots\ldots (5)$$

$$Idle_{ij} \in \{0,1\}, \qquad i \in \text{Active},\ j \le \text{TSlots} \qquad \ldots\ldots (6)$$

$$\sum_{j=1}^{j=|\text{P}|} T_{ij} \le 1, \qquad i \le \text{TSlots} \qquad \ldots\ldots (7)$$

$$\sum_{m=1}^{m=i} \frac{1}{2}\left(A_{km} + A_{lm} - D_{km} - D_{lm}\right) \ge T_{ij} \qquad i \le \text{TSlots},\ j \in \text{P} \qquad \ldots\ldots (8)$$

$$B_{ij} \le B_{\max} \qquad i \in \text{Active},\ j \le \text{TSlots} \qquad \ldots\ldots (9)$$

$$Idle_{ij} = \Omega\left(T_{jk}\right) \qquad i \in \text{Active},\ j \in \text{TSlots}, k \in \text{P}, i \in k \qquad \ldots\ldots (10)$$

$$A_{ij} = \Psi_1\left(T_{jk}\right) \qquad i \in \text{Active},\ j \in \text{TSlots}, k \in \text{P}, i \in k \qquad \ldots\ldots (11)$$

$$D_{ij} = \Psi_2\left(T_{jk}\right) \qquad i \in \text{Active},\ j \in \text{TSlots}, k \in \text{P}, i \in k \qquad \ldots\ldots (12)$$

In the formulation the objective function in (2) is to minimize the total energy consumed by the sensors due to idle time and energy consumed due to transitions between active and sleep states. The constraints in (3), (4), (5) and (6) indicate that entries in decision matrices are binary. The constraint (7) implies that in a particular cluster no more than one sensor can transmit and only one sensor can receive in each slot. However in distinct clusters each gateway will run its own scheduling algorithm and slots can be reused in different clusters after ensuring the absence of inter-cluster collision. In addition, we implicitly constrain the number of slots in a frame to TSlots so that the frame size would not grow larger while optimizing the number of transitions and idle slots.

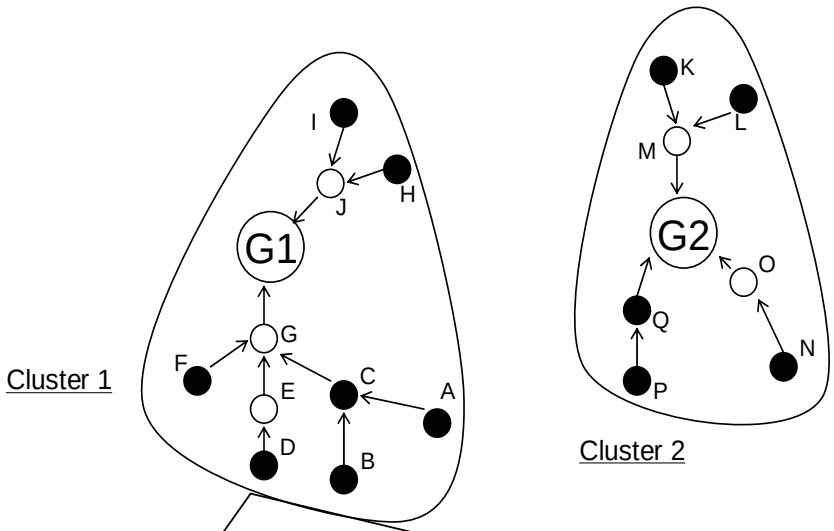
Constraint (8) implies that two nodes *k* and *l* must be activated before slot *i* if they have to communicate during slot *i*. If both nodes have been activated more than the number of times they have been deactivated from the beginning of the frame till slot *i*, it will mean that they are in active state and can be assigned slot *i*. Constraint (9) indicates that the total number of packets in the buffer for any sensor should be always less than the maximum buffer size. Equations (10), (11) and (12) derive the idle state and transition indicators for a node from the current schedule T. Such derivation is intuitive and can simply be done by parsing the schedule and generating the perspective idle slots and the transition patterns for each active sensor. If the difference between two non-contiguous slots assigned to a particular node is greater than the sum of activation and deactivation times, for going to and out of the sleep mode, the node can make a transition to sleep state, otherwise it remains in idle state until next slot assigned to it. It is also worth noting that the functions Ω, Ψ_1, and Ψ_2 are not linear with respect to the schedule *T*.

While the information captured in matrices *A* and *D* can be combined in only one matrix with the use of tri-state indicators, we have found that sticking to Boolean values simplifies the implementation. Given the non-linearity of the optimization problem and the large number of variables, analytical solution is not attainable and search heuristics should be pursued. We show that proposed search algorithm is a good match to such slots scheduling problem. We partition the routing tree into sub-trees and optimize the schedule within each sub-tree. The process is then repeated for scheduling each sub-tree. Like Tabu-search methodology local optima is stored in the memory after every intermediate step. The property of storing local optima as the search progress expedites the convergence. The detailed search process is explained in the next subsection.

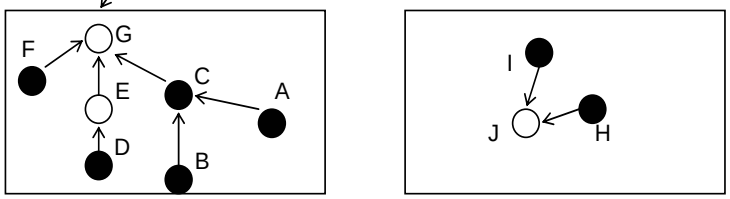
3.2 Search Heuristic

We will use Fig. 3 to illustrate our approach for slot scheduling. Fig. 3a represents a sample sensor network topology. It consists of two clusters. Each cluster has its own gateway that is responsible for sensor organization, route setup and assigning transmission slots to the sensors in its cluster. The gateway will establish a multi-hop route, assign a transmission slot for every active sensor to send the collected data and designate time slots for relay sensor to forward the data to the gateway. Sensors, which are not engaged by the gateway in probing the environment, switch to low-power sleep mode. In Fig. 3a, nodes A, B, C, D, F, H and I are sensing nodes that generate their own packets, while nodes E, G and J are relays. Node C generates its own data and also relays the data forwarded by its leaf nodes. We will use cluster #1 to illustrate our slot scheduling approach. The gateway of cluster #2 follows a similar methodology.

a)



b)



c)

Slot No.	1	2	3	4	5	6	7	8	9	10	11	12	13
Receive	**E**	**G**	C	C	G	G	Gateway	Gateway	Gateway	G	G	Gateway	Gateway
Transmit	**D**	**E**	A	B	C	C	G	G	G	C	F	G	G

d)

Slot No.	1	2	3	4	5	6	7	8	9	10	11	12	13
Receive	C	C	G	G	G	Gateway	Gateway	Gateway	**E**	**G**	G	Gateway	Gateway
Transmit	A	B	C	C	C	G	G	G	**D**	**E**	F	G	G

Fig. 3: a) Initial topology represented as trees b) Cluster-1 after partitioning into distinct trees c) Node G is selected at level-2 of Tabu Search d) At level-3, slots of branch 1 (A, B and C) are exchanged with slots of branch 2 (D and E).

In Fig. 3a, the arrows indicate the direction of packet flow from the sensors to the gateways. It is clear from the figure that the data routes within a cluster form a tree rooted at the gateway. Such a routing tree can be envisioned as a set of distinct smaller trees (branches), each connected to the gateway with a dedicated link. For example, the data routes of cluster #1 can be partitioned to smaller trees, as illustrated in Fig. 3b. The paths followed by packets in each of these sub-trees will be independent of the other sub-trees. This partitioning of the routing tree will significantly reduce the complexity of the slot-scheduling problem. By using this approach, the gateway can break the problem into smaller sets and deal with each sub-tree independent of other sub-trees instead of simultaneously dealing with all the sensors in the cluster. In the following discussion we will use the term tree to simply refer to a branch of the gateway node.

We use a modified version of BFS to generate an initial solution to our search heuristics. Slots are assigned such that nodes get contiguous transmission and reception slots. To avoid packet drops, nodes are assigned contiguous reception slots that do not exceed its buffer size**.** We illustrate this approach in Fig 3c. Let's assume that the size of a node's buffer is 3. After applying our approach nodes D (the node with highest depth) and E are assigned slots 1 and 2 respectively. Node G is not assigned the next transmission slot since it has not yet received all the packets destined to it. Therefore, nodes A and B that have the highest depth among node G's remaining branches are assigned slots 3 and 4, while node C is assigned slots 5, 6.

Node C is not assigned slot number 7 even though it has a packet to transmit because the buffer of node G is full. Instead, node G is assigned slots 7, 8 and 9 since it has consumed all its buffering capacity. The modified assignment of slots will prevent the buffer overflow at node G since the buffer will be flushed as soon it gets filled up. Finally node C and F are allocated slots 10 and 11 while node G is allocated slots 12 and 13.

Once the initial solution is generated, the slot' scheduling algorithm is applied to find the optimal slot assignment. The search process examines different combination of slot assignments until either an optimum is reached or a terminating condition is met. To increase the efficiency of the search, a divide and conquer scheme is employed. The search is partitioned into three distinct levels namely; Tree level, Node Level and Branch level. Each level maintains a list, called the memory or Tabulist of that level, for storing the recent combinations of slot schedules that have already been tried so that they are not exercised again. The optimization process starts at the tree level and considers some (or all) routing trees in the cluster. This level ensures that the slots assigned to trees are optimized for minimal energy consumption. As an example, for cluster #1 in Fig. 3 search at this level can be performed twice, once for tree#1 and once for tree#2. The selection of a tree for optimization is prioritized based upon the ratio of energy consumed due to transitions and idle time to the number of slots required by the tree. The tree with highest ratio (Tree #1 in example 3) is selected for optimization and is added to the Tabulist so that it is not considered in future iterations. For non-complex topology such as the one in our illustrative example, it will make sense to try all trees. However, for large networks a subset can be picked in order to limit the search complexity.

```
INITIAL SOLUTION()
1.    slot_count = 0
2. For (i = 0; i < MaxTreeCount; i++)
3.     slot_count = AssignSlots(RootNode, slot_count)
4. endFor

AssignSlots(Node, slotcount)
5. If(!(all children of the Node have been allocated slots))
6.    slot_count=AssignSlots(Random child node not allocated slots, slot_count)
7.    slot_count = slot_count + 1;
8.    Node_slot[Node] = slot_count;
9. return slot_count

SEARCH HEURISTIC ()
10.  Set best_solution = initial solution.
11.  While (Tabulist1 < Max_Iterations)
12.  Choose the tree with highest energy consumption to optimize;
13.  If (tree is in Tabulist1)
14.     Go to 2.
15.  else
16.     Add the tree to Tabulist1;
17.     Set Iteration_Count_1=0;
18.     While (Iteration_Count_1 < Max_Iterations_1 and |Tabulist2| < node_count)
19.          Choose the node with highest energy consumption of the selected tree;
20.          If (selected node is in Tabulist2)
21.             Go to 9;
22.          else
23.             Add the node to Tabulist2;
24.             Iteration_Count_1++; Iteration_Count_2=0;
25.             While (!(No improvement For X iterations) and Iteration_Count_2 <
                Max_Iteration_2)
26.                Iteration_Count_2++;
27.                Choose 2 random inbound branches of the selected node to swap;
28.                If (selected branches are in Tabulist3)
29.                   Go to 16;
30.                else
31.                   Swap branches and calculate energy;
32.                If (energy > energy of best solution)
33.                   Update Tabulist3;
34.                   Go to step 29;
35.                else if (energy < energy of  best solution)
36.                   Update Tabulist3;
37.                   best solution=current solution;  go to 16;
38.                Undo the swap move; go to 16;
39.                endWhile
40.             Empty Tabulist3;
41.       endWhile
42.      Empty Tabulist2;
43.  endWhile
```

Fig 4: The intra-cluster slot assignment algorithm.

All the nodes of the selected tree are passed to the next (Node) level. Similar to the tree counterpart, the node level is repeated until either the optimum solution is found or the maximum number of iterations is completed. In each iteration the node whose branches have the highest ratio of energy consumed due to transitions and idle time to the number of slots is selected and passed to the third (branch) level for optimization. The selected node (Node G in the example) is then added to the

memory (Tabulist) of this level and is retained in memory for number of iterations equal to the size of the Tabulist. Any node that is in the memory for the current iteration is assumed to be optimized and thus cannot be passed to the third level for optimization. The implementation of the Tabulist is a circular queue. When a node leaves the Tabulist, it can again be reconsidered for optimization at the branch level.

The branch level search strives to minimize the energy consumed by the currently considered node. Different combinations of slot schedules are tried for this node by swapping the transmission slots allocated to the inbound branches and then retaining the best allocation. For example slots assigned to nodes A, B and C are swapped with that of nodes D and E in Fig 3c to get a new schedule in Fig 3d. The resultant schedule reduces the number of transitions for node C. It also reduces the idle time for node G. It is worth noting that the slots are swapped such that the resulting schedule does not result in buffer overflow. After swapping, the energy consumed by the new solution is compared to the energy of the current (best) solution. If new solution consumes less energy, it is saved as the current solution and the swap move is saved in the memory of this level so that it is not repeated again until it is in the memory. This level is terminated either when all the swap moves are in the memory or maximum numbers of iterations is reached. The algorithm is sketched in Fig. 4.

Table 1 shows a transmission schedule when applying our approach. For the sake of comparison we include the schedule using BFS and DFS in tables 2 and 3 respectively. Comparing table 1 to tables 2 and 3 demonstrates the superiority of our approach to BFS and DFS, both in terms of packet drop count and energy consumption due to both transitions and idle time. More elaborate performance's comparison will follow in section 4.

Table 1: Using proposed approach, there is no packet drop and a total of 13 transitions. Sensor G is in idle mode for one slot (Tr = Transmit, Rec = Receive, Sl = Sleep)

Slot No. / Sensor ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
A	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
B	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
C	↑ Rec	Rec	Tr	Tr	Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
D	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl
E	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	↑ Rec	Tr	↓ Sl	Sl	Sl
F	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	↑ Tr	↓ Sl	Sl
G	Sl	Sl	↑ Rec	Rec	Rec	Tr	Tr	Tr	**Idle**	Rec	Rec	Tr	Tr
Gateway	-	-	-	-	-	Rec	Rec	Rec	-	-	-	Rec	Rec

Table 2: Using BFS, some packets will not reach the gateway (max. buffer size = 3). A total of 17 state transitions are needed. Sensor G is idle for two slots. (Tr = Transmit, Rec = Receive, Sl = Sleep)

Slot No. / Sensor ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
A	Sl	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
B	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
C	Sl	↑ Rec	Rec	↓ Sl	Sl	↑ Tr	Tr	Tr	↓ Sl	Sl	Sl	Sl	Sl
D	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
E	↑ Rec	↓ Sl	Sl	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
F	Sl	Sl	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
G	Sl	Sl	Sl	↑ Rec	Rec	Rec	**Rec(Drop)**	**Rec(Drop)**	Tr	Tr	Tr	**Idle**	**Idle**
Gateway	-	-	-	-	-	-	-	-	Rec	Rec	Rec	-	-

Table 3: Using DFS, total of 15 transitions take place. Sensor D is idle for one slot while sensor G is idle for two slots. (Tr = Transmit, Rec = Receive, Sl = Sleep)

Slot No. / Sensor ID	1	2	3	4	5	6	7	8	9	10	11	12	13
A	Sl	Sl	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
B	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl
C	↑ Rec	Tr	**Idle**	Rec	Tr	↓ Sl	Sl	Sl	Sl	↑ Tr	↓ Sl	Sl	Sl
D	Sl	Sl	Sl	Sl	Sl	Sl	↑ Tr	↓ Sl	Sl	Sl	Sl	Sl	Sl
E	Sl	Sl	Sl	Sl	Sl	Sl	↑ Rec	Tr	↓ Sl	Sl	Sl	Sl	Sl
F	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	Sl	↑ Tr	↓ Sl
G	Sl	↑ Rec	Tr	**Idle**	Rec	Tr	**Idle**	Rec	Tr	Rec	Tr	Rec	Tr
Gateway	-	-	Rec	-	-	Rec	-	-	Rec	-	Rec	-	Rec

3.3 Inter-Cluster Collision Avoidance

Collisions will occur when a sensor can hear multiple transmissions in a particular time slot. Using our TDMA based scheme collisions will not occur between sensors of the same cluster since only one sensor is scheduled to transmit in a particular slot. However the operation of the different clusters are simultaneous and inter-cluster collisions cannot be ruled out. Inter-cluster collisions may degrade the performance of the network and diminish the advantage of the distinct slot assignment within each individual cluster. Although the inter-cluster collisions can be eliminated via partitioning the frequency band among clusters and designating unique range for each cluster, dividing the frequency band leads to an increase in both the sensor active time and energy consumption 8. In addition, while adopting sequential operation of clusters will also prevent inter-cluster collision, it can extend the TDMA frame size and cause unacceptable data latency for the application. Therefore, an alternative scheme is thought.

A possible approach for preventing inter-cluster collisions is to ensure that the gateways in adjacent clusters assign different slots to sensors that are close to the inter-cluster boundary and whose transmission ranges overlap. Such a precautionary measure requires each gateway to be aware of the schedule of adjacent clusters. However, the knowledge of schedules of neighboring clusters does not solve the problem. To form a collision free schedule, each gateway has to validate the allocation of every slot in its cluster with the corresponding slot in neighboring clusters. For a large network, the number of comparisons required can be large. Moreover, after the comparisons, the gateway will have to modify some slots to eliminate collisions. Finding the appropriate slot reassignment can be NP-hard. Modification in the schedule may induce additional collisions and the slots will have to be compared again and so on. Additionally, modifying the schedules might also negate the energy efficiency achieved by our slot' scheduling algorithm.

A closer look at the intra-cluster slot allocation scheme provides insight to a good approach to deal with this problem. It has been shown in Fig. 3 that each cluster is composed of a number of disjoint trees rooted at the gateway. The schedules of these trees are independent of each other. For example we can have two optimum slot schedules for cluster #1. Assigning slots 1 through 10 to tree #1 followed by slots 11 through 13 to tree #2 is one such solution. An alternate solution could be to allocate slots 1 through 3 to tree #2 followed by slots 4 through 13 to tree #1. Energy consumption due to both these solutions is identical. Thus, if the first solution leads to collision with an adjacent cluster and the second one does not lead to collisions, we could employ the second solution without sacrificing the optimality of the slot schedule. Our scheme to avoid collisions with sensors in adjacent clusters exploits this characteristic. The approach, which is detailed in the remainder of this subsection, maintains the energy efficiency of the schedule computed by intra-cluster slot allocation algorithm and achieves a quasi-optimal solution at reasonable complexity. Even with a bounded frame size only a few modifications are required to the schedule. In addition, our proposed approach limits the number of comparisons performed.

The proposed scheme works as follows. After calculating the transmission schedules for sensors within their respective clusters, the gateways elect one of the gateways as the head gateway. The head gateway is one of the gateways in the network picked in a round robin fashion. A new head gateway is selected for each arbitration phase. The head gateway collects transmission schedules from all the clusters and then scans through these schedules to detect potential inter-cluster collisions. In case of a collision, the head gateway attempts to avoid this collision by replacing all the slots allocated to the affected tree with those designated to a different tree in the same cluster. We call such a step a tree swap. As explained earlier, such a

swapping of slots does not affect the optimality of the intra-cluster schedule. However, it may not completely solve the issue and may generate new transmission conflicts.

A conflict that remains unresolved through tree swapping can be handled in one of two mechanisms. In the first, all the sensors in the affected tree are allocated slots at the end of the current transmission schedule. This mechanism will maintain the optimality of the intra-cluster schedule and will be pursued only if the frame size would not get extended. Assuming F to be the TDMA frame size, F_C to be the length of the intra-cluster schedule and S_t is the number of slots required by the tree causing the collision, it will be acceptable to re-allocate these S_t slots at the end of F_C as long as $F \geq F_C + S_t$. If not possible, the slots required by such tree will be satisfied using non-contiguous slots in the schedule possibly sacrificing the optimality criteria for that particular tree. Clearly the first mechanism for handling unresolved conflicts is preferred. To limit the potential for applying the second mechanism, we start consider trees in a decreasing order of their required slots so that trees remaining unresolved through swapping at the end will have the smallest possible S_t and thus increasing the possibility of successful collision prevention through the first scheme.

We illustrate the collision avoidance approach with the example network shown in Fig. 5. The considered network consists of three clusters. We assume that the gateways have calculated the transmission schedules for the trees in their cluster using our slot' scheduling algorithm detailed in the previous subsection. We also assume that the gateways have transmitted these schedules to the head gateway. The head gateway scans these schedules to detect and eliminate the possibility of collisions.

Schedule scanning consists of several steps. First, the trees in each cluster are sorted in a decreasing order in the number of time slots they require. The trees of various clusters are then grouped based on their positions in the sorted lists. Fig. 6a, shows the sorted lists and the groups derived from them. Trees #1, #4 and #6 require the maximum number of slots in their respective clusters and hence are grouped together. Similarly, trees #2, #5 and #8 are grouped together into group #2 while trees #3 and #7 fall into group #3. As mentioned earlier, the wisdom of sorting the trees and forming inter-cluster groups is to mitigate the impact of the worst-case scenario when collisions can only be avoided by allocating slots to some trees at the end of the current intra-cluster transmission frame, hoping that if needed only small trees are moved.

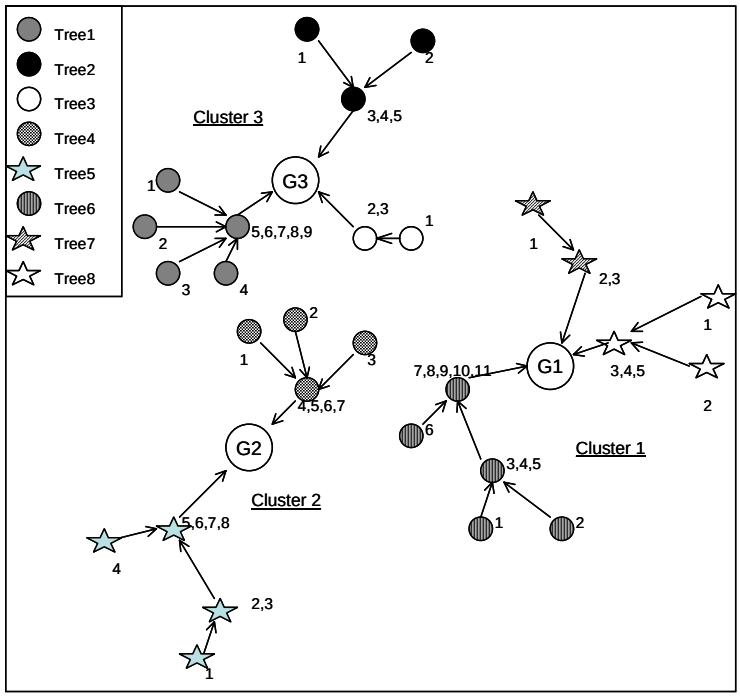


Fig. 5: An example topology for sensor networks

In the next step, the trees that fall in the same group are compared for potential collisions. In case of a collision the tree that requires the least number of slots is replaced with another tree in its cluster. Figures 6b through 6f illustrate the approach application to the network of Fig. 5. The procedure starts with group #1 since it requires the maximum number of slots. Trees of this group are picked in the decreasing order of slots required. Tree #6 is selected first and is retained in group #1 as there are no other trees for comparison as shown in Fig. 6b. Tree #1 is selected next and its slots are checked for collision with tree #6. Since no collisions are detected it is also retained with group #1 as shown in Fig. 6c. Tree #4 is selected next and compared with trees #1 and #6. Collisions are detected between tree #4 and tree #1, as shown in fig 6d. Therefore tree #4 cannot remain with group #1. Tree #5, which requires the next largest number of slots in tree #4's cluster, is then picked and compared with trees #1 and #6. Since there are no collisions, tree #5replaces tree #4 of group #1. Tree #4 is then moved to the next group (group #2). The maximum number of slots required by group #2 thus becomes equal to 8. After group # 1 has been made free of collisions, trees in groups #2 and #3 are scanned for collisions. Since no more collisions are detected all the trees retain their positions as shown in Fig 6f. After the scan is complete and all the potential collisions have been eliminated, the slots are assigned to trees based upon their group. This is illustrated in Fig. 6g. Trees with group #1 get first 11 slots, which is equal to the maximum required by any tree in this group. Similarly group #2 gets 12 through 19 (maximum 8 Slots) and group #3 gets slots 20 through 22 slots (maximum 3 Slots).

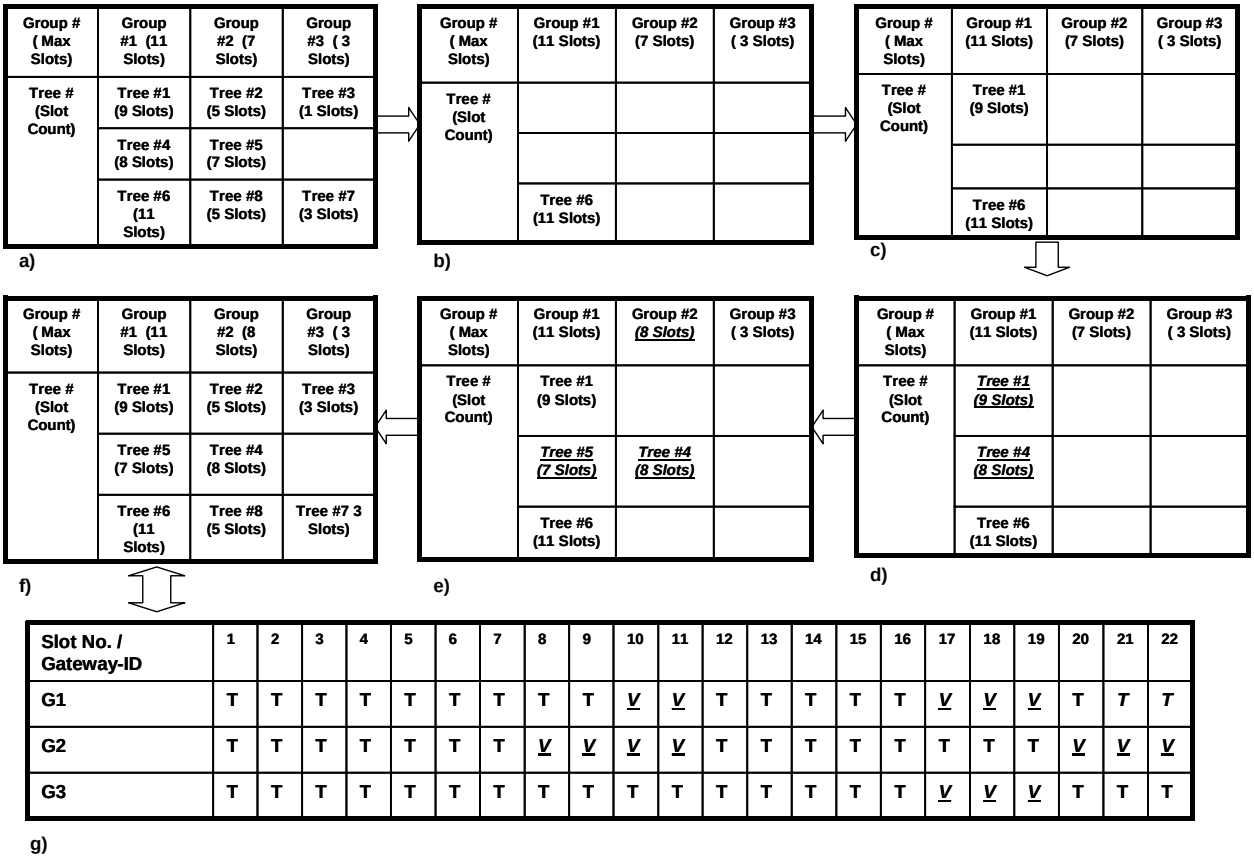


Fig. 6: Step by step procedure followed by head gateway to form collision free schedules (V=Vacant, T=Transmit).

In case a tree cannot find a collision free schedule in any of the available groups, it will be allocated slots at the end of current schedule of its cluster. To illustrate the methodology to deal with this issue, let's assume that the topology is different from one shown in Fig. 7 and schedule of tree #3 collides with schedules of all the trees in cluster #3. Therefore it cannot be placed in any of the groups #1, #2 and #3. In such a situation a new group is inserted at the end and tree #3 is added to that group. This is illustrated in figures 7a and 7b. There might also be situations when the maximum frame size is reached and no new groups can be added to the end of the schedule. In such a scenario remaining trees fill in the vacant slots allocated to its cluster. As an example, let's assume that the maximum frame size is 22 in the above scenario and no new groups can be added to the end of the schedule. In this situation, Tree #3 is allocated slots in any of the vacant slots in its clusters schedule. Thus, any of the vacant slots numbered 10, 11, 17, 18, 19, 21 and 22 as shown in Fig. 6g can be used by tree #3. It should be noted that this scenario might also happen if the swapping of trees results in exceeding the frame boundary, e.g. when the size of trees in the first group are dominantly larger than other trees in the their respective clusters.

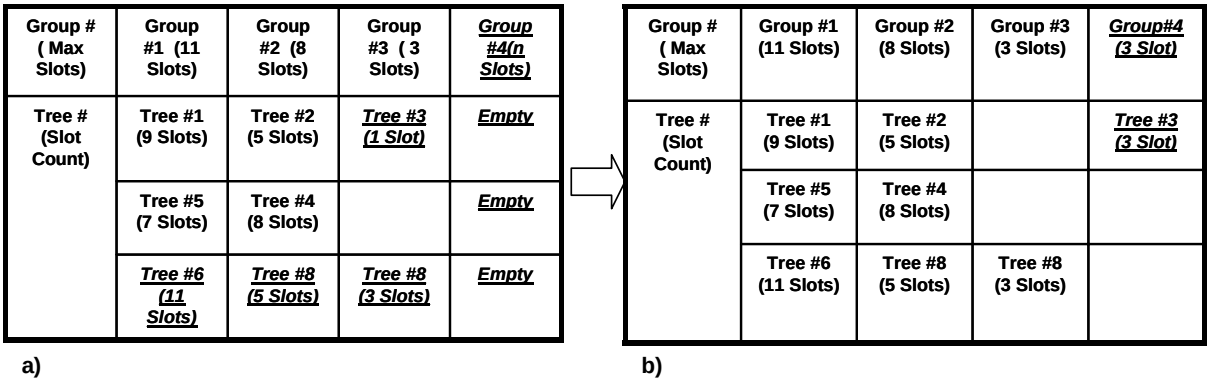


Fig. 7: Additional slots are added in case no collision free schedule can be found

The algorithm for computing collision free schedules is detailed in Fig. 8. The number of messages exchanged in the course of executing the inter-cluster collision avoidance algorithm is linear in the number of gateways. Assuming m is the number of clusters in the network and q is the maximum number of sensors in any tree, the "for" loop on line 1 of the "compare" function is executed a maximum of m time. The complexity of the comparison of slots in line 2 is $O(q^2)$. Thus, the total execution time complexity of the "compare" function is $O(mq^2)$. In the worst case all the trees in the cluster will collide with the current tree, therefore the inner "while" loop will be executed in $O(m)$ time. Similarly the worst case performance of the outer "while" loop is $O(nm)$, where n is the maximum number of trees in any cluster. The running time of outer and inner for loops in FillVacantSlots is $O(m)$ and $O(n)$ respectively, while running time of inner while loop is $O(nq)$. Therefore, the worst case performance of the FillVacantSlots module is $O(mn^2q)$. Thus the overall complexity of the algorithm is $O(nm^3q^2 + mn^2q)$.

```
1.  Initialize Done = 0;GatewayCount = 1; CurrentGroup = 1;Collision = 1;
2.  Sorts the trees in descending order of number of slots required.
3.  Sort the groups in the descending order of the number of slots required.
4.  While(!Done)
5.    While(Collision == 1)
6.      Collision = COMPARE(GatewayCount, CurrentGroup, CurrentTree);
7.      if(collision == 1)
8.        if(No Tree is Available)
9.          Insert Vacant Slots
10.         Insert CurrentTree at the end of list of trees for the cluster
11.         Collision = 0;
12.     else
13.       replace the position of tree with that of next non-colliding tree in the cluster
14.   endWhile
15.   GatewayCount++;
16.   if(GatewayCount = MaxGatewayCount && CurrentGroup = MaxGroupCount)
17.     Done = 1;
18.   else if(GatewayCount == MaxGatewayCount)
19.     GatewayCount = 0;
20.     CurrentGroup++;
21. endWhile
22. if ( Frame Size has exceeded)
23.   FillVacantSlots(TreesExceedingFrame size,VacantSlotList)

COMPARE (GatewayCount, CurrentGroup, CurrentTree)
1.  For(i = GatewayCount; i < MaxGatewayCount; i++)
2.    Compare all slots in the Trees for Collision
3.    if(collision)
4.      return 1;
5.  endFor
6.  return 0;

FillVacantSlots(TreesExceedingFrame,VacantSlotList)
1.  For(i = 0; i < MaxGatewayCount; i++)
2.    For(j = 0;j < Gateway[i].TreeExceedingFrameCount; j++)
3.      k=0,m=0;
4.      While(k < Tree[j].SlotCount or m < VacantSlotList[i].size)
5.        Check if use of slot at index m leads to collision.
6.        if (no collision)
7.          allocate vacant slot to current sensor
8.          remove slot at index m from VacantSlotList[i]
9.          k++;
10.       else
11.         m++;
12.     endWhile
13.   endFor
14. endFor
```

Fig. 8: Inter Cluster Collision Avoidance Algorithm.

4 Experimental Validation

The effectiveness of our approach is validated through simulation. This section describes the simulation environment, performance metrics and experimental results.

4.1 Setup and Metrics

In the experiments varying number of nodes are randomly placed in a 1000×1000 meter square area. The gateway is randomly positioned within this area. A free space propagation channel model is assumed 29 with the capacity set to 2Mbps. Packet lengths are 10 Kbit for data packets. For a node in the sensing state, packets are generated at a constant rate of 1 packet/sec 18. The time taken in making a transition between the sleep and active states is assumed to be 470μsec. The power consumed at the circuit level in transmission and reception of a packet is set to 81mW and 180mW respectively 8. It should be noted that though we have based out results on one particular radio model the other models exhibit similar characteristics 22. The energy consumed in the transition is obtained by multiplying the transition time by the average of the power consumed by the radio circuit in active and sleep states 8. In the implementation, a radio circuit in a sleep mode is assumed not to consume any power. Energy consumption due to the radio amplifier in case of the transmission depends on the distance and is optimized through the use of multi-hop routing. Routes are computed based upon Dijkstra's shortest path algorithm, such that each sensor will transmit the information to its closest distance neighbor. The probability of a sensor being on in a particular frame was varied from 0.1 to 1. The set of sensing nodes changes every reroute phase. The clustering was done such that each sensor falls in the cluster of a gateway that is nearest to the sensor. Clock drifts were assumed to be 1 μsec every second. A packet was assumed to be lost in case of a collision and is not retransmitted.

The following metrics are used to measure the performance of the proposed scheme:

- *Energy Consumed by the sensors in idle mode and due to transitions:* This measures energy consumed by sensors while turning the radio circuitry on and staying idle. It also measures energy dissipated due to transitions between active and sleep modes. This should be minimized to extend sensor lifetime.
- *Throughput:* It measures the number of packets reaching the gateway per time unit. Throughput is important because if large numbers of packets are dropped, the gateway cannot form the correct vision for the activities in the covered area.
- *Average delay per packet:* It measures the average time a packet takes to reach the gateway from the instant it is generated. Long delays will result in packets reaching the gateway when the information is of no use.
- *Energy Consumed by the Gateway:* The gateways have sufficient energy resources. However they do not have infinite resources and this metric will be a good indicator of the capabilities a gateway should have.
- *Energy Consumed by the Sensors due to collisions:* This metric measures the amount of energy wastage by the sensors due to collisions, which should be completely eliminated.

4.2 Experiment Results

Performance of the clock synchronization scheme: We conducted a set of experiments to compare the synchronization schemes proposed in section 2 with a scenario where all sensors were synchronized periodically via explicit synchronization messages (we call this scheme SYNC). Fig. 9 shows the results of theses experiments with time between successive synchronization captured in terms of interleaving data cycles. The results for the optimized SYNC, in which only active sensors receive synchronization messages while the remaining sensors are synchronized only in the reroute phase, show a reduction of up to 280% in comparison to SYNC. When the synchronization messages are included in the reroute phase (SYNC in reroute) the energy consumption related to synchronization is minimal. In addition, Fig. 9 indicates that a considerable reduction in energy consumption can be achieved if the frequency of synchronization messages can be decreased through better clock circuitry or a relaxed application delay requirements.

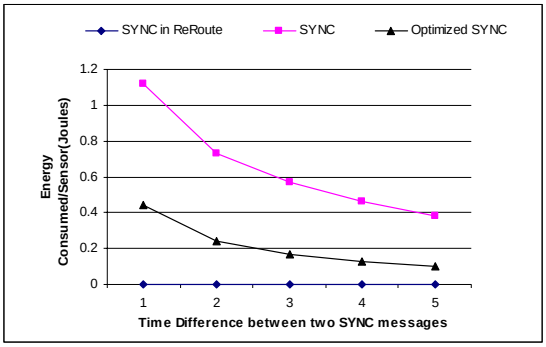


Fig. 9: Performance of the clock synchronization schemes under varying re-synch frequency (captured in terms of the number of data cycles between successive synchronization)

Comparison between the time slot assignment algorithms: We ran a set of experiments to compare the performance of our slot' scheduling algorithm (SLA) to DFS and BFS. The results are shown in figures 10 and 11. Fig. 10 displays the average

energy consumed by a sensor due to transitions. The result corroborates the practicality of our approach, since it combines the advantages of the other two approaches. Moreover, for less number of sensors DFS performs better than BFS but for larger number of sensors BFS gives superior results. This is expected since as the number of sensors increases the depth of the tree increases in comparison to its breadth. Consequently the difference between slots assigned to sensors in DFS widens extending the sensor's active time. In the second experiment we varied the packet sizes and observed its effect on the energy consumed by the system due to transitions. As the packet sizes were reduced the energy contributed due to transitions increased manifold. Therefore, as shown in Fig. 11 for smaller packet sizes effect of transitions becomes more conspicuous and the significance of our approach increases.

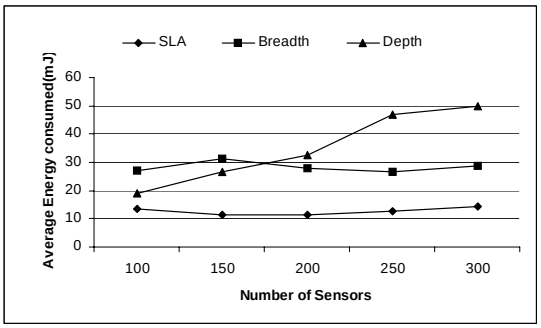


Fig. 10: Effect of number of sensors on the average energy consumed by a sensor in idle state

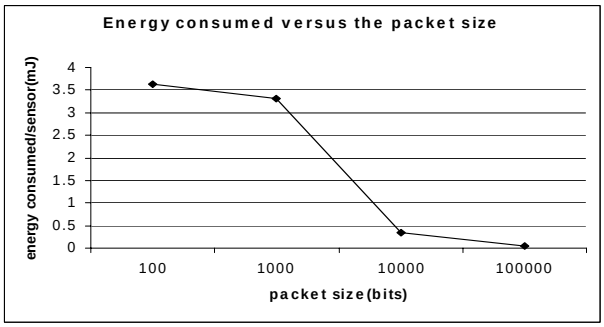


Fig. 11: Effect of packet size on transition energy

Performance impact for the inter-cluster arbitration scheme: To quantify the impact of the inter-cluster arbitration on network performance metrics and the gateway's and sensor's energy, we have considered a multi-cluster setup. The reported results in Fig. 12 are based on experiments that ran for a duration for which the gateways were able to perform five inter-cluster arbitration cycles. As shown in Fig 12 the average energy consumed by each gateway in a 5 clusters setup is .018 joules. This is not significant considering the fact that gateways have abundant energy resources. The figure further shows that the average energy consumed by the gateways increases with the increase in the number of gateways. This is expected since the number of exchanged messages in the arbitration phase is proportional to the number of gateways. The energy consumed by the network when number of gateways is increased from 4 to 6 does not show an increase due to the random placement of gateways. Although the number of messages exchanged still grows when the gateways are increased from 4 to 6, the gateways are placed closer to each other in a 6 gateways scenario, reducing the energy consumed in transmissions. The computation energy has been deemed negligible compared to communication energy and thus has not been considered in the measurements 22.

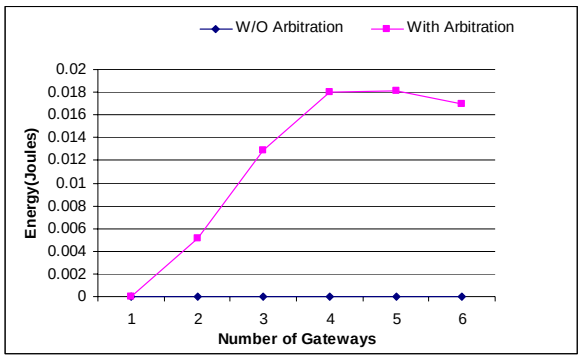


Fig. 12: Average gateway energy consumed in performing the inter-cluster medium arbitration

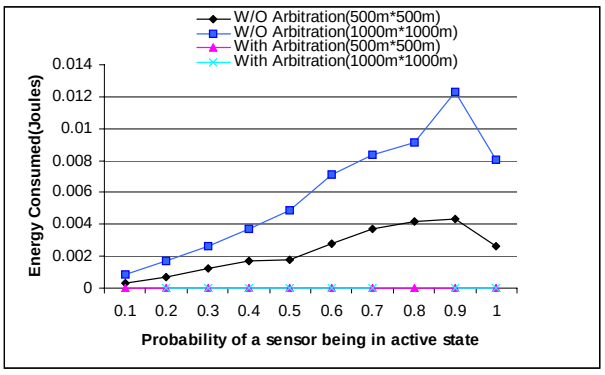


Fig. 13: Average excess energy consumed by sensors due to collisions

Fig. 13 investigates the impact of collisions on consumed sensor's energy with and without the arbitration. The results are based on a setup of 3 clusters and 4000 sensors. Initially, increasing the number of active sensors increases the number of collisions, leading to an increased energy wastage. However, such rise in collision related energy consumption ceases when most sensors are on. The decrease is due to reduction in distances between communicating sensors, with an increase in sensor density. The transmission energy decrease due to the shorter distance between nodes and their next hop neighbors. It should

be noted that the number of collision continues to rise with the increase in the number of active sensors. However, the energy wastage due to these collisions stops rising due to the reduced sensor-sensor transmission energy. On the other hand, when inter-cluster arbitration is employed, the collisions are completely eliminated. The figure also shows the energy consumed is less when the deployment area is small. This is again because of the smaller transmission distances in the smaller region.

Since the inter-cluster arbitration scheme may insert vacant slots in the schedule, the average delay per packet can increase. If this delay is large the proposed scheme becomes ineffective. Therefore we have measured the delay with and without arbitration for two scenarios. In the first scenario we kept the number of sensors constant (= 500) and the probability of a sensor being on was taken as 1. As indicated in Fig. 14, as the number of gateways increases the delay experienced by the packets in both schemes decreases. This happens because increasing the number of gateways decreases the number of sensors per cluster and therefore the tree sizes decrease. Consequently the number of hops (tree depth) also decreases. As seen from Fig. 14, on an average the delay is 0-10% more than that experienced by packets without arbitration and in the worst-case it is 27% more than those experienced when the arbitration scheme is not employed. One interesting fact to note is that, increasing the number of gateways tends to mute the impact of the inter-cluster arbitration on the average packet delay. This is because as the number of gateways increases the trees sizes decrease. Therefore even if some trees leads to extension in frame size that extension is not much in comparison to the scenario when the arbitration scheme is not employed.

The second scenario fixes the number of gateways and studies the variation in delay as the number of sensors increases. As shown in Fig. 15 there is a sharp increase in delay for both schemes, as the number of sensors increases. For smaller number of sensors both schemes perform almost identically. The gap between delays experienced by the packets in the two schemes widens for larger number of sensors. For up to 400 sensors the delay experienced by the arbitration scheme is 0-1% more than that experienced without employing the arbitration scheme. However in the case of 800 sensors the delay experienced by arbitration scheme is 28% more. This is because the potential of collisions increase for large number of sensors and therefore, the arbitration scheme inserts more vacant slots in order to avoid collisions**.** Figures 14 and 15 present the results of experiments conducted when all the sensors were transmitting data. Therefore these figures also reflect the scalability of the proposed scheme under heavy traffic conditions.

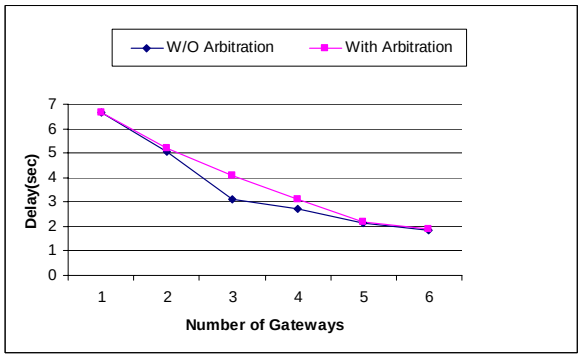


Fig. 14: Avg. delay/packet (with varying gateways)

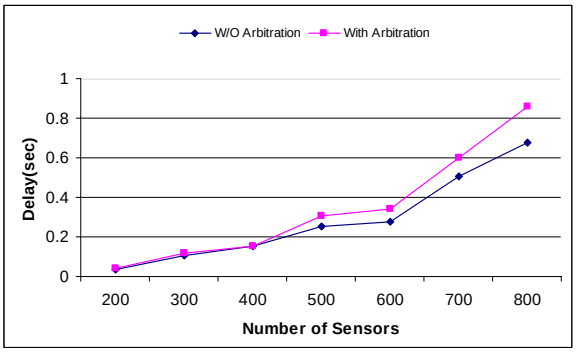


Fig. 15: Avg. delay per packet (with varying sensors)

Fig. 16 captures the throughput achieved by both schemes. In these experiments we have assumed reliable communication among the sensors and that there is no packet drops due to any reason other than collisions. The arbitration-based scheme always achieves 100% throughput. However the throughput for the scheme without arbitration fluctuates between lower bounds. Such fluctuation is due to the variation in the number of collisions, which depends upon the number and location of active sensors at a particular time. When multiple sensors in close proximity are active, the collision rate increases. On average a gain of 10% in throughput can be achieved using our inter-cluster collision avoidance technique.

Fig. 17 shows the variation in the delay experienced by packets for different levels of network traffic controlled by changing the probability of the sensors being active. The experiment is based on 400 sensors partitioned among 3 clusters. The results show that as the probability of the sensor being in active state increases the delay also increases. This is expected since the number of slots in the schedule will increase to accommodate more data sources. It is worth noting that the delay increase resulting from the employment of the inter-cluster arbitration scheme is not very large. At a low rate of data generation, small probability for a sensor being active, the inter-cluster arbitration could successfully handle most collisions without extending the delay. Even in the worst-case when all the sensors are on, the average delay a packet experiences is only 8% more than that experienced when arbitration is not employed. The results confirm that the advantages of the inter-cluster collision avoidance scheme can be achieved with minimal impact on packet delay.

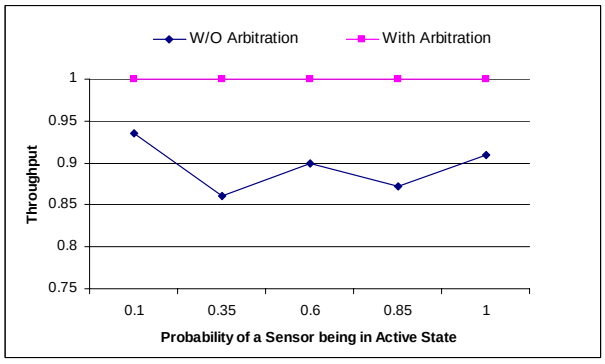


Fig. 16: Average throughput across the network

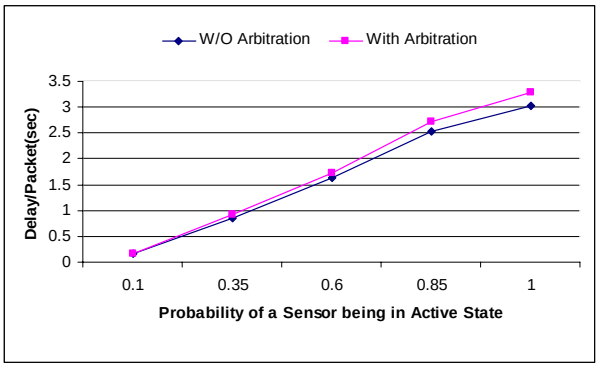


Fig. 17: Relationship between average delay per packet and the number of active sensors

Fig. 18 depicts the impact of restricting the frame size, on the arbitration scheme. This experiment was conducted in a setup consisting of 3 gateways and 800 sensors. The frame size was assumed to be 1 second. The arbitration algorithm was able to successfully include all the trees in the given frame size as long as the probability of the sensor's being on was less than .8. At higher probabilities some of the slots could not be included in the frame and had to be inserted into the vacant slots generated by inter-cluster avoidance algorithm. However, even in the worst-case only 12% of the schedules had to be modified. At probabilities higher than .95 slots did not fit in to the frame size even without applying the arbitration scheme. These results demonstrate that the proposed algorithm is able to produce efficient schedules across the network, with only a slight decrease in performance under heavy load condition.

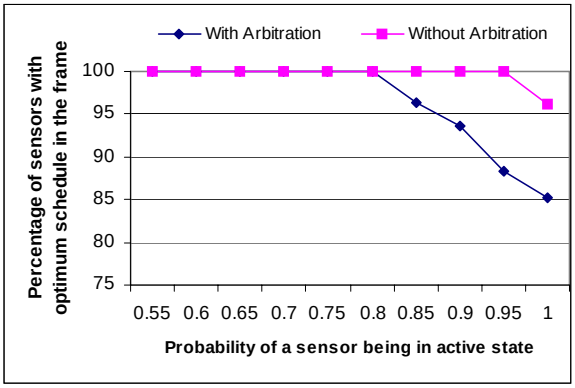


Fig. 18: Impact of the inter-cluster collision avoidance on the frame size

5 Conclusion

Wireless sensor networks have been drawing increased attention in recent years. Sensors in such systems are typically disposable and expected to last until their energy drains. Therefore, energy is a very scarce resource for such sensor systems and has to be managed wisely in order to extend the life of the sensors for the duration of a particular mission. Time based operation of sensor networks can conserve sensor's energy since it allows sensors to switch to low energy sleep mode while not transmitting and receiving messages.

In this paper we have presented an energy efficient, scalable and collision free MAC layer protocol for sensor networks. Scalability is achieved through network clustering. The approach promotes time-based arbitration of medium access in order to limit signal interference among the transmission of sensors. Time slots are optimally assigned to communicating sensors within the cluster to achieve efficient utilization of the energy resources. The presented slot assignment mechanism conserves sensor's energy by minimizing the number of transition between active and sleep modes and the duration in which active sensors are idle. Moreover, the slot assignment mechanism observes the buffering limitation at sensor's node and prevents packet drop. In addition, we have proposed an arbitration scheme that prevents collisions among the transmission of sensors that belong to different clusters. The proposed MAC protocol has been validated through simulation and shown to have positive impact on energy consumption and other contemporary network performance metrics.

References

1. I. F. Akyildiz et al.,Wireless sensor networks: a survey. *Computer Network.* March 2002; 38: 393-422.
2. D. Estrin, et al., Next Century Challenges: Scalable Coordination in Sensor Networks. *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99);* August 1999; Seattle: Washington.
3. G. J. Pottie and W. J. Kaiser, Wireless integrated network sensors. *Communications of the ACM*; May 2000; 43[5]; 51 – 58.

4. K. Sohrabi, et al., Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*; October 2000; 7[5]; 16-27.
5. R. Min, et al., Low Power Wireless Sensor Networks. *Proceedings of International Conference on VLSI Design*; January 2001; Bangalore, India.
6. J.M. Rabaey, et al., PicoRadio supports ad hoc ultra low power wireless networking. *IEEE Computer*;July 2000; 33: 42-48.
7. A. Woo and D. Culler, A transmission control scheme for medium access in sensor networks. *Proceedings of the 7th ACM Mobile Computing and Communication (MobiCom 2001)*;July 2001; Rome, Italy.
8. E. Shih, et al., Physical Layer Driven Algorithm and Protocol Design for Energy-Efficient Wireless Sensor Networks. *Proceedings of the 7th ACM Mobile Computing and Communication (MobiCom 2001)*; July 2001; Rome, Italy.
9. C-K. Toh, Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks. *IEEE Communications Magazine*; June 2001.
10. J.-H. Chang, L. Tassiulas, Energy Conserving Routing in Wireless Ad-hoc Networks. *Proceedings of the 19th International Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000),* 2000.
11. C. Röhl, H. Woesner, and A. Wolisz, A Short Look on Power Saving Mechanisms in the Wireless LAN Standard Draft IEEE 802.11. *Proceedings of the 6th WINLAB Workshop on third generation Wireless Systems*; March 1997; New Brunswick, New Jersey.
12. S. Singh and C.S. Raghavendra, PAMAS: Power Aware Multi-Access protocol with Signaling for Ad Hoc Networks; *ACM Computer Communications Review*; July1998.
13. P. Havinga, G. Smit, Energy-efficient TDMA medium access control protocol scheduling. *Proceedings of the Asian International Mobile Computing Conference (AMOC 2000);* November 2000.
14. G. Gupta, M. Younis, Fault-Tolerant Clustering of Wireless Sensor Networks. *Proceedings of the IEEE Wireless Communication and Networks Conference (WCNC 2003);* March 2003; New Orleans, Louisiana.
15. G. Gupta, M. Younis, Load-Balanced Clustering in Wireless Sensor Networks. *Proceedings of the International Conference on Communication (ICC 2003)*; May 2003; Anchorage, Alaska.
16. M. Younis, M. Youssef, K. Arisha, Energy-Aware Routing in Cluster-Based Sensor Networks. *Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS2002)*; October 2002; Fort Worth, Texas.
17. National Semiconductor Corporation, *LMX3162 Evaluation Notes and Datasheet*, April 1999.
18. Data sheet for the Acoustic Ballistic Module, SenTech Inc., http://www.sentech-acoustic.com/; April 2002.
19. V. Bhagwan, et al, MACAW: A Media Access Protocol for Wireless LANs. *Proceedings of SIGCOMM Conference*; 1994; 212-225.
20. http://grouper.ieee.org/groups/802/11/main.html; April 2002.
21. Bluetooth. http://www.bluetooth.com ; April 2002.
22. V. Raghunatham, et al., Energy-Aware Wireless Micro sensor Networks. *IEEE Signal processing magazine*; March 2002; 40-50.
23. E. Shih, et al., Energy-Efficient Link Layer for Wireless Microsensor Networks. *Proceedings of the Workshop on VLSI 2001 (WVLSI '01)*; April 2001; Orlando, Florida.
24. A. Wang, et al., Energy-Efficient Modulation and MAC for Asymmetric Microsensor Systems. *Proceedings of ISLPED 2001*; August 2001; Huntington Beach, CA.
25. W. Ye, J. Heidemann, D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks" in the *Proceedings of INFOCOM 2002*; June 2002; New York City, New York.
26. K. Arisha, M. Youssef, M. Younis, Energy-Aware TDMA-Based MAC for Sensor Networks. Proceedings of the IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT 2002); May 2002; New York City.
27. J. Elson and K. Romer, Wireless Sensor Networks a new regime of time synchronization. *Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I)*; October 2002; Princeton, New Jersey.
28. John R. Vig. Introduction to quartz frequency standards. *Technical report SLCET-TR-92-1, Army Research Laboratory, electronics and power sources directorate,* October 1992. Available at http://www.ieee-uffc.org/freqcontrol/quartz/vig/vigtoc.htm; April 2002.
29. J.B. Andresen, T.S. Rappaport, and S. Yoshida, Propagation Measurements and Models for Wireless Communications Channels. *IEEE Communications Magazine*; January 1995; 33[1]; January 1995.

30. F. Glover, Tabu Search, *Part I. ORSA Journal on Computing* 1; 1989; 190-206.

31. F. Glover, Tabu Search, *Part II. ORSA Journal on Computing 2*; 1990; 4-32.

32. A Dell'Amico, A Trubian, Applying Tabu Search to the Job-shop Scheduling Problem. *Journal of Annals of Operation Research*; 1993; 41.

33. A Hertz, Finding a Feasible Course Schedule Using Tabu Search. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*; 1992; 35.

34. A M. Laguna, A J. W. Barnes, A F. Glover, Tabu Search Methodology for a Single Machine Scheduling Problem. *Journal of International Manufacturing;* 1992; 2: 63-74.