DGRAM: A Delay Guaranteed Routing and MAC Protocol for Wireless Sensor Networks

Chilukuri Shanti, *Student Member*, *IEEE*, and Anirudha Sahoo, *Member*, *IEEE*

Abstract—This paper presents an integrated MAC and routing protocol called Delay Guaranteed Routing and MAC (DGRAM) for delay-sensitive wireless sensor network (WSN) applications. DGRAM is a TDMA-based protocol designed to provide deterministic delay guarantee in an energy-efficient manner. The design is based on slot reuse to reduce latency of a node in accessing the medium, while ensuring that the medium access is contention-free. The transmission and reception slots of nodes are carefully computed so that data is transported from the source toward the sink while the nodes could sleep at the other times to conserve energy. Thus, routes of data packets are integrated into DGRAM, i.e., there is no need for a separate routing protocol in a DGRAM network. We provide a detailed design of time slot assignment and delay analysis of the protocol. We have simulated DGRAM using ns2 simulator and compared the results with those of FlexiTP, which is another TDMA protocol that claims to provide delay guarantee, and with those of a basic TDMA MAC. Simulation results show that the delay experienced by data packets is always less than the analytical delay bound for which the protocol is designed. Also, the TDMA frame size with DGRAM is always lesser compared to that of FlexiTP, which makes the maximum possible delay much lesser than that of FlexiTP. The average delay experienced by packets and the average total energy spent in the network are much lesser in a network using DGRAM than that using FlexiTP or the basic TDMA MAC.

Index Terms—WSNs, delay guarantee, energy efficiency, MAC, TDMA.

1 Introduction

Wireless Sensor Networks (WSNs) are an emerging technology with a wide range of potential applications such as environment monitoring, earthquake detection, patient monitoring systems, etc. Sensor networks are also being deployed for many military applications, such as target tracking, surveillance, and security management. WSNs typically consist of small, inexpensive, resource-constrained devices that communicate among each other using a multihop wireless network. Each node, called a sensor node, has one sensor, embedded processors, limited memory, and low-power radio, and is normally battery operated. Each sensor node is responsible for sensing a desired event locally and for relaying a remote event sensed by other sensor nodes so that the event is reported to the end user. Sensors have limited energy and they continue to operate until their energy is exhausted. Therefore, applications and protocols for WSNs should be carefully designed in an energy-efficient manner so that the lifetime of sensors can be longer. The sensing element of a sensor probes the surrounding environment. If an *interesting* event is detected, after performing signal processing of the observed data, sensors communicate these data to the sink or base station using a radio link. This communication happens in a single or multihop fashion depending on the location of the sensing node. The node has to access the medium and then transmit the data. Thus, in a distributed system like a WSN, medium access control (MAC) protocol plays an important role. MAC protocols can be broadly divided into two categories: contention-based and TDMA-based. In contention-based MAC, the nodes can transmit without having any predetermined time assigned to them. Thus, this may result in collision. Hence, the protocol should provide mechanism for resolving collision. TDMA-based protocols are collision-free because each node has a designated time slot in which only that particular node transmits. As stated before, these MAC protocols should be energy efficient. In addition, if the sensor network is to be used for real-time applications, the MAC protocol should provide QoS (e.g., delay) guarantee.

In this paper, we propose a TDMA-based energy-efficient integrated MAC and routing protocol, called Delay Guaranteed Routing and MAC (DGRAM) protocol, which provides deterministic delay guarantee. Traditional TDMA MAC protocols suffer from high latency. Most of them like [1] consider a centralized slot allocation based on graph-coloring approach to reuse slots beyond two-hop neighbors. However, this approach is not scalable and requires slot allocation messages to be passed by the base station, resulting in wastage of energy. FlexiTP reported in [2] is a TDMA MAC protocol that has a loose slot structure and claims guaranteed end-to-end data delivery. In this protocol, the nodes run a neighbor discovery phase and then slot assignment is done so that data flow toward the sink. Both the neighbor discovery and slot assignment phases require message passing using CSMA/CA. On the other hand, DGRAM requires a short beacon exchange phase for gathering node location information. It then uses slot reuse technique to reduce latency between two successive medium accesses by a sensor node, with a slot allocation strategy that does not require exchange of control messages, which makes the deployment *self-configuring*. DGRAM only requires a sensor network to be deployed with uniform

• *The authors are with the Department of Computer Science and Engineering, IIT Bombay, Powai, Mumbai-400076, India. E-mail: {shanti, sahoo}@cse.iitb.ac.in.*

Manuscript received 9 Jan. 2009; revised 13 July 2009; accepted 23 Jan. 2010; published online 1 June 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-01-0012. Digital Object Identifier no. 10.1109/TMC.2010.107.

node density. Each node runs a short beacon exchange phase to learn about the topology of the network. Data packets are then transmitted/received following a logical topology. While traditional TDMA MACs require a routing protocol to run on top of them, DGRAM has the routing mechanism built into the MAC, using coordinated sleep and wake-up cycles. Further, DGRAM allows sensors to go to sleep when they are not communicating (neither transmitting nor receiving), and hence, it conserves energy. Since DGRAM can provide delay guarantee, it is suitable for real-time applications like detection of radioactive radiation, earthquake, etc. We present the method by which time slots are assigned to sensor nodes and show how the slots are reused by nodes that are noninterfering. Then we present the delay analysis of DGRAM to show that the delay is bounded in DGRAM. Routing of data from source to sink is integrated into DGRAM by carefully designing transmission and reception slots of nodes. When a node in an outer tier transmits in its transmission slots, certain node in the inner tier wakes up to receive the data in those slots (which are its receive slots). This enables the flow of data from source toward the sink. Thus, a separate routing protocol is not required for DGRAM. This indirectly saves energy, which otherwise would have been expended in determining the route of packets from source to sink. We present our simulation results, which show that our analytical delay bound is always guaranteed by the protocol and that there is no packet loss as long as the event rate is below the designed event rate. We compare the performance of DGRAM with FlexiTP, which is also a TDMA protocol that claims to provide end-to-end delivery guarantee and energy efficiency and show that DGRAM outperforms FlexiTP in terms of delay, energy consumption, and number of packets meeting deadline. We also compare DGRAM with a basic TDMA MAC that is bundled with ns2. It is seen that DGRAM outperforms FlexiTP and the basic TDMA MAC in all the above aspects for different network sizes and event rates.

The rest of the paper is organized as follows: In Section 2, we present some related work in the area of MAC protocols for WSNs. Section 3 describes the topology in which DGRAM is deployed, location of sensor nodes in DGRAM network and a few assumptions made in designing DGRAM. A detailed description of TDMA time slots assignment at different hierarchy is presented in Section 4. Section 5 analyzes the energy consumption by nodes running DGRAM. We present our simulation results in Section 6. Finally, we conclude the paper in Section 7.

2 Related Work

WSNs generally use MAC protocols that are either TDMA-based or are contention-based. In contention-based protocols, multiple nodes may access the medium simultaneously, resulting in collision. The MAC protocol then provides a mechanism to resolve collision. Akyildiz et al. [3] give an excellent overview of the various MAC protocols for sensor networks. The IEEE 802.11 distributed coordination function (DCF) is one such protocol. Woo and Culler [4] have studied different configurations of CSMA and proposed an adaptive rate control mechanism to achieve fair bandwidth allocation to all nodes. S-MAC [5] is a popular MAC protocol for WSNs that conserves energy by having listen and sleep cycles so that idle listening time is minimized. S-MAC conserves energy, but has high latency. Several extensions and modifications to S-MAC like WiseMAC [6] and DMAC [7] have been proposed to improve the protocol performance. Cohen and Kapchits [8] propose an algorithm to determine the wake-up frequency of nodes depending on their proximity to the sink so that the overall energy spent and delay are minimized. However, it does not give a schedule for the wake-up cycles.

TDMA-based protocols are contention-free protocols in which sensor nodes transmit only in their assigned time slot. Sohrabi and Pottie proposed a self-organizing MAC for sensor networks in which each node maintains a TDMA-like frame called the superframe [9]. Interference between adjacent links is avoided by using FDMA and CDMA in potentially interfering links. Chandrakasan et al. [10] propose a TDMA protocol in which clusters are formed and each cluster elects a regular sensing node as a cluster head. All the nodes in the network are assumed to have enough radio power to communicate with the base station directly. Cluster heads communicate directly with the base station and other nodes communicate directly only with the cluster head. The slot assignment is randomized. Wu and Biswas [11] present a self-reorganizing slot allocation mechanism for TDMA-based MAC in multicluster sensor networks. The primary contribution of the paper is to demonstrate that with adaptive slot allocation, it is possible to reduce intercluster interference under low load conditions. The second contribution is the design of a feedback-based adaptive allocation reorganization protocol that significantly reduces the intercluster interferences without relying on any global synchronization mechanism. In [12], the authors propose a new MAC protocol referred to as DEE-MAC, which reduces energy consumption by making the idle nodes sleep to reduce idle listening using synchronization at cluster heads. Each cluster is dynamically formed and all nodes contend for the position of cluster head. Nodes can join or leave the cluster any time they want to. RT-Link [1] focuses on reducing the communication delay between a node and the base station. It uses a centralized slot computing mechanism, using the distance-k node coloring approach (slot scheduling), to minimize the number of collisions along each transmission hop in a multihop wireless network. It also proposes a delay-sensitive schedule. Logical connectivity graph formation and slot assignment are done by the base station centrally. FlexiTP [2] is another TDMA protocol, where the nodes run a neighbor discovery algorithm after which they mark transmission, forwarding, and receive slots. The main goal is to reassign slots upon the occurrence of a fault, which is detected by the absence of a packet to be received.

There have been some recent papers [13], [14] that study real-time routing in sensor networks. However, their focus is only on reducing the routing delay, ignoring the delay at the MAC level. In [13], the authors discuss the design of a stateless routing protocol for real-time communication called SPEED. While SPEED is highly efficient and scalable, it is limited to routing and does not take into account the MAC layer delays. Though the protocol tries to find the best possible route along which data have to be forwarded in order to minimize delay, the delay at the MAC layer can be

quite large as it is built on the 802.11 MAC protocol. The number of nodes considered by the authors is large, but simulation and experimental results are presented for a very few number of flows. Whether the protocol can guarantee end-to-end delay bound when all nodes generate packets (e.g., in case of a fire or biohazard) is doubtful.

TDMA protocols need highly accurate time synchronization for correct operation. Out-of-band, hardware-based synchronization has been suggested for this purpose in [1]. Other mechanisms of time synchronization for TDMA protocols are presented in [15], [16], [17]. Kulkarni et al. [18] use a logical tier structure similar to that of DGRAM, but it is contention-based and may not provide delay guarantee when there is a burst of data. In [19], we discussed a routing and MAC scheme with a logical tier structure to provide delay guarantee and compared it with [5]. This work, however, requires data aggregation at each node as it does not allot any slots for forwarding data packets.

DGRAM requires each node to know its position relative to the sink. This can be achieved by programming the nodes with their location at the time of deployment or using any localization algorithm reported in the literature. There are many distributed algorithms in the literature to find the coordinates of the nodes in a sensor network. Many localization systems depend on having direct distance estimates to globally accessible beacons like the Global Positioning System (GPS). Recently, there has been some research in localizing a wireless sensor network, where there are no globally accessible beacons. Savvides et al. [20] describe a distributed algorithm that recursively infers the positions of sensors with unknown positions from a set of sensors with known positions, using intersensor distance estimates. While algorithms like this need seed nodes that know their position, algorithms that do not need such seed nodes have also been proposed. Priyantha et al. [21] propose a localization algorithm that does not need any anchor nodes. Khan et al. [22] describe a localization algorithm that requires a single anchor in the network. For our protocol, this single anchor can be the sink.

3 Deployment of the WSN

3.1 Topology

We assume a circular sensing area and that the sensing area has a base station or sink, which is wired and is not power-constrained. The sensor nodes (henceforth referred to as just *nodes*) sense the event of interest and transmit these data to the sink using a MAC described in the paper. The nodes are deployed with uniform density all around the sink, with the sink at the center of the area as shown in Fig. 1. Depending on the distance of a node from the sink and the transmission range of the nodes, data have to traverse single or multiple hops before being received by the sink.

Our protocol is designed with the following assumptions:

- The nodes and the sink are stationary.
- Each node knows its location relative to the sink.
- Each node is programmed with the total number of nodes in the network.
- The sink is at the center of the circular sensing area.
- The clocks of sensor nodes are synchronized by using out-of-band time synchronization. A similar mechanism as used in [1] may be used for this purpose.

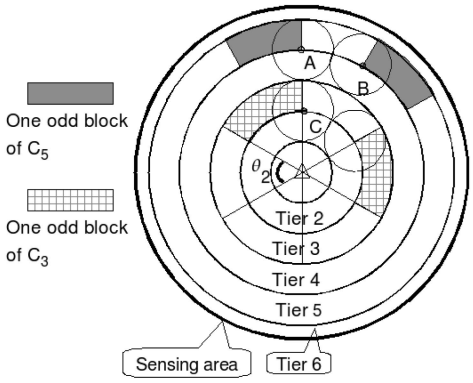

Fig. 1. Depiction of tiers and blocks (for $\alpha = 1.0$).

- The transmission and interference ranges of the nodes are generally not exactly circular. Hence, we consider the maximum interference radius and the minimum transmission radius when the transmission and reception cycles are calculated (details are given in Sections 4.7.1 and 4.7.2). This ensures that no packets are lost.

3.2 Location of the Sensors

DGRAM is most optimal when there is uniform density of sensors in the sensing area. One of the main assumptions made above is that each node knows its position with respect to the sink. As mentioned before, this can either be achieved by manually programming the sensors with their coordinates or by using some known distributed algorithm [20], [21], [22]. The position of a node is represented in rectangular coordinates with the sink as the origin.

4 Time Slot Assignment

In this section, we present a detailed discussion of time slot assignment in DGRAM. The notations used in the discussion are given in Table 1.

4.1 Logical Organization of Nodes

To facilitate multihop communication and time slot reuse, the sensing area is divided into tiers and blocks. The sensing area is organized into tiers, based on the radial distance of different nodes from the sink. Then each tier is divided into blocks based on the angular distance of nodes (measured in clockwise direction) from the sink with respect to the geographical North axis passing through the sink. This is depicted in Fig. 1. Each tier is of radial width αR. The parameter α determines the minimum node density in the area for the network to remain connected (more details on α are given in Section 4.5). The tier closest to the sink is the first tier and is identified as C_1. There are H number of tiers in the sensing area. Thus, the tier farthest from the sink is identified as C_H. The division of the network into tiers helps in multihop transmission of data toward the sink and helps in slot reuse. The division into blocks also facilitates reuse of time slots. The outermost tier C_H may not have the full tier width of αR. For example, if the deployment area has a radius of 150 m, i.e., $RADIUS$ is 150 m and R and α are 100 m and 0.51, respectively, the network would have two full tiers (that cover a radius of $2 * 0.51 * 100\ \mathrm{m} = 102\ \mathrm{m}$) and an

TABLE 1
Explanation of Various Notations

Notation	Definition
$RADIUS$	Radius of the circular deployment area
λ	Uniform node density expressed in nodes per unit area
M	total number of nodes in the deployment area
Ψ	Table having M entries built by every node. The M entries carry location information of all the nodes.
d_{worst}	worst case delay of an event from the time it occurred to the time it is delivered to the sink
p_i	number of nodes in the tier C_i
q_{ij}	number of nodes in j^{th} block of tier C_i
R	Transmission range of a node
H	Total number of tiers in the sensing area
F	Number of inner tiers that can transmit to the sink directly
N	Number of subframes in a superframe
α	Ratio of radial width of a tier to the transmission radius of a node ($\alpha \leq 1$)
I	Interference range of a node
β	Ratio of interference range to the transmission radius of a node ($\beta \geq 1$)
C_i	ID of the i^{th} tier
Z_i	Number of blocks in the tier C_i
θ_i	Angle subtended (in radians) by a block in the tier C_i at the sink
a_i	Number of transmission slots per node in tier C_i
B_{ij}	ID of the j^{th} block in the tier C_i
S_i	Total number of slots assigned to the i^{th} subframe of a superframe
S_i'	The size of the subframe for the i^{th} tier (as per calculation)
D_{ijk}	Radial distance of node k belonging to tier C_i and block B_{ij} from the sink
ϕ_{ijk}	Angular distance of node k belonging to tier C_i block B_{ij} from the sink with reference to geographical North passing through the sink (in clockwise direction)
G	Number of tiers that are not divided into blocks
P_i	number of slots allotted in a superframe for nodes in tiers outer to C_i
Q_{ij}	number of slots allotted for nodes in blocks up to $(j-1)$ in tier C_i
U_{ijk}	number of slots allotted for nodes up to $(k-1)$ in block B_{ij} of tier C_i
T	Number of slots in a superframe

outermost tier that has a tier width of the remaining radial distance to be covered, i.e., $150\ \mathrm{m} - 102\ \mathrm{m} = 48\ \mathrm{m}$. Based on the position of the node and the sink, the radial distance of the node from the sink is calculated by each node. Based on this distance and α, each node calculates the tier C_i to which it belongs. If the radial distance is between $(i-1)\alpha R$ and $i\alpha R$ ($i > 0$), the node determines itself to be in the ith tier C_i. Data from any node in this tier flow radially inward and hop from outer tier to inner tier and reach the sink. We merge all the innermost tiers that can reach the sink directly into a single tier C_1 because nodes in these tiers can communicate with the sink in a single hop. The number of innermost tiers that can be merged into a single tier is F, where

$$F = \left\lfloor \frac{1}{\alpha} \right\rfloor. \tag{1}$$

Note that the tier C_1 has a radius of $F\alpha R$. Hence, data from tier C_i reach the sink in $(i - F + 1)$ hops if the node's radial distance from the sink is greater than $F\alpha R$ and in a single hop if the radial distance is less than or equal to $F\alpha R$.

After merging of the innermost F tiers, the number of tiers into which a network of radius $RADIUS$ is divided is given by

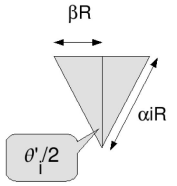

Fig. 2. Calculation of θ' for the ith tier ($i > G$).

$$H = \begin{cases} \left\lceil \dfrac{RADIUS}{\alpha R} \right\rceil - F + 1, & \text{if } \left\lceil \dfrac{RADIUS}{\alpha R} \right\rceil > F, \\ 1, & \text{otherwise.} \end{cases}$$

Each tier is further divided into blocks for concurrent transmission of data within a tier. A block is an angular area covered by the interference range of a node in a given tier (see Fig. 1). The division of a tier into blocks helps in slot reuse within a tier, which, in turn, enables concurrent transmission within a tier. A node in the extreme right of block B_{ij} can interfere with transmission by a node in block $B_{i(j+1)}$, but cannot interfere with a node to the extreme left of the block $B_{i(j+2)}$. Thus, nodes in alternate blocks can transmit in the same slots without interfering with each other. For example, nodes A and B shown in Fig. 1 can transmit simultaneously. This calculation is done based on the angle θ_i that each block in tier C_i subtends at the center (the sink). To keep the number of nodes per block approximately the same, the angle θ_i is made smaller for tiers farther away from the sink. As shown in Fig. 2, this angle for any tier C_i is calculated by the node as follows:

$$sin(\theta_i'/2) = \frac{\beta R}{\alpha i R} = \frac{\beta}{\alpha i}. \tag{2}$$

Since $0 \leq sin(\theta_i'/2) \leq 1$, from (2), we have $0 \leq \frac{\beta}{\alpha i} \leq 1$. The first condition $0 \leq \frac{\beta}{\alpha i}$ is always satisfied, since β, α, and i all take positive values. For the second condition $\frac{\beta}{\alpha i} \leq 1$ to be true, $i \geq \frac{\beta}{\alpha}$. Hence, all tiers with index less than $\frac{\beta}{\alpha}$ are not divided into blocks. Let G' be the number of tiers that are not divided into blocks before merging the first F tiers into a single tier. Hence,

$$G' = \left\lceil \frac{\beta}{\alpha} \right\rceil.$$

Since we merge the first F tiers into a single tier, the effective number of tiers that are not divided into blocks is G, where

$$G = G' - F + 1.$$

Note that G' is always $\geq F$, since $\beta \geq 1.0$. Hence, G is always positive and the first (merged) tier is never divided into blocks. In the rest of the paper, whenever we refer to tier C_i, the tier index i is the index of the tier after merging the first F tiers into a single tier.

Solving for θ_i', we get

$$\theta_i' = sin^{-1} \frac{2\beta\sqrt{(\alpha^2(i+F-1)^2 - \beta^2)}}{\alpha^2(i+F-1)^2}. \tag{3}$$

Since we allow alternate blocks to use the same half of a subframe (explained in Section 4.2), the number of blocks should be even. However, the number of blocks in tier C_i

depends on the angle θ'_i of that tier and may not always be an even number. To ensure that the number of blocks Z_i in the tier C_i is an even number, blocks are assigned in the half circle region, i.e., the angle for each block is assigned from π. Thus,

$$Z_i = 2\left\lfloor \frac{\pi}{\theta'_i} \right\rfloor. \quad (4)$$

Note that we have used the floor function in the above equation so that the area of adjacent blocks will be greater than or equal to that required to make them noninterfering. Keeping the above constraints in view and using (3) and (4), the angle subtended by a block at the sink is modified as follows:

$$\theta_i = \theta'_i + \frac{\frac{\pi}{\theta'_i} - \left\lfloor \frac{\pi}{\theta'_i} \right\rfloor}{Z_i/2}. \quad (5)$$

Now, knowing the value of θ_i and the angular distance of the node from the sink, the node can find out the block to which it belongs. If the angular distance is between $(j-1)\theta_i$ and $(j\theta_i)$ $(j > 0)$, then the node belongs to jth block, B_{ij}. Thus, B_{i1} is the first block in tier C_i, which is between 0 and θ_i (in clockwise direction) with respect to geographical North passing through the sink.

The number of nodes in the ith tier is denoted by p_i and is given by

$$p_i = \sum_{j=1}^{Z_i} q_{ij}, \quad (6)$$

where q_{ij} is the number of nodes in the jth block of tier C_i and is computed while a node executes Algorithm 1.

Algorithm 1. Node_level_slot_assignment(D_{ijk}, ϕ_{ijk})

/* $rad\_distance(w)$ and $angle\_distance(w)$ are radial and angular distances, respectively, of the node (from the sink) whose location is stored in the wth entry of Ψ */
/* M is the number of nodes in the deployment area */
/* x_{ijk} and y_{ijk} are, respectively, the X and Y coordinates of the node trying to get its index in the subsubframe */
/* u, v and w are running indices */
$u = 1, v = 1, w = 1, k = 1$
/* Initialize the number of nodes in each block and tier to 0*/
while $u \leq H$ **do**
  $p_u = 0$
  while $v \leq Z_u$ **do**
    $q_{uv} = 0$
    $v = v + 1$
  end while
  $u = u + 1$
end while
while $(w \leq M)$ **do**
  $u = \left\lceil \frac{rad\_distance(w)}{\alpha R} \right\rceil$
  if $(u > F)$ **then**
    /* since innermost F tiers are merged into one, adjust the tier index accordingly */
    $u = u - F + 1$
  else
    $u = 1$
  end if
  $p_u = p_u + 1$
  $v = \left\lceil \frac{angle\_distance(w)}{\theta_u} \right\rceil$
  $q_{uv} = q_{uv} + 1$
  if $((u == i)\ AND\ (v == j))$ **then**
    if $(rad\_distance(w) < D_{ijk})$ **then**
      $k = k + 1$
    else if $(rad\_distance(w) == D_{ijk})$ **then**
      if $(angle\_distance(w) < \phi_{ijk})$ **then**
        $k = k + 1$
      end if
    end if
  end if
  $w = w + 1$
end while
/* the node now knows values of k (this is the index of the node in its tier and block), p_i and q_{ij} for all $i \in [1, H], j \in [1, Z_i]$ */

Each node needs transmission slots to relay the data received from nodes of the outer tiers and to transmit its own data. Assuming a slot size large enough to transmit a single data packet, each node thus needs as many transmission slots as the number of data packets to be forwarded by it, in addition to one slot to transmit its own data. The number of slots required per node in the ith tier for relaying the outer tier's data depends on the ratio of the number of nodes in the $(i+1)$th tier to the number of nodes in this tier and on the number of slots per node in the outer tier $(a_{(i+1)})$. Hence,

$$a_i = \begin{cases} 1 + \left\lceil \frac{p_{(i+1)}}{p_i} \right\rceil a_{i+1}, & \text{for } 1 \leq i < H, \\ 1, & \text{for } i = H. \end{cases} \quad (7)$$

The numerator in the second expression of the first part of the above equation gives the number of nodes in the $(i+1)$th tier and the denominator is the number of nodes in the ith tier. In the outermost tier C_H, a_H is one, as each node needs a slot to transmit its own data and no relay slots are needed.

4.2 Slot Assignment at the Tier and Block Level

A node in DGRAM can be in one of the two states: *active* or *sleeping*. A node is in active state in its allotted transmission slot if it has pending data to send. A node switches itself into the active state at the beginning of each of its receive slots. If it senses a preamble, it remains in active state to listen for the rest of the slot duration to receive data from its previous tier. Otherwise, it goes back to sleeping state. At all other time, the node stays in sleeping state. This careful design of active and sleeping states helps in minimizing energy consumption, while making sure that data are carried from the source node to the sink.

Slot assignment in DGRAM happens in three levels of hierarchy. At the highest level, the time frame is called the

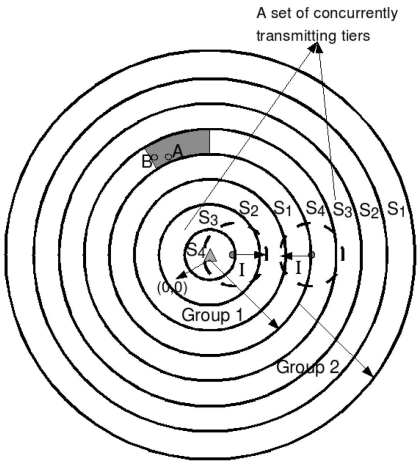

Fig. 3. Counting number of nodes in a tier.

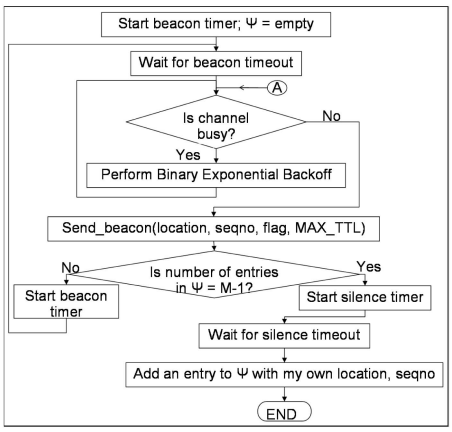

Fig. 4. Flowchart for sending periodic beacons.

superframe. The superframe is divided into subframes, which are assigned to different tiers. The subframes are reused across the tiers such that the transmissions do not interfere. Nodes in two different tiers can transmit at the same time when they are at least $2I$ apart, as shown in Fig. 3. Note that the merging of the first F tiers does not effect the calculation of N, since we consider the distance $2I$ from the edge of C_1, to take care of the worst case. Each tier (other than tier C_1) is αR wide. If N is the number of subframes, N tier widths should encompass the $2I$ length. This means that $N = 2I/\alpha R$. But we need to take care of border cases of tiers, i.e., a node on the border between Nth and $(N+1)$th tiers and belonging to $(N+1)$th tier could interfere with the first tier. Thus, taking an extra tier width into the calculation and the fact that $I = \beta R$, we have

$$N = \left\lceil \frac{2\beta}{\alpha} + 1 \right\rceil. \tag{8}$$

Thus, the tiers can be categorized into groups of N tiers each. Nodes that belong to different groups, but transmit in the same subframe, form a set of concurrently transmitting nodes. The tiers that transmit concurrently with tier C_i are

$$C_{\lceil i+j(1+2\beta/\alpha)\rceil},\ j > 0. \tag{9}$$

Thus, each tier has a subframe allotted to it. Further, each subframe is split into two halves called *subsubframe*. One half is used by all odd-numbered blocks whereas the other is used by all even-numbered blocks in the tier. This is possible because the even (odd)-numbered blocks are not within interference range of each other. Thus, slots of subsubframes are reused at the block level also. Note that reuse of slots helps in reducing the size of the superframe, which leads to reduced latency of the network. The innermost G tiers, i.e., tiers C_1 to C_G, are not divided into blocks.

4.3 Node-Level Slot Assignment

Once the subframe allocation is done for tiers and subsubframe allocation is done for blocks, the node-level slot assignment has to be done. Nodes only transmit data during the slots assigned to them in the subsubframe of the block to which they belong.

Each node goes through a short beacon exchange phase and then runs a simple algorithm (Algorithm 1) to determine its transmission time slots within the subsubframe assigned to it. To know when to transmit, the node needs to know the following:

- The organization of superframe.
- Its tier and block.
- Its index in its tier and block.

A node needs to know the number of nodes in each tier and block of the network in order to get the structure of the superframe. To obtain this information, a node needs to have location information of every other node in the network. Each node in the network goes through a short beacon exchange phase after deployment to collect location information of all other nodes in the network.

4.3.1 Beacon Exchange Phase

As mentioned before, each node finds out location of every other node in the network by broadcasting periodic beacons. In this section, we present the details of the beacon exchange phase. Every node maintains a table Ψ, which contains the location information of every other node. This table is populated during this phase. An entry in Ψ is a two-tuple of *{location, seqno}*. *seqno* field in this table is used to determine *freshness* of a received beacon with respect to a beacon received earlier. Initially, each node knows its location and the total number of nodes in the network, M, and the table Ψ is empty. Each node originates periodic beacon message as per the flowchart shown in Fig. 4. A beacon message contains four fields: *{location, seqno, flag, TTL}* and is broadcast using CSMA protocol. $flag$ field is set to 1 (by the originator) if Ψ has $(M-1)$ entries, which means that the originator node has received location information of all other nodes in the network; otherwise, it is set to 0. The TTL field is set to MAX_TTL by the originator. MAX_TTL is chosen suitably so that the message reaches all the nodes before it is dropped. Typically, MAX_TTL should be chosen little more than the diameter of the network (in terms of number of hops). When the node receives location information from all other nodes, Ψ will contain $(M-1)$ entries, and hence, it will not originate any periodic beacon (see Fig. 4). Instead, it will start a *silence timer*. During this time, this node waits for beacon from any other node. If it does not receive a beacon with $flag = 0$ before this time-out, that means all other nodes are done with this phase, and hence, this node also ends this phase.

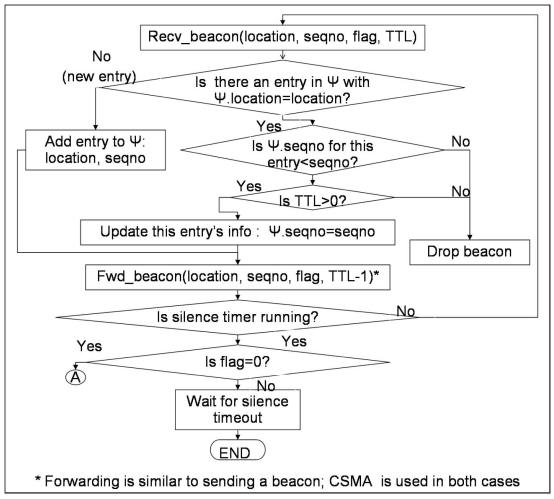

Fig. 5. Flowchart for receiving and forwarding beacons.

When a node receives a beacon message from its neighbor, it may drop it or forward it further. The flowchart for this case is shown in Fig. 5. If the location information is not there in its table Ψ, then it adds the entry and forwards the beacon after decrementing its TTL value by 1. Otherwise, if the location information carried in the beacon is already present in Ψ table, then the beacon is forwarded only if the $seqno$ is more than that in the Ψ table and the TTL value is more than 0. In other cases, the beacon message is dropped. Dropping the beacon message based on the above criteria reduces the extent of flooding of beacons and reduces the convergence time of the beacon exchange phase considerably. If the $flag$ field in the beacon is 0, then it follows a part of the flowchart in Fig. 5 (indicated by point A in the two figures). This path ensures that the node receiving this beacon sends its own beacon (since some node in the network still has not received all $(M-1)$ location information). Note that silence timer value should be long enough to ensure that no node is still transmitting beacons with $flag = 0$. This guarantees that every node gets the location information of all other nodes in the network.

4.3.2 Determination of Slots at the Node Level

A node knows its position in the deployment region in rectangular coordinates expressed as (x_{ijk}, y_{ijk}). However, the node does not initially know its tier i, block j, or index k in that tier and block. The location of a node k in tier C_i and block B_{ij} in polar coordinates is essentially an ordered pair (D_{ijk}, ϕ_{ijk}), where D_{ijk} is the radial distance of the node from the sink and ϕ_{ijk} is the angular distance of the node with respect to the geographical North (in clockwise direction) passing through the sink. The values of (D_{ijk}, ϕ_{ijk}) can easily be calculated based on the position (x_{ijk}, y_{ijk}) of the node in the deployment area and are useful in calculating the tier and block to which the node belongs. Now, the tier to which the node belongs can be calculated as follows:

$$i = \begin{cases} \left\lceil \dfrac{D_{ijk}}{\alpha R} \right\rceil - F + 1 & \text{if } \left\lceil \dfrac{D_{ijk}}{\alpha R} \right\rceil > F, \\ 1, & \text{otherwise.} \end{cases} \tag{10}$$

For example, for $F = 1$, if the tier width (αR) is 51 m, a node that has a radial distance of 10 m from the sink lies in the tier $\lceil \frac{10}{51} \rceil$, i.e., tier 1. Similarly, a node with a radial distance of 110 m from the sink lies in the tier $\lceil \frac{110}{51} \rceil$, i.e., tier 3. The width of the outermost tier may not be αR, as discussed in Section 4.1. Nevertheless, the above equation applies to all nodes in the deployment area. Further, the block to which the node belongs can be calculated as

$$j = \left\lceil \frac{\phi_{ijk}}{\theta_i} \right\rceil. \tag{11}$$

For example, if the value of θ for that tier is 1.3 radians and the node is at an angle of 2.1 radians to the geographical North, the node is in block $\lceil \frac{2.1}{1.3} \rceil$, i.e., block 2. Hence, each block includes all the nodes between its left (in clockwise direction) boundary and the right boundary, including those on the right boundary, but not those on its left boundary. The division of tiers into blocks is such that slots can be reused to the maximum extent possible without interfering transmissions. In a particular slot, no two nodes belonging to a block in a tier can transmit (in order to avoid interfering transmissions), but two nodes belonging to alternate blocks in the same tier may transmit in the same slot.

The index of a node (in its tier and block) is obtained by running Algorithm 1. A node (e.g., B in Fig. 3) running Algorithm 1 counts the number of nodes (from the entries in Ψ) that belong to each tier and block of the deployment area. Knowing this information, the node can get the organization of the superframe as discussed in Section 4.4. Since the node knows its tier (based on the radial distance from the base station) and block (based on the angular distance from the base station), it knows the subframe and subsubframe in which to transmit. To know the slots in which to transmit within this subsubframe, the node calculates its index within the subsubframe in course of the algorithm.

While running the algorithm, if the node finds another node that lies in the same block and tier as itself and has a radial distance lesser than itself from the base station, it increments the $index$ by 1 (Step 27). This is because nodes are assigned indices in the increasing order of their radial distances from the sink. In case of a tie in radial distance, the node with lower angular distance gets lower index. By the end of the algorithm, node B knows its index within its block and tier (given by k). It then uses this information to mark its transmission slots. For example, in Fig. 3, node B runs the algorithm and finds that node A has lesser radial distance than itself from the sink. Hence, it marks its index to be one greater than that of A. If no other node in the block has lesser radial distance than B, this would become the final index of B. Since there cannot be two nodes with the same radial and angular distances from the sink, this method gives an unique index to each node. Based on this unique index, the node chooses unique transmission slots in the subsubframe of that block.

Each node in the network runs Algorithm 1. After running this algorithm, each node knows the following:

- The number of nodes in each tier and block in the entire network.
- Its index within its tier and block.

Using these values, it can calculate the number of slots per node in each tier, i.e., a_i, as per (7). By now, the node has

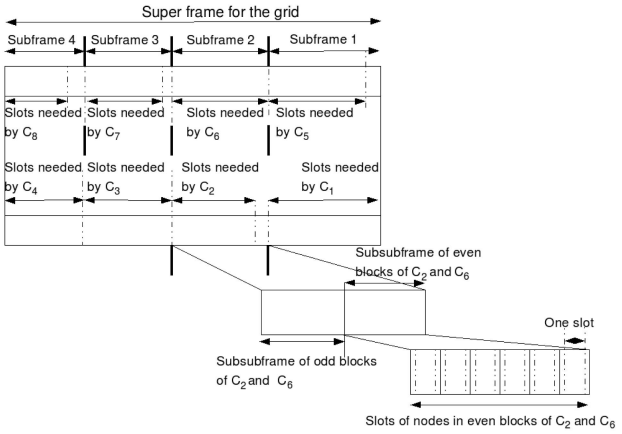

Fig. 6. Time slot assignment at different hierarchy.

enough information to calculate the size of each subframe (S_i), and hence, the size of the superframe (T), as discussed in Section 4.4. Though each node independently runs the algorithm, all nodes obtain the same values of subframe and superframe sizes for the network. Then, the node can calculate its transmission and reception slots as discussed in Sections 4.7.1 and 4.7.2.

Fig. 6 shows the time slot assignment at various levels when there are eight tiers. Note that this algorithm and the subsequent calculation of the transmission and reception slots are done only once and do not need any message passing.

4.4 Calculation of Size of Superframe

In DGRAM, we make sure that there is no queuing in the nodes. This makes it easier to provide deterministic delay guarantee. Since the nodes are deployed with uniform density and the tiers and blocks are angular, the number of nodes in two blocks in the ith tier may not be the same. Hence, the size of the subsubframe of the ith tier is determined by the block that has the maximum number of nodes. Also, all odd blocks of the ith tier transmit in the same subsubframe and all the even blocks of the ith tier transmit in the same subsubframe. Each node in the ith tier has a_i slots, where a_i is given by (7). Hence, the subframe length for an inner tier C_i is given by

$$S_i^{'} = 2 \,.\, a_i \,.\, \max_l(q_{il}),\ l \in [1, Z_i],\ i \in (G, H]. \tag{12}$$

Note that since the outermost tier C_H has one slot per node, a_H is one. Since tiers C_1-C_G have a single block, the corresponding parameters for these tiers are given by

$$\begin{aligned} S_1' &= a_1 \,.\, q_{11} \\ S_2' &= a_2 \,.\, q_{21} \\ &\vdots \quad \vdots \\ S_G' &= a_G \,.\, q_{G1}. \end{aligned}$$

It is obvious from the above discussion that the numbers of slots allotted for concurrent tiers from different groups are not equal. Hence, we consider the largest number of slots required by a tier in a concurrent group of tiers to be the subframe size for that set of tiers. For example, for a network with eight tiers and two groups, if C_1, C_2, C_3, and C_4 belong to the first group, C_5, C_6, C_7, and C_8 form the next group, and C_1 and C_5 transmit concurrently and form part of a set of tiers that transmit concurrently. Similarly, C_2 and C_6 transmit concurrently, C_3 and C_7 transmit concurrently, and C_4 and C_8 transmit concurrently. The size of the subframe in which C_1 and C_5 transmit should be the larger of number of transmission slots needed by C_1 (S_1') and C_5 (S_5'). Likewise, if C_2 has a larger number of slots compared to C_6 with which it transmits concurrently, the size of the second subframe is taken to be that of C_2. Let the ith subframe of the superframe be denoted by S_i. Thus,

$$S_i = \max_l(S'_{i+l.N}), \quad i \in [1, N], \quad l \in \left[0, \left\lceil \frac{H}{N} - 1 \right\rceil\right]. \tag{13}$$

Hence, the size of the subframe is decided by the tier that has the maximum number of slots among a set of tiers that transmits concurrently in that subframe. If C_x and C_y are a set of tiers that transmits concurrently and S'_x and S'_y are their respective subframe sizes such that $S'_x > S'_y$, the subframe size for both these tiers is taken to be S'_x. This may, however, result in some slots being left over after all the nodes of C_y have been assigned slots. All the nodes of C_y sleep during these slots so that no energy is wasted during these slots. Note that these extra slots are necessary in the subframe because some nodes in C_x transmit during these slots. DGRAM can work with any topology, as long as connectivity is assured as per the discussion in Section 4.5. However, uniform density results in minimum wastage of slots due to unequal subframe size in a set of concurrently transmitting tiers. N is the number of tiers in each group and is given by (8). Thus, the number of slots in a superframe is given by

$$T = \sum_{i=1}^{N} S_i. \tag{14}$$

As an example, the division of slots in T for an area with eight tiers, with $\beta = \alpha = 1$, is shown in Fig. 6.

4.5 Dimensioning of DGRAM Parameters

For a network with given β and H, the value of T depends on the node density λ. The minimum node density that can keep the network connected, in turn, depends on the value of α, as described by Kulkarni et al. [18]. The authors [18] study the effect of α for a given λ on the connectivity, which is the number of nodes that can potentially relay data from a given node. In Fig. 7, consider node A transmitting data. In this figure, we show two different cases of overlap region, by showing only half of the deployment area on either side of the vertical line. As per DGRAM, the only nodes that can relay data from this node are those in the shaded region. The shaded area is the region of overlap of two circles (in the left half of the figure, for $\alpha \geq 0.5$): The first is a circle centered at the node with radius R and the second is a tier circle centered at the sink with radius $\alpha(n-1)R$. Let this area be $\zeta_{(n-1)}$. The value of α has to be less than unity for a node in the nth tier to be able to relay data to the $(n-1)$th tier. The number of nodes in this region of overlap is a measure of connectivity of the network, since only nodes in this region will ultimately relay data.

In Fig. 7, the depicted region of overlap is the minimum area possible for all nodes in tier n, since the sender node is at

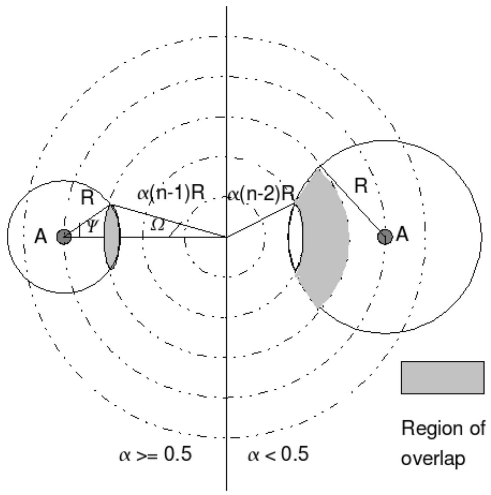

Fig. 7. Minimum density for connectivity of the network.

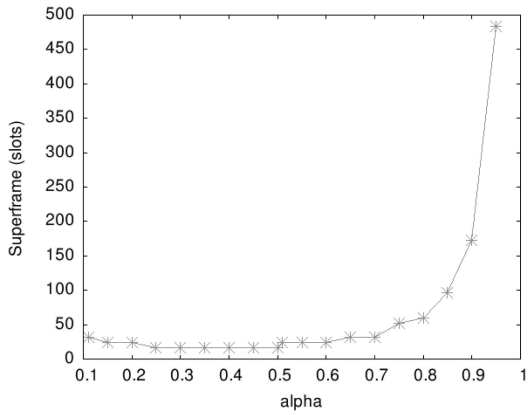

Fig. 8. Effect of α on the superframe size.

the edge of the nth tier. This area is minimum with respect to n when n is as small as possible, i.e., at $n_{min} = \lfloor \frac{1}{\alpha} + 1 \rfloor$. This is because all nodes from the first tier to the $\lfloor \frac{1}{\alpha} \rfloor$th tier are within a distance R from the sink and do not depend on the connectivity since they can communicate directly with the sink. The $\lfloor \frac{1}{\alpha} + 1 \rfloor$th tier is the first tier that would require multihop transmission. The shaded area (of Fig. 7) will be minimum for this tier among all the tiers that need multihop transmission. It is because similar shaded area for higher tiers will be larger, since their circles are larger. By the cosine rule of triangles, the area of the shaded region (in the left half of the figure, for $\alpha \geq 0.5$) is given by

$$\zeta_{(n_{min}-1)} = R^2(\psi + (n_{min} - 1)^2\alpha^2\Omega - n_{min}\alpha sin\psi).$$

When $\alpha < 0.5$, the area of overlap is as shown on the right half of Fig. 7. Let $\zeta_{(n_{min}-2)}$ be the area of intersection of the two circles: the first is a circle centered at the node with radius R and the second is a tier circle centered at the sink with radius $\alpha(n-2)R$. The value of $\zeta_{(n_{min}-2)}$ can be calculated in a similar manner as $\zeta_{(n_{min}-1)}$ using cosine rule of triangles. The area of the shaded region A_{conn} is given by

$$A_{conn} = \begin{cases} \zeta_{(n_{min}-1)} - \zeta_{(n_{min}-2)} & \text{if } \alpha \in (0, 0.5), \\ \zeta_{(n_{min}-1)} & \text{if } \alpha \in [0.5, 1.0). \end{cases} \quad (15)$$

The average number of nodes in this area is given by $A_{conn}\lambda$. For a connectivity of one, the minimum value of the node density for a given α is $1/A_{conn}$.

Fig. 8 shows the relationship between various values of α and the corresponding superframe size (in slots) for a network of radius 150 m. For each value of α, we calculated A_{conn} using (15) and the node density λ is set to $1/A_{conn}$, so that a connectivity of one is provided. From this figure, we observe that superframe size decreases as α increases from 0.1 to 0.5, but it starts increasing rapidly beyond the value of 0.5. Small values of α give rise to a large number of tiers, and hence, several relaying slots have to be allocated for each data unit. This results in a large superframe size. Large values of α result in lesser number of relaying slots for each packet, but the tiers are wider and A_{conn} is smaller. This requires a high node density λ, which, in turn, has the effect of increasing the superframe size. A minimal value of the superframe size can be observed between $\alpha = 0.3$ and $\alpha = 0.5$. We observed a similar trend for networks of other radii. For simulation, we have chosen a value of 0.5 for α to minimize the number of tiers while keeping the superframe size small. The value of λ chosen is $\frac{1}{400}$, which gives a connectivity of 8, which is more than the required connectivity of one, and thus, ensures that there are no reception outages.

4.6 Calculation of Worst-Case Delay

The worst-case delay occurs when an event occurs in the outermost tier C_H and when a node in that tier which senses the event just misses its assigned transmission slot. So it has to wait for time T before it can transmit the packet. Once the data are transmitted out of the source node in tier C_i, these are transported in $\lceil i/N \rceil$ cycles by the nodes in the inner tiers. $\lceil i/N \rceil$ is the group number to which the source node belongs. Once the data are on their way, these are transmitted in subsequent subframes in a continuous manner by the node in the respective tier until they reach the sink. When source node is in the outermost tier C_H, this translates to total number of subframes assigned to all the tiers in the network, which is H/N. But the number of tiers H may not be an exact multiple of the number of subframes N in a superframe. Hence, the worst-case delay is given by

$$d_{worst} = T + \lceil H/N \rceil T. \quad (16)$$

For a network with a given α, and whose maximum delay is to be $T + \lceil H/N \rceil T$, the parameter λ has to be chosen appropriately.

4.7 Sleep and Wake-Up Cycle of Nodes

4.7.1 Transmission Cycle of a Node

Since DGRAM follows a TDMA schedule, each node transmits only during its assigned slots in a superframe. Each node in a tier C_i has a_i slots per superframe to transmit the data it has sensed and any data it has to relay from the outer tiers. Consider a node belonging to the tier C_i and block B_{ij}. If k is the index of the node in its tier and block, the slots it should transmit in are calculated as follows: Let the number of slots assigned to the outer tiers (as subframes) before the tier C_i in the superframe be P_i. This is given by

$$P_i = \begin{cases} \sum_{(l=(i+1) mod N)}^{N} S_l & \text{if } (i \bmod N) \neq 0, \\ 0, & \text{otherwise.} \end{cases} \quad (17)$$

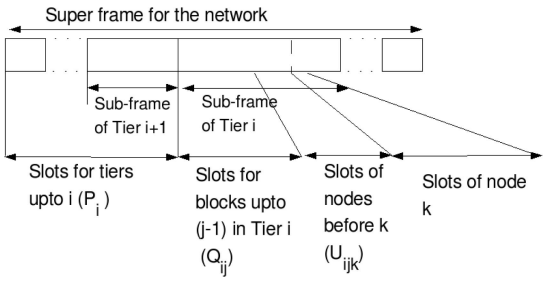

Fig. 9. Transmit cycle of the kth node in B_{ij}.

Let the number of slots assigned to the blocks before block B_{ij} in the subframe assigned to C_i be Q_{ij}. Q_{ij} is given by

$$Q_{ij} = ((B_{ij} - 1).\ S_i/2)\ mod\ S_i, \quad i \in [1, H], \quad j \in [1, Z_i]. \tag{18}$$

For example, consider the constitution of superframe of a network with eight tiers as shown in Fig. 6. A node in the 4th block of the 7th tier, i.e., in B_{74}, has to skip slots corresponding to the subframe meant for the 8th tier. Hence, P_7 is the size of the subframe of C_8, i.e., the subframe for C_7 should start after subframe C_8. As the subframes are reused across tiers, the subframe of C_8 is S_4. Then, the node has to leave slots for three blocks before it in its tier. The subframe size for C_7 is S_3, as the same subframe is allotted to both C_7 and C_3. Since each block takes $S_3/2$ slots, the node has to leave $3{*}S_3/2$ slots for blocks before it. However, $S_3/2$ slots in a subframe are reused by the odd and even blocks in this tier. Thus, Q_{ij} has a value of $S_3/2$.

Now let us find the number of slots that a node has to skip in its subsubframe. Let the number of slots before this node in the subsubframe of its block be U_{ijk}. Since this is the kth node in its block B_{ij} and every node in this tier needs to have a_i slots, U_{ijk} is given by

$$U_{ijk} = (k - 1)\ .\ a_i. \tag{19}$$

Hence, this node transmits for a_i slots starting from the slot $(P_i + Q_{ij} + U_{ijk} + 1)$ as shown in Fig. 9. If the node in the above example has an index of nine and the tier C_7 has three slots per node, U_{ijk} for this node would be $8{*}3$, which is 24. Hence, it allots itself three slots from slot number 25, counting from the beginning of its subsubframe. From the beginning of the superframe, the node's transmission slots are three slots starting from $(S_4 + S_3/2 + 25)$.

4.7.2 Receive Cycle of a Node

A node in tier C_i receives data from its previous tier C_{i+1}, where $(1 \leq i < H)$. This node should receive from a maximum of $(a_i - 1)$ slots, where a_i is given by (7).

For relaying data, senders choose their respective receivers so that there are no outages. Hence, to compute the reception cycle of a node A belonging to tier C_i, the node computes the receivers of all the senders in tier C_{i+1}. Then it would come to know the particular sender(s) for which it would be the receiver.

To compute the receivers of all the senders in the tier C_{i+1}, each node in tier C_i runs Algorithm 2. There is no control message passing between any of the nodes regarding the receive cycles. Hence, it is essential that all the nodes in tier C_i map senders to their respective receivers identically, i.e., all of them must converge to the same result after running the algorithm. So all receiver nodes of C_i have to run the above algorithm in the same order of potential senders. This order in which potential senders are considered from tier C_{i+1} while running the *for* loop in Step 7 of Algorithm 2 is imposed by means of the ordered list Γ. Γ has potential senders sorted in the nonincreasing order of their radial distance from the sink, and in case of a tie, by the nonincreasing order of their angular distance from the geographical North.

Algorithm 2. Identify_sender_nodes (A)

1: /* find sender node(s) of receiving node A belonging to tier C_i. This algorithm is run by every receiving node in tier C_i. */
2: $\Gamma =$ ordered list of nodes in tier C_{i+1} sorted in the nonincreasing value of its radial distance from the sink (in case of a tie, sorted in the nonincreasing order of angular distance from geographical North)
3: **for** each node Y in tier C_i **do** /*initialize the maximum number of receiving slots of each receiver node in C_i */
4: $num\_rcv\_slots[Y] = a_i - 1$
5: **end for**
6: $\Lambda = NULL$
7: **for** each node X in Γ **do** /*traverse the list Γ in the order in which the elements are stored in Γ */
8: $receiver\_found = FALSE$
9: **for** each node Y in tier C_i **do**
10: Let $\delta_{XY} =$ distance between X and Y
11: **if** $\delta_{XY} > R$ **then** /*since Y is outside of transmission range, it cannot be a receiver for X */
12: continue
13: **end if**
14: Insert (Y, δ_{XY}) into a temporary array Δ
15: **end for**
16: sort Δ in nonincreasing order of δ_{XY} elements in Δ are in the order of farthest to nearest w.r.t. X */
17: $l = 0$
18: **while** $receiver\_found == FALSE$ **do**
19: $\delta_{XY} = \Delta[l] \cdot \delta_{XY}$ /* get the distance between X and Y */
20: $Y = \Delta[l] \cdot Y$
21: **if** $num\_rcv\_slots[Y] \geq a_{i+1}$ **then** /*there are enough reception slots available at the receiving node Y */
22: $num\_rcv\_slots[Y] -= a_{i+1}$
23: /*Y is the receiver of X */
24: **if** $Y == A$ **then** /*the receiving node A (which is running this algorithm) is the receiver of X */
25: insert X into Λ
26: **end if**
27: $receiver\_found = TRUE$
28: **else**
29: l^{++}
30: **end if**
31: **end while**
32: **end for**
33: /* Now Λ contains the list of senders for which A is the receiver */

Each node in the tier C_i can receive in maximum of up to $(a_i - 1)$ reception slots. To find the receiver of a potential sender X, the distance between X and every other node (say, Y) in its receiving tier C_i is calculated. Note that A is the node in tier C_i that runs the algorithm and Y is a variable to denote any node of tier C_i. All the nodes which are out of the transmitting radius of X are discarded, as these cannot receive from X. Nodes within transmitting distance of X are entered into a temporary array Δ. Then, the nodes in Δ are sorted in the nonincreasing order of their distance from X. Now, each element in Δ (in the order from first to last) is checked for being a receiver of X. To check if X can choose Y as its receiver, it is checked if Y has at least a_{i+1} reception slots that are unassigned. Further, if Y is the same as the node A which is running the algorithm, then X is marked as the sender for A.

By the end of the algorithm, Λ contains the set of senders for A. Now, A can calculate the transmission slots of these nodes as described in Section 4.7.1. Hence, node A remembers to go into active state only for those slots to receive data. In receiving slots, node A listens for a short period of time (preamble) and then remains in active mode for the entire duration of slot only if there is a transmission in that slot. The node can sleep during slots other than its transmission or reception slots. These are called *idle* slots. From the discussion above, it can be seen that a sender node always tries to choose its farthest potential receiver as its receiver first. This ensures that all nodes (including those close to the outer boundary of a tier) are assigned receivers. Also, since a receiver is chosen from the list of receivers that are all within the node's transmission radius, there can be no reception outages. Each node in the ith tier is assigned a maximum of $(a_i - 1)$ reception slots. This spreads the power usage across all the nodes in a tier uniformly. Nodes in the tier C_H need not receive data from any other node, and hence, do not have a receive cycle.

This distributed slot calculation without control message passing is a major advantage of the DGRAM, which distinguishes it from other TDMA protocols for WSNs, which typically have a centralized slot assignment. Also note that the coordination of sleep and wake-up cycles helps in transporting data from source to the sink across the tiers, thereby eliminating the need for a separate routing protocol. Thus, DGRAM does not require a separate routing protocol to transport data from source to the sink. This also means that energy which otherwise would have been spent in finding routes is conserved in a DGRAM network.

4.8 An Example

As an example, let us consider a sensing area of 250 m radius. Let the minimum transmission radius of each sensor be $R = 100\ \mathrm{m}$ and the maximum interference radius $I = 100\ \mathrm{m}$. Hence, for this network, $\beta = I/R = 1.0$. Considering a sensing radius of 30 m, for proper coverage of the sensing area, the node density is chosen to be one node per 400 square meters. As per Section 4.5, α is fixed to be 0.5. This gives $F = \lfloor \frac{1}{\alpha} \rfloor = 2$, which means that the first two tiers are combined to form a single first tier. Hence, the number of tiers in the network is $H = \lceil \frac{250}{\alpha * R} \rceil - F + 1 = 4$. The number of tiers in each group is 5 for this network (using (8)), i.e., $N = 5$. Hence, the example network has a single group with four tiers. Calculation of the number of blocks in each tier is done as per Section 4.1. Thus, C_1 and C_2 have one block and C_3 and C_4 have four blocks each. Each node in C_4 has one slot, a node in C_3 has three slots, and a node in C_2 has six slots, as per (7). Nodes in the innermost tier C_1 have the maximum number of slots per node, which is 13 slots, as they have to relay data from all outer tiers to the base station. The superframe size is $T = 1{,}892$ slots. The worst-case delay for this network calculated as per (16) is 3,784 slots.

TABLE 2
Notation Used in Calculation of Power Consumption

Notation	Definition
ρ	duration of a single time slot
T_{grx}	duration of preamble at the beginning of a reception slot
N_{ijk}	ID of a node in tier C_i, block B_{ij} and index k in the block
d_{ijk}	maximum number of slots N_{ijk} should receive from
T_{idle}^{ijk}	Total duration of idle slots of N_{ijk} in a superframe
P_{idle}	Power consumption when the radio is idle
P_{rx}	Power consumption when the radio is receiving
P_{tx}	Power consumption when the radio is transmitting
$E_{rx} = P_{rx} * \rho$	Energy spent by a node to receive in a single slot
$E_{tx} = P_{tx} * \rho$	Energy spent by a node to transmit in a single slot
E_{max}^{ijk}	Maximum energy spent by node N_{ijk} per superframe
E_{min}^{ijk}	Minimum energy spent by a node N_{ijk} per superframe
$E_{grx} = P_{rx} * T_{grx}$	Energy spent in checking the preamble at the beginning of a reception slot
$E_{idle}^{ijk} = P_{idle} * T_{idle}^{ijk}$	Energy spent by node N_{ijk} in idle slots in a superframe

5 Energy Consumption in DGRAM

In this section, we calculate the maximum power consumption of a node in the network running DGRAM. The notation used in this calculation is given in Table 2. In a WSN, energy consumption can be because of event sensing, transmitting events to the next hop, receiving events from the previous hop, and energy consumed while being idle. We do not consider the energy consumption due to sensing, since this component is common to all protocols. The maximum number of slots in which a node in tier C_i has to receive is d_{ijk}, where

$$d_{ijk} = a_i - 1. \tag{20}$$

The minimum energy spent by a node during a single TDMA frame is the energy spent by it when it has no packets to receive or send. Consider a node N_{ijk}, which belongs to tier C_i, block B_{ij}, and has an index k in its tier and block. It is supposed to receive from a maximum of d_{ijk} slots and transmit in a_i slots. So $T_{idle}^{ijk} = T - \rho(d_{ijk} + a_i)$. To compute the minimum energy spent by this node, we note that it has to check the channel in the beginning of d_{ijk} slots for a possible reception and there should be no transmission or reception. Hence, the minimum energy this node spends is given by

$$E_{min}^{ijk} = d_{ijk} * E_{grx}^{ijk} + E_{idle}^{ijk} + P_{idle} * (d_{ijk} * (\rho - T_{grx}) + \rho a_i). \quad (21)$$

The first term on the right-hand side of the above equation is the energy spent in checking for preamble in the d_{ijk} receive slots. The second term represents the idle energy spent in time other than the transmission and receive slots. The third term accounts for the idle energy spent during the transmission and reception slots. In this, the term $d_{ijk} * (\rho - T_{grx})$ is the time for which the node is idle in each receive slot, after it senses the channel for T_{grx} and then goes to sleep for the rest of the slot time. The term ρa_i is the time for which the node is idle during its transmission slots.

The maximum energy is spent by this node in a TDMA frame when it receives in all its d_{ijk} slots and transmits in all its a_i slots. Thus, the maximum energy spent by this node is given by

$$E_{max}^{ijk} = E_{idle}^{ijk} + d_{ijk} * E_{rx} + a_i * E_{tx}. \quad (22)$$

For every superframe, energy spent by a node N_{ijk} is computed as follows: Let us say that there are ev_{ijk} number of events to be transmitted by this node and it receives in rcv_{ijk} ($rcv_{ijk} \leq d_{ijk}$) number of slots. So energy spent in a superframe by this node is given by

$$\begin{aligned} E_{superframe}^{ijk} = {} & (d_{ijk} - rcv_{ijk}) * E_{grx}^{ijk} + rcv_{ijk} * E_{rx} \\ & + (d_{ijk} - rcv_{ijk}) * (\rho - T_{grx}) * P_{idle} \\ & + \ min(a_i, ev_{ijk}) * E_{tx} \\ & + (a_i - min(a_i, ev_{ijk})) * \rho * P_{idle} + E_{idle}^{ijk}. \end{aligned} \quad (23)$$

The first three terms on the right side of (23) together give the total energy spent in the d_{ijk} receive slots. The next two terms denote the energy spent in the a_i transmission slots. The last term represents the energy spent in the idle period of the superframe.

The lifetime of a node is dependent on the TDMA frame time and the number of TDMA slots for which the node remains active for transmission or reception. We compute the lifetime of the most constrained node in the network. Among all the nodes in the network, we track the node which dies first and denote the lifetime of that particular node as the *lifetime of most constrained node*. This is an important parameter since death of a node might lead to outage.

Some protocols employ Low-Power Listen CSMA (LPL-CSMA) approach in which a node listens for preamble in a low-power mode at random times. But the problem with this approach is that the receiver has no way of knowing when to listen for preamble. Hence, it might require the sender to send long preamble so that when the receiver listens for preamble, there is a high probability of receiver detecting the preamble. This adversely affects the energy consumption of nodes and the end-to-end delay of packets. DGRAM does not exhibit this problem as the receiver knows that it has to listen for preamble at the beginning of every reception slot. This helps in reducing energy consumption as well as latency.

6 Simulation Results

We simulated DGRAM, FlexiTP, and a basic TDMA MAC using ns2 for different radii of the sensing area and for different event rates. FlexiTP is a TDMA-based protocol that has a loose slot structure. The protocol has an initial neighbor discovery phase, where each node discovers its neighbors using HELLO messages with CSMA/CA. Then slot assignment phase is executed, where nodes allot themselves time slots to transmit, forward, and receive data, and announce these slots to their neighbors. Once the slots are decided, each node transmits, forwards, and receives data in these slots only, and goes to sleep at all other times. When a node detects the absence of a packet in its receive slot, it treats this as a fault and the protocol switches to fault tolerance mode, where a node again tries to find a neighbor using HELLO messages. FlexiTP can work in any of the two modes: with slot reuse ON (where the slots are reused by nodes beyond two hops) and without slot reuse ON (where slots are not reused). The basic TDMA MAC (henceforth called the TDMA MAC) we have considered is the MAC that comes bundled with ns2, extended for multihop communication by us. In this protocol, each TDMA cycle is preceded by a preamble in which the wake-up slots for nodes for reception are advertised by the sending nodes. These wake-up slots are decided by the routing protocol. Each node is given one slot for transmission of data in a TDMA frame. For all our simulation experiments, we used the parameter values given in Table 3. Each simulation experiment was run for 2,000 simulation seconds. The values of the physical parameters of the nodes are taken from [23], which contains representative values for Mica2 Motes. We have assumed a data rate of 19.2 Kbps for our experiments. Duration of each time slot (ρ) is taken to be the duration for which packet of 64 bytes can be transmitted over a link of 19.2 Kbps bandwidth, i.e., $\rho = \frac{64 \times 8}{19.2}$ ms $= 27$ ms. We considered duration of the preamble to be 1 ms. The simulation is done for a circular region with the sink at the center. The sensor nodes are uniformly distributed with a density $\lambda = 1$ node per 400 square meters. Each node knows its position with respect to the base station, the node density λ, the transmission radius R, the interference radius I, the network deployment radius *RADIUS*, and the total number of nodes in the network M. The work presented in [19] assumes uniform density of nodes in the deployed area and uses a *centralized* algorithm to compute superframe structure. Depending on the area under consideration, calculating the number of nodes based on uniform density may not always be accurate. This is because the uniform density of nodes does not hold for arbitrary granularity and shape of

TABLE 3
Physical Parameters Used in Simulation

Parameter	Value
λ	1/400 nodes/m^2
ρ	27 ms
R	100m
β	1.0
P_{rx}	63mW
P_{tx}	30mW
P_{idle}	30mW
P_{sleep}	0.003mW
$TransitionPower$	30mW
$TransitionTime$	2.45ms
$InitialNodeEnergy$	54,000J

TABLE 4
Simulation Results for Beacon Exchange Phase for Networks of Different Radii ($\text{Node Density} = 0.0025\ \text{Nodes}/m^2$)

Network Radius(m)	Avg. energy spent (J)		Convergence time (s)		% Error in no. of nodes
	Distributed	Centralized	Distributed	Centralized	Centralized
50	1.106	0.254	31.4	25.4	25%
75	17.32	2.45	449.5	249.5	1.25%
100	33.55	8.76	676.7	374.8	1.25%
125	43.24	18.91	1016.69	553.1	2.5%
150	119.4	18.99	3152.97	553.2	2.9%

the area. Hence, there may be an error in computing the number of nodes based on uniform density. To mitigate this problem, this paper proposes a beacon exchange phase discussed in Section 4.3.1, where the topology information is gathered by every node in a *distributed* manner. We ran simulation to compare the above two methods. For the beacon exchange phase, MAX_TTL was set to one more than the diameter of the network (in terms of number of hops), *silence timer* time-out value was set to five times the interbeacon time. The results of this experiment are presented in Tables 4 and 5. It can be seen that the distributed scheme consumes more energy and takes more time for beacon exchange, but is 100 percent accurate in topology gathering, and hence, in the calculation of subframe lengths. On the other hand, the centralized scheme ends up with an error in calculating the number of nodes. This may result in data loss due to wrong subframe lengths and mismatched sleep/wake-up cycles.

During the beacon exchange phase discussed in Section 4.3.1, DGRAM spends much lesser time and energy for networks of all radii and node densities, as compared with FlexiTP. For example, it is noted that the time and energy for FlexiTP's *neighbor discovery* phase for a network of 100 m radius and $0.0025\ \text{nodes}/\text{m}^2$ are 1,072 sec and 32.63 J, respectively. In addition, FlexiTP took 881.28 sec and spent 26.71 J for slot assignment. For the same network scenario, DGRAM takes 676.7 sec and spends 33.55 J for beacon exchange.

We are interested in finding out the performance of the protocol in the worst-case scenario. Hence, events were generated such that the interevent time is uniformly distributed with the desired average rate. Further, when an event generation time is up, then all the nodes in the network are handed down an event with a time stamp, which is uniformly distributed with the time of generation of the event being the mean of the uniform distribution.

TABLE 5
Simulation Results for Beacon Exchange Phase for Networks of Different Node Densities ($\text{Network Radius} = 150\ \text{m}$)

λ (nodes/m^2)	Avg. energy spent (J)		Convergence time (s)		% Error in no. of nodes
	Distributed	Centralized	Distributed	Centralized	Centralized
0.0005	19.32	6.8	549.52	100.53	12.5%
0.001	43.42	10.82	720.35	149.5	4.41%
0.0015	58.41	12.31	1499.4	309.4	5.36%
0.002	71.17	15.18	1800.21	426.43	4.05%
0.0025	119.4	18.99	3152.97	553.2	2.9%

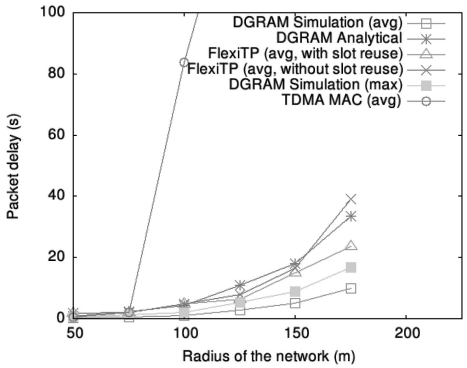

Fig. 10. Effect of the network radius on the delay.

This scenario captures the worst-case scenario when a phenomenon occurs, which affects the entire sensing area (e.g., a biohazard that encompasses the entire sensing area), and hence, all the nodes in the network would sense an event almost at the same time. We have considered constant bit rate (cbr) traffic and a lossless channel. Hence, when we consider an interevent time that is 100 percent of the superframe (which is, say, 8.964 sec), the interevent time is 8.964 sec and data generation rate is $\frac{1}{8.964}$ packets/sec.

6.1 End-to-End Delay Guarantee

In Fig. 10, as expected, the average delay increases as the radius of the network increases. This graph is very useful in designing the radius of the sensing area so that the maximum delay is less than a certain desired deadline. It is also observed that the analytical maximum delay value is always greater than the average delay obtained through simulation. Also, the maximum delay experienced while simulating DGRAM is always less than the analytical maximum delay, though this is not clearly seen in the graph. A similar simulation with FlexiTP and the TDMA MAC shows that the average delay is much higher compared to that of DGRAM for any network radius. This is because the slot allocation in FlexiTP always leads to a larger superframe size than DGRAM and the superframe size is the largest in the case where the slot reuse is not done (Fig. 11). The superframe size for the TDMA MAC is the lowest (equal to the number of nodes) as there are no relaying slots, but the packet delay is very high due to this absence of relaying slots.

To study the effect of different event rates on the average packet delay, energy spent, and number of packets

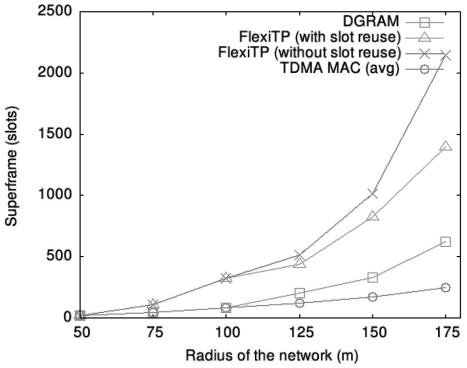

Fig. 11. Effect of the network radius on superframe.

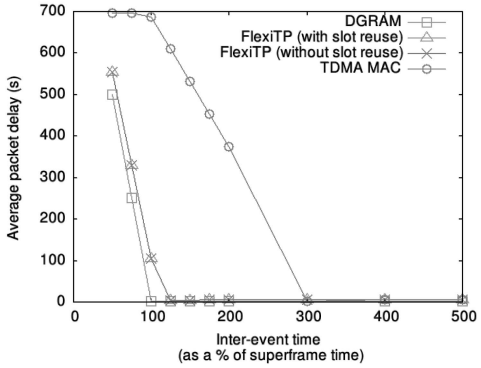

Fig. 12. Effect of event rate on the average delay (network radius = 100 m).

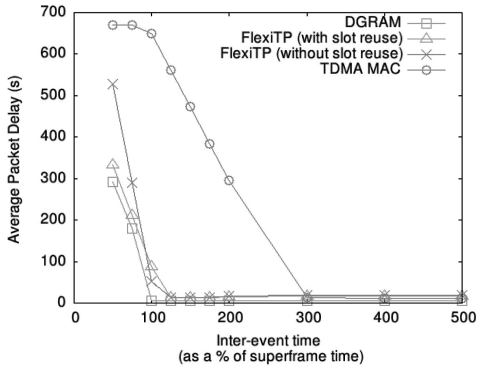

Fig. 14. Effect of event rate on the average delay (network radius = 150 m).

delivered to the base station, we compare FlexiTP, TDMA MAC, and DGRAM for three different scenarios and network radii (100, 125, and 150 m). Figs. 12, 13, and 14 show the variation of average delay as the interevent time is varied, for a network of 100, 125, and 150 m radius, respectively. We have plotted the graph for interevent times, which are expressed as a percentage of the superframe size. As expected, DGRAM gives a delay lower than the analytical worst-case delay, for interevent times greater than or equal to 100 percent of the superframe size (given by (14)), since events will not stay in queue beyond the worst-case delay. We have not shown the worst-case delay in those graphs so as not to make them cluttered. But we did verify that the average delay of DGRAM is lower than the worst-case delay when interevent time is above the superframe size. For the simulation parameters considered, the worst-case delay is $2T$ for all the three networks. For instance, superframe size for network radius of 150 m is 8.964 sec, and hence, the maximum delay is 17.928 sec. In most of the cases, an event reaches the cluster head within a superframe duration, since the first term in (16) accounts for the case when the node just misses its turn, which happens very rarely. That is why in these graphs, we notice that the average delay drops to below the worst-case delay when interevent time is around 100 percent of the superframe size. For interevent times lesser than the superframe size, buffering of events takes place and the average delay sometimes exceeds the worst-case delay. The average delay is more or less constant as the event rate decreases, which is to be expected since the nodes always transmit in fixed slots irrespective of the event rate. Hence, DGRAM is suitable for interevent time greater than or equal to the superframe size, when it provides deterministic delay guarantee.

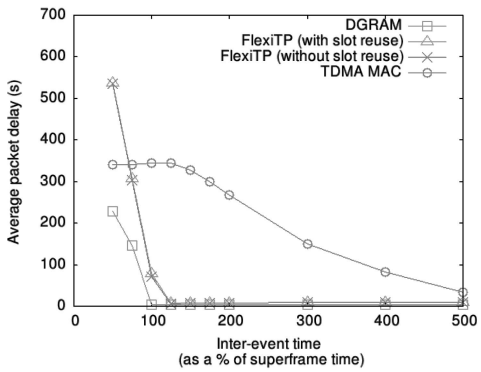

Fig. 13. Effect of event rate on the average delay (network radius = 125 m).

FlexiTP, on the other hand, does not provide any end-to-end delay guarantee. For interevent times lower than its superframe size, the delay is large due to buffering. For such cases, it is seen from these figures that the delay for FlexiTP with and without slot reuse is much higher than that for DGRAM. For the network radius of 100 m, FlexiTP has the same slot allocation, and hence, same delay, both with and without slot reuse. This is because slot reuse can only be done beyond the transmission range, which is 100 m. For a network radius of 125 m, the slot allocation is so similar for FlexiTP in the slot reuse and no slot reuse cases that the difference in delay for these two cases is very less and cannot be discerned clearly in the graph. For higher interevent times where one would expect a guaranteed delivery of packets at the base station, FlexiTP fails due to its fault tolerance mechanism taking over. In this case, when a node does not receive a packet during its receive slot, it treats it as a missed transmission and sends a retransmission request, whereas the receive slot is actually empty because there is no event to receive. With each superframe, such retransmission requests grow exponentially and the number of packets actually reaching the base station falls rapidly. This behavior is discussed in more detail in the next paragraph. The basic TDMA MAC simulated by us also does not provide any end-to-end delay guarantee. As it does not have relaying slots (each node is allotted a single slot), buffering of data takes place and high packet delays are seen for low interevent times. With higher interevent times, the delay decreases, but this is the delay for the small number of packets that reach the base station. Some packets still remain in the buffer, waiting to be relayed. The average delay for FlexiTP and TDMA MAC shown in the graphs, Figs. 12, 13, and 14, is actually the delay for the very small percentage of packets that reach the base station. DGRAM provides 100 percent packet delivery for higher interevent time. Thus, though FlexiTP claims to provide end-to-end delay guarantee when slots are not reused, it is seen that delivery guarantee exists neither for small interevent times (where buffering of events, and hence, buffer overflow may take place) nor for large interevent times (where absence of an event is taken as a missed event and the fault tolerance mechanism takes over, resulting in unnecessary loss of energy and packets), making DGRAM an ideal choice for both cases for different network sizes.

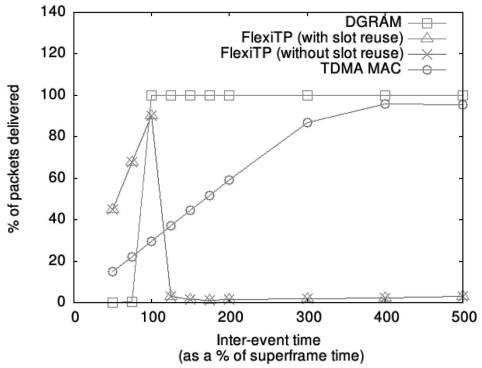

Fig. 15. Number of packets missing deadline as event rate changes (network radius = 100 m).

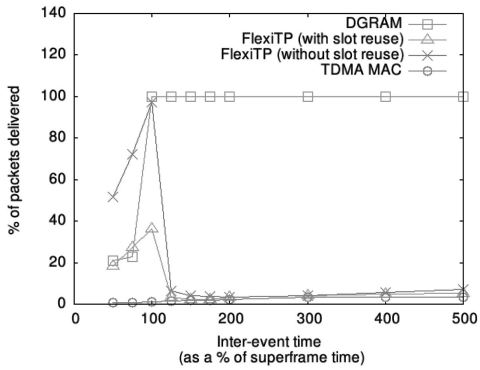

Fig. 17. Number of packets missing deadline as event rate changes (network radius = 150 m).

In [24], the authors have reported a deployed fire-monitoring system, where the maximum end-to-end packet delay should be less than 30 sec. For such an application, a DGRAM network of 150 m radius can be deployed, as it can be seen from the graphs that such a network guarantees packet delivery within 17.928 sec.

Figs. 15, 16, and 17 show how DGRAM guarantees delivery of events (packets) to the sink within the deadline in networks of 100, 125, and 150 m radius. As interevent time increases, the percentage of packets with delay less than the maximum allowable delay, i.e., packets meeting deadline, increases. We have labeled these as *packets delivered* because for FlexiTP and TDMA MAC, packets are actually lost, but for DGRAM, packets miss deadline. When interevent time is less than superframe size, the DGRAM network has packets missing deadline. As long as interevent time is less than the superframe size, DGRAM is designed in such a way that the packets flow toward the sink without exceeding the maximum possible delay. If two or more events are generated in a duration less than this value (8.964 sec for the 150 m network radius), then there will be queuing at the source node, which will have a cumulative queuing effect at the nodes in subsequent tiers, eventually leading to packets missing deadline. As the interevent time approaches superframe size T, packets meeting their deadline (labeled as packets delivered) fast approach 100 percent and when it is greater than or equal to this superframe size, all the packets are delivered before this maximum delay. Thus, DGRAM guarantees delivery of packets within the maximum delay as long as interevent time is greater than or equal to the superframe size. On the contrary, as discussed before, FlexiTP has packets lost due to buffering for low interevent times and due to fault tolerance mechanism getting activated when the interevent times are high. Hence, FlexiTP does not provide guaranteed delivery of packets in any of the cases. For TDMA MAC, when the network radius is small (100 m), most of the packets reach the sink directly without being relayed. Hence, the packet delivery approaches 100 percent for large interevent times, as seen in Fig. 15. As the network radius increases, a lot of packets are lost in the buffer for low interevent times, due to the lack of relaying slots. Though the number of packets lost in the buffer decreases with increasing interevent times, there is no guaranteed delivery of packets.

6.2 Energy Consumption

We ran the simulation to evaluate the energy consumption by FlexiTP, TDMA MAC, and DGRAM for different event rates (see Figs. 18, 19, and 20). In DGRAM, the average energy spent decreases as the interevent time increases, since increase in interevent time implies that fewer events need to be sent to the base station. When FlexiTP is used, the average energy spent remains constant for interevent times greater than the superframe size. This is again due to the fault tolerance mechanism of FlexiTP that takes over when there is no event to be received. Since very few packets reach the base station and much of the energy is spent for fault tolerance in all such cases, energy spent remains almost constant. The average energy spent using FlexiTP, both with and without slot reuse schemes, is much

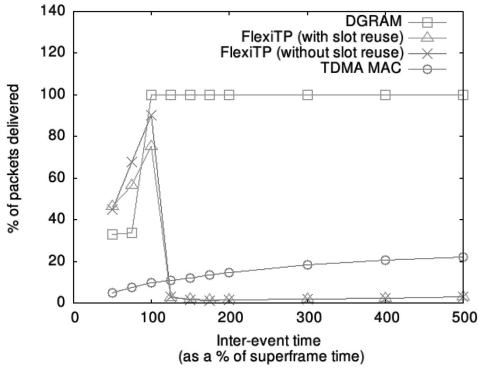

Fig. 16. Number of packets missing deadline as event rate changes (network radius = 125 m).

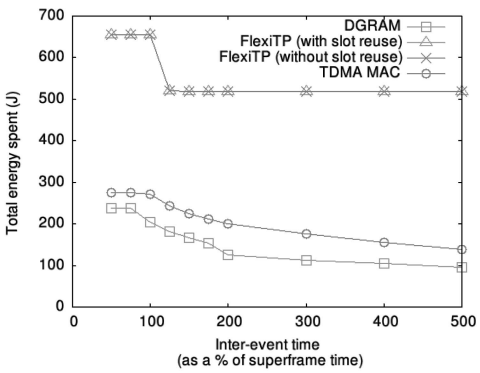

Fig. 18. Total energy consumption versus event rate (network radius = 100 m).

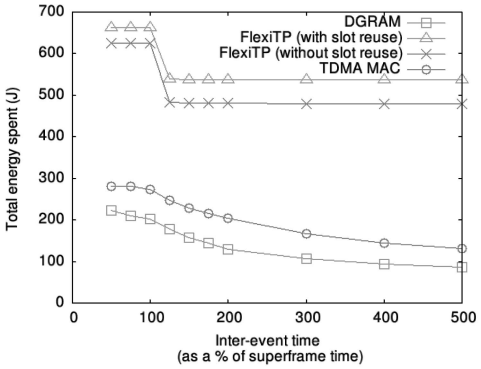

Fig. 19. Total energy consumption versus event rate (network radius = 125 m).

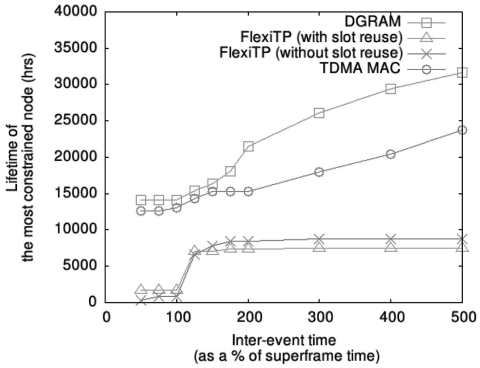

Fig. 21. Lifetime of the most constrained node versus event rate (network radius = 150 m).

higher for any event rate than DGRAM. TDMA MAC also has higher energy expenditure than DGRAM for all network radii and event rates. Note that only three plots are visible in Fig. 18 because the plots for FlexiTP with and without slot reuse coincide with each other.

6.3 Lifetime of the Most Constrained Node

We also wanted to compare the lifetime of the most constrained node at different event rates for DGRAM, TDMA MAC, and FlexiTP. For this purpose, we observe the node that is first to die and measure the duration for which that particular node was alive, which we refer to as *lifetime of the most constrained node* in the network. As seen in Fig. 21, this parameter is much higher for DGRAM than for FlexiTP or the TDMA MAC network for all event rates. This makes DGRAM a better choice not only with respect to the average energy spent by a node, but also based on the lifetime of the most constrained node.

7 Conclusion and Future Work

We have presented a new contention-free TDMA-based integrated MAC and routing protocol named DGRAM, which can provide deterministic delay guarantee. Nodes in DGRAM go through a short beacon exchange phase to learn the location of other nodes. DGRAM is fully self-configuring and slot assignment is done without exchange of any control messages. We presented the detailed design of time slot assignment, transmission and reception cycles of nodes. We also provided the worst-case delay analysis of DGRAM. We compared our protocol with another TDMA protocol called FlexiTP and a basic TDMA MAC using simulation, which showed that our protocol is a much better choice in terms of delay, packets meeting their deadlines, and energy consumption. Our simulation results also validated that the actual delay is always less than the analytical worst-case delay bound for which DGRAM is designed. Thus, DGRAM can be used for hard real-time applications such as biohazard detection, radioactive emission control, etc. DGRAM is designed to handle interevent time greater than or equal to the superframe size. That is, DGRAM can guarantee delay bound and zero packet loss as long as interevent time is greater than or equal to the superframe size. This characteristic of DGRAM can be exploited while choosing various operating parameters of the protocol.

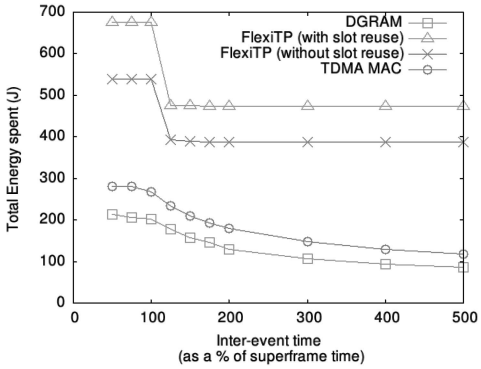

Fig. 20. Total energy consumption versus event rate (network radius = 150 m).

Time synchronization is of paramount importance in systems with TDMA protocols. We present a high-level description of the synchronization protocol for DGRAM network. After every x (where x is a tunable parameter) TDMA cycles, the network enters into a short synchronization phase, which consists of a synchronization superframe. At the beginning of this phase, the sink broadcasts a synchronization (sync) message with its clock value. All the nodes in C_1 stay awake to listen to this message. Once they receive the sync message, they adjust their clocks and then relay the message in predefined TDMA slots. This process is then repeated by the nodes in C_2, then by C_3, and so on, till the outermost tier receives the sync message.

This synchronization scheme has a clear advantage of fewer sync messages (and hence, lesser energy expenditure) over flooding synchronization protocols. Each time slot (in the DGRAM superframe) is then made long enough for guard bands before and after the transmission of a packet, to take care of the maximum possible clock drift between two synchronization superframes. This ensures that no data are lost due to clock drift before the nodes are synchronized again.

Death of a node in the DGRAM protocol has serious implication in terms of reporting and transporting data from that region to the sink. Hence, we intend to incorporate mechanism into DGRAM so that the base station would come to know when and where a node has become defunct. In a DGRAM network with a synchronization phase after every x TDMA cycles, a node counts the number of sync messages it receives from each node in its inner tier. If it does not receive sync messages from a particular node for y (again a tunable parameter) or more sync cycles, it assumes that that node is

dead. It then sends this report to the sink in a slot assigned to it in a "fault tolerance frame." After every y sync frames, there is a fault tolerance frame, which is organized similar to the DGRAM superframe, except that tierwise slot reuse is not done and each node stays awake for the entire duration of the subframe of its immediate outer tier. Hence, a node in C_i would stay awake for the entire duration of $S_{(i+1)}$ and would receive all the reports from $C_{(i+1)}$. These reports reach the sink by the end of the fault tolerance cycle so that the dead nodes can be identified and replaced, or other nodes can be made to receive in their receive cycles if possible. Note that this scheme relies on the reception of sync messages rather than reception of data messages to declare a node dead, and can work correctly even with low data rates, unlike FlexiTP.

We are currently working on the details of time synchronization and fault tolerance aspects of DGRAM and intend to incorporate those into DGRAM simulation.

There exists an optimal value of the tier width coefficient α that would result in the optimal superframe size and minimal energy consumption. We found this value of α empirically and aim to calculate this value using optimization techniques. There are many applications of WSNs that do not need a hard delay bound, but can tolerate a probabilistic delay guarantee. We plan to look at modifications necessary in DGRAM to provide probabilistic delay guarantee, which would make it useful for a wider spectrum of applications.

REFERENCES

[1] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A Global Time-Synchronized Link Protocol for Sensor Networks," *Ad Hoc Networks,* vol. 6, no. 8, pp. 1201-1220, 2008.

[2] W.L. Lee, A. Datta, and R. Cardell-Oliver, "FlexiTP: A Flexible-Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 6, pp. 851-864, June 2008.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks,* vol. 38, no. 4, pp. 393-422, 2002.

[4] A. Woo and D.E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," *Proc. ACM MobiCom,* pp. 221-235, 2001.

[5] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Trans. Networking,* vol. 12, no. 3, pp. 493-506, June 2004.

[6] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multi-Hop Wireless Sensor Networks," *Proc. First Int'l Workshop Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS),* pp. 18-31, 2004.

[7] G. Lu, B. Krishnamachari, and C.S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks," *Proc. 18th Int'l Parallel and Distributed Processing Symp. (IPDPS '04),* pp. 224-233, 2004.

[8] R. Cohen and B. Kapchits, "An Optimal Algorithm for Minimizing Energy Consumption While Limiting Maximum Delay in a Mesh Sensor Network," *Proc. IEEE INFOCOM,* pp. 258-266, 2007.

[9] K. Sohrabi and G.J. Pottie, "Performance of a Novel Self-Organization Protocol for Wireless Ad-Hoc Sensor Networks," *Proc. IEEE Vehicular Technology Conf.,* pp. 1222-1226, 1999.

[10] A.P. Chandrakasan, A.C. Smith, and W.B. Heinzelman, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Comm.,* vol. 1, no. 4, pp. 660-670, Oct. 2002.

[11] T. Wu and S. Biswas, "Reducing Inter-Cluster TDMA Interference by Adaptive MAC Allocation in Sensor Networks," *Proc. First Int'l IEEE WoWMoM Workshop Autonomic Comm. and Computing (ACC '05),* pp. 507-511, 2005.

[12] S. Cho, K. Kanuri, J.-W. Cho, J.-Y. Lee, and S.-D. June, "Dynamic Energy Efficient TDMA-Based MAC Protocol for Wireless Sensor Networks," *Proc. IEEE Joint Int'l Conf. Autonomic and Autonomous Systems and Int'l Conf. Networking and Services (ICAS-ICNS),* p. 48, 2005.

[13] T. He, J.A. Stankovic, C. Lu, T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '03),* p. 46, 2003.

[14] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 6, pp. 738-754, June 2006.

[15] D. Lucarelli and I.-J. Wang, "Decentralized Synchronization Protocols with Nearest Neighbor Communication," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* pp. 62-68, 2004.

[16] S. Ganeriwal, R. Kumar, and M.B. Srivastava, "Timing-Sync Protocol for Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* pp. 138-149, 2003.

[17] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* pp. 39-49, 2004.

[18] S. Kulkarni, A. Iyer, and C. Rosenberg, "An Address-Light, Integrated MAC and Routing Protocol for Wireless Sensor Networks," *IEEE/ACM Trans. Networking,* vol. 14, no. 4, pp. 793-806, Aug. 2006.

[19] S. Chilukuri and A. Sahoo, "DGRAM: A Delay Guaranteed Routing and MAC Protocol for Wireless Sensor Networks," *Proc. Ninth Int'l Conf. World of Wireless, Mobile and Multimedia Networks (WoWMoM '08),* pp. 1-9, 2008.

[20] A. Savvides, C.-C. Han, and M.B. Strivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," *Proc. ACM MobiCom,* pp. 166-179, 2001.

[21] N.B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-Free Distributed Localization in Sensor Networks," Technical Report TR-892, MIT Laboratory for Computer Science, Apr. 2003.

[22] H.M. Khan, S. Olariu, and M. Eltoweissy, "Efficient Single-Anchor Localization in Sensor Networks," *Proc. Second IEEE Workshop Dependability and Security in Sensor Networks and Systems (DSSNS),* pp. 35-43, 2006.

[23] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrasekharan, "Physical Layer Driven Algorithm and Protocol Design for Energy-Efficient Wireless Sensor Networks," *Proc. ACM MobiCom,* pp. 272-287, 2001.

[24] H. Xu, L. Huang, J. Wu, Y. Wang, B. Xu, J. Wang, and D. Wang, "Wireless Fire Monitoring System for Ancient Buildings," *Proc. Second Int'l Conf. Scalable Information Systems (InfoScale '07),* pp. 1-4, 2007.

Chilukuri Shanti received the BE and MTech degrees from Andhra University, Visakhapatnam, in 1994 and 1999, respectively. She is currently working toward the PhD degree at the Indian Institute of Technology Bombay. She was a lecturer at GITAM University from October 1999 to June 2001 and at GVP College of Engineering, Visakhapatnam, from August 2003 to June 2006. Her research interests include computer networks and distributed computing. She is a student member of the IEEE.

Anirudha Sahoo received the BSc (Engg) degree in electrical engineering from NIT, Rourkela, and the MS and PhD degrees in computer science from the University of Lousiana and Texas A&M University, respectively. He is currently an associate professor in the Department of Computer Science and Engineering, Indian Institute of Technology Bombay (IIT Bombay). Prior to joining IIT Bombay, he worked as a software engineer in Intergraph Corporation, Alabama, for five years, and then as a senior software engineer in Cisco Systems, California, for five years. He is a member of the IEEE.