2010

A battery aware high-throughput MAC layer protocol in sensor networks

Mohamed Watfa
University of Wollongong in Dubai

Samir Salmen
Stanford University

Hovig Denkilkian
American University of Beirut

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2010, Article ID 259809, 13 pages
doi:10.1155/2010/259809

Research Article

A Battery-Aware High-Throughput MAC Layer Protocol in Sensor Networks

Mohamed K. Watfa,[1] Samir Selman,[2] and Hovig Denkilkian[3]

[1] *Faculty of Computer Science and Engineering, University of Wollongong in Dubai, Internet City, KV 15, # 104, Dubai, UAE*
[2] *Electrical & Computer Engineering Department, Stanford University, Stanford, CA 94305, USA*
[3] *Electrical & Computer Engineering Department, American University of Beirut, Beirut 1107 2020, Lebanon*

Correspondence should be addressed to Mohamed K. Watfa, mohamedwatfa@uowdubai.ac.ae

Received 21 February 2009; Revised 15 April 2009; Accepted 4 August 2010

Energy savings have always been the primary concern in wireless sensor network protocols, however there are applications where latency and throughput are prioritized over energy efficiency and are so significant that the application would not be able to satisfy its requirements without them. The communication unit and the antenna operation consume most of the battery-powered energy of the node. Thus, the access to the medium must be controlled in a very strict manner in order to avoid collisions which result in lost transmissions and have a dramatic impact on the lifetime of the network. Although existing duty cycle MAC protocols are power efficient, they introduce significant end-to-end delivery latency and provide poor throughput. In this paper, we propose SN-MAC, a CDMA-based power controlled medium access protocol that uses both transmitter-based and receiver-based CDMA inside a formed cluster, and uses a TDMA schedule to make the cluster heads communicate with the base station. Our algorithm targets latency and throughput needs in addition to its ability to increase the overall network lifetime. We provide a head-to-head comparison with other protocols through extensive simulations focusing on the performance in terms of latency, throughput, and energy consumption.

1. Introduction

The communication unit and the antenna operation consume most of the battery-powered energy of a sensor node. This means that the access to the medium must be controlled in a very strict manner in order to avoid collisions which result in lost transmissions and have a dramatic impact on the lifetime of the network. CDMA systems allow for concurrent transmissions at the same frequency to occur through separating the signals by their corresponding spreading codes. Each terminal after joining the network receives a code through the code assigning protocol which it uses to expand the bandwidth of its signals that need to be transmitted, thus allowing for multiple transmissions from different users to occur at the same time and in the same frequency. These spreading codes can be transmitter-based, receiver-based, or a hybrid of both. In the first, the signal sent is spread using the code of the transmitter which allows multiple transmissions to be directed to the same receiver. However the receiver is supposed to monitor the whole code set of its neighbors so that it can be able to despread all received signals. In the second, the spreading is done using the code of the receiver. This simplifies the receiver design which now needs only to monitor its own code instead of the whole code set but requires the transmitter to store the codes of all of its neighbors in its memory. Also, for a receiver-based code, multiple transmissions to the same receiver result in collisions because the same code is used for spreading of all the signals destined to the same receiver. The third approach for spreading codes is to use a hybrid-based (receiver-transmitter) spreading as proposed in [1]. In this scheme, the transmitter spreads the packet header containing the source and destination addresses by the receiver's code and the rest of the packet by its code. The receiver on the other hand monitors its code until it receives the packet header and retrieves the sender's address. Then it switches to the transmitters code and despreads the rest of the packet.

One advantage of CDMA systems is that they allow users to send at any time without being confined by a certain allocated time slot or frequency channel. This leads

to significant improvement in system performance in both latency and throughput measures. Also since CDMA systems use spread spectrum modulation, they are resistant to jamming and provide self-interference suppression which is due to multipath propagation and multiaccess interference suppression from other users. But these systems require sophisticated correlation filters that increase the complexity and the cost of the receiver node, and since they use spreading codes, they are usually not scalable. In our paper, we solve the first problem by having only specific designated nodes having this complex circuitry "super nodes" while the rest are relatively simple. The second problem is tackled though our scheduling algorithm that assures spreading code reuse by making nodes with the same allocated spreading code have orthogonal wakeup/sleep schedule. We also make use of spatial reuse of codes.

Previously proposed MAC protocols for wireless sensor networks aim at minimizing the energy consumption of the nodes and this is done at the expense of degraded throughput and latency performance. There are many applications in wireless sensor networks that have stringent latency and high throughput requirements such as medical monitoring, intruder detection, and battlefield surveillance. In the last for example, the data gathered by the sensors need to be transmitted effectively and under no delay conditions since it contains timed information about movement of explosives and car bombs that will signal the soldiers to act upon detection of enemy presence. Also, when a sensor network is being used to track an object, out-of-date information is of no value because the object that is being tracked is no longer in the vicinity of the sensing node when the information is received at the base station. Thus our protocol design came to balance the considerations of energy efficiency, latency, throughput, and fault-tolerance in sensor networks.

SN-MAC uses a combination of DS-CDMA and TDMA on the MAC layer and reduces channel interference by using a power control mechanism and a separate channel for control packets. The network is divided into clusters (*formed initially after deployment*), where each node could be any hop away from the cluster head, that are kept intact for the whole lifetime of the network because our goal behind clustering is to construct a logical hierarchy in the network rather than assuring that each node is part of a cluster and that the cluster head role is dynamically rotated to distribute energy fairly among nodes. Our algorithm can run on top of previously proposed clustering algorithms; yet we develop our own simple clustering algorithm (SCA) to show that our protocol does not need complex clustering and works fine even if only the basic requirements are met. The algorithm targets the MAC layer and provides through a cross-layer design an optimum routing strategy that gives a best effort design to deliver data from the sensors towards the base station. The information flow traverses several nodes within a cluster reaching the cluster head which in turn delivers the data to the base station. The clusters are divided into levels where each node chooses its best neighbor which is one level away from it, based on considerations of the battery state of the node and packet transmission information which are represented in the form of a priority function. SN-MAC assigns PN codes and makes the nodes adopt a sleep/wakeup schedule that in addition to minimizing the power wasted on idle listening reduces the end-to-end delay of messages and enhances the network throughput. Also, since robustness is one of the desired characteristics of sensor networks, our algorithm reacts favorably upon the addition or failure of nodes and which could severely affect the performance of the network. Most of the work in the field proposes ways to increase the lifetime of the nodes by means of power-aware techniques, such as, using an optimal transmission power or by switching of the nodes when idle. Though these protocols try to increase the lifetime of the network, they do not directly consider the behavior of the batteries. In our protocol, we also propose to exploit the chemical properties of the battery to increase their lifetime. Our protocol shows that a uniform discharge of the nodes of the network can increase their lifetime. This ultimately postpones the death of individual nodes and hence increases the network lifetime.

The rest of the paper is organized as follows. In Section 2 we present the work that has been done in this area. Section 3 describes the battery model used in the Mac algorithm. Section 4 describes in detail SN-MAC. Section 5 presents the simulation results. We conclude this paper in Section 6.

2. Related Work

In the literature, many MAC protocols were proposed to achieve different requirements for sensor networks. S-MAC [2] was designed for tackling the idle listening problem, which is a dominant source of energy waste in sensor networks, through the adoption of periodic sleep and wakeup schedules. The duty cycle of S-MAC is fixed and predefined which makes it nonadaptive to traffic conditions in the network. Thus T-MAC [3] was designed to improve S-MAC by dynamically changing the idle listening intervals and hence improving throughput and end-to-end latency. On the other hand, B-MAC provided a different approach for minimizing energy on the MAC layer by using a low-power listening scheme and allowing the application to develop its own MAC protocol through a well-defined interface. B-MAC outperforms both S-MAC and T-MAC in terms of throughput and energy efficiency.

In general, MAC protocols can be classified as either contention-based or contention free protocols. TDMA is an example of a contention-free MAC protocol whereby each node is allocated a specific time slot in which it can send or receive. CSMA on the other hand is an example of a contention-based MAC protocol which makes the nodes transmit RTS/CTS packets to gain access to the shared medium. In [4], a hybrid MAC scheme called Z-MAC was developed which combines the strengths of both TDMA- and CSMA-like protocols. Z-MAC behaves either as CSMA or as TDMA based on the level of contention in the network. The advantage of Z-MAC over the other existing protocols is in the ability to overcome problems related to synchronization and topological changes. Also, several MAC protocols proposed the design of CDMA-based protocols for wireless sensor networks. In [5], CDMA Sensor MAC "CSMAC" was developed to minimize latency in addition

to reducing energy consumption. This was achieved through the use of a direct sequence spread spectrum CDMA system. It further used frequency division to reduce multiple-access interference. However, it assumed that each node is aware of its location and that the application requires the delivery of low latency and high fault-tolerance under high-load conditions. Along the same lines, [6] introduced a cross-layer analysis for CDMA-based wireless sensor networks, that examined analytically the multi-access Interference (MAI) problem and shed light on the tradeoff between interference and connectivity by using three deterministic topologies and one random topology for analysis of the problem. In [7], Liu and Asada proposed an energy efficient DS-CDMA system for sensor networks that controlled the MAI by using spreading codes with more reduced bits and employing an on-off keying data transmission scheme. In [8], a CDMA-based MAC protocol was designed for wireless ad hoc networks where an out-of-band RTS/CTS handshake was used to dynamically determine the transmission power of a node that will not result in collisions at neighboring receivers. The exchanged RTS/CTS packets included information that controlled the MAI resulting from multiple concurrent transmissions.

All of the above-mentioned protocols do not take into consideration the battery behavior when minimizing the energy consumption in the network. But, [9] proposed a novel battery-aware MAC protocol which schedules transmissions of different nodes in a round-robin fashion, based on the battery state of the contending nodes. However, this protocol does not take into consideration the energy consumed due to idle listening and uses a simple round-robin scheduler that is ineffective since it does not adapt to the needs of the transmitting nodes. In BAMAC, a node which urgently needs to transmit a detected event must wait its turn although there may be other nodes that might not need to transmit any new data. Also, BAMAC neglects the power consumed on the transmission of control packets (RTS, CTS, ACK etc.) which implies that it makes no effort on minimizing protocol overhead which consumes a significant amount of power. A novel protocol named TP-MAC was described in [10] that achieved synchronized low-power listening with rapid fast path establishment by the propagation of short wake-up tones. The results of this paper show that TP-MAC can achieve very low duty cycles for the same target latency when compared with pure SCP-MAC [11]. On the other hand, L-MAC in [12] is a contention-based MAC protocol that targets low-latency, energy-constrained applications. L-MAC assumes that the network is divided into levels where nodes execute an adaptive sleeping schedule allowing those with lower traffic to have longer time to sleep in order to save more energy. L-MAC delays the transmission and reception of packets on a hop-by-hop basis, so that when a node is in the sending mode, its lower-hop node is in the receiving mode. The simulations performed in this paper show that L-MAC achieves lower energy consumption and latency than the traditionally used contention-based MAC protocol. Level-based scheduling was used in DMAC [13] which presents an adaptive duty cycle protocol that is designed for data gathering trees in sensor networks. DMAC uses topology knowledge in order to stagger nodes' schedules according to their position and depth in the routing tree, so that packets flow continuously from source nodes to the sink, minimizing end-to-end delay significantly. Light-weight MAC (LMAC) [14] is one scheme where each node controls a unique slot. However, nodes still have to contend to transmit data to an intended receiver in its time slot. The receiver (slot controller) is responsible for settling contention and deciding who it receives data from. Contention often leads to collisions and therefore such protocols require some form of carrier sense multiple access (CSMA).

Our paper presents SN-MAC, a comprehensive framework on the MAC and Routing layers to be adopted by sensor nodes. SNMAC further provides functionalities that can ease the design of upper layer protocols, especially clustering. Also, since CDMA code assigning protocols is essential in all CDMA systems, SN-MAC is able to integrate any code assignment protocol to the presented algorithm. Strict synchronization and power control can also be supported although not required by SN-MAC itself through the flexibility it offers. It presents a battery-aware CDMA-based MAC protocol that will serve a low latency and high throughput demanding application. In addition, the protocol will strive to minimize the energy consumption by tackling the problems of idle listening, overhearing, collisions, and protocol overhead using its scheduling algorithm. SN-MAC also provides the upper layers a routing strategy through a proper cross-layer design.

3. An Online Battery Model

Sensor nodes are battery-powered, and, as was mentioned earlier, protocols aim to minimize the power consumed in these batteries. But designing to reduce the average power consumption does not necessarily lead to optimum battery lifetime, since battery behavior highly depends on the discharge profile experienced by the battery. Hence, the actual behavior of the battery must be represented through a model that takes into account the processes that govern its operation in order to be able to determine the actual capacity of the battery which allows a node to assess its lifetime correctly so that it adapts its behavior to maximize this lifetime accordingly. Since energy minimization is a major constraint in sensor networks, our algorithm forces the nodes to track the state of their batteries, assess their participation in the network, and react to changes in their battery states by changing their routing decisions. Most battery models that are currently used for analyzing and simulating the energy efficiency of protocols are linear models, which assume that the maximum capacity of the battery is unaffected by the discharge rate. Unfortunately linear battery models are only a rough estimate of the actual battery behavior, which must take into consideration both the *rate capacity effect* and the *recovery capacity effect* of the battery. Moreover, the lifetime of a battery depends on the discharge profile, and hence protocols should behave in a way that takes this into account so that they can achieve the lowest energy consumption and hence the longest possible network lifetime.

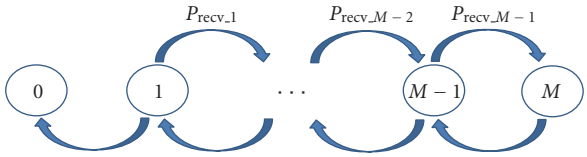

Figure 1: A Markov Chain representation of the battery model.

In our model, we represent the battery behavior using a discrete time Markov process (the discrete time model is an accurate representation because of the packetized nature of communications). We divide the time axis into slots each of fixed size. The Markov chain contains "M" states where the zero state represents the state when the battery becomes dead and unable to recover whereas state "M" represents a fully recovered battery (Figure 1). A battery in state "i" ($i \neq 0$) when left idle for a single time slot will move back to state "M" with a probability $0 < P_{\text{rec\_}i} < 1$ whereas if left idle for a time Δ (recovery time) will return to state M with probability $P_{\text{rec\_total}} \approx 1$. In our algorithm, a node's goal is to evaluate Δ in order to take decisions on how long to remain idle. Thus a node and upon participating in a transmission should first realize the state it has transferred to by estimating the energy dissipated in the battery (Φ) during the process of transmission. After calculating this energy and knowing its initial state a node can estimate its current state and the probability of recovering thus estimating Δ as a result. Assume that the transmission process takes place within the time slot "n". Then according to [15], Φ_n which is the energy dissipated in time slot "n" is

$$\Phi_n = I_n \times f(T, n\sigma, (n+1)\sigma, \beta), \tag{1}$$

where

$$f(T, n\sigma, (n+1)\sigma, \beta) = \left(\sigma + \frac{\pi^2}{3\beta^2}\right) \times \left[e^{-\beta^2(T-(n+1)\sigma)} - e^{-\beta^2(T-n\sigma)}\right], \tag{2}$$

T is Lifetime of the battery if the energy was fully consumed, β is a chemical parameter experimentally determined and specific to each battery, and I_n is discharge current through the battery during time slot "n".

Assuming the initial capacity of the battery before the transmission during time slot n was α_0 and it was in state "N" then we can estimate its final state "i" in the Markov chain using: $i = \lfloor((\alpha_0 - \Phi_n)/\alpha_0) \times N\rfloor$. Next, we compute the probability of the battery recovering in the next slot (slot n + 1) from state "i". Assuming that the packet arrival process has a Poisson distribution with mean μ_q then the probability that no packet arrives is:

$$\Pr(x = 0) = \left(\frac{\left((\mu_q)^x e^{-\mu_q}\right)}{x!}\right]_{x=0} = e^{-\mu_q} \tag{3}$$

Thus the probability of the battery recovering to the higher state in the next slot $P_{\text{rec\_}i}$ can be modeled as:

$$P_{\text{rec\_}i} = e^{-\mu_q} \times e^{-\eta(N-i)} = e^{-(\mu_q + \eta(N-i))}. \tag{4}$$

Therefore the probability of the battery recovering back to state "N" assuming independent events is:

$$P_{\text{rec}} = \prod_{z=i}^{N-1} P_{\text{rec\_}i} \tag{5}$$

Notice that this model takes into account that the recovery capability of the battery decreases as the capacity decreases and thus the lower the state "i" is, the less the probability to recover in the next time slot. Note that η is an internal battery parameter which signifies its recovery capability. η depends on the internal battery resistance and the current the battery is discharged at. Finally, the amount of time needed for the battery to recover (Δ) can be approximated by

$$\Delta \approx (1 - P_{\text{rec}}) \times \text{Max}(\Delta). \tag{6}$$

Max(Δ) is the maximum recovery time needed, that is, the time of recovery needed if the battery were left to recover just before it dies. This value is determined experimentally and is specific for each type of batteries. Our algorithm will use the value of Δ as will be demonstrated below in order to compute a node's priority in being selected as a routing candidate towards the base station.

4. SN-MAC Design and Analysis

After initial deployment, each node will be in the setup phase in which it will run a simple clustering algorithm that achieves leveling, neighbor discovery, choice of schedule and PN-code exchange in addition to the formation of the clusters. Notice that all these functions are done in one step and only at startup and hence the overhead incurred will last for the whole lifetime of the network. After finishing the setup phase nodes will use CDMA as their basic MAC protocol to communicate with other nodes in a cluster. Our algorithm implements an adaptive TDMA schedule between cluster heads to allow them to communicate with the base station.

4.1. Code Assignment Protocol. Since our algorithm uses CDMA as the basic MAC protocol, a distributed code assignment protocol becomes a must. This code assignment protocol should offer spatial reuse and aim at assigning nodes with PN codes such as guaranteeing that no logically neighboring nodes use the same PN code. Several code assignment protocols have been designed as in [16, 17] and all tackle the above goal. In this paper, we assume that a code assignment protocol is present at a higher layer, yet SN-MAC design provides great opportunities of code reuse through its scheduling algorithm that tends to have neighboring equilevel nodes adopt different schedules. Thus these nodes are now able to use the same spreading code without interfering with each other since their wake-up schedules are different.

4.2. Network Formation. Our algorithm can run on top of previously proposed clustering algorithms; yet we develop

our own simple clustering algorithm (SCA) to show that our protocol does not need complex clustering and works fine even if only the basic requirements are met. We also use SCA to leverage the overhead inherently present in any clustering algorithm to perform neighbor discovery, leveling, schedule selection and exchange of PN codes at the same time. By this we would have decreased the overhead to a minimum. This is run only at startup and hence this minimal overhead is incurred only once in the network's lifetime. We leave the scheduling till Section 4 where it will be discussed in details so what follows will only describe the remaining functionalities handled by SCA.

We assume there are two kinds of nodes in our network. The first we will call "*super node*" and is supposed to have higher capabilities than the rest of the nodes in that they are more energy abundant and have high communication ranges, that is, can directly communicate to the base station. They also have relatively more complex circuitry in order to receive packets from their one-hop neighbors that will be sent using transmitter-based CDMA. Yet not too complex since a node needs only to monitor the codes of its higher-level neighbors and not the whole code set. The second type is the "*normal node*" which constitutes most of the nodes in the network. When a super node needs to send packets to normal nodes, it uses the value of the power equal to a normal node's maximum power. This will allow us to assume equal forward and reverse gains between any pair of nodes including cluster heads. On the other hand, super nodes are allowed to raise their power when they need to communicate with the base station.

In SCA, we aim at forming multihop clusters with super nodes as cluster heads. However in wireless sensor networks, the nodes are often randomly deployed and hence there might exist a part of the network which is not connected to any super node. By connected we mean that it is a k-hop neighbor to it, that is, there exists a path from it to any super node. This case though will be rare given a sufficient number of super nodes is present. We allow nodes that can directly reach the base station and that after the network formation phase were not part of any cluster to elect themselves as cluster heads and form clusters. Note that remaining nodes which are neither connected to any super node nor to any normal node which is itself connected to the base station are considered partitioned from the remaining of the network and hence cannot be handled in any way since their data can never reach the base station. The overhead incurred in the formation of clusters is also used for neighbor discovery, level discovery, schedule formation, and exchange of PN codes.

After deployment, each normal node waits for a predefined period T which depends on the total number of nodes in the network "N". N can be decided and configured in the nodes prior to deployment. Yet this does not affect the ability of our protocol to support addition and removal of nodes since the time "T" is needed only once and that is during the network formation phase. In this time T, super nodes are allowed to form clusters. Each super node forms an invitation packet and includes in it a cluster ID, a level field and a PN-code field. The level field in the packet is set to 0 whereas the levels initially stored in the nodes have a value of INFINITY until they get updated by the reception of an invitation packet. This invitation packet is first sent to the super node's one-hop neighbors with a power equal to a normal node's maximum power. The one-hop neighbors in turn are supposed to store the cluster ID which is now their cluster, increment the level field in the packet and store it as their level to replace the INFINITY value. They will also include the PN-code they listen at in the PN-code field and rebroadcast the packet to their one-hop neighbors. However before doing so they wait for a certain amount of time also a function of "N" to make sure that all super nodes have sent their invitation packets to their one-hop neighbors. Next, the two-hop neighbors of the super node now receive the invitation packet from their lower-level neighbors. Upon receiving the packet, a node looks at its level and if it is greater than the level in the packet plus one, it updates its level by setting it equal to the packet level plus one then joins the advertised cluster. Also it stores the address of the lower-level neighbor that sent the packet along with its PN-code. On the other hand if the node's level is equal to the level in the packet plus one, it only updates its table if the advertised cluster ID in the received packet is the same as its current cluster ID. Hence, it forms a table of its lower-level one-hop neighbors and their corresponding PN codes. These nodes in turn will then replace the PN-code of the received packet by their own PN-code, increment the level field and rebroadcast the packet again after waiting for the sufficient time "$T2$" to allow their lower-level nodes to finish broadcasting. Note that the delay in the network formation process resulting from waiting "$T1$" and several "$T2$" seconds is acceptable since the latency constraint we are targeting is in the application and thus after network formation. The process continues until all possible nodes which are able to join a cluster do so.

A node which does not belong to any cluster yet will try to contact the base station. If it succeeds, it elects itself as a cluster head and floods an invitation packet as before; however, this time indicating that it is not a super node and hence does not have the complex circuitry needed to receive messages using transmitter-based CDMA. Its one-hop neighbors will thus resort to a centralized TDMA schedule managed by the advertised cluster head. We suspect such a case to be rare if enough super nodes are present, yet we present this patch to our algorithm to solve such unpredictable cases and decrease the chance of having partitioned sections in the network. Figure 2 illustrates the operation of our algorithm after the setup phase.

4.3. Power Control Mechanism. In SN-MAC, we use a combination of transmitter-based and receiver-based CDMA. All level 1 nodes of a cluster (*i.e., nodes one-hop away from the cluster head*) use transmitter-based CDMA to send packets to the cluster head whereas nodes with a level of 2 or above use a receiver-based CDMA to communicate with upper level nodes. CDMA suffers from multi access interference (MAI) and one of the ways of reducing the effect of MAI is through power control. Moreover, power control significantly reduces energy consumption which is the highest during transmissions; however, it also decreases

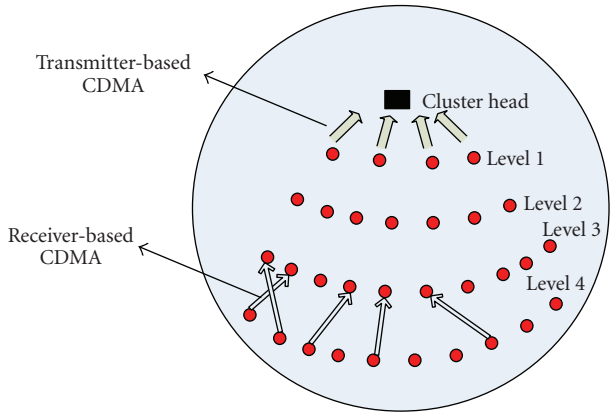

Figure 2: An Illustration of our algorithm behavior inside a cluster.

the signal to interference-plus-noise ratio (SINR) at the targeted receiver. This results in a tradeoff between reducing interference at nontargeted nodes and battery consumption on one hand, and reducing the SINR at the targeted node on the other hand. Since the applications we are considering (Intrusion detection, medical monitoring, animal tracking, etc.) possess the requirement of having all sensors that detect an event transmit their data with the lowest possible latency, there would be instances in the network lifetime where there is a burst of data that needs to be transmitted. The sensors would be carrying different types of information as well as different data within every type, and all need to reach the base station with minimum delay. Therefore our power control mechanism will prioritize reducing MAI at neighboring nodes which we expect will be concurrently receiving transmissions as well. The scheme we use will not totally prevent collisions from occurring though will reduce their occurrence drastically and does not require a node to keep listening on any channel for the whole time. Note that SNMAC can support other tight and more restricted power control schemes through the flexibility it provides by its use of RTS/CTS packets which allow for packet-level power control to occur smoothly and without significant extra overhead.

In our scheme, a node i wishing to send to another node j will first send it an RTS on the control channel at maximum power $P_{\max}$. Node j will then calculate the minimum power that node i can use to send data to it. Hence node j would reply back with a CTS which includes the minimum power that i can use to send to j. Consequently, node i will send at the power at which it received the CTS. We assume that all nodes initially agree and know the value of $P_{\max}$. Node j on the other hand receives the RTS with a certain power P_r and can thus compute the channel gain $G = P_r/P_{\max}$. We assume that the packet duration (Δt_p) is small compared to the coherence time of the channel (τ_c). Therefore the channel is slowly fading and G can be assumed constant for the duration of the packet transmission. Let $\Omega_j = E_b/N_{0\text{eff}}$ be the effective bit energy-to-noise spectral density ratio at node j. Let Ω_j^* be the effective bit energy-to-noise spectral density ratio at node j that is needed to achieve the target bit error rate. Hence at node j we require that $P_{\text{recv}}/(P_{\text{Thermal}} + P_{\text{MAI}}) \geq \Omega_j^*$, where P_{recv} is the power received by terminal j, P_{Thermal} is the thermal noise power and P_{MAI} is the total power due to multiaccess interference at the receiver. Therefore $P_{\text{recv}}^{\min} = \Omega_j^* \times (P_{\text{Thermal}} + P_{\text{MAI}})$, where $P_{\text{recv}}^{\min}$ is the minimum received power needed by terminal j in order to correctly decode the packet. Finally $P_{\min} = P_{\text{recv}}^{\min}/G$ is the minimum power needed for the transmitter to send at in order to allow the receiver to properly decode the packet. Node j places the minimum power calculated above as an additional field in the CTS packet and sends it back. Node i consequently will send data to node j at $P_{\min}$. Hence by using power control, we were able to reduce MAI since the effect of a certain transmission on another nearby one is much less (*the power used for transmission is much less*). Moreover we were able to significantly save energy by reducing the power used by nodes for transmission.

4.4. Wakeup/Sleep Schedule. Since idle listening is a major source of energy wastage in sensor networks, a wakeup/sleep schedule becomes very essential in prolonging a network's lifetime. Hence, the goal behind the scheduling algorithm is to minimize the energy consumed by noncluster head nodes in the network, but provided that the penalty incurred on the end-to-end packet latency is tolerated by the delay requirements of the application running in the network. The scheduling scheme adopted in SN-MAC provides the nodes which are closer to the cluster head with the priority of determining the wakeup/sleep schedule, since the nodes that are closer to the cluster head experience a higher amount of traffic due to the uni-directionality of the generated data.

In addition, the scheduling scheme tries to make neighboring nodes at the same level adopt different schedules and try to make nodes at level "n" adopt the same schedule as their neighboring nodes on levels "$n-1$" and "$n+1$". The intuition behind these goals is that we try to make nodes adopt the same schedule as their upper and lower-level neighbors since they will be responsible for relaying the packets from the lower-level neighbors and through the upper level ones towards the cluster head. On the other hand, we try to make nodes on the same level adopt different schedules as much as possible so that if a lower-level node gets data and wants to forward it towards the cluster head, it will find an awake upper-level neighbor which will forward the packet for it. Initially, all the nodes will run the SN-MAC algorithm without sleep scheduling in a phase called the initiation phase which allows the higher-level nodes to compute the priorities for their neighbors that are one level below. The goal of this phase is to set up the schedules that will be used during the steady-state phase, and it can be repeated to enhance the performance in the steady state phase. In order to obtain orthogonal schedules for nodes lying on the same level, the cluster head defines orthogonal schedules for the lowest level nodes (equal division of slots in a round-robin fashion) and the upper level nodes try to adopt the schedules initiated by the lower-level nodes. The lower-level nodes will start by computing the sleep duration knowing that minimizing the sleep time T_{sleep}

beyond a certain threshold results in rapid switching between the on and off modes that leads to large energy wastage due to the transient characteristics of hardware circuits and maximizing it can result in not meeting the latency and throughput requirements of the application.

The lower-level nodes will periodically transmit a special "sleep" packet that contains the time the node will sleep for the coming frames, where the duration of a frame F is defined by a number m of active/sleep cycles $F = m \times T_{\text{cycle}}$, where the active period of a cycle is large enough to receive RTS and reply with a CTS or receive a data packet. However, the "sleep" packet will be transmitted only if the difference between the new "sleep time" and the previously calculated is larger than a certain threshold determined by the energy wasted on the transmission of the "sleep" packet and its reception by the one level away neighbors $(T_{\text{sleep}(2)} - T_{\text{sleep}(1)})/T_{\text{sleep}(1)} \gg \alpha$, where α is determined from: $(E_{tx} + nE_{rx})/E_{\text{residual(neighbours)}}$ Where n is the number of neighbors that have a one level higher-level inside the cluster and E_{residual} is the remaining battery energy of a sensor node.

Hence, the power consumption due to wake-ups can be given as: $P_0 = mxE_0/F$ where E_0 depends on the active period and the hardware circuit. The sleep time is a function of the recovery time of the battery state τ generated by the battery model algorithm, and the number of packets received for the last frame. The latter depends on the average packet generation rate λ in each sensing node during its nonsleep period (depending on the application data rate) and the number of upper level neighbors. The weighing factor for these components depends on the latency and throughput requirements that can be given as the maximum delay tolerated/message of the application. We will consider δ as the tuning parameter of the application.

$$T_{\text{sleep}} = \tau\left(1 + \left(\frac{\lambda nF}{P_{\text{avg}}}\right) - \delta\right), \tag{7}$$

where P_{avg} is the average number of packets received during the last 5 frames. The upper-level neighbor that will receive the schedule will choose the schedule of the node which has the highest priority Pr among its neighbors (one level below). Initially, some nodes will not receive any packets because their upper level neighbors will adopt different schedules but these nodes will be highly prioritized for their neighbors for the future frames. The performance of the scheduling algorithm is improved by determining the optimal frame size based on simulations and fixing it for the lifetime of the network. Finally, it is important to mention that although SN-MAC does not design a CDMA code distribution algorithm, the scheduling algorithm makes the nodes on the same level adopt different schedules, which allows SN-MAC to perform a time reuse of the CDMA codes in order to improve the scalability of the algorithm.

4.5. Priority Assignment. In SN-MAC, a node's goal is to sense and deliver packets successfully to the base station. It uses multihop communication to achieve this since most nodes' communication ranges will not cover the base station. Thus after a node senses data it will need to forward it to a certain neighboring node which is at a lower-level than itself (*i.e., closer to the cluster head*). Therefore a node needs to select which lower-level neighbor among the possible candidates it will choose as its intermediate node towards the cluster head. This choice will be priority-based, that is, it will choose the lower-level nonbusy node with the highest priority. The priority function aims at reducing the latency on the routing level and maximizing the network's lifetime through the proper choice of the next hop forwarders and doing load balancing. Three components determine the priority of a node to be chosen as the next hop forwarder.

4.5.1. The Battery Model Described in Section 3. As a node's battery capacity increases, its priority will increase. This provides load balancing and prevents the formation of holes in the network. Also as the time of last transmission decreases, the computed priority will increase. Thus we aim at choosing the neighbor whose last transmission was the farthest in time. This is to allow for nodes to recover and thus make efficient use of the capacity recovery effect in the battery model, hence also increasing network lifetime.

$$\text{Pr}_1 = \frac{\text{BatteryCap}}{\text{Initial Battery Cap}}\left(1 - e^{-((\text{CT}-\text{TOLT})/\tau)}\right), \tag{8}$$

where CT is current time, TOLT is time of last transmission and τ is average time to recover.

4.5.2. The Distance of the Candidate Node from the Sender Node. As this distance decreases, the power needed to transmit a packet to that node will also decrease. This will preserve energy and also reduce MAI. Thus a neighboring node closer to the sender will be given a higher Pr_2 value than another neighboring node farther yet still in its communication range.

$$\text{Pr}_2 = \frac{d_{\text{cand}}}{d_{\max}}, \tag{9}$$

where d_{cand} is the distance of the candidate node whose priority is being computed to the sender node. $d_{\max}$ is the maximum distance between the sender node and any of the candidate nodes. d_{cand} can be estimated by a node from the power of the received CTS packet using the Free Space Pathloss Channel Model.

$$\frac{P_{\text{recv\_CTS}}}{P_{\max\_\text{CTS}}} = \frac{G_t G_r \lambda^2}{16\Pi^2 d_{\text{cand}}^2}, \tag{10}$$

with G_t and G_r being the antenna gains of the transmitter and receiver, respectively. λ is the wavelength of the transmitted signal. $P_{\max\_\text{CTS}}$ is the power by which the CTS packet was transmitted. This power is the maximum power that the nodes initially agree on as described earlier. Finally $P_{\text{recv\_CTS}}$ is the power of the CTS packet when it was received. Note that other models such as Ray Tracing Pathloss Model, Empirical Pathloss Model, or even models incorporating shadowing and fading effects can be used for the distance estimation; however, more complex computations would be needed in the sensor node.

4.5.3. Estimated Congestion at the Candidate Nodes Is the Third Component of the Priority Function. Since congestion maps directly to latency, nodes tend to pick the neighbor who is least likely to be congested and hence can forward the packet with minimum delay. Furthermore if a candidate node is always congested, then attempts to forward the packet to it might result in failures since it will be busy processing another reception. The sender node can estimate the amount of congestion to each candidate node through a weighted average of the proportion of failed attempts to forward a packet to this designated candidate node, that is, a weighted average of the ratio of RTS packets which did not result in CTS replies to the total attempts of sending RTS packets. The weights will depend on the time when these RTS packets where sent (the more recent, the higher the weight given). Thus, the time axis is divided into frames. Assume that the congestion estimation will take into account RTS failures in the last three frames. Then the corresponding component of the priority function would be:

$$\mathrm{Pr}_3 = \sum_{i=1}^{m} \frac{n_{\mathrm{fail\_RTS}\_i}}{n_{\mathrm{attempts\_RTS}\_i} \times i}, \tag{11}$$

where "i" designates the current frame number. Hence as the frame number increases, the information being used for estimation is older and thus is given lower weight. $n_{\mathrm{fail\_RTS}\_i}$ is the number of RTS packets sent and did not result in a CTS reply during frame i. $n_{\mathrm{attempts\_RTS}\_i}$ is the total number of attempts to send RTS packets during frame i.

Finally, the total priority is

$$\mathrm{Pr} = \frac{\mathrm{Pr}_1}{\mathrm{Pr}_2 \times \mathrm{Pr}_3}. \tag{12}$$

Each node keeps a table in its memory containing its lower-level neighboring nodes and their corresponding priorities. Since these priorities depend on the current time, they should be continuously calculated and the table updated. Updates must also come from newly overheard values of the battery capacity and TOLT in the RTS and CTS packets. The format of these packets is shown in Figure 3(a). Therefore, nodes rely on their history table to estimate the battery state of their lower-level neighbors when calculating the corresponding priorities. These history tables will be relatively accurate since nodes update them on every overheard RTS or CTS packet sent by their lower-level neighbors. In addition, since the wakeup/sleep schedule tends to make nodes adopt similar wakeup periods as their lower-level neighbors, any RTS/CTS packet sent by these nodes will be overheard by their neighbors who will consequently update their tables. The "Battery Cap" in the RTS/CTS packets represents the value of the current battery capacity of the sending node before the transmission (case of overheard RTS) or reception (overheard CTS) that is going to occur. The "Dur" field describes the duration of time needed to finish the data transmission and thus allows overhearing nodes to estimate the final battery capacity and time of last transmission (TOLT) according to the following equations:

$$\begin{aligned} \mathrm{BatteryCap} &= \mathrm{BatteryCap}_{\mathrm{RTS/CTS}} - i_{\mathrm{avg}} \times \mathrm{Dur}; \\ \mathrm{TOLT} &= \mathrm{RX\_time}_{\mathrm{RTS/CTS}} + \mathrm{Dur}, \end{aligned} \tag{13}$$

where "$\mathrm{RX\_time}_{\mathrm{RTS/CTS}}$" be the time of reception of the overheard RTS or CTS packets.

4.6. SN-MAC. The topology is now divided into clusters. Each cluster contains n levels of nodes. The level of a node within a cluster is defined by the number of hops the node lies away from the cluster head. Each node has PN-codes which it can use to spread any signal it needs to send. The goal of the nodes within a cluster is to sense and forward data towards the base station through cluster heads and intermediate nodes within the cluster. The first time a node has data to send, will broadcast an RTS on the control channel. All of its awake lower-level neighbors will wait for a random time then reply with a CTS. This random waiting is aimed to avoid packet collisions. After receiving the CTS packets, the node builds up a table of its lower-level neighbors. Notice that the table has in addition to the priority field, an NAV field which indicates the duration for which the node in the corresponding entry will remain busy. This NAV field is updated after overhearing an RTS or a CTS of the corresponding node and hence deducing that it will be busy for the time indicated by the duration field in theses packets. This table will be regularly updated with every overheard RTS and CTS. Since in our scheduling algorithm we aim at giving nodes the same schedule as their lower-level intermediate neighbors, we can expect that most of the RTS and CTS packets sent by nodes will be overheard by their higher-level neighbors who will then be able to update the corresponding priorities and NAV fields. Next, we give the steps taken by the nodes to successfully route data all the way towards the base station.

(1) When a node "n_i" at *level n* has data to send, it must first choose an intermediate lower-level neighbor to send its packet to and which in turn will further forward it to lower-level nodes. The first choice would be the nonbusy neighbor with the highest priority (Suppose that node is node B). n_i will send an RTS to node B at maximum power. Node B in turn will reply with a CTS packet except in the following cases.

   (i) Node B is busy (In this case n_i would have missed the RTS or CTS that should have told it that node B will be involved in a transmission and hence n_i would not have sent the RTS in the first place).

   (ii) Node B is asleep.

   (iii) Node B is both awake and nonbusy; however, when it received the RTS and computed the minimum power at which n_i needs to send at (so that the packet can be successfully recovered), this power came out to be larger than

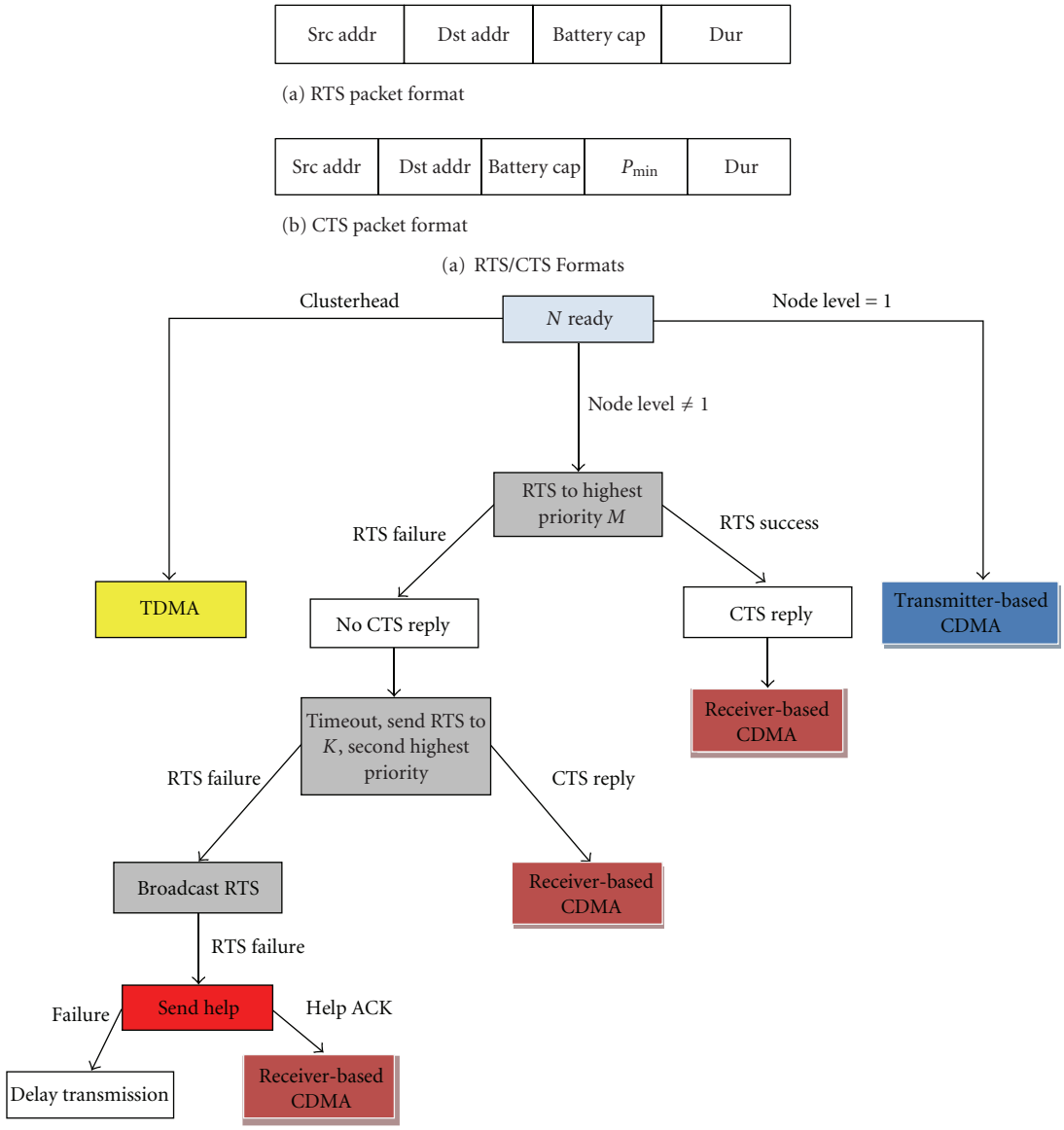

Figure 3:

the maximum power a node can use to send due to the very large MAI around the receiver. Therefore B will refrain from replying with a CTS packet.

(2) If node B does not reply for the above reasons, n_i will choose its nonbusy neighbor with second highest priority as its intermediate neighbor and will hence send it an RTS. On the other hand it will update its table after node B did not reply by changing the priority of node B to a value equal to the minimum priority in the table, that is, n_i will place node B at the bottom of its table. This is to avoid high energy neighbors from being constantly requested and hence stay busy all the time.

(3) If for the second time no CTS was received then n_i will broadcast an RTS. Several CTS packets will be received and the one with the highest priority is to be chosen. The broadcasting after two failures is done because of the latency requirement imposed by the application running our algorithm.

(4) If even after broadcasting the RTS, still no CTS packets arrive, this means that all lower-level neighbors of n_i are currently unavailable. n_i will then try to forward its packet through a neighbor with the same level by sending a *help* message. There are three cases in which a node might ask for help from another node on its level:

(i) node after broadcasting an RTS did not get a reply;

(ii) all the node's neighbors have an NAV > 0;

(iii) if there are only two or less available neighbors and have not replied on the RTS unicast.

The *help* message is a modified RTS with a help bit set to 1. The receiver of a help message checks the corresponding level against the node's level. If they match the node will reply with a help-ack message (*modified CTS*) in case it had lower-level nonbusy neighbors which it can route through. Further failure to receive replies will cause the node to delay its transmission to a later time.

(5) Upon receiving a CTS, a level n node ($n \neq 1$) switches its transceiver to the data channel, spreads the packet it wishes to send using the PN code of the desired receiver, and then transmits it using a receiver-based CDMA. The receiver in turn uses the same PN code to despread the packet sent.

(6) The packet will continue to be forwarded upstream (to lower-levels) using a receiver-based CDMA until it reaches a node with level one. Level one nodes transmit to the cluster head using transmitter-based CDMA. Notice that the cluster head also performs power control through the CTS packets it sends to its one-hop neighbors.

(7) The packet has successfully reached the cluster head. Cluster heads in turn communicate to the base station using a centralized TDMA schedule. This is done to provide load balancing in addition to fairness between all regions of a network.

5. Simulation Results

We successfully implemented our protocol in the Network Simulator (ns2). We simulated the DSSS (Directed Sequence Spread Spectrum) by adding a PN code attribute to the packet header. Hence, each time a packet is received, its PN code is checked against the PN codes monitored by the receiver. For comparison, we simulated SMAC using ns2. We decided to evaluate its effectiveness and relative performance in comparison to existing protocols such as S-MAC through simulations. Also, we compared SN-MAN to well-known CDMA-based MAC protocols such as CSMAC. The network in the simulations was subjected to change in size (number of nodes), change in average traffic (number of data packets originated per node) and topology change to plot the performance graphs of the three protocols in terms of network lifetime and data gathered over the network lifetime. We conducted different types of simulations. The first experiment was run to analyze the end-to-end latency. The second experiment analyzes the average network lifetime of the network using each MAC protocol. The third experiment analyzes the network throughput and the total data collected at the sink. The nodes take one of the following actions in a single time period: sense (sensor read), idle listen (where a node enables its transceiver so that it is ready to receive data or carrier sense), transmit a single packet, receive a single packet and sleep. All actions have a set power consumption value affixed to them. The radio propagation model in the simulation was assumed to be symmetric. We decided to ignore the processing action of the node due to its near negligible power consumption. Specifically, Table 1 shows the typical energy consumption of each action. All nodes were initialized with an energy capacity of 1000 Joules. The basic functionalities of S-MAC were incorporated in the simulation with the presence of both the message passing module and periodic listen and sleep. The sleep time for S-MAC was set to 600 ms. For a fair comparison between the three protocols, nodes sleep (and sense) for most of the time but only communicate during set predefined windows during which all three MAC protocols execute. In addition, we ensured that data traffic in the network was not constant so that neither protocol could gain advantage. We also made nodes move about randomly in the network to force a topology change from time to time. Finally, in order to obtain statistically significant results, we report average results of 10 simulations in each of the experiments carried out. The confidence level is 95%.

Table 1: Simulation parameters.

Routing	AODV
Propagation model	Log distance path loss
Path loss exponent	3.5
Antenna gain	1
System loss	1
Rx threshold	$1e-10$ W
Carrier sense threshold	$1e-11$ W
Max radio power	10 mW
ISM frequency band	2.4–2.4835 GHz
Power amplifier efficiency	33.33%
Number of nodes	*200*
Percentage of super nodes	*10%*
Topology	*Random*
Packet size	*36 bytes*
Initial energy/node	*100 J*
Transmission per packet	*0.5 J*
Reception per packet	*0.15 J*
Idle transceiver on	*0.105 J*
Sense	*0.3 J*
Sleep per second	*105 μJ*

5.1. Experiment I (Latency Analysis). Figure 4 shows the results of our *latency* evaluation for scenarios using SN-MAC, S-MAC and CS-MAC. Delivery latency in all the protocols under study increases as the hop count of the path increases. However, delivery latency in S-MAC increases at a much faster rate showing the benefit of SN-MAC's capability of multihop delivery within a single cycle and multiple transmissions in the vicinity of the receiver are possible using transmitter-based-CDMA. CSMAC is comparable to SN-MAC; however, the use of clustering in SN-MAC minimizes the latency. Also, CSMAC assumes that each node is aware

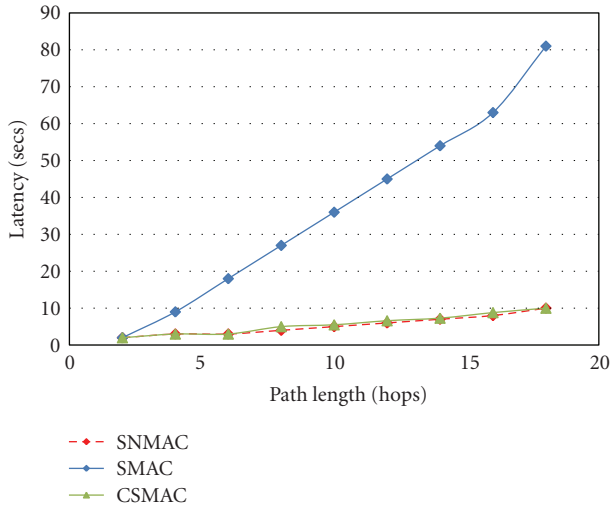

Figure 4: The delivery latency as the path length increases.

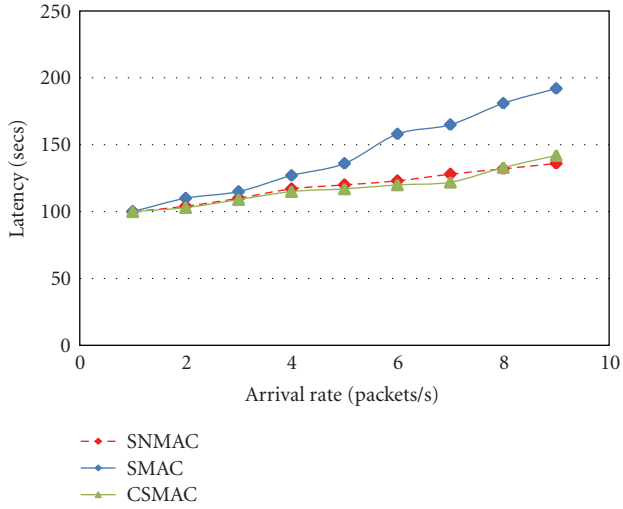

Figure 5: The delivery latency as the arrival rate increases.

of its own location which would be an extra overhead in analyzing the performance. We also simulated the *latency* as a function of the packet arrival rate of both SN-MAC and S-MAC shown in Figure 5. The difference between the latency incurred by SMAC and our protocol is much higher as the packet arrival rate increases and this is due to the use of DSSS for the transmission of data packets adopted by our algorithm.

5.2. Experiment II (Network Lifetime Analysis). In this experiment, we simulated the three protocols to observe how they affected the network lifetime. Usually, network lifetime is defined as the time span from deployment to the instant when the network is considered nonfunctional. However, at what point in time should a network be considered non functional is application-specific. We define network life time as the time taken for 85% of the nodes to die assuming after that point the network might be disconnected. The

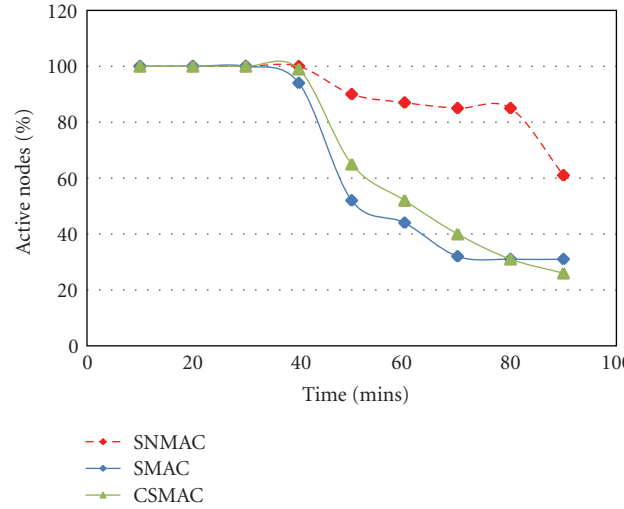

Figure 6: The network lifetime as time varies.

significance of our algorithm is evident in the overall network lifetime simulation as depicted in Figure 6. Our online battery model embedded with SN-MAC results in spreading the energy load on the whole network which results in increasing the overall lifetime of the network. The simulation was run for 1000 seconds and random events are generated at a rate of 5 events/second. Initially all the nodes were alive, but after only 41 and 50 minutes, less than 85% of the nodes are alive using S-MAC and CSMAC respectively however using SN-MAC it takes the network more than 80 minutes for the number of active nodes to go below 85% and thus highlighting the effect of SN-MAC on the network lifetime extension.

Also, from a different perspective (Figure 7), we studied the lifetime as we increase the number of deployed nodes. We can see an expected decline in all three graphs as more nodes are introduced in the network. However SNMAC clearly outperforms CSMAC and S-MAC. The latter two are decentralized and distributed in nature and this is the main cause of their underperformance, especially in larger networks. More nodes competing for the same medium during the contention phase results in an increase in the number of collisions even with carrier sense. Consequently, many nodes are prevented from getting access to the medium but at the same time are depleted of their energy whilst contending instead of transmitting data itself. This characteristic tends to shorten the network lifetime considerably in comparison to SNMAC where the contention is minimized.

5.3. Experiment III (Throughput Analysis). In Figure 8, we evaluated the *network throughput* using both SN-MAC and SMAC. Although network throughput is not a crucial metric in typical sensor networks, it is important when the traffic can potentially come in a burst. For both cases, the output rate follows the input rate when the input rate is low and finally the output rate reaches its peak point. Using SMAC, if we continue injecting more packets into the system, after the output has peaked, the input creates more contention

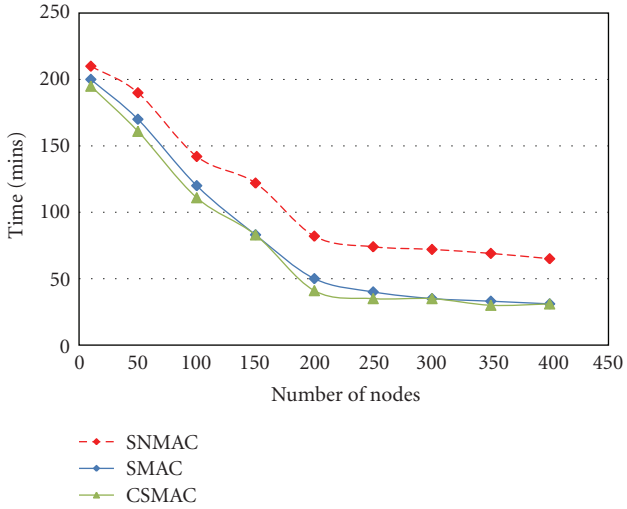

Figure 7: The network lifetime as the number of deployed nodes vary.

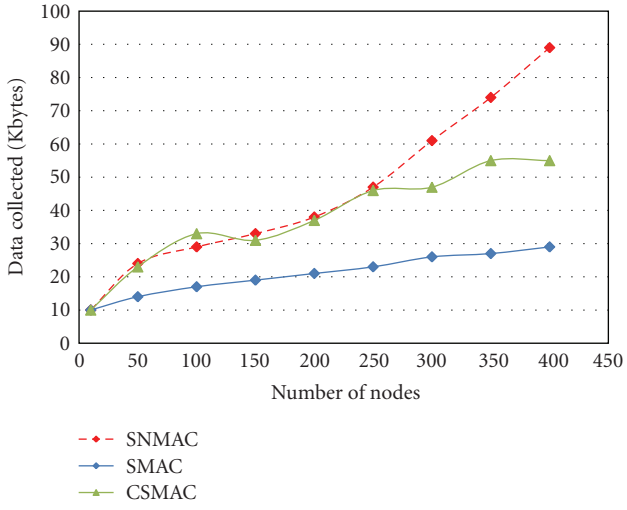

Figure 8: Total data collected at the sink over the lifetime of the network.

in the system and decreases the throughput slowly until the throughput reaches a steady-state value. Although the medium is saturated when the load is high, SN-MAC packets can still be forwarded whenever it is possible and thus uses its medium access opportunity more efficiently than with RTS/CTS in S-MAC. That is because of SN-MAC's capability of multihop delivery within a single cycle and multiple transmissions in the vicinity of the receiver are possible using transmitter-based-CDMA.

We also simulated the three protocols to observe how much data the sink (anchor) nodes gathered over the network lifetime. The results of simulations are shown in Figure 9. The plot shows the mean number of data packets collected over the network lifetime using each protocol. It can be seen that the three protocols perform similarly for extremely small sized networks involving a handful of nodes.

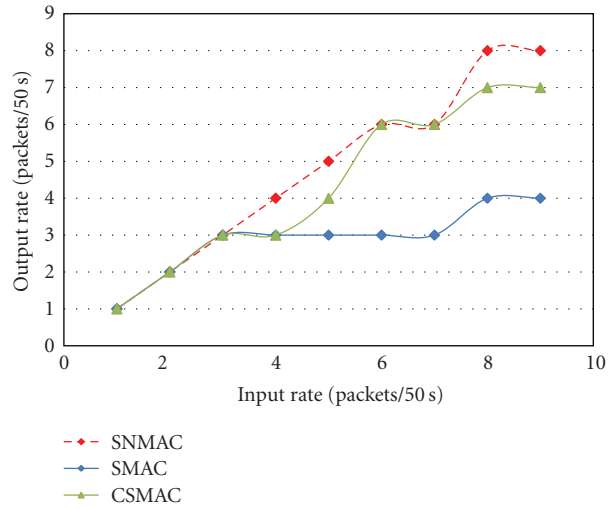

Figure 9: The analysis of the network throughput with the input rate.

However, as the number of nodes in the network begin to increase, the distinction between the three graphs becomes clearer where SNMAC starts outperforming CSMAC and SMAC. The shorter network lifetime of CSMAC and S-MAC ensues that less data is collected at the sink. The S-MAC network suffers most because of two reasons. Firstly, there is no presence of a slot (listen period) controller. Secondly, most S-MAC nodes tend to follow the same listen-sleep schedule. These two factors combine to create a bottle neck effect where intended receivers (relaying nodes) also compete for the medium to transmit their own data. This results in less data being transmitted at a high cost of collisions.

6. Conclusion

Sensor networks have always given energy efficiency much more importance than other requirements like latency and throughput. Duty cycle mechanisms have been used in sensor networks to improve energy efficiency, but they also introduce significant increase in end-to-end delivery latency and poor contention handling as well. In this paper, we have achieved a low latency delivery of data from sensing nodes towards the base station taking into consideration sources of energy wastage and successfully minimizing them. SN-MAC decreases the delivery latency and increases the throughput while extending the overall network lifetime. We believe that simulating accurately the DSS could result in better energy efficient results since our protocol gains energy efficiency by adopting a sleep/wakeup schedule, using the battery capacity of the nodes and minimizing the number of data collisions through CDMA with a separate data channel.

References

[1] E. S. Sousa and J. A. Silvester, "Spreading code protocols for distributed spread-spectrum packet radio networks," *IEEE Transactions on Communications*, vol. 36, no. 3, pp. 272–281, 1988.

[2] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, pp. 1567–1576, New York, NY, USA, June 2002.

[3] T. Van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 171–180, Los Angeles, Calif, USA, November 2003.

[4] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a hybrid MAC for wireless sensor networks," in *Proceedings of the International conference on Embedded Networked Sensor Systems*, pp. 90–101, San Diego, Calif, USA, 2005.

[5] B. H. Liu, N. Bulusu, H. Pham, and S. Jha, "CSMAC: a novel DS-CDMA based MAC protocol for wireless sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference Workshops (GLOBECOM '04)*, pp. 33–38, Dallas, Tex, USA, December 2004.

[6] S. De, C. Qiao, D. A. Pados, M. Chatterjee, and S. J. Philip, "An integrated cross-layer study of wireless CDMA sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1271–1285, 2004.

[7] C.-H. Liu and H. H. Asada, "A source coding and modulation method for power saving and interference reduction in DS-CDMA sensor network systems," in *American Control Conference*, pp. 3003–3008, May 2002.

[8] A. Muqattash and M. Krunz, "CDMA-based MAC protocol for wireless ad hoc networks," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '03)*, pp. 153–164, June 2003.

[9] S. Jayashree, B. S. Manoj, and C. S. R. Murthy, "On using battery state for medium access control in ad hoc wireless networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom '04)*, pp. 360–373, Philadelphia, Pa, USA, October 2004.

[10] A. Grilo, M. Macedo, and M. Nunes, "Tone-propagated MAC (TP-MAC): a low duty cycle low latency MAC protocol for wireless sensor networks," in *Proceedings of the Performance Control in Wireless Sensor Networks Workshop*, Coimbra, Portugal, 2006.

[11] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 321–334, Boulder, Colo, USA, November 2006.

[12] L. Wang and K. Liu, "An adaptive energy-efficient and low-latency MAC protocol for wireless sensor networks," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '07)*, pp. 2440–2443, September 2007.

[13] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS '04)*, pp. 224–231, Santa Fe, NM, USA, April 2004.

[14] L. F. W. van Hoesel and P. J. M. Havinga, "Design aspects of an energy-efficient, lightweight medium access control protocol for wireless sensor networks," Tech. Rep. TR-CTIT-06-47, Centre for Telematics and Information Technology, 2006.

[15] Y. Yang and C. Ma, "Battery aware routing for streaming data transmissions in wireless sensor networks," in *Proceedings of the 2nd International Conference on Broadband Networking (ICBN '05)*, vol. 1, pp. 464–473, 2005.

[16] A. A. Bertossi and M. A. Bonuccelli, "Code assignment for hidden terminal interference avoidance in multihop packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 441–449, 1995.

[17] L. Hu, "Distributed code assignments for CDMA packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, pp. 668–677, 1993.