AN ENERGY-EFFICIENT MAC PROTOCOL FOR WIRELESS SENSOR NETWORKS FOR WIDE AREA LARGE SCALE ENVIRONMENTAL MONITORING

by

Miguel Angel Erazo Villegas

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2006

Approved by:

Domingo Rodríguez, Ph.D. Member, Graduate Committee	Date
Manuel Rodríguez, Ph.D. Member, Graduate Committee	Date
Yi Qian, Ph.D. President, Graduate Committee	Date
Edgar Acuña, Ph.D. Representative of Graduate Studies	Date
Isidoro Couvertier, Ph.D. Chairperson of the Department	Date

ABSTRACT

An Energy-Efficient MAC Protocol for Wireless Sensor Networks for Wide Area Large Scale Environmental Monitoring

(May 2006)

Technical improvements in sensor networks have resulted in low cost sensor nodes that are suitable for a large number of applications. Environment monitoring, target tracking and border surveillance are just some of the many applications for today's sensor networks. For different application areas, there are different technical issues that researchers are currently solving.

Sensor nodes have various energy and computational constraints because of their inexpensive nature and ad-hoc method of deployment. Considerable research has been focused at overcoming these deficiencies through more energy efficient routing, localization algorithms and system design. Even though many issues of old sensor networks have been already solved, high energy consumption continues to be a hot topic in research papers. This is because it is hard and impractical to charge or replace exhausted batteries of nodes. Protocols that save more energy than its predecessors and at the same time comply with its computation and communication functions are desirable for wireless sensor networks.

This Thesis presents a proposal for the development of a Medium Access Control (MAC) protocol for wireless sensor networks that is appropriate for environmental monitoring purposes. An energy aware MAC protocol is proposed for wireless sensor networks. This protocol saves energy by substantially reducing idle-listening. It is a reactive protocol which permits nodes to transmit packets every time they wake-up but at the same time proposes a mechanism to transmit packets that contain interesting data when nodes should not normally be awake.

The proposed protocol uses two techniques to reduce energy consumption for wide area large scale environmental monitoring applications. First, it uses a sleep/listen schedule in which nodes awake when a sample from the environment is taken. Second, it uses a mechanism through which a node that has urgent packets to send wakes up other nodes to convey data to Base Station fur further processing. This Thesis presents simulation results comparing proposed protocol against another existing protocol. Results show that proposed protocol achieves less energy consumption than the other protocol.

ACKNOWLEDGEMENTS

During the development of my graduate studies in the University of Puerto Rico several persons collaborated directly or indirectly with my research. Without their support it would have been impossible for me to finish my work. That is why I wish to dedicate this section to recognize their support.

First of all I want to thank my family in Bolivia for their unconditional support, inspiration and love during all this time. At the same time, Adriana was a great source of inspiration and support all this years. I love you all.

I want to express a sincere acknowledgement to my advisor, Dr. Yi Qian because he gave me the opportunity to research under his guidance and supervision. I received motivation; encouragement and support from him during all my studies. I also want to thank Professor Domingo Rodriguez for giving me the opportunity to be part of his project WALSAIP and accomplish my Master's degree through his financing. The Grant from NSF 0424546 provided the funding and the resources for the development of this research. Finally, I want to thank Professor Manuel Rodriguez for his support and help.

TABLE OF CONTENTS

TABLE LIST

FIGURE LIST

Figures **Page**

1. INTRODUCTION

1.1 Wireless Sensor Networks

Wireless sensor networks are dense networks of small, low-cost sensors, which collect and disseminate data. Wireless sensor networks facilitate monitoring and controlling of physical environments from remote locations [1]. A sensor network is composed of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it. The position of sensor nodes need not be engineered or predetermined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this also means that sensor network protocols and algorithms must possess self-organizing capabilities. Another unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an onboard processor. Instead of sending the raw data to the nodes responsible for the fusion, they use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data.

1.2 MAC protocols for wireless sensor networks

The MAC is a sublayer of the Data Link Layer of the Open Systems Interconnection (OSI) model. Among other things, this layer is responsible for controlling the access of nodes to the medium to transmit or receive data. In traditional wireless voice or data networks, each user desires equal opportunity and time to access the medium, i.e., sending or receiving packets for their own applications. Per-hop MAC level fairness is, thus, an important issue. However, in sensor networks, all nodes cooperate for a single common task. At any particular time, one node may have dramatically more data to send than some other nodes. In this case fairness is not important as long as application-level performance is not degraded [5].

In wireless sensor networks, MAC protocols control how sensor nodes access a shared radio channel to communicate with neighbors. Traditionally, this problem is known as the channel allocation or multiple access problem. Though MAC protocols have been extensively studied in traditional areas of wireless voice and data communications (e.g. Time division multiple access (TDMA), frequency division multiple access (FDMA) and code division multiple access (CDMA) [8], ALOHA and carrier sense multiple access (CSMA) [9]), sensor networks require a MAC protocol that differs from those of traditional wireless voice or data networks in several ways. First of all, most nodes in sensor networks are likely to be battery powered and it is often very difficult to change batteries for all the nodes. Second, nodes are often deployed in an ad-hoc fashion rather than with careful pre-planning. Hence, after deployment the sensor nodes must quickly organize themselves into a communication network. Third, many applications employ large numbers of nodes. Finally, most traffic in the network is triggered by sensing events, and it can be extremely bursty. All these characteristics suggest that traditional MAC protocols employed in the past wireless networks are not suitable for wireless sensor networks without modifications.

Since our protocol is designed to detect extreme events (e.g. natural disasters like flooding), the system thus has to remain operational for months or years. Once a flooding is detected, this information must be forwarded to the system management quickly and accurately.

Contention based MAC protocols are suitable for wireless sensor networks because the required synchronization allows higher clock drifts; while TDMA requires a tight synchronization. Nodes do not have assigned a particular time or frequency to transmit so a node could sleep until a particular event happens that triggers nodes to active state.

1.3 Literature Review

Rappaport [8] provides a survey of MAC protocols for traditional and contention based networks. Though MAC protocols have been extensively studied in traditional areas of wireless voice and data communications (e.g. Time division multiple access (TDMA), frequency division multiple access (FDMA) and code division multiple access (CDMA), ALOHA and carrier sense multiple access (CSMA)), sensor networks require MAC protocols that differ from those used in traditional wireless voice or data networks in several ways [1]. First of all, most nodes in sensor networks are likely to be battery powered and it is often very difficult to change batteries for all the nodes. Second, nodes are often deployed in an ad-hoc fashion rather than with careful pre-planning. Hence, after deployment the sensor nodes must quickly organize themselves into a communication network. Third, many applications employ large numbers of nodes. Finally, most traffic in the network is triggered by sensing events, and it can be extremely bursty. All these characteristics suggest that traditional MAC protocols employed in the past wireless networks are not suitable for wireless sensor networks without modifications.

Recent advances in wireless communications and electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks. Sensor networks represent a significant improvement over traditional networks. A sensor network is composed of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it. The position of sensor nodes need not be engineered or predetermined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this also means that sensor network protocols and algorithms must possess self-organizing capabilities. Another unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an onboard processor. Instead of sending the raw data to the nodes responsible for the fusion, they use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. [1].

Wireless sensor networks are dense wireless networks of small, low-cost sensors, which collect and disseminate data. Wireless sensor networks facilitate monitoring and controlling of physical environments from remote locations with better accuracy. They have applications in a variety of fields such as environmental monitoring, military purposes and gathering sensing information in inhospitable locations. Sensor nodes have various energy and computational constraints because of their inexpensive nature and ad hoc method of deployment. Considerable research has been focused at overcoming these deficiencies through more energy efficient routing, localization algorithms and system design. Our survey attempts to provide an overview of these issues as well as the solutions proposed in recent research literature [2].

Wireless sensor networks are appealing to researchers due to their wide range of application potential in areas such as target detection and tracking, environmental monitoring, industrial process monitoring, and tactical systems. However, lower sensing ranges result in dense networks, which bring the necessity to achieve an efficient medium access protocol subject to power constraints. Various MAC protocols with different objectives were proposed for wireless sensor networks. Demirkol et al. [3] outlines the sensor network properties that are crucial for the design of MAC layer protocols. Then, they describe several MAC protocols for sensor networks emphasizing their strengths and weaknesses. Finally, they point out open research issues on MAC layer design.

Dam et al. [4] proposed T-MAC, a contention-based Medium Access Control protocol for wireless sensor networks. Applications for these networks have some characteristics (low message rate, insensitivity to latency) that can be exploited to reduce energy consumption. To handle load variations in time and location, T-MAC introduces an adaptive duty cycle in a novel way: by dynamically ending the active part of it. This reduces the amount of energy wasted on idle listening, in which nodes wait for potentially incoming messages, while still maintaining a reasonable throughput. They discuss the design of T-MAC, and provide a head-to-head comparison with classic CSMA (no duty cycle) and S-MAC (fixed duty cycle) through extensive simulations. Under homogeneous load, T-MAC and S-MAC achieve similar reductions in energy consumption (up to 98 %) compared to CSMA. In a sample scenario with variable load, however, T-MAC outperforms S-MAC by a factor of 5. Preliminary energy-consumption measurements provide insight into the internal workings of the T-MAC protocol.

Ye et al. [5] proposed S-MAC, a MAC protocol designed for wireless sensor networks. Wireless sensor networks use battery-operated computing and sensing devices. A network of these devices will collaborate for a common application such as environmental monitoring. These characteristics of sensor networks and applications motivate a MAC that is different from traditional wireless MAC such as IEEE 802.11 in several ways: energy conservation and self-configuration are primary goals, while per-node fairness and latency are less important. S-MAC uses a few novel techniques to reduce energy consumption and support self-configuration. It enables low-duty-cycle operation in a multihop network. Nodes form *virtual clusters* based on common sleep schedules to reduce control overhead and enable traffic-adaptive wake-up. S-MAC uses in-channel signaling to avoid overhearing unnecessary traffic. Finally, S-MAC applies *message passing* to reduce contention latency for applications that require in-network data processing. [5] presents measurement results of S-MAC performance on a sample sensor node, the UC Berkeley Mote, and reveals fundamental tradeoffs on energy, latency and throughput. Results show that S-MAC obtains significant energy savings compared with an 802.11-like MAC without sleeping.

Mainwaring et al. [7] provides an in-depth study of applying wireless sensor networks to real-world habitat monitoring. A set of system design requirements are developed that cover the hardware design of the nodes, the design of the sensor network, and the capabilities for remote data access and management. A system architecture is proposed to address these requirements for habitat monitoring in general, and an instance of the architecture for monitoring seabird nesting environment and behavior is presented. The deployed network consisted of 32 nodes on a small island off the coast of Maine streaming useful live data onto the web.

1.4 Summary of Following Chapters

We first give a brief introduction and literature survey in Chapter 1. Chapter 2 gives a overview of most related MAC protocols for sensor networks. Then, our Energy-Efficient MAC Protocol for Wireless Sensor Networks for Wide Area Large Scale Environmental Monitoring is presented in detail in Chapter 3. Chapter 4 gives simulation study and analysis of the proposed protocol, and compares it with previous protocols. Conclusions are presented in Chapter 5.

2. THE BACKGROUND

2.1 Wireless Sensor networks for environmental monitoring

Researchers in the Life Sciences are becoming increasingly concerned about the potential impacts of human presence in monitoring plants and animals in field conditions. At best it is possible that chronic human disturbance may distort results by changing behavioral patterns or distributions [7]. Disturbance effects are of particular concern in small island situations, where it may be physically impossible for researchers to avoid some impact on an entire population. In addition, islands often serve as refuge for species that cannot adapt to the presence of terrestrial mammals, or may hold fragments of once widespread populations that have been extirpated from much of their former range. These considerations are of particular importance for projects like Wide Area Large Scale Automated Information Processing (WALSAIP) which will be deployed in the island of Puerto Rico because of the sensitive existing flora and fauna. For purposes of automation of data collection and reduction of human intervention in areas of interest, sensor networks have been considered. It is expected that deployed sensor networks in Puerto Rico do not interfere with existing life.

Recent developments in wireless network technology and miniaturization now make it possible to realistically monitor the natural environment. These systems can provide new data for environmental science, such as climate models, as well as vital hazard warnings such as flood alerts [13]. Instrumenting natural spaces with numerous networked microsensors can enable long-term data collection at scales and resolutions that are difficult, if not impossible, to obtain otherwise [7]. The intimate connection with its immediate physical environment allows each sensor to provide localized measurements and detailed information that is hard to obtain through traditional instrumentation. The combination of storage and in-node processing enable them to perform triggering functions suitable for some applications and protocols.

2.1.1 Generic Architecture

Sensor networks are designed to transmit data from an array of sensors to a data repository or server that will store it for further analysis. Figure 1 shows a generic architecture for an environmental sensor network. Sensor nodes gather data autonomously, and the network passes this data to one or more base stations, which forward the data to a server. The wireless sensor networks which have sensing, computation and communication functions to move packets from sensor nodes to final servers, consume quantities of energy that must be taken into account to forecast the life cycle of a network and maximize it.

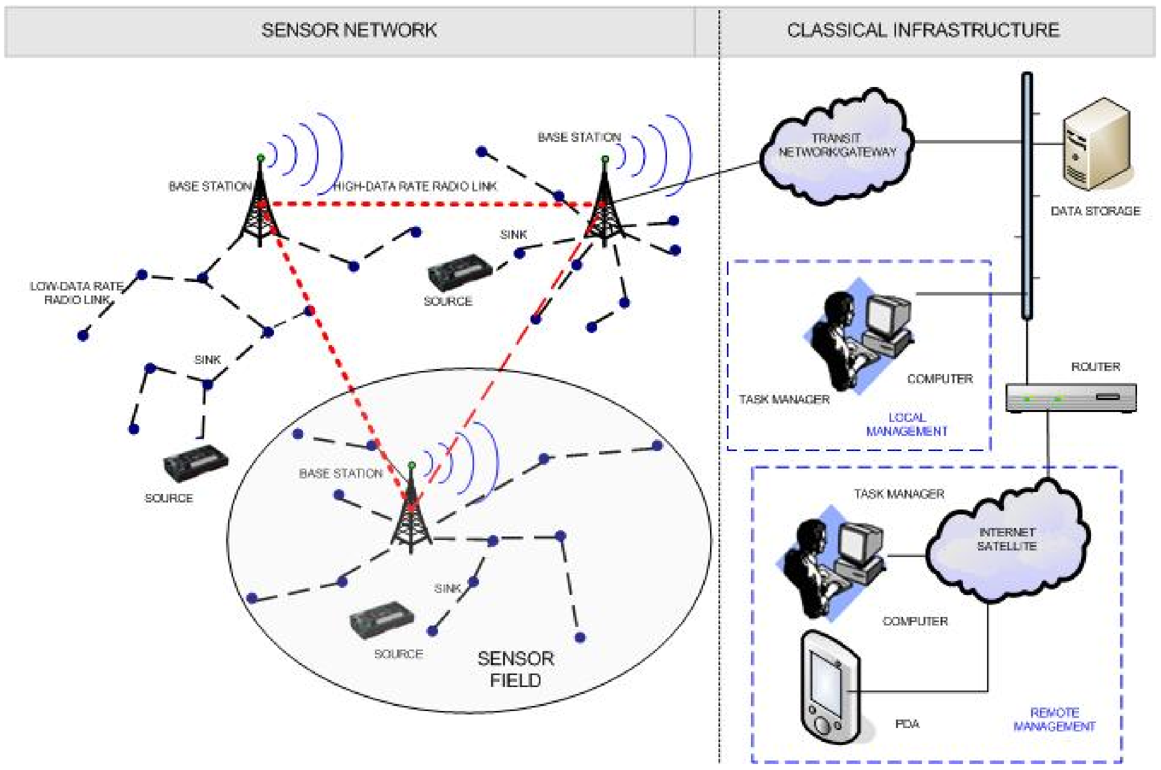


Figure 1: Illustration of sensor network and backbone infrastructure

Sensor nodes are the network components that will be sensing and delivering the data. According to the system application requirements, nodes may do some computations. After computations, it transmits its data to its neighboring nodes or simply passes the data as it is to the Task Manager. Sensor nodes can act as a source or sink in the sensor field. The function of a source is to sense and deliver the desired information (see Figure 1). Thus, a source reports the state of the environment. On the other hand, a sink is a node that is interested in some information a sensor in the network might be able to deliver.

Gateways allow the scientists and system managers to access nodes through personal computers (PCs), personal digital assistants (PDA) and Internet. In a nutshell, gateways act as a proxy for the sensor network on the Internet. Gateways could be classified as active, passive, and hybrid. Active gateway allows the sensor nodes to actively send its data to the gateway server. Passive gateway operates by sending a request to sensor nodes. Hybrid gateway combines capabilities of the *active* and *passive* gateways.

The Task Manager will connect to the gateways via some media like Internet or satellite. Task Managers comprise of data service and client data browsing and processing. These Task Managers can be visualized as the information retrieval and processing platform. All information (raw, filtered, processed) data coming from sensor nodes is stored in the task managers for analysis. Users can use any display interface (i.e. PDA, computers) to retrieve or analyze data either locally or remotely (see Figure 1).

2.1.2 Habitat monitoring applications

Habitat-monitoring applications consist of multiple software components implementing core system services. Because they require ways to specify and deliver data of interest, they need a routing and tasking service [14]. Besides, long-term operation dictates that the system operate in low-power mode; current applications achieve this goal via duty cycling, or changing the amount of time the subsystem is active during any given period, at several levels.

The routing service in habitat-monitoring networks delivers the queries to the sensor nodes and reports the data of interest; that data is either sampled (such as humidity sampled every certain time) or triggered (such as when an animal enters the area of interest). The service copes with poor-quality links and dynamic topology changes; all these features need to be robust and consume only minimal resources on the constrained nodes. Fortunately, in many cases the actual deployment simplifies the general routing problem. For example, on Great Duck Island [7], it was sufficient to provide tree-based routing for data collection and simple flooding for parameter dissemination. The data to be gathered was specified ahead of time; sensor motes self-organized into a tree rooted at the patch gateway. A set of commands setting sampling rates, reporting immediate status, and invoking calibration procedures was flooded through the network, and the acknowledgements flowed using the tree-based routing.

2.1.3 Design Factors

The following factors are important as a guideline to design a protocol for wireless sensor networks [1]:

- Fault tolerance. The failure of some sensor nodes must not affect the overall task of the network.

- Scalability. The protocol must be able to work in dense wireless sensor networks.

- Production costs. The cost of producing a node must be kept low.

- Hardware constraints. A sensor node is made up of four basic components: a *sensing unit*, a *processing unit*, a *transceiver unit*, and a *power unit.* Apart from size, there are some other stringent constraints for sensor nodes. These nodes must consume extremely low power, operate in high volumetric densities, have low production cost, be dispensable and autonomous, operate unattended, and be adaptive to the environment [1].

- Sensor network topology. Wireless sensor network protocol must be able to maintain network topology so the network can accomplish its objectives.

- Power consumption. In a multihop wireless sensor network, each node plays the dual role of data originator and data router. The malfunctioning of a few nodes can cause significant topological changes and might require rerouting of packets and reorganization of the network. Hence, power conservation and power management take on additional importance. Our protocol will focus on reducing network energy consumption.

2.2 Related MAC protocols for Wireless Sensor Networks

In the following paragraph S-MAC and T-MAC are presented. It is important to describe protocol that will serve as base for ours. Our new MAC protocol is developed based on many S-MAC characteristics.

2.2.1 S-MAC

2.2.1.1 Basic characteristics

A wireless sensor network MAC protocol must have the characteristics described below in order to be suitable for applications on these sorts of networks [5]:

- Energy efficient in such a way that nodes spend the least amount possible of energy in their communication and computation functions. Communication procedures consume much more energy than those of computation, so this protocol concentrated on communication issues;
- Scalable to allow the network which is running it to grow without compromising too much its performance;
- Adaptable to changes not only due to new nodes that enter the network but also due to nodes that fail in their normal operation;

Major sources of energy wastage are identified in [3] and [5] and described below:

- Collisions: packets discarded have to be retransmitted, which increase energy consumption;
- Overhearing: a node listens to packets destined to other nodes;
- Control packet overhead: transmission and reception of control packets consumes energy;
- Idle listening: Listening to receive possible packets that are not sent;
- Overemitting: Transmission of a packet when destination node is not ready.

S-MAC takes into account characteristics described above taking more emphasis on reducing energy consumption by reducing idle listening and control packet overhead. This protocol trades off energy consumption with latency. Low latency is a desirable feature for traditional networks that do not have the constraint of energy.

2.2.1.2 Synchronization

To minimize the idle listening energy consumption, S-MAC uses a sleep-listen schedule where a node remains inactive for a long time and them wakes up to transmit or receive packets. When slept, a node turns off its radio saving a lot of energy. Figure 2 shows S-MAC sleep-listen schedule. A *frame* is a complete cycle of listen and sleep times. The *duty cycle* is defined as the ratio of the listen interval to the frame length.

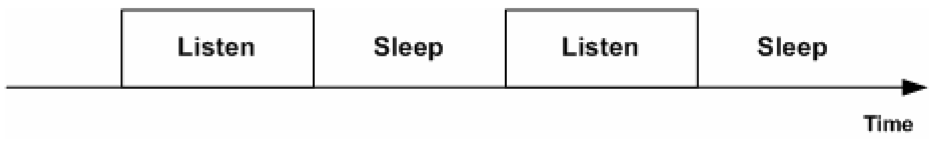


Figure 2. Basic Sleep-Listen schedule

Nodes synchronize each other by periodically broadcasting Synchronization (SYNC) packets. A node could follow more than one schedule if received more than one SYNC packet when synchronizing for the first time. In such case, the node follows both schedules. SYNC packets are very short and include the address of the sender and the time of its next sleep. The time of the next sleep is relative to the moment that the sender starts to send its SYNC packet rather than absolute. SYNC packets and data ones are sent on different time slots as shown in Figure 3.

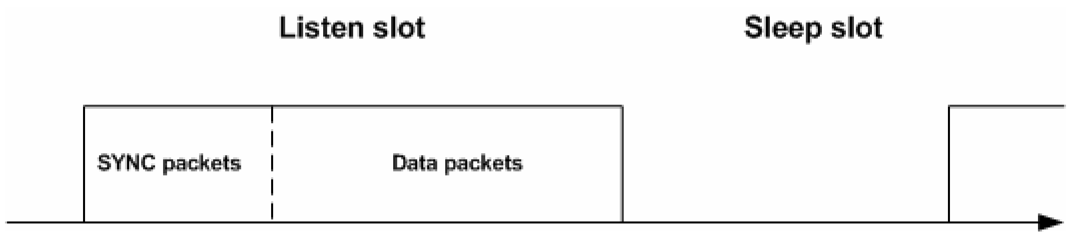


Figure 3. The Listen slot is divided

.

2.2.1.3 Collision avoidance

To avoid collisions, S-MAC uses the 802.11 mechanism: RTS/CTS/DATA/ACK. Only broadcast packets do not follow RTS/CTS/DATA/ACK sequence. A second mechanism that S-MAC uses to avoid collision is the vector called Network Allocation Vector (NAV). In such vector, it keeps the duration of transmission of node's neighbors. In this way, the node knows when the medium will be most likely idle to transmit a data packet. This mechanism is called virtual carrier sense. Furthermore, S-MAC uses Carrier Sense (CS) before transmitting a SYNC or data packet to verify if there are current transmissions or not, this mechanism is a physical sense.

Within a frame, a node can transmit only a SYNC packet, only a data packet or both. This is shown in Figure 4. The figure highlights the procedure RTS/CTS/DATA that S-MAC uses to send a data packet avoiding collisions.

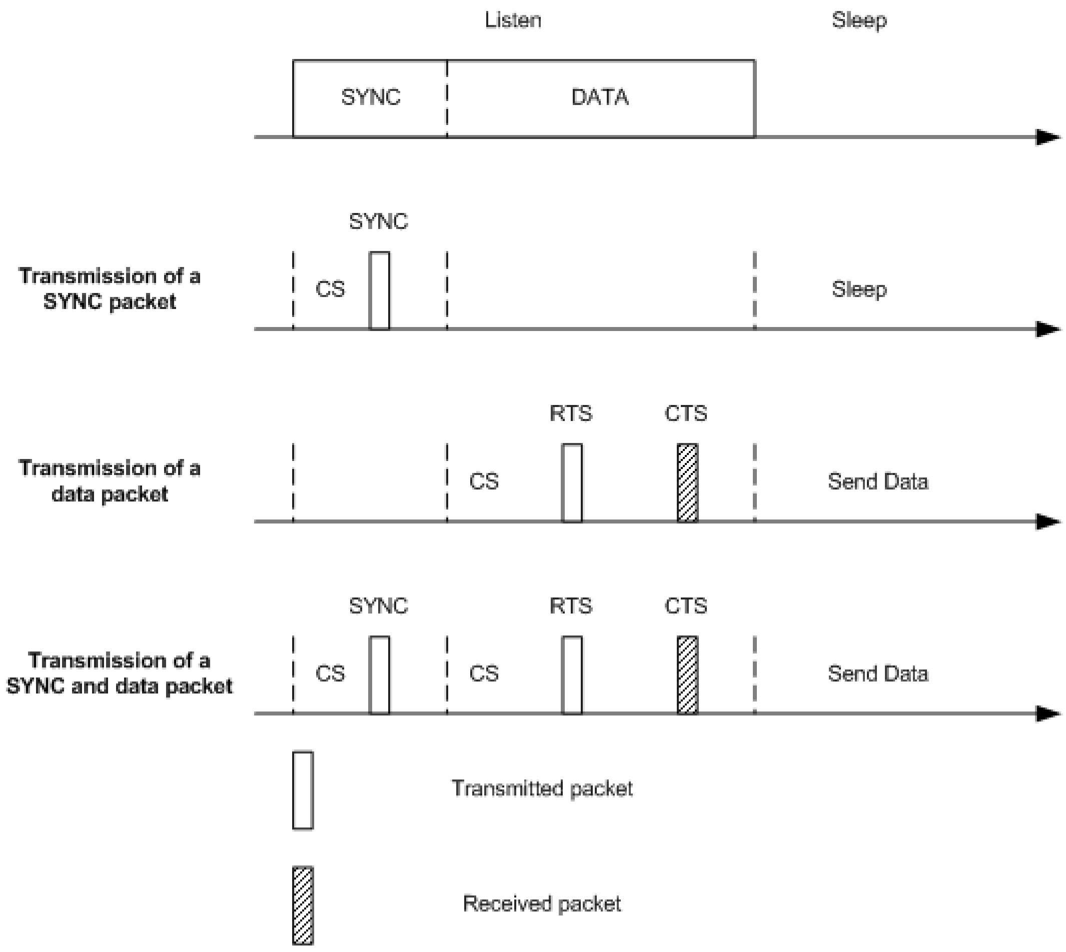


Figure 4. Ways to send a data packet and synchronize other nodes.

2.2.1.4 Adaptive Listening

Adaptive listening is a mechanism to turn nodes from low-duty cycle to a more active state to reduce latency trading it off with more energy consumption. The basic idea is to let the node who overhears its neighbor’s transmissions (ideally only RTS or CTS) wake up for a short period of time at the end of the transmission. In this way, if the node is the next-hop node, its neighbor is able to immediately pass the data to it instead of waiting for its scheduled listen time. If the node does not receive anything during the adaptive listening, it will go back to sleep until its next scheduled listen time. If the next-hop node is a neighbor of the sender, it will receive the RTS packet. If it is only a neighbor of the receiver, it will receive the CTS packet from the receiver. Thus, both the neighbors of the sender and receiver will learn about how long the transmission is from the duration field in the Request-to-Send (RTS) and Clear-to-Send (CTS) packets. In this way, they are able to adaptively wake up when the transmission is over.

It should be noted that not all next-hop nodes can overhear a packet from the previous transmission, especially when the previous transmission starts adaptively, i.e., not at the scheduled listen time. So if a sender starts a transmission by sending out an RTS packet during the adaptive listening, it might not get a CTS reply. In this case, it just goes back to sleep and will try again at the next normal listen time.

2.2.1.5 Overhearing avoidance

Inspired by PAMAS [10], S-MAC tries to avoid overhearing by letting interfering nodes go to sleep after they hear an RTS or CTS packet. Since DATA packets are normally much longer than control packets, the approach prevents neighboring nodes from overhearing long DATA packets and following ACKs. All immediate neighbors of both the sender (node C) and receiver (node D) should sleep after they hear the RTS or CTS until the current transmission is over, as indicated by “X ” in Figure 5.

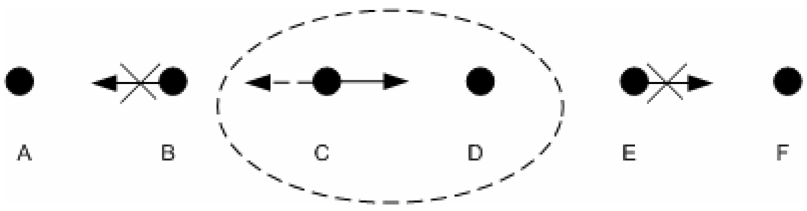


Figure 5. Nodes that should go to sleep when C transmits to D

As stated in section 2.2.1.3, every node when overhears a transmission updates its NAV. To avoid overhearing, a node goes to sleep when NAV is not zero (indicating transmission in progress). Node awakes again as normally when its NAV is zero.

2.2.1.6 Message Passing

Message Passing consists in fragmenting the long message into many small fragments, and transmitting them in a burst. Only one RTS and one CTS are used. They reserve the medium for transmitting all the fragments. Every time a data fragment is transmitted, the sender waits for an ACK from the receiver. If it fails to receive the ACK, it will extend the reserved transmission time for one more fragment, and re-transmit the current fragment immediately. If a neighboring node hears an RTS or CTS packet, it will go to sleep for the time that is needed to transmit all the fragments. Each data fragment or ACK also has the duration field. In this way, if a node wakes up or a new node joins in the middle of a transmission, it can properly go to sleep no matter if it is the neighbor of the sender or the receiver. If the sender extends the transmission time due to fragment losses or errors, the sleeping neighbors will not be aware of the extension immediately. However, they will learn it from the extended fragments or ACKs when they wake up.

2.2.1.7 Comparison of Proposed Protocol with S-MAC

S-MAC was chosen as the base for our Proposed Protocol because of the following reasons:

- First, S-MAC focus on energy savings while other protocols like D-MAC concentrates more on reducing latency, which is not our objective. Second, S-MAC achieves better throughput that T-MAC [4], which is very important for us since it is expected that no packets are lost, especially in event-triggered situations. Finally, although no direct comparisons have been made between S-MAC with adaptive listening and T-MAC, S-MAC with adaptive listening has greatly improved energy efficiency compared to the first version. In this way, it is expected that S-MAC in its final version have similar energy consumption that T-MAC. However, S-MAC has other advantages as described below. In the future, our protocol could adopt characteristics of any other protocols as long as they leverage more energy savings.

- Availability of code for ns-2. S-MAC code is widely used in the ns-2 community. Furthermore, authors of this protocol provide users quick support for it. These two factors shortened the development of our protocol.

- TinyOS, in its different versions, provides code for S-MAC. Additionally, application layer code is also provided that facilitates the development of applications that use S-MAC characteristics.

2.2.2 IEEE 802.11 MAC protocol

IEEE 802.11, the Wi-Fi standard, denotes a set of Wireless LAN/WLAN standards developed by working group 11 of the IEEE LAN/MAN Standards Committee (IEEE 802) [11].

This section describes most important characteristics of the standard 802.11 that are also used in S-MAC protocol and in our proposed protocol.

2.2.2.1 Medium Access

The basic medium access protocol of IEEE 802.11 is a Distributed Coordination Function (DCF) that allows for automatic medium sharing between compatible Physical layers (PHY) through the use of CSMA/CA and a random backoff time following a busy medium condition.

For medium reservation, nodes involved in communication, exchange RTS and CTS frames prior to the actual data frame. The RTS and CTS frames contain a Duration/ID field that defines the period of time that the medium is to be reserved to transmit the actual data frame and the returning Acknowledge (ACK) frame. All nodes within the range of the originating or the receiving node shall learn of the medium reservation. Another means of distributing the medium reservation information is the Duration/ID field in Directed frames. This field gives the time that the medium is reserved.

2.2.2.2 Carrier sense mechanism

Physical and virtual carrier sense functions are used to determine the state of the medium. When either function indicates a busy medium, the medium shall be considered busy; otherwise, it should be considered idle.

The physical carrier sense shall be provided by the PHY while virtual carrier sense shall be provided by the MAC. The NAV maintains a prediction of future traffic on the medium based on duration information that is announced in RTS/CTS frames prior to the actual exchange of data. The carrier sense mechanism combines the NAV state and the node’s transmitter status with physical carrier sense to determine the busy/idle state of the medium. The NAV may be thought of as a counter, which counts down to zero at a uniform rate. When the counter is zero, the virtual carrier sense indication is that the medium is idle; when nonzero, that it is busy. The medium shall be determined to be busy whenever the node is transmitting.

2.2.3 T-MAC

Timeout-MAC (T-MAC) [4] is proposed to enhance the performance results of S-MAC protocol under variable traffic load. In T-MAC, *listen period* ends when no activation event has occurred for a time threshold (TA). This operation makes T-MAC's schedule variable instead of the fixed schedule proposed in S-MAC. Figure 6 depicts the basic operation of T-MAC.

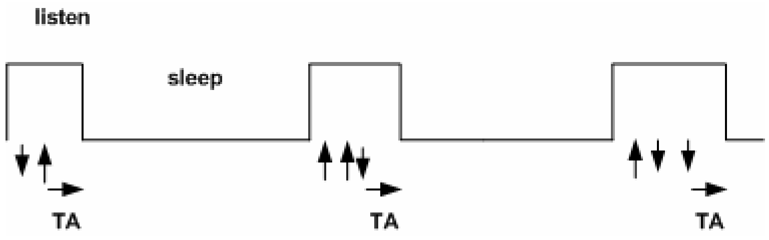


Figure 6. T-MAC basic operation

T-MAC synchronization is similar to that of S-MAC [5]. SYNC packets are exchanged between nodes to form virtual clusters that share the same synchronization. A node can run more than one synchronization scheme.

The scheme used to contend for the medium is the well known RTS/CTS/DATA/ACK. However, T-MAC proposes a change in this scheme that is used to avoid the early sleep problem [4]. The early sleep problem is the excessive contention for a node that wants to transmit to its neighbors. To avoid this problem T-MAC proposes two solutions: *Future request to send* and *priority on full buffers* [4].

Simulations have shown that the T-MAC protocol introduces a way of decreasing energy consumption in a volatile environment where the message rate fluctuates, either in time or in location. Implementation of the T-MAC protocol has shown that, during a high load, nodes communicate without sleeping, but during a very low load, nodes will use their radios for as little as 2.5% of the time, saving as much as 96% of the energy compared to a traditional protocol.

2.2.4 D-MAC

The main objective of DMAC [12] is to achieve very low latency and still be energy efficient. For that purpose it is designed to overcome S-MAC problems: increased delivery latency because a packet can not reach the sink in a single listen period, fixed duty cycles that do not adapt to traffic changes and the increased possibility of collisions due to synchronous duty cycle.

Lu et al. [12], identifies a problem that exists in implicit *sleep delay* reducing protocols: S-MAC [5] and T-MAC [4]. *Sleep delay* refers to the time that a packet suffers since it is transmitted from originating node to the sink (Base Station). This delay refers to the time that transmitting node has to wait for intended receiving node to wake up and receive the packet. It is shown in [12] that S-MAC and T-MAC only solve this problem for a two-hop path. If a network is a multi-hop one with more than 2 hops, the solutions provided by S-MAC and T-MAC are not appropriate [12]. This problem is identified as *Data forwarding Interruption* (DFI). Lu et al. [12] identified that the limited overhearing range due to radio sensitivity is the origin of this DFI problem.

The solution proposed is a *staggered active/sleep schedule* shown in Figure 7. Low latency is achieved by assigning subsequent slots to nodes that are successive in the data transmission path. With this scheme it is expected that a packets do not suffer from sleep delay at all because the next intended receiving node must always be awake when transmitting node wants to transmit a packet to it.

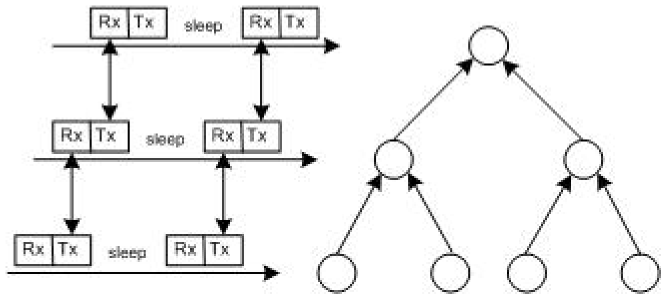


Figure 7. Staggered active/sleep schedule in D-MAC

DMAC staggers the active/sleep schedule of the nodes in the data gathering tree according to its depth in the tree. This allows continuous packet forwarding flow in which all nodes on the multi-hop path can be notified of the data delivery in progress as well as any duty-cycle adjustments. Simulation results have shown that DMAC achieves both energy savings and low latency when used with data gathering trees in wireless sensor networks.

3. THE PROPOSED ENERGY-EFFICIENT MAC PROTOCOL

3.1 Protocol operation

As explained in section 2.1.2, there are two types of data: sampling and triggered. *Sampling* data is obtained by sampling a certain parameter a given number of times every day while *triggered* data is disseminated after a certain event has happened. For energy saving purposes, it is important to differentiate between these two types of data.

The proposed energy-efficient MAC protocol is proposed to exploit the advantages that sampling data has from an energy saving perspective and, at the same time, cope with latency requirements of triggered data.

Sampling data has two great advantages: first, the number of samples to take in a given period of time is known in advance and second, instants to take the samples are also known. This fact leads us to the idea that between two consecutive sample instants, the communication functions of two nodes is almost null. In this way, the proposed protocol exploits this fact to save energy turning off its radio between two consecutive sample instants for data transmissions. Significant energy savings can be achieved by this operation if we take into account that idle listening is the most energy consuming operation of all [3][5]. However, if the radio is simply turned off, no triggered packets can be transmitted from originating nodes to the base station in a reasonable time. In such situation, triggered packets would be queued up and would also wait for the next available active time slot to be transmitted; what would create a long delay for triggered data, which would ideally have to be transmitted without delay. Furthermore, collisions would increase dramatically because all nodes in the network would content for the medium when the next time slot started.

Proposed energy-efficient MAC protocol meets inherent requirements of triggered packets by creating a mechanism to wake-up nodes so they are able to move the packet through the network. Specifically, it is proposed a third time slot inside the listen interval within a frame [5]. This time slot must be large enough to permit nodes to transmit and receive a very small packet and much smaller than data slot times. This time slot will be useful to signal other nodes to wake up at instants they are supposed to be sleeping to avoid idle listening.

The proposed energy-efficient MAC protocol inherits some characteristics of S-MAC with coordinated adaptive sleeping. The RTS/CTS/ACK mechanism is used for exchange of packets between nodes. In the following paragraphs, the proposed energy-efficient MAC protocol is explained step by step.

3.2 Basic listen/sleep schedule to avoid idle listening

To have an energy-efficient protocol that transmits packets only every certain period of time, the radio should be turned off between two consecutive samples. It is therefore conceived a sleep/listen schedule whose listen periods coincide with specific instants of time when samples are taken. Figure 8 shows the most basic intended sleep/listen schedule to save great amounts of energy in idle listening.

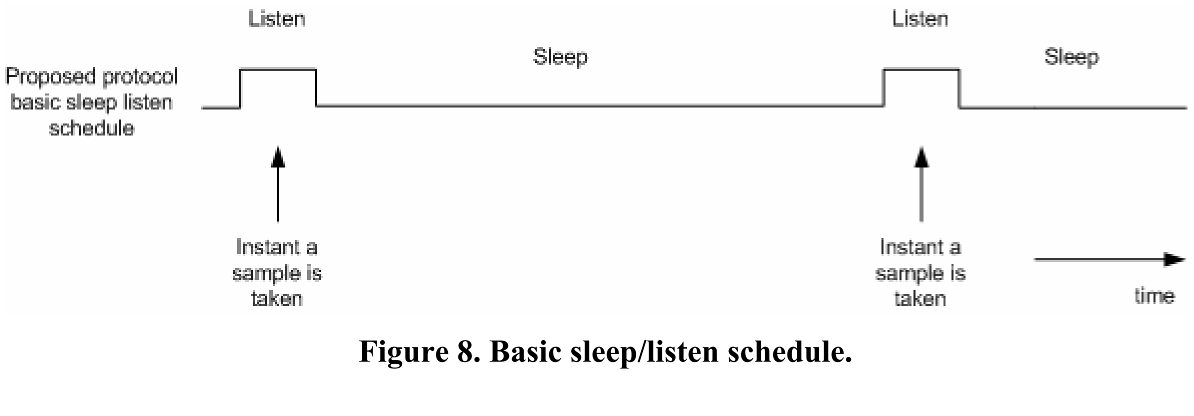

Listen
Sleep
Listen
Sleep
Proposed protocol basic sleep listen schedule
Instant a sample is taken
Instant a sample is taken
time

Figure 8. Basic sleep/listen schedule.

For remote management purposes and to satisfy the flexible requirements, it is desirable to change listen/sleep schedule anytime. Such operation must not be traumatic for network operation since useful samples could be lost due to long synchronization intervals of time. Our protocol proposes that initial synchronization between nodes continues to run but at a lower level than actual synchronization (Figure 9).

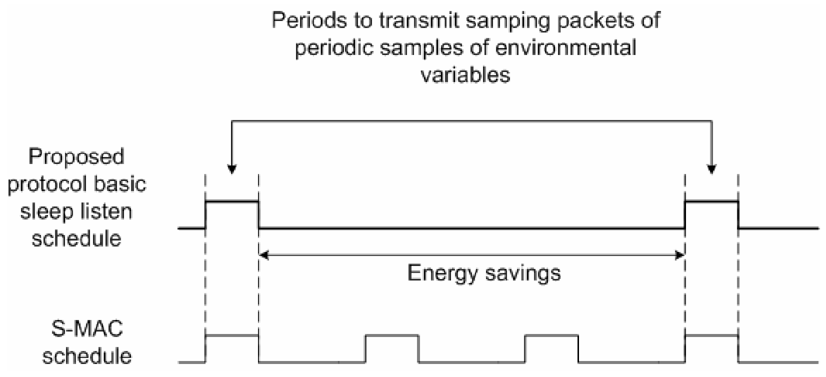

Periods to transmit samping packets of periodic samples of environmental variables
Proposed protocol basic sleep listen schedule
Energy savings
S-MAC schedule

Figure 9. Two schedules in each node

Sleep/listen schedule can be easily changed to meet scientists' requirements by the dissemination of appropriate control packets through the network. Such control packets ought to contain, al least, information of:

- *Time between Consecutive sets of Active Listen Periods (TCALP).* Parameter defined by the desired granularity in samples and strongly dependent on the variable sampled from environment.

- *Number of S-MAC Listen Periods in a set (NSLP).* A single listen period may not be enough if large quantities of data are expected. This parameter is subject to the type of environmental variable being measured. For example, temperature sampling contains much less data and originates less correspondent packets than a video sensor network for environmental monitoring [15].

With information provided as those described above, the synchronization can be configured as desired by network administrators as required by scientists to fulfill their needs to study the environment. Figure 10 shows a typical synchronization scheme for our proposed protocol for sampling packets and at the same time avoid idle listening.

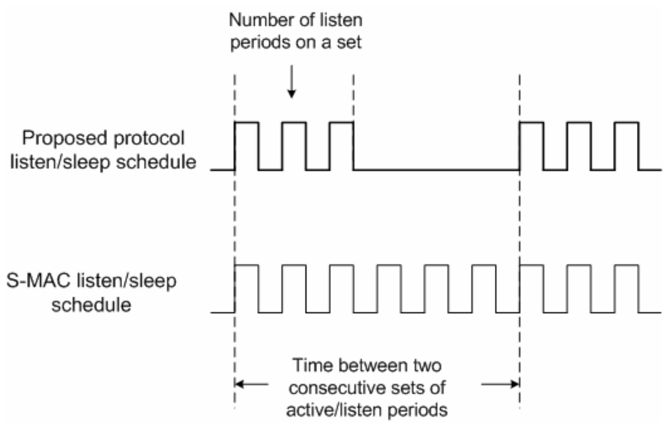

Number of listen periods on a set
Proposed protocol listen/sleep schedule
S-MAC listen/sleep schedule
Time between two consecutive sets of active/listen periods

Figure 10. Proposed protocol synchronization

3.3 Tone Time

In the following sections, the issues of the proposed sleep/listen schedule are described as well as the challenges of managing triggered packets. Later, improvements made to the sleep/listen schedule to meet requirements of triggered packets are described.

3.1.1 Basic listen/sleep schedule drawbacks

The proposed sleep/listen schedule presented shown in Figure 10 and described in above paragraphs is appropriate to avoid idle listening. The nodes of the wireless sensor network awake when a sample is going to be taken and then they sleep until the next sample. Much energy is saved with this operation but this schedule is not the most efficient for triggered packets because of the following reasons:

- *High Latency*. If triggered packets are generated between two consecutive sets of active listen periods, they would have to wait an interval of time between an S-MAC frame and a TCALP to be transmitted. If the granularity of sampling is not high, then TCALP would be large and therefore triggered packets would have to wait too long to be disseminated through the network. It is a basic objective of wireless sensor networks that latency for triggered packets be minimized since they contain crucial information that must be analyzed as soon as possible.

- *Queue overflow*. Triggered packets not transmitted are queued up and expect for next active available listen slot to be transmitted. A massive event like a sudden flooding would generate large number of triggered packets that would be put in the queue. If so many packets are generated, the queue limit can be exceeded with the consequent loss of crucial packets. A whole analysis of a sudden event can be damaged because of a chunk of packets lost.

- *Collisions*. Even if queued up packets do not overflow the queue of a node, they will wait for the next available listen time slot to contend with other node's packets to be transmitted. This situation is even worse if it is considered that all nodes will try to gain the medium at the same time when they awake in the next listen period. Thus, it is important that all triggered packets be transmitted as soon as possible to decrease latency, decrease use of queue and avoid collisions.

It is concluded that the sleep/listen schedule presented in previous paragraphs is indeed useful to lower idle listening wastage of energy but neither is it appropriate to decrease latency nor to avoid collisions. It is needed a sleep/listen schedule to address, at the same time: idle listening, latency, queue overflowing and collisions.

3.1.2 Modified sleep/listen schedule

As explained on above paragraphs, the initially proposed sleep/listen schedule is indeed effective to avoid idle listening; however the schedule proposed does not address the requirements of triggered packets. This section describes the additions made to initially proposed sleep/listen schedule to meet the requirements of triggered packets and at the same time avoid idle listening.

As stated in Section 2.2.1.3, S-MAC sleep/listen schedule contains two intervals: SYNC and DATA. SYNC interval is a time slot used to send synchronization information and DATA is used for RTS/CTS/DATA sequence. In the proposed protocol, between two sets of active listen periods the radio is turned off since the very beginning of the SYNC interval. Therefore, there is no way to signal other nodes to wake up is a sudden event occurs. Our approach will use the in-channel signaling and not a two-radio approach as other protocols do [10] [16].

The proposed energy efficient MAC protocol introduces a new time interval that will be used for a node that has detected an extraordinary event and generated its correspondent packets to signal other nodes in the wireless sensor network that there are triggered packets that need to be disseminated through the network as soon as possible. With this scheme, triggered packets will be sent in the next active listen period of the underlying S-MAC synchronization (see Figure 9). The proposed approach is depicted if Figure 11.

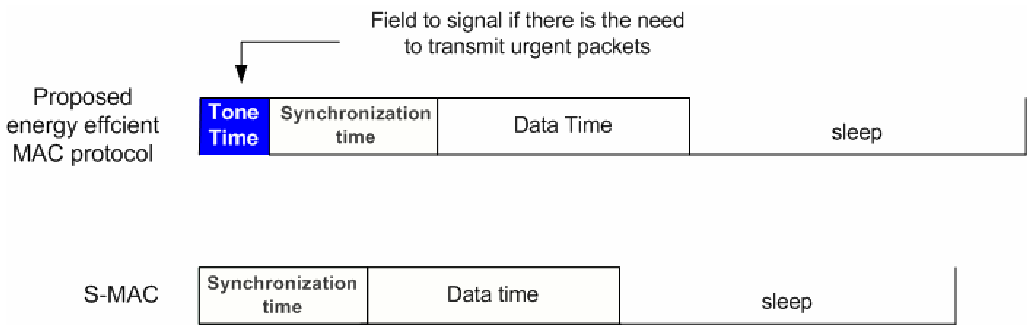
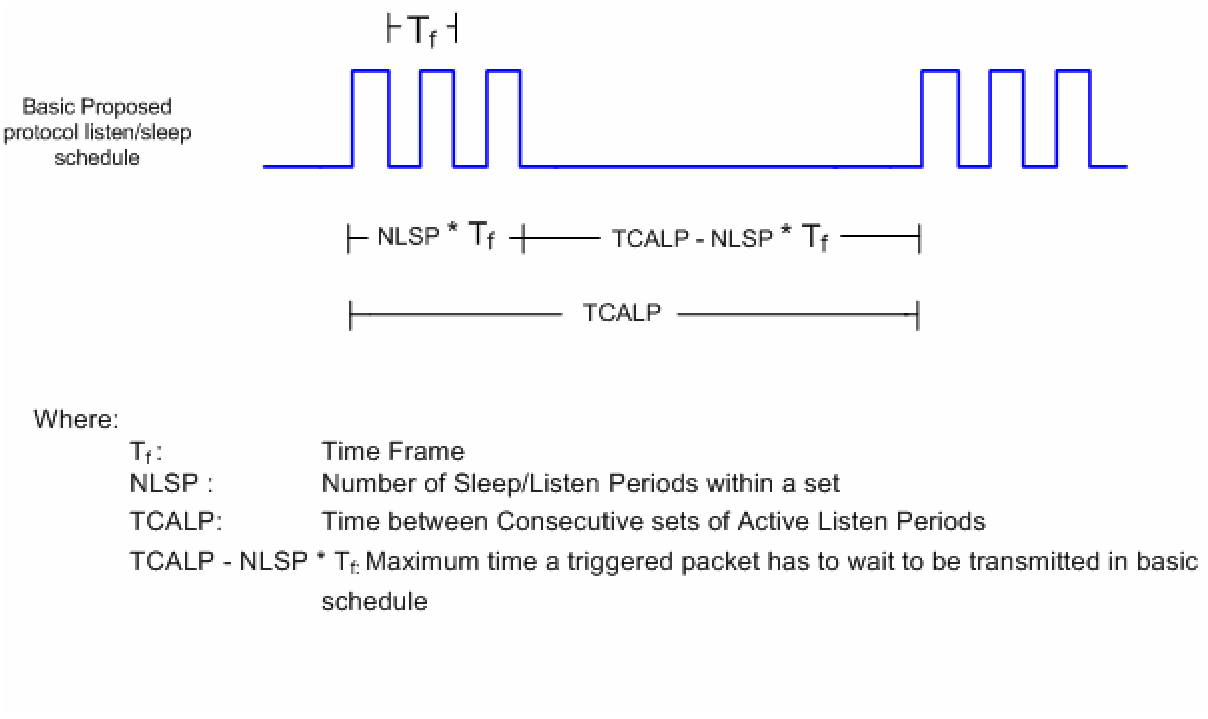

Field to signal if there is the need to transmit urgent packets
Proposed energy effcient MAC protocol
Tone Time
Synchronization time
Data Time
sleep
S-MAC
Synchronization time
Data time
sleep

Figure 11. Proposed energy-efficient protocol vs. SMAC listen- time structure

The proposed energy efficient MAC protocol novel listen-time structure will overcome initially proposed sleep/listen schedule drawbacks in the following ways:

- *Decreased latency*. Triggered packets will not wait a time between a Time Frame (T_f) and TCALP-NLSP*T_f (Figure 12).



T_f
Basic Proposed protocol listen/sleep schedule
NLSP * T_f
TCALP - NLSP * T_f
TCALP
Where:
T_f: Time Frame
NLSP : Number of Sleep/Listen Periods within a set
TCALP: Time between Consecutive sets of Active Listen Periods
TCALP - NLSP * T_f: Maximum time a triggered packet has to wait to be transmitted in basic schedule

Figure 12. Time intervals in basic proposed listen/sleep schedule

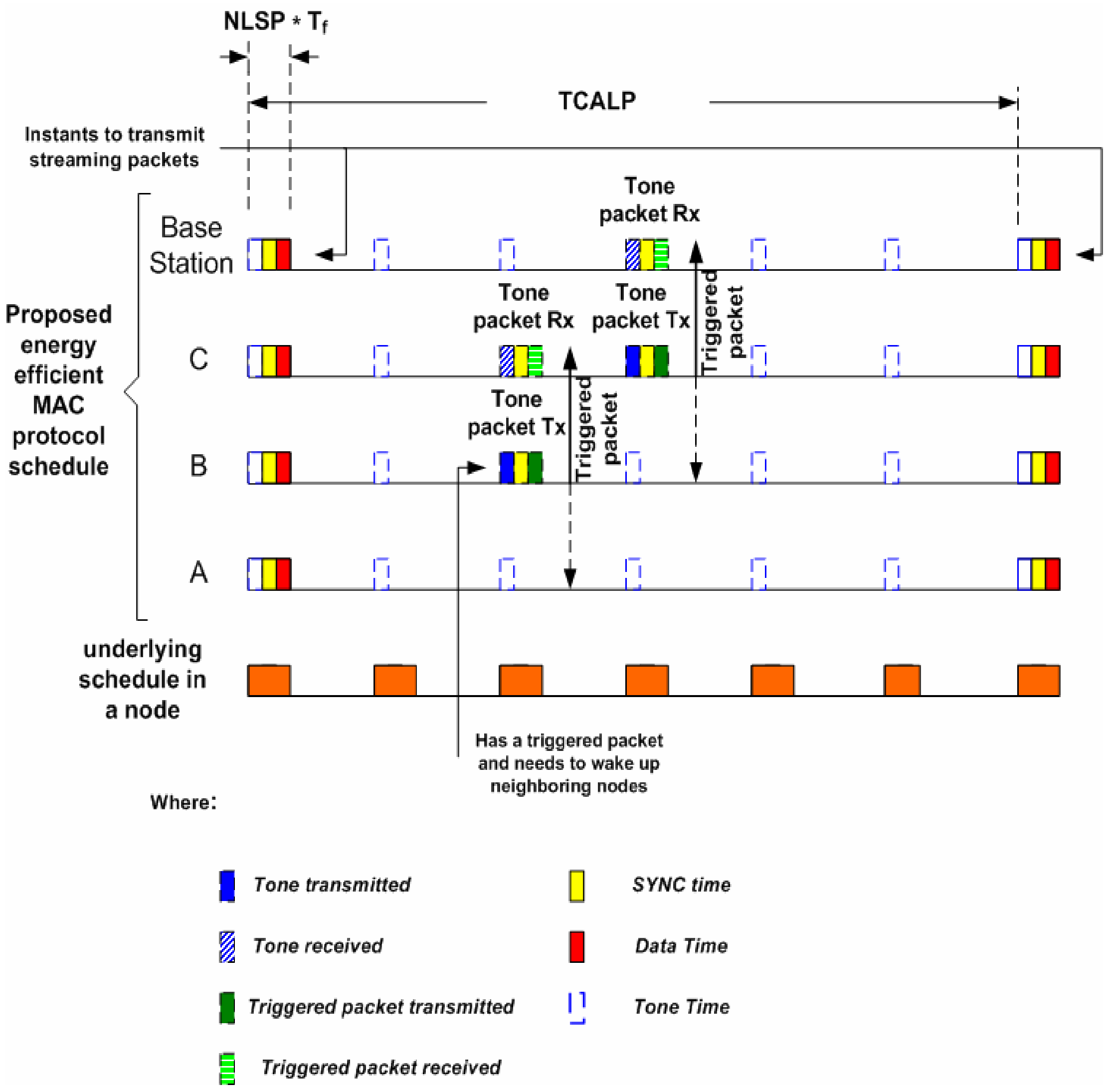
- *Decreased queue length*. Since triggered packets will be sent in the next active listen period of the underlying S-MAC synchronization, nodes will not queue up triggered packets for more than a T_f and therefore queue length is lowered.

- *Less number of collisions*. The number of packets to transmit are much less than if they were accumulated in the TCALP-NLSP*T_f time. Thus, the probability of collisions decreases dramatically.

3.1.3 Tone Time signaling mechanism

As stated in previous section, the Tone Time interval will be used by nodes which have sensed an interesting event and thus generated triggered packets that need to reach the *Task*

Manager for urgent analysis (see Figure 1). Figure 13 depicts how this approach works for the proposed energy efficient MAC protocol.



NLSP * T_f
TCALP
Instants to transmit streaming packets
Base Station
Tone packet Rx
Tone packet Rx
Tone packet Tx
Triggered packet
C
Tone packet Tx
Triggered packet
B
A
Proposed energy efficient MAC protocol schedule
underlying schedule in a node
Has a triggered packet and needs to wake up neighboring nodes
Where:
Tone transmitted
Tone received
Triggered packet transmitted
Triggered packet received
SYNC time
Data Time
Tone Time

Figure 13. Energy efficient MAC protocol listen/sleep schedule optimized to avoid idle listening and decrease latency, collisions and queue length for triggered packets.

The proposed sleep/listen mechanism depicted in the Figure above is explained in the following paragraphs:

- The proposed energy efficient protocol sleep/listen schedule is based on the underlying schedule that resulted from initial synchronization between nodes. This underlying synchronization is based on the most simple sleep/listen schedule of S-MAC. The important parameters here are *Duty Cycle*, *Sync Time duration* and *Data Time Duration* [5].

- We met the requirements of sampling packets to schedule behaves as explained in section 3.2. Two important parameters TCALP and NLSP are introduced.

- Within the TCALP-NLSP*T_f, radio is turned off in Data and SYNC times. However. All nodes turn on its radio at Tone Time to listen for possible requests from other nodes, to wake up and transmit triggered packets. Since all nodes have its radio turned on in Tone Time, some energy is consumed in idle listening. Therefore, it is crucial to design the Tone Time as small as possible compared to SYNC and DATA times; but large enough to allow a node to transmit/receive a very small Tone-packet.

- Whenever a interesting event is sensed, e.g. a sudden raise in temperature or another monitored phenomenon, the node generates a corresponding triggered packet that must reach the Task Manager with minimum latency.

- Once a node has generated a triggered packet it broadcasts a Tone-packet indicating that it needs the next-hop node to wake up so the packet can be disseminated though the network until it reaches the Base Station.

- Neighboring nodes receive Tone-packet and check its destination address. If the packet is intended for it, they do not go to sleep when the Tone Time is over.

- The node that originated the Tone packet sends the triggered packet. After sending the packet it goes to sleep following the original schedule to accommodate the sampled packets.

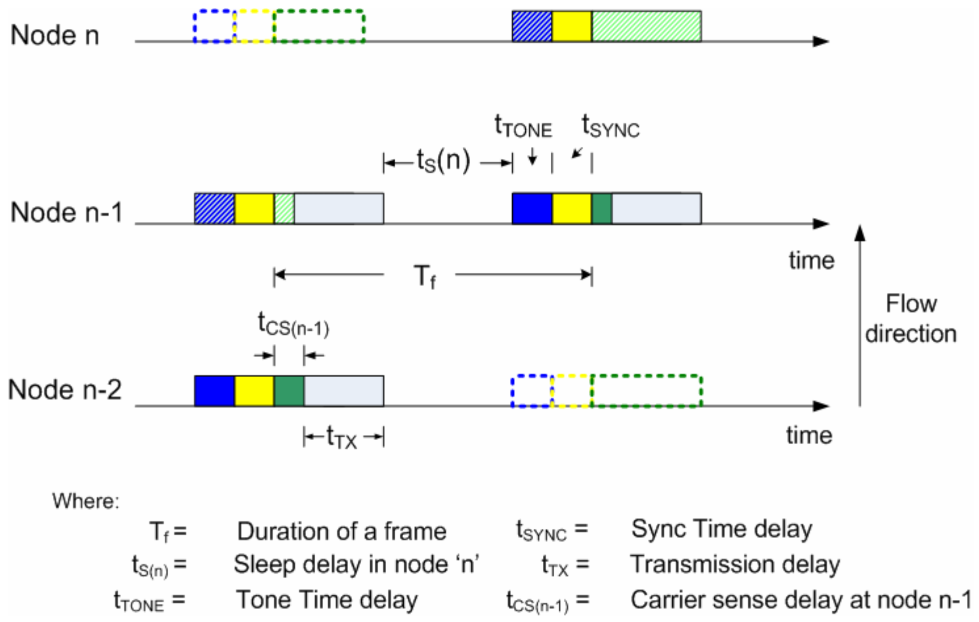
- The process is repeated as many times as number of nodes are in packet's route to the nearest Base Station.

3.4 Latency analysis

In this section, the latency that a triggered packet suffers for the proposed energy efficient MAC protocol is analyzed.

3.4.1 Triggered packets latency

We will use Figure 14 to derive the latency analysis for triggered packets.



Node n
t_{TONE}
t_{SYNC}
$t_S(n)$
Node n-1
time
T_f
Flow direction
$t_{CS(n-1)}$
Node n-2
t_{TX}
time
Where:
T_f = Duration of a frame
$t_{S(n)}$ = Sleep delay in node 'n'
t_{TONE} = Tone Time delay
t_{SYNC} = Sync Time delay
t_{TX} = Transmission delay
$t_{CS(n-1)}$ = Carrier sense delay at node n-1

Figure 14. Identification of existing delays to transmit triggered packets

The delay experienced in node *n* since the reception of a packet to the time it is forwarded is:

$$D(n) = t_s(n) + t_{TONE} + t_{CS}(n) + t_{TX} + t_{SYNC} \tag{1}$$

There is an exception for delay at hop number 1 because a packet can be generated at any time within a frame. Therefore, the sleep delay at node 1 is a random variable whose value lies in $(0,T_f)$ with mean $T_f/2$.

$$D(1) = t_s(1) + t_{TONE} + t_{CS}(1) + t_{TX} + t_{SYNC} \tag{2}$$

According to Figure 14, T_f can be expressed as:

$$T_f = t_s(n) + t_{SYNC} + t_{TONE} + t_{TX} + t_{CS}(n-1) \tag{3}$$

From (3), the sleep delay at hop n is:

$$t_s(n) = T_f - t_{SYNC} - t_{TONE} - t_{TX} - t_{CS}(n-1) \tag{4}$$

The total delay D_N:

$$D_N = \sum_{n=1}^{N} D(n) = D(1) + \sum_{n=2}^{N} D(n)$$

Where:

N : *number of hops from the node that originated the triggered packet to the nearest Base Station*

Substituting (1) and (2) into the last expression:

$$D_N = t_S(1) + t_{TONE} + t_{CS}(1) + t_{TX} + t_{SYNC} + \sum_{n=2}^{N} t_s(n) + t_{TONE} + t_{CS}(n) + t_{TX} + t_{SYNC}$$

Replacing (4) into the last expression:

$$D_N = t_S(1) + t_{TONE} + t_{CS}(1) + t_{TX} + t_{SYNC} + \sum_{n=2}^{N} \left(\left(T_f - t_{SYNC} - t_{TONE} - t_{TX} - t_{CS}(n-1) \right) + t_{TONE} + t_{CS}(n) + t_{TX} + t_{SYNC} \right)$$

$$D_N = t_S(1) + t_{TONE} + t_{CS}(1) + t_{TX} + t_{SYNC} + \sum_{n=2}^{N} \left(\left(T_f - t_{CS}(n-1) \right) + t_{CS}(n) \right)$$

$$D_N = t_S(1) + t_{TONE} + t_{CS}(1) + t_{TX} + t_{SYNC} + \sum_{n=2}^{N} Tf + \left(t_{CS}(2) + t_{CS}(3) + \ldots + t_{CS}(n) \right) - \left(t_{CS}(1) + t_{CS}(2) + \ldots + t_{CS}(N-1) \right)$$

Eliminating some terms:

$$D_N = t_S(1) + t_{TONE} + t_{CS}(1) + t_{TX} + t_{SYNC} + \sum_{n=2}^{N} \left(Tf \right) + t_{CS}(N) - t_{CS}(1)$$

Simplifying:

$$D_N = t_{TONE} + t_S(1) + t_{TX} + t_{SYNC} + \sum_{n=2}^{N} \left(Tf \right) + t_{CS}(N)$$

$$D_N = t_{TONE} + t_S(1) + t_{TX} + t_{SYNC} + (N-1) \times T_f + t_{CS}(N) \qquad (5)$$

Taking averages:

$$E[D_N] = E\left[t_{TONE} + t_S(1) + t_{TX} + t_{SYNC} + (N-1) \times T_f + t_{CS}(N)\right]$$

Let’s identify the distributions of each one:

- t_{TONE} is a constant for triggered packets because no matter what happens, each packet has to wait the entire t_{TONE} to be transmitted.
- $t_S(1)$ is a uniformly distributed variable between 0 and T_f because the packet in node 1 can be generated at any time
- t_{TX} as well as t_{SYNC} are constants
- T_f is constant as well as N
- $t_{CS}(N)$ is not constant and depends on parameters defined in the 802.11 protocol. However, its average is constant [11] and we called it t_{CS}.

Then:

$$E[D_N] = t_{TONE} + E[t_S(1)] + t_{TX} + t_{SYNC} + (N-1) \times T_f + E[t_{CS}(N)]$$

$$E[D_N] = t_{TONE} + \frac{T_f}{2} + t_{TX} + t_{SYNC} + (N-1) \times T_f + t_{CS}$$

$$E[D_N] = N \times T_f - \frac{T_f}{2} + t_{TX} + t_{SYNC} + t_{TONE} + t_{CS} \qquad (6)$$

Equations (5) and (6) predict the delay that a triggered packet will experience for the proposed energy efficient MAC protocol. It is import to notice that the delay and the mean delay are proportional to the number of hops.

3.5 Energy consumption analysis

3.5.1 S-MAC energy analysis

Following are the assumptions made for energy analysis:

- Sampled packets are small enough to be transmitted in a single listen interval.
- Only one node in the network generates sampled packets.
- Single route to Base Station.
- Each node has only two neighbors.
- There are no collisions.
- There are no retransmissions.

Figure 15 shows the configuration used for analysis (see section 4 for details):

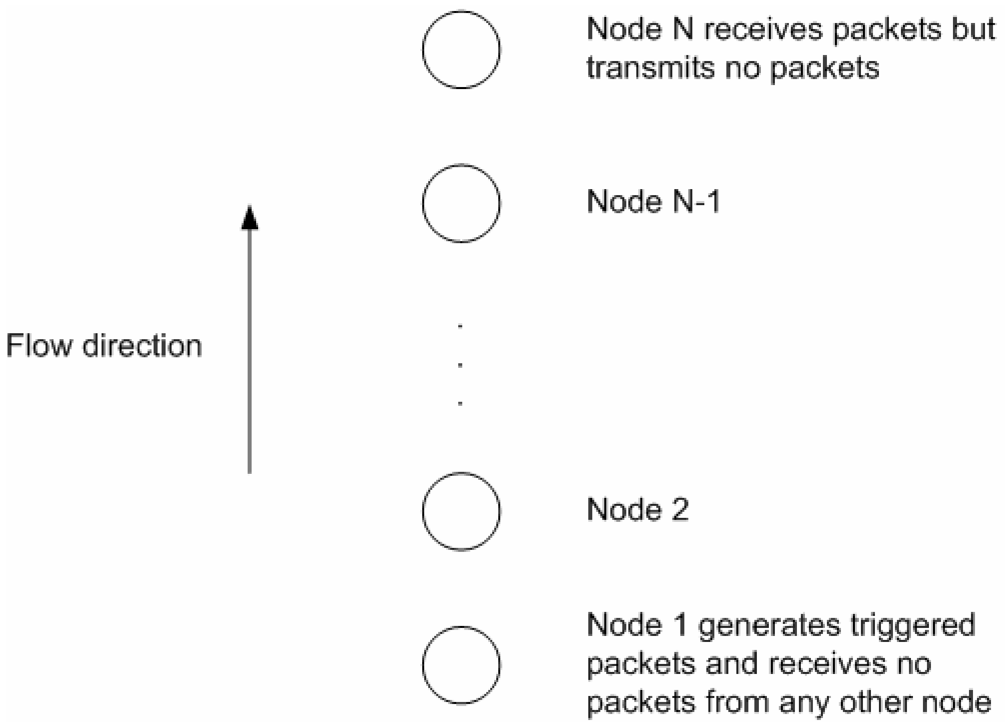

Node N receives packets but transmits no packets
Node N-1
Flow direction
Node 2
Node 1 generates triggered packets and receives no packets from any other node

Figure 15. Configuration used for analysis

The parameters used are the following:

- $\rho(n)[\frac{packets}{s}]: Rate \quad of \quad packets \quad sampled \quad by \quad second \quad in \quad node \quad n$
- $E_{IL}(n)[joules]: Energy \quad spent \quad in \quad idle \quad listening \quad in \quad a \quad T_f \quad in \quad node \quad n$
- $E_{TX}(n)[joules]: Energy \quad spent \quad in \quad transmitting \quad a \quad packet \quad in \quad node \quad n$
- $E_{RX}(n)[joules]: Energy \quad spent \quad in \quad receiving \quad a \quad packet \quad in \quad node \quad n$
- $E_{ON}[joules]: Energy \quad spent \quad to \quad turn \quad on \quad radio$
- $T_0(n): Period \quad of \quad observation$
- $E_S(n)[joules]: Energy \quad spent \quad on \quad sleep \quad time \quad within \quad a \quad T_f \quad in \quad node \quad n$
- $N: Number \quad of \quad nodes \quad in \quad the \quad network$
- *E(n):Energy consumed in node n in the period of observation T_0*

Now that the parameters and assumptions have been identified, we can calculate the energy consumed by S-MAC:

E(n) = Energy-transmit-packets + Energy-idle-listening + Energy-turn-on-radio + Energy-receive-packets + Energy-sleeping

More specifically:

$$E(n) = \rho(n)\times T_0 \times E_{TX} + E_{IL}(n)\left(\frac{T_0}{T_f} - 2\times\rho(n)\times T_0\right) + E_{ON}\times\frac{T_0}{T_f} + \rho(n)\times T_0\times E_{RX}(n) + \\ + \frac{T_0}{T_f}\times E_s(n) \qquad (7)$$

Where:

$\rho(n)\times T_0$ = *Number of packets received or transmitted by node n*

$\frac{T_0}{T_f}$ = *number of listen periods in T_0*

$\frac{T_0}{T_f} - 2\times\rho(n)\times T_0$ = *Number of listen periods in which the node is not in idle listening state*

The total energy consumed by the network is:

$$E_N = \sum_{n=1}^{N} E(n)$$

The energy expressions for initial node 1 and final node N are different, so:

$$E_N = E(1) + E(N) + \sum_{n=2}^{N-1} E(n) \tag{8}$$

Grouping common terms from (7):

$$E(n) = T_0 \rho(n)\left(E_{TX}(n) - 2E_{IL}(n) + E_{RX}(n)\right) + \frac{T_0}{T_f}\left(E_{IL}(n) + E_{ON} + E_S(n)\right) \tag{9}$$

Using (9) and taking into consideration the assumptions listed in the beginning of this section:

$$E(1) = T_0 \rho(1)\left(E_{TX}(1) - E_{IL}(1)\right) + \frac{T_0}{T_f}(E_{IL}(1) + E_{ON} + E_S(1)) \tag{10}$$

$$E(N) = T_0 \rho(N)\left(-E_{IL}(N) + E_{RX}(N)\right) + \frac{T_0}{T_f}\left(E_{IL}(N) + E_{ON} + E_S(N)\right) \tag{11}$$

Replacing (9), (10) and (11) into (8):

$$\begin{aligned} E_N = {} & T_0 \rho(1)\left(E_{TX}(1) - E_{IL}(1)\right) + \frac{T_0}{T_f}(E_{IL}(1) + E_{ON} + E_S(1)) + T_0 \rho(N)\left(-E_{IL}(N) + E_{RX}(N)\right) + \frac{T_0}{T_f}\left(E_{IL}(N) + E_{ON} + E_S(N)\right) + \\ & + \sum_{n=2}^{N=1}\left[T_0 \rho(n)\left(E_{TX}(n) - 2E_{IL}(n) + E_{RX}(n)\right) + \frac{T_0}{T_f}\left(E_{IL}(n) + E_{ON} + E_S(n)\right)\right] \end{aligned} \tag{12}$$

Equation (12) can be used to calculate the total energy consumed by the network. We can get the average energy consumed by taking averages on both sides of equation (12):

$$E[E_N] = E[T_0\rho(1)(E_{TX}(1) - E_{IL}(1)) + T_0\rho(N)(-E_{IL}(N) + E_{RX}(N))] + E\left[\frac{T_0}{T_f}(E_{IL}(1) + E_{ON} + E_S(1)) + \frac{T_0}{T_f}(E_{IL}(N) + E_{ON} + E_S(N))\right] +$$

$$+ \sum_{n=2}^{N=1}\left\{E[T_0\rho(n)(E_{TX}(n) - 2E_{IL}(n) + E_{RX}(n))] + E\left[\frac{T_0}{T_f}(E_{IL}(n) + E_{ON} + E_S(n))\right]\right\} \qquad (13)$$

We can identify in equation (13):

- *$E_{TX}(n)$ is constant since it is intended that all packets have the same length*
- *E_{ON} is a constant*
- *$E_{RX}(N)$ is a constant*
- *T_0 is a constant*
- *$\rho(n)$ is constant since it is assumed that there are no retransmissions or collisions, so the packets that a node generates are the same the other node receives*

- *$E_{IL}(n)$ is a random variable since the energy spent on idle listening is not constant because a node may overhear a transmission not intended for it so it may go to sleep early. In this way, $E_{IL}(n)$ can be expressed as:*

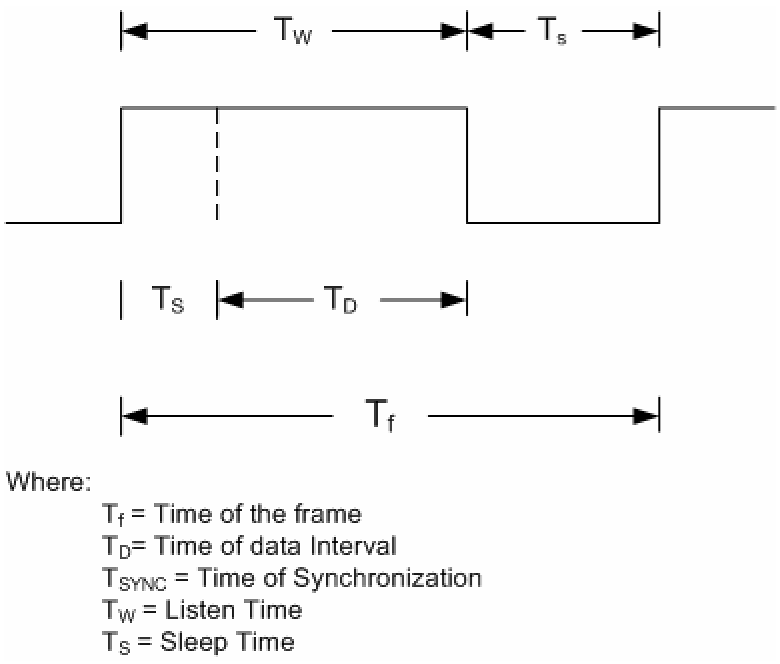
  $E_{IL}(n) = \rho_{IL} \times t_{IL}$

  Where:

  ρ_{IL} : Density of energy spent on idle listening $\left[\frac{Joules}{s}\right]$

  t_{IL}: time on idle listening on a Tf. t_{IL} is identically distributed in the range of $[0, T_D]$ (see Figure 16). Idle listening can last the complete T_D period in case the node does not overhear any transmission or it could be zero. Thus, its average is $\frac{T_D}{2}$

Thus, $E[E_{IL}(n)] = \rho_{IL}\left(\frac{T_D}{2}\right)$



T_W
T_s
T_S
T_D
T_f
Where:
T_f = Time of the frame
T_D= Time of data Interval
T_{SYNC} = Time of Synchronization
T_W = Listen Time
T_S = Sleep Time

Figure 16. S-MAC schedule parameters

- *$E_S(n)$ is a random variable which can be expressed as:*

$E_S(n) = \rho_S \times t_S(n)$

Where:

ρ_S *: Density of energy sent on sleeping* $\left[\frac{Joules}{s}\right]$

$t_s(n)$: Time that a node sleeps. According to equation (4),

$t_s(n) = T_f - t_{SYNC} - t_{TONE} - t_{TX} - t_{CS}(n-1)$, *but for S-MAC analysis we do not consider t_{TONE} and t_{SYNC}, so,* $t_s(n) = T_f - t_{TX} - t_{CS}(n-1)$

In this way, $E[t_s(n)] = E[T_f - t_{TX} - t_{CS}(n-1)] = T_f - t_{TX} - t_{CS}$

Finally, E[$E_S(n)$] = $\rho_S \times [T_f - t_{TX} - t_{CS}]$

However, because of practical considerations in simulations, ρ_S *is not considered because it is too small compared to idle listening so we will take into consideration the energy consumed when node is sleeping.*

Equation (13) is modified taking into account all considerations made above:

$$E[E_N] = T_0\rho E_{TX} - T_0\rho\rho_{IL}\frac{T_D}{2} - T_0\rho\rho_{IL}\frac{T_D}{2} + T_0\rho E_{RX} + 2\frac{T_0}{T_f}E_{ON} + 2\frac{T_0}{T_f}\rho_{IL}\frac{T_D}{2} +$$

$$+\sum_{n=2}^{N=1}\left\{T_0\rho E_{TX} + T_0\rho E_{RX} - 2T_0\rho\rho_{IL}\frac{T_D}{2} + \frac{T_0}{T_f}\rho_{IL}\frac{T_D}{2} + \frac{T_0}{T_f}E_{ON}\right\}$$

$$E[E_N] = T_0\rho(E_{TX} + E_{RX}) - T_0\rho\rho_{IL}T_D + 2\frac{T_0}{T_f}E_{ON} + \frac{T_0}{T_f}\rho_{IL}T_D +$$

$$+(N-2)\times\left\{T_0\rho E_{TX} + T_0\rho E_{RX} - T_0\rho\rho_{IL}T_D + \frac{T_0}{T_f}\rho_{IL}\frac{T_D}{2} + \frac{T_0}{T_f}E_{ON}\right\}$$

$$E[E_N] = T_0\rho(N-1)(E_{TX} + E_{RX}) - (N-1)T_0\rho\rho_{IL}T_D + N\frac{T_0}{T_f}E_{ON} + \frac{N}{2}\frac{T_0}{T_f}\rho_{IL}T_D \qquad (14)$$

3.5.2 Proposed protocol energy analysis

In this section the energy consumed by the proposed protocol is analyzed.

The energy consumed in each node is similar to that of S-MAC as well as the assumptions made in section 3.5.1. However a new term is added in idle listening that is the energy consumed in the tone-time. We have:

E(n) = Energy-transmit-packets + Energy-idle-listening + Energy-turn-on-radio + Energy-receive-packets + Energy-sleeping

Expressing each term and discarding the *Energy-sleeping* term because it is much smaller than other terms:

$$E(n) = \rho T_0 E_{TX} + E_{IL}(n)\left(\frac{T_0}{TCALP}NLSP - 2\rho T_0\right) + \frac{T_0}{T_f}E_{TONE} + \frac{T_0}{TCALP}E_{ON} + \rho T_0 E_{RX} \qquad (15)$$

Where:

TCALP : Time between Consecutive sets of Active Listen Periods (see Figure 12)

NLSP : Number of Listen Periods within a TCALP (see Figure 12)

E_{TONE} *: Energy consumed by a node in idle listening in a Tone Time (see Figure 13)*

$\frac{T_0}{TCALP}NLSP$ *: Number of active listen periods on* T_0

$\frac{T_0}{TCALP}NLSP - 2\rho T_0$ *: Number of active listen periods in which the node does not overhear a transmission*

It is important to observe that in equation (15) of the proposed protocol schedule, nodes are always listening for incoming transmissions in the Tone-Time, therefore the number of Tone-Time time slots is the same as the number of listen periods in S-MAC protocol within the period of observation T_0. Different is the number of activations a node suffers because a node sleeps more time than S-MAC, it changes from OFF status to ON less times.

As explained in section 3.5.1, nodes 1 and N have slightly different expressions for the energy consumed:

$$E(1) = T_0 \rho \left(E_{TX}(1) - E_{IL}(1) \right) + \frac{T_0}{TCALP} NLSP(E_{IL}(1) + E_{ON}) + \frac{T_0}{T_f} E_{TONE} \qquad (16)$$

$$E(N) = T_0 \rho \left(- E_{IL}(N) + E_{RX} \right) + \frac{T_0}{TCALP} NLSP(E_{IL}(N) + E_{ON}) + \frac{T_0}{T_f} E_{TONE} \qquad (17)$$

Replacing equations (15), (16) and (17) in (8) we get the total energy consumed:

$$E_N = T_0 \rho \left(E_{TX} - E_{IL}(1) - E_{IL}(N) + E_{RX} \right) + \frac{T_0}{TCALP} NLSP(E_{IL}(1) + E_{IL}(N) + 2E_{ON}) + 2\frac{T_0}{T_f} E_{TONE} +$$

$$+ \sum_{2}^{N-1} \left[\rho T_0 \left(E_{TX} + E_{RX} \right) + E_{IL}(n) \left(\frac{T_0}{TCALP} NLSP - 2\rho T_0 \right) + \frac{T_0}{T_f} E_{TONE} + \frac{T_0}{TCALP} NLSP \times E_{ON} \right] \quad (18)$$

Where:

$E_{IL}(n)$ is a random variable whose average is $\rho_{IL} \times \frac{T_D}{2}$ *(see section 3.5.1)*

Equation (18) is the general expression to calculate the energy spent in a sensor network that runs the proposed MAC protocol. In order to compare to SMAC energy consumption, it is necessary to derive the expected values of (18):

$$E\left[E_N \right] = T_0 \rho \left(E_{TX} - \rho_{IL} T_D + E_{RX} \right) + \frac{T_0}{TCALP} NLSP(\rho_{IL} T_D + 2E_{ON}) + 2\frac{T_0}{T_f} E_{TONE} +$$

$$+(N-2)\left(\rho T_0\left(E_{TX}+E_{RX}\right)+\rho_{IL}\times\frac{T_D}{2}\left(\frac{T_0}{TCALP}NLSP-2\rho T_0\right)\;+\frac{T_0}{T_f}E_{TONE}+\frac{T_0}{TCALP}NLSP\times E_{ON}\right)$$

Finally:

$$E[E_N]=(N-1)T_0\rho(E_{TX}+E_{RX})-(N-1)T_0\rho\rho_{IL}T_D+\frac{N\times T_0}{TCALP}NLSP\times E_{ON}$$
$$+\frac{N\times T_0}{TCALP}NLSP\frac{\rho_{IL}T_D}{2}+N\frac{T_0}{T_f}E_{TONE} \tag{18}$$

If we compare equations (18) and (14) we can get the following conclusions:

- ✓ The energy consumed in transmitting and receiving the packets is the same: $(N-1)T_0\rho(E_{TX}+E_{RX})$

- ✓ The energy saved by overhearing a packet and sleeping early is the same: $-(N-1)T_0\rho\rho_{IL}T_D$

- ✓ The proposed protocol saves energy by making less transitions than S-MAC: $\frac{N\times T_0}{TCALP}NLSP\times E_{ON}$

- ✓ The energy consumed in idle listening is considerably less than that of S-MAC: $\frac{N\times T_0}{TCALP}NLSP\frac{\rho_{IL}T_D}{2}$

- ✓ A small amount of energy is added to proposed protocol because of the idle listening in the Tone Time: $N\frac{T_0}{T_f}E_{TONE}$

It can be seen that for the proposed protocol to consume less energy than that of S-MAC the following inequality must be complied:

$$\frac{NLSP}{TCALP} < \frac{1}{T_f} \qquad (19)$$

4. SIMULATIONS AND PERFORMANCE ANALYSIS

First, this section describes the algorithms and the implementations of the protocol. Then, simulation scenario used in the network simulator ns-2 to test the proposed protocol features is described. Finally, the results obtained from simulations are analyzed.

4.1 Algorithms

In this section, the algorithms and the implementations of the proposed protocol are explained here. The corresponding code for these algorithms is described in Appendix B.

4.1.1 Algorithm for proposed sleep/listen schedule

Figures 17 and 18 depict the algorithms that are used for the implementation of proposed protocol schedule. The algorithm to implement the listen/sleep schedule works as described below:

- ➢ When a node is created it sets its parameters to:
    - *Status = 1*. This means that the node will follow at the beginning the original S-MAC schedule.
    - *NLSP = 0*. Since the node is following the S-MAC schedule at the beginning, it does not need to set NLSP to any value because the node will wake up periodically anyway (see Figure 12).

- ➢ *Status* is a variable that will control the number of periods a node stays slept once the node follows the proposed protocol schedule. Thus, every time a new cycle (Tone->Sync->Data.>Sleep) begins with a Tone Time, this variable's value must be decreased by one.

- A node stops to follow the S-MAC protocol schedule and starts to follow the proposed protocol one when it receives the first ACK packet from its neighbor indicating that it successfully received the data packet (see Appendix B for ACK synchronization). After receiving the first ACK packet the node sets its parameters to:
    - *Status = X*. Status is set to a specific value so the node will not turn its radio on in the next period because the sampling packet has already been sent (due to ACK packet received). Status variable is decreased on each iteration.
    - *NLSP = Y*. Independently of the value of Status variable, NLSP will make the node turn its radio on when different than zero.

  Every time an ACK packet is received, the node sets these values again. Status and NLSP are the variables that determine the configuration of the new schedule (Figure 12).

- If *Status* = 0 then that node is either following the S-MAC schedule or ending its configured number of periods to sleep, so it must turn on its radio in the Tone Time. Otherwise, the node must sleep to satisfy the number of periods to sleep set on the *Status* variable (Figure 18).

- If NSLP is different than zero then that node has already synchronized with an ACK packet and must turn its radio on for NLSP number of periods before it goes to sleep (see subsection B.4 for ACK synchronization). In each iteration NLSP is decreased (Figure 18).

- After a node 'decides' to turn on its radio on or not, it must follow the natural cycle: Sync->Data.>Sleep->Tone.

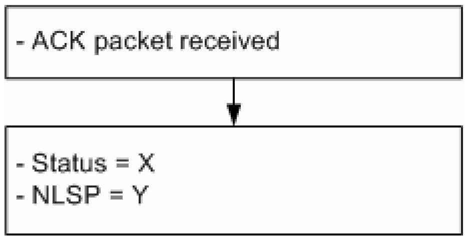


Figure 17. When An ACK packet is received the node sets new values

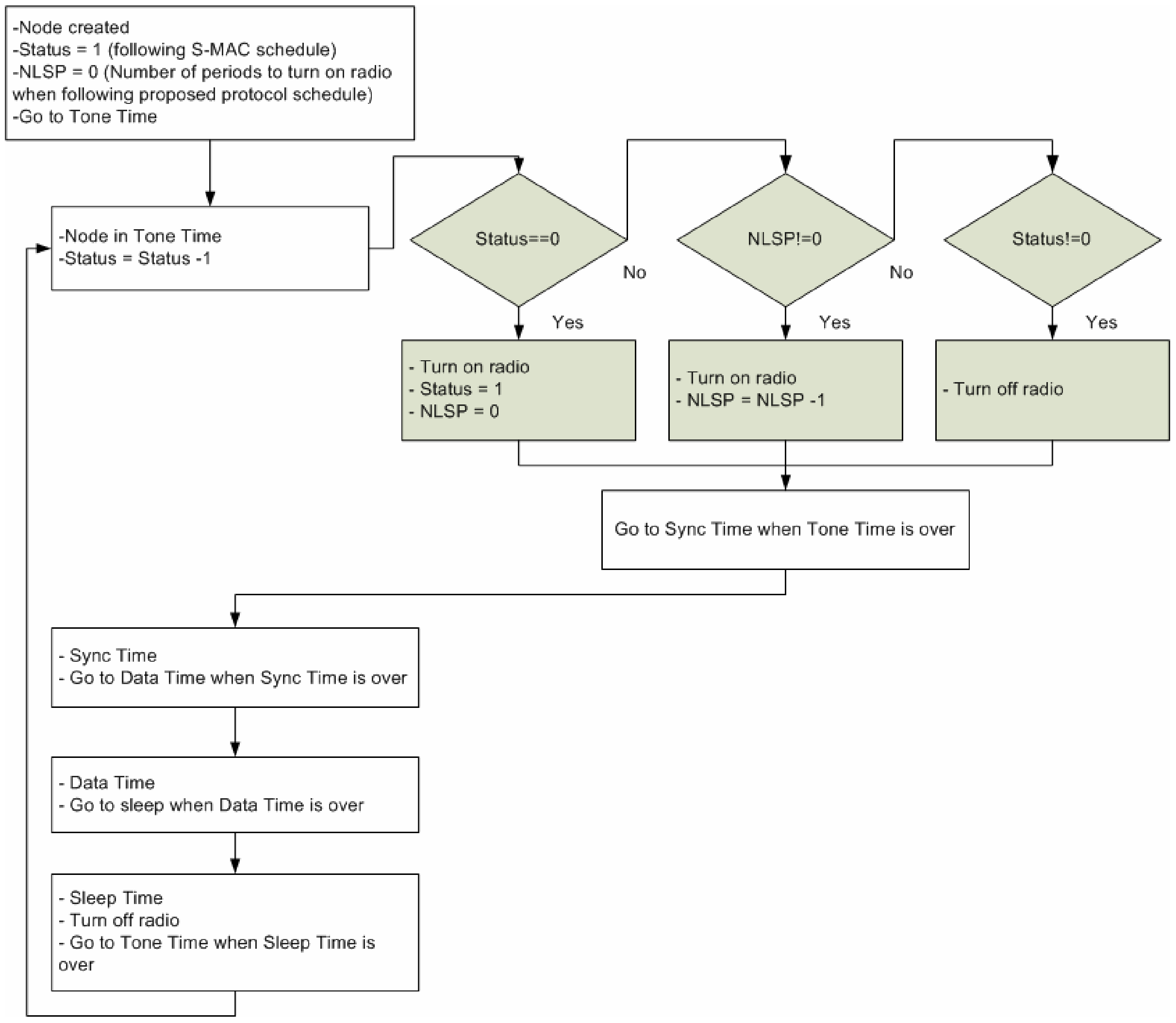

-Node created
-Status = 1 (following S-MAC schedule)
-NLSP = 0 (Number of periods to turn on radio when following proposed protocol schedule)
-Go to Tone Time
-Node in Tone Time
-Status = Status -1
Status==0
NLSP!=0
Status!=0
No
No
Yes
Yes
Yes
- Turn on radio
- Status = 1
- NLSP = 0
- Turn on radio
- NLSP = NLSP -1
- Turn off radio
Go to Sync Time when Tone Time is over
- Sync Time
- Go to Data Time when Sync Time is over
- Data Time
- Go to sleep when Data Time is over
- Sleep Time
- Turn off radio
- Go to Tone Time when Sleep Time is over

Figure 18. Proposed protocol scheme to create and use a second schedule in top of existing one

4.1.2 Algorithms for Tone packets mechanism

Figures 19 and 20 depict the algorithms that are used for the implementation of proposed protocol Tone packets scheme. The algorithm to implement the sleep/listen schedule works as described below:

- When a node is created there is no need to send Tone packets since nothing has been sensed yet.
- A node follows its normal cycle (Tone->Sync->Data->Sleep) when there is no need to transmit Tone packets.
- If an interesting event is sensed at any time, a corresponding triggered packet is generated and then, the node knows it must send Tone packets.
- If there is a need to transmit a Tone packet, the node turns on its radio no matter the values of variables *Status* or *NLSP* described in previous subsection. It contents for the medium and after it has gained the medium, it broadcasts the packet.
- After broadcasting Tone packets the node supposes the other nodes received it and than is ready to send its triggered packets.
- Triggered packets are sent in Data time.

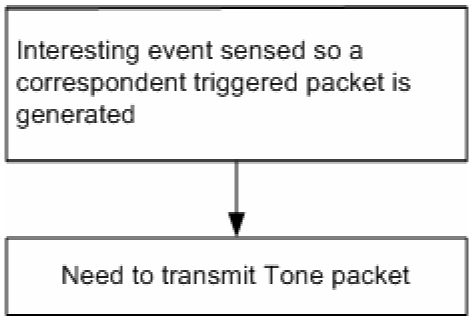

Interesting event sensed so a correspondent triggered packet is generated
Need to transmit Tone packet

Figure 19. A need to transmit a Tone packet is generated when an interesting event is sensed.

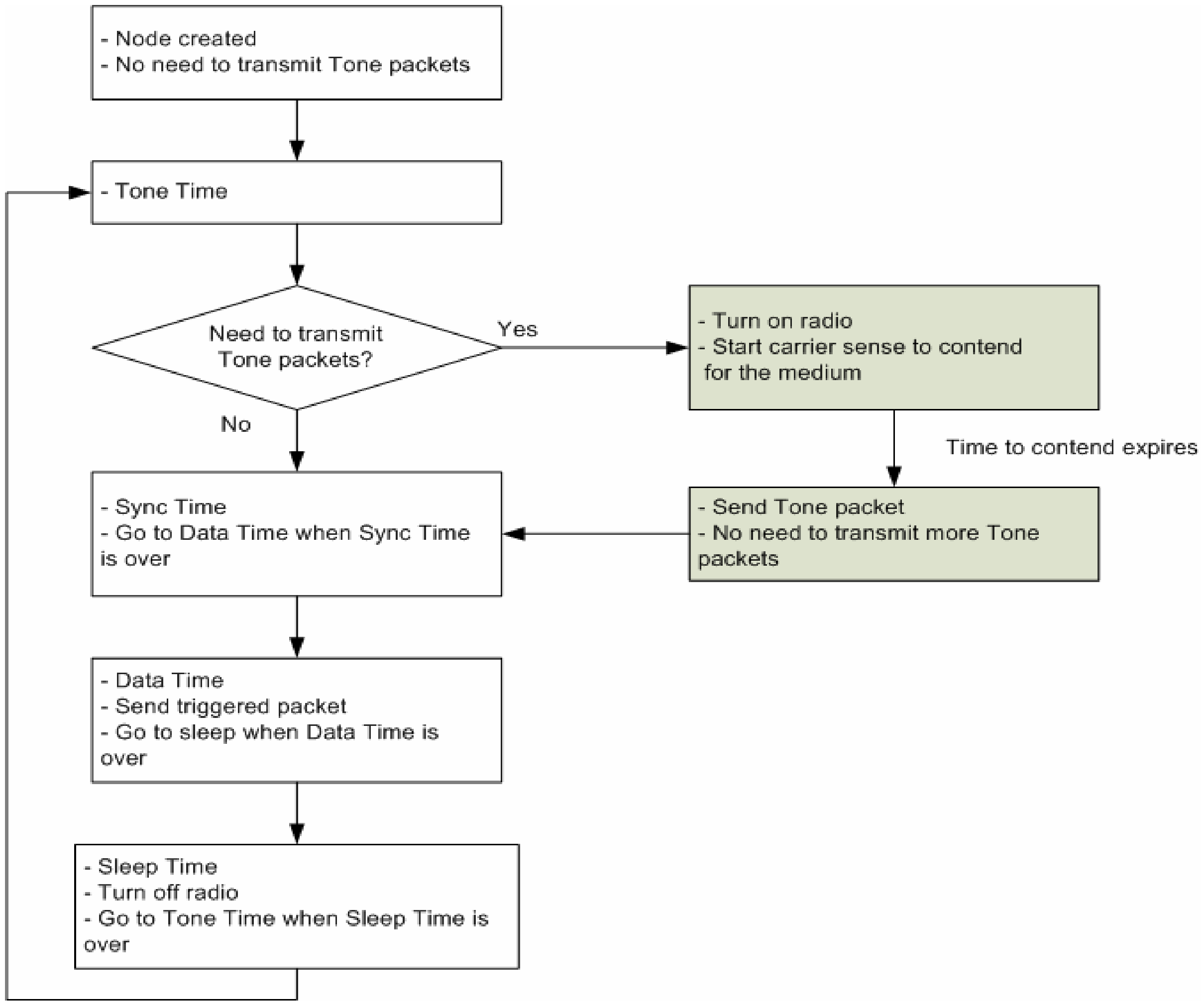

- Node created
- No need to transmit Tone packets
- Tone Time
Need to transmit Tone packets?
Yes
No
- Turn on radio
- Start carrier sense to contend for the medium
Time to contend expires
- Send Tone packet
- No need to transmit more Tone packets
- Sync Time
- Go to Data Time when Sync Time is over
- Data Time
- Send triggered packet
- Go to sleep when Data Time is over
- Sleep Time
- Turn off radio
- Go to Tone Time when Sleep Time is over

Figure 20. Mechanism to send a triggered packet by sending Tone packets.

4.2 Simulation scenario

The simulation was designed to test both the new proposed schedule and the mechanism of sending tone packets to disseminate triggered packets when generated in a time between two consecutive samples.

Network configuration for both tests is shown in Figure 21.

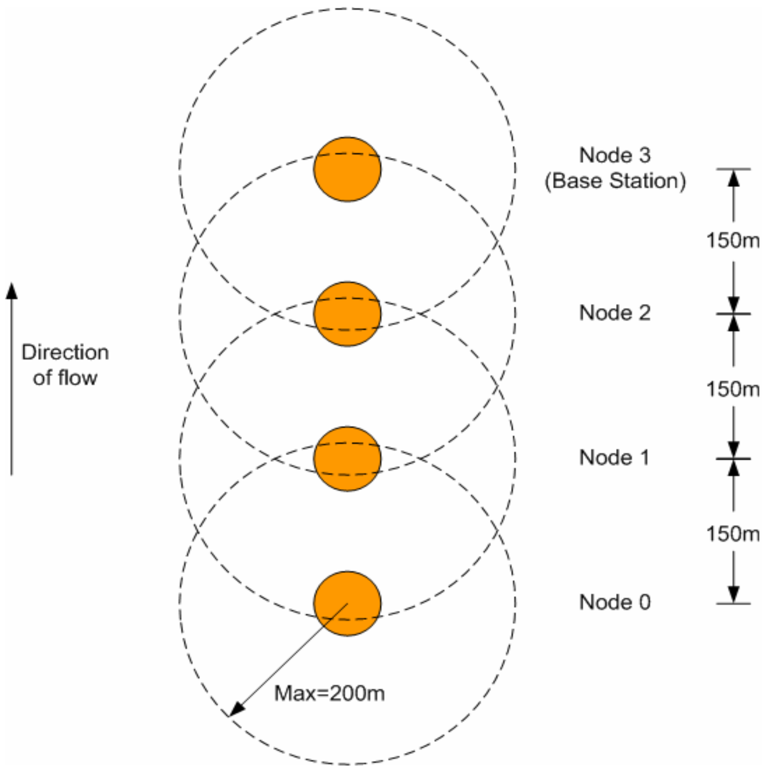


Figure 21. Network configuration of simulation

Network configuration shown in Figure 21 has the following characteristics:

- The four nodes (0,1,2,3) are on a straight line with 150m in apart.

- Node 0 can reach only node 1, 1 can reach 0 and 2, 2 can reach 1 and 3 and 3 only 2.

- The objective of each node is to transmit its data packets to node 3 (Base Station). The synchronization and control information is also exchanged between neighbors.

Parameters used in the simulation are the following:

Simulation time (**T_0**):	Time to send 100 packets[s], max 100000[s]
Duty cycle (**DC**):	variable
idlePower(**E_{IL}**):	1.0
rxPower(**E_{RX}**):	1.0
txPower(**E_{TX}**):	1.0
sleepPower(**E_S**):	0.001
transitionPower(**E_{ON}**):	0.2
transitionTime:	0.005
Routing Protocol:	AODV
Queue type:	Queue/DropTail/PriQueue
ns-2 energy model:	EnergyModel
Propagation model:	Propagation/TwoRayGround
Initial energy of each node:	100,000 [mJ]

Simulation in ns-2 is described in the following paragraphs:

- Nodes delay 50 seconds approximately in synchronizing themselves. Due to the number of nodes used in the simulation, all nodes must have the same listen/sleep schedule, forming a single virtual cluster.

- At 100 second, node 0 starts sending 10-byte data packets with a Poisson traffic generator at a mean *sending rate*. This rate of transmission can also be expressed as the mean time between consecutive packets that we call *message inter-arrival time*. If the radio of node 0 is *on* when the packet is generated the node supposes all nodes are awake as well so transmits the packet immediately. Otherwise, node 0 uses the tone packet technique described in Chapter 3 to signal other nodes to wake up.

- Nodes internally follow the proposed second schedule of the protocol with the following parameters:

    - NLSP =5
    - StatusModifiedProtocolTimer = 20
    - syncTime = difs + slotTime * SYNC_CW + durSyncPkt + guardTime;
    - toneTime = 0.6*syncTime;

- All logs are saved in the trace file. The total energy consumed is computed adding the energy consumed by all four nodes.

- Energy consumed includes all types of energy consumption described in chapter 3.

Each simulation is run at constant DC. For each given constant rate the duty cycle is changed from 10% to 50%. At the end of each simulation, the remaining energy in each node is saved for further computations. Then, to compute the total energy consumed in each simulation, the remaining energy is subtracted from initial energy configured to get the energy consumed in each node. Then, the energy consumed in each node is added to get total energy consumed in the network.

4.3 Results and analysis

The results obtained are organized into Tables 1 to 4. In such tables, for each message inter-arrival time (I) five simulations results are listed. The remaining energy in [mJ] in each node for S-MAC and for the proposed protocol is depicted in Tables 1 and 2 respectively. Tables 3 and 4 depict the energy consumed in each node for S-MAC and for the proposed protocol respectively. To compute the energy consumed in each simulation, the energy consumed by each node is added.

Table 1. Remaining energy in each node at the end of simulation after 100 packets have been disseminated through the network using the SMAC protocol

Remaining energy in each node with S-MAC protocol					
Message inter-arrival time node 0 = I =100	**DC=10%**	**DC=20%**	**DC=30%**	**DC=40%**	**DC=50%**
Node 0	98339	97407	96533	95530	94563
Node 1	98185	97352	96544	95571	94619
Node 2	98169	97357	96541	95554	94613
Node 3	98348	97399	96531	95522	94556
I = 200					
Node 0	97093	95143	93214	91316	89239
Node 1	96899	95091	93232	91379	89345

Node 2	96898	95087	93231	91363	89338
Node 3	97101	95140	93211	91302	89233
I = 300					
Node 0	95748	92851	90244	87241	83372
Node 1	95537	92792	90270	87337	83530
Node 2	95550	92788	90277	87322	83524
Node 3	95782	92855	90250	87230	83367
I = 400					
Node 0	94476	90297	86692	82328	78587
Node 1	94217	90225	86733	82457	78788
Node 2	94220	90225	86726	82442	78780
Node 3	94478	90294	86681	82317	78580
I = 500					
Node 0	93160	87966	83265	77871	72483
Node 1	92285	87879	83314	78027	72738
Node 2	92881	87880	83305	78016	72734
Node 3	93170	87961	83251	77861	72476

Table 2. Remaining energy in each node at the end of simulation after 100 packets have been disseminated through the network using the Proposed Protocol

Remaining energy in each node with Proposed Protocol					
Message inter-arrival time node 0 = I =100	**DC=10%**	**DC=20%**	**DC=30%**	**DC=40%**	**DC=50%**
Node 0	98854	98120	97247	96\276	95660
Node 1	98605	98056	97217	96276	95683
Node 2	98630	98044	97203	96261	95652
Node 3	98855	98094	97211	96225	95606
I = 200					
Node 0	97798	96022	94693	93063	91586
Node 1	97560	95957	94676	93069	91643
Node 2	97534	95856	94625	93060	91605
Node 3	97832	96026	94662	92995	91531

I = 300					
Node 0	97022	94556	91973	89623	87287
Node 1	96786	94472	91911	89642	87369
Node 2	96842	94497	91901	89614	87328
Node 3	96994	94571	91943	89579	87228
I = 400					
Node 0	95846	93148	89406	86543	82734
Node 1	95696	93030	89441	86623	82829
Node 2	95622	92937	89424	86599	82805
Node 3	96010	93050	89409	86531	82684
I = 500					
Node 0	95260	91271	87302	82334	78481
Node 1	95071	91128	87306	82424	78605
Node 2	95052	91163	87228	82406	78593
Node 3	95282	91252	87219	82321	78437

Table 3. Energy consumed in each node at the end of simulation after 100 packets have been disseminated through the network using the SMAC protocol

Energy consumed in each node with S-MAC protocol					
Message inter-arrival time node 0 = I =100	**DC=10%**	**DC=20%**	**DC=30%**	**DC=40%**	**DC=50%**
Node 0	1661	2593	3467	4470	5437
Node 1	1815	2648	3456	4429	5381
Node 2	1831	2643	3459	4446	5387
Node 3	1652	2601	3469	4478	5444
TOTAL	**6959**	**10485**	**13851**	**17823**	**21649**
I = 200					
Node 0	2907	4857	6786	8684	10761
Node 1	3101	4909	6768	8621	10655
Node 2	3102	4913	6769	8637	10662
Node 3	2899	4860	6789	8698	10767

TOTAL	**12009**	**19539**	**27112**	**34640**	**42845**
I = 300					
Node 0	4252	7149	9756	12759	16628
Node 1	4463	7208	9730	12663	16470
Node 2	4450	7212	9723	12678	16476
Node 3	4218	7145	9750	12770	16633
TOTAL	**17383**	**28714**	**38959**	**50870**	**66207**
I = 400					
Node 0	5524	9703	13308	17672	21413
Node 1	5783	9775	13267	17543	21212
Node 2	5780	9775	13274	17558	21220
Node 3	5522	9706	13319	17683	21420
TOTAL	**22609**	**38959**	**53168**	**70456**	**85265**
I = 500					
Node 0	6840	12034	16735	22129	27517
Node 1	7715	12121	16686	21973	27262
Node 2	7119	12120	16695	21984	27266
Node 3	6830	12039	16749	22139	27524
TOTAL	**28504**	**48314**	**66865**	**88225**	**109569**

Table 4. Energy consumed in each node at the end of simulation after 100 packets have been disseminated through the network using the Proposed Protocol

Energy consumed in each node with Proposed Protocol					
Message inter-arrival time node 0 = I =100	**DC=10%**	**DC=20%**	**DC=30%**	**DC=40%**	**DC=50%**
Node 0	1146	1880	2753	3724	4340
Node 1	1395	1944	2783	3724	4317
Node 2	1370	1956	2797	3739	4348
Node 3	1145	1906	2789	3775	4394
TOTAL	**5056**	**7686**	**11122**	**14962**	**17399**
I = 200					
Node 0	2202	3978	5307	6937	8414
Node 1	2440	4043	5324	6931	8357
Node 2	2466	4144	5375	6940	8395
Node 3	2168	3974	5338	7005	8469
TOTAL	**9276**	**16139**	**21344**	**27813**	**33635**

I = 300					
Node 0	2978	5444	8027	10377	12713
Node 1	3214	5528	8089	10358	12631
Node 2	3158	5503	8099	10386	12672
Node 3	3006	5429	8057	10421	12772
TOTAL	**12356**	**21904**	**32272**	**41542**	**50788**
I = 400					
Node 0	4154	6852	10594	13457	17266
Node 1	4304	6970	10559	13377	17171
Node 2	4378	7063	10576	13401	17195
Node 3	3990	6950	10591	13469	17316
TOTAL	**16826**	**27835**	**42320**	**53704**	**68948**
I = 500					
Node 0	4740	8729	12698	17666	21519
Node 1	4929	8872	12694	17576	21395
Node 2	4948	8837	12772	17594	21407
Node 3	4718	8748	12781	17679	21563
TOTAL	**19335**	**35186**	**50945**	**70515**	**85884**

The plots shown in Figures 22-26 correspond to the data in Tables 3 and 4.

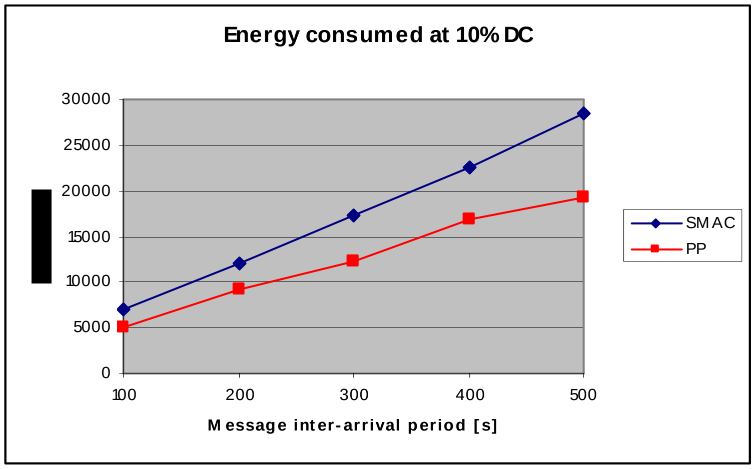


Figure 22. Energy consumed at a DC of 10%

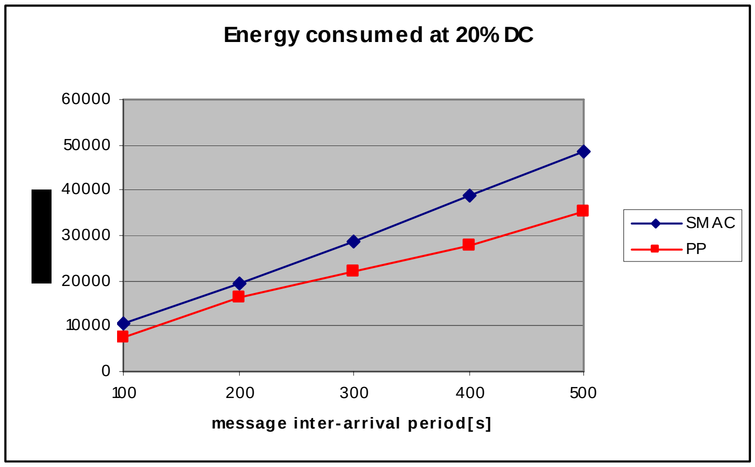


Figure 23. Energy consumed at a DC of 20%

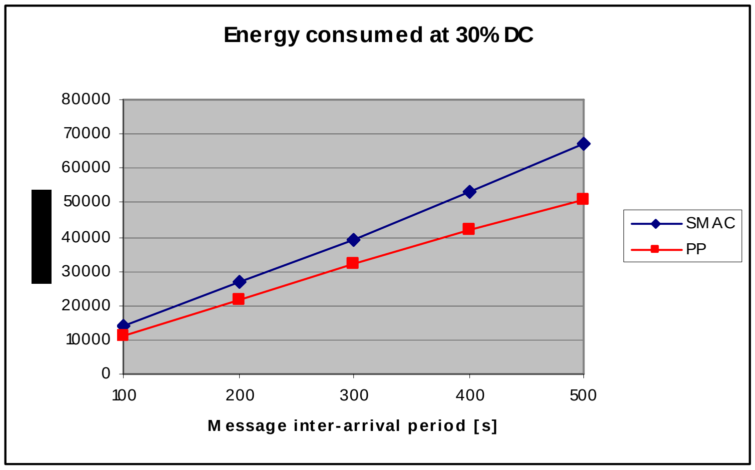


Figure 24. Energy consumed at a DC of 30%

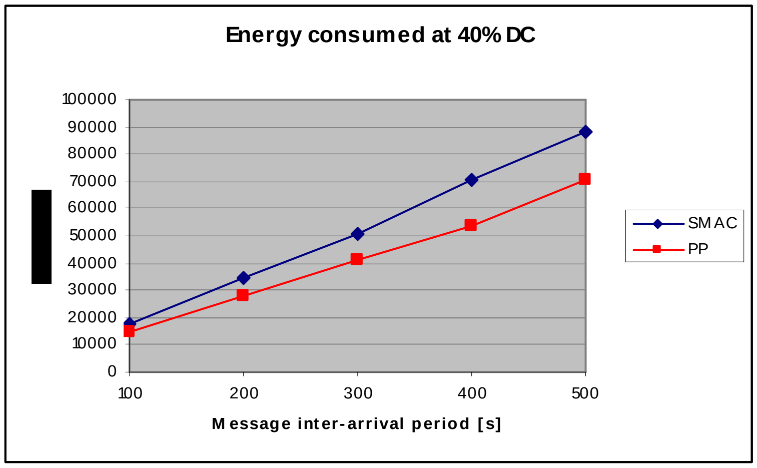


Figure 25. Energy consumed at a DC of 40%

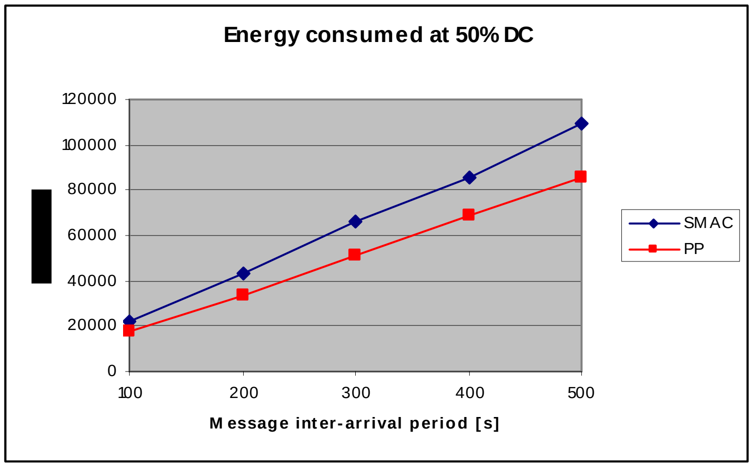


Figure 26. Energy consumed at a DC of 50%

The *message inter-arrival period* can help us calculate the *mean sending rate* at which node 0 sends its 10-byte packets:

$$Rate\left[\frac{bits}{s}\right] = \frac{1}{\text{message inter-arrival period}}\left[\frac{packet}{s}\right] \times \frac{80\ bits}{1\ packet} =$$

$$\frac{80}{message\ inter-arrival\ \ period} \times \left[\frac{bits}{s}\right]$$

Figure 27 is obtained by converting each *message inter-arrival time* to *mean sending rate* in Tables 3 a 4 and plotting the data. In this, PP stands for Proposed Protocol.

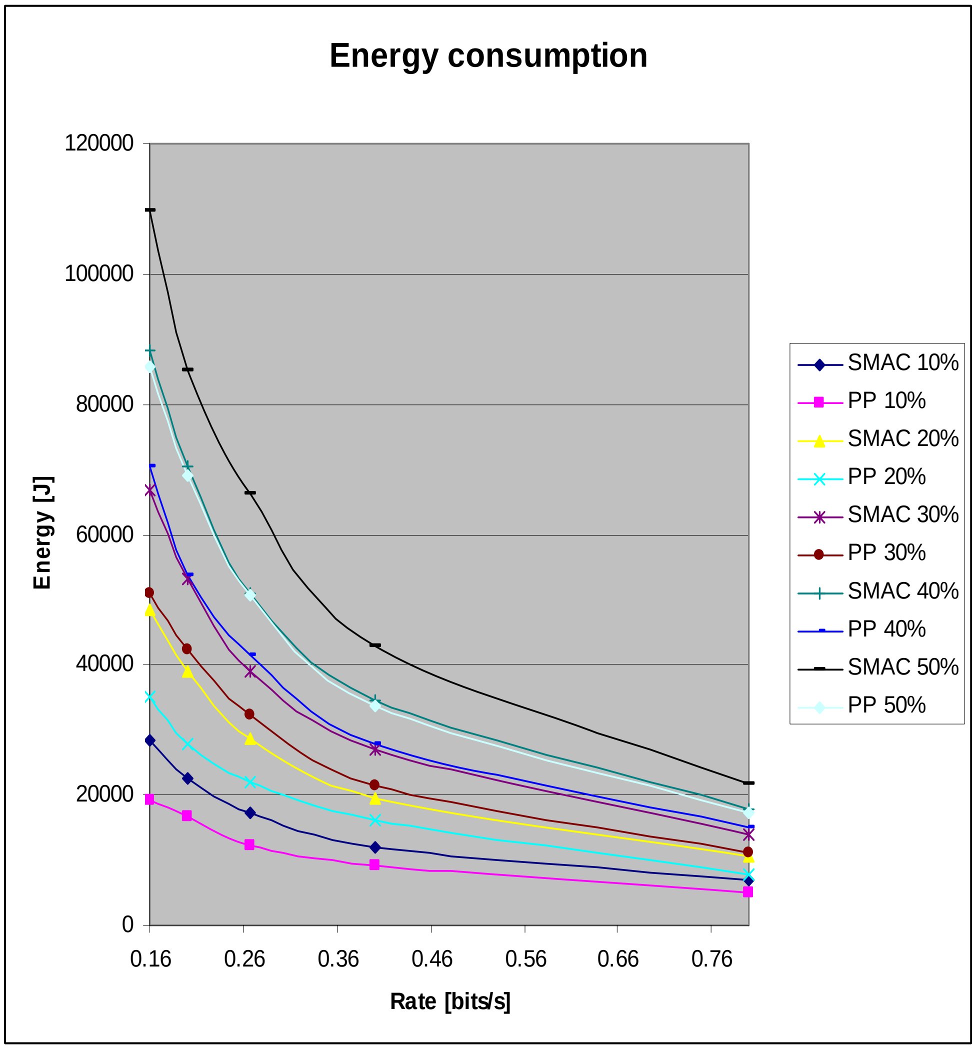


Figure 27. Energy consumption vs. rate of transmission

As it is shown in Figures 22-26 , proposed protocol outperforms SMAC at every duty cycle at every message inter-arrival rate delivering the same throughput.

The following obervations can be made from the figures and results shown above:

- ✓ In Figure 27 it is observed that the higher the rate of transmission of packets (sampling packets) the less the consumption of energy. This may be a contradiction because the higher the rate the more energy consumed in transmitting and receiving packets. Equation (18) in subsection 3.5.2 is helpful to analyze this result. According to this equation, if rate of transmission increases (ρ), the first term of the equation $(N-1)T_0\rho(E_{TX}+E_{RX})$ increases. However, the protocol has an anti-overhearing mechanism in which a node goes to sleep if listens a transmission not intended for it and also a node that transmit a packet simply goes to sleep before the end of the cycle so the higher ρ the less energy is consumed in idle listening, this fact is expressed in the second term of the equation: $-(N-1)T_0\rho\rho_{IL}T_D$.

  Thus, the reason why the higher the mean long term rate of transmission the less energy is consumed is because the term $\rho_{IL}T_D$ is more dominant than $(E_{TX}+E_{RX})$. Remember that the majority of energy is consumed in idle listening and not in transmission or reception of packets.

  Conversely, if the rate increases dramatically, there will appear some other phenomena in the network such as retransmissions due to collisions which have not been included in the analysis. In this case the energy consumption may increase instead of reduce.

- ✓ At a fixed Duty Cycle (DC), the higher the message inter-arrival period the greater the difference between Proposed Protocol and SMAC (the better the proposed protocol is). Figures 22 to 26 show clearly this fact.

In Figure 12 it is observed that, the higher the message-inter-arrival period, the higher the Time between Consecutive sets of Active Listen Periods (TCALP), and remembering that Tf is constant and so is NLSP, the “better“ the inequality (19) satisfies. Inequality (19) is the most basic requisite for energy savings. In fact, the more the difference between the two sides of the inequality the more energy savings will result in the wireless sensor network.

5. CONCLUSIONS AND FUTURE WORK

We proposed an energy aware MAC protocol for wireless sensor networks that gathers data for wide-area large scale environmental monitoring. The scheme saves energy by organizing the networks usage changing the running synchronization. Specifically, the proposed protocol uses:

- A sleep/listen schedule running in top of a previously negotiated one. In this schedule, nodes wake up only when a sample is to be taken. This schedule is intended to save more energy by avoiding idle listening.

- A mechanism to wake-up nodes when a node has the urgency to transmit a triggered packet. We called this mechanism Tone-Time; which met the requirements of triggered packets of low latency.

According to simulation results, the proposed scheme is observed to perform better in terms of achievable network lifetime as compared to similar existing schemes like S-MAC. Furthermore, simulations show that:

- The higher the rate of transmission of packets (sampling packets) the less the consumption of energy.

- At a fixed Duty Cycle (DC), the higher the message inter-arrival period the greater the difference of consumed energy between Proposed Protocol and SMAC.

In our future work, we will implement this proposed protocol on a Mote-based sensor network platform and evaluate its performance through real experiments. Then, we will add more features to the sleep/listen schedule in order to make it more energy efficient.

We also plan to define a specific interface between the MAC and Network layers to use some crucial features of the latter. We believe that a MAC protocol can be more efficient is it has some information available in Network layer such as number of hops to the Base Station.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramanian, E. Cayirci, “A Survey on Sensor Networks”, IEEE Communications Magazine, August 2002.

[2] Archana Bharathidasan, Vijay Anand Sai Ponduru, “Sensor Networks: An Overview”, Department of Computer Science, Technical Report, University of California, Davis, 2003.

[3] Ilker Demirkol, Cem Ersoy, Fatih Alagöz, “MAC Protocols for Wireless Sensor Networks: a Survey”, IEEE Communications Magazine, in press, 2006.

[4] T.V. Dam, K. Langendoen, “An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks”, The First ACM Conference on Embedded Networked Sensor Systems (Sensys‘03), Los Angeles, CA, USA, November, 2003.

[5] W. Ye, J. Heidemann, D. Estrin, “Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks”, IEEE/ACM Transactions on Networking, Volume: 12, Issue: 3, Pages:493 - 506, June 2004.

[6] “The network simulator”, http://www.isi.edu/nsnam/ns/

[7] A. Mainwaring, J. Polastre, R. Szewczykt, D. Culler, J. Anderson, “Wireless Sensor Networks for Habitat Monitoring”, First ACM Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, USA, September 28, 2002.

[8] T. S. Rappaport, “Wireless Communications, Principle and Practice”, Prentice Hall, 1996.

[9] L. Kleinrock, F. Tobagi, “Packet switching in radio channels: Part 1-carrier sense multiple access modes and their throughput delay characteristics”, IEEE Transactions on Communications, Pages: 1400-1416, December 1975.

[10] S. Singh and C. S. Raghavendra, “PAMAS: Power aware multi-access protocol with signalling for ad hoc networks,” ACM Comput. Commun. Rev., vol. 28, no. 3, pp. 5–26, July 1998.

[11] LAN MAN Standards Committee of the IEEE Computer Society, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications”, IEEE Std 802.11-1997.

[12] G. Lu, B. Krishnamachari, C.S. Raghavendra, “An adaptive energy efficient and low-latency MAC for data gathering in wireless sensor networks”, Proceedings of 18th International Parallel and Distributed Processing Symposium, Pages: 224, 26-30 April 2004.

[13] K. Martinez, J. K. Hart and R. Ong, “Environmental Sensor Networks”, IEEE Computer magazine, Volume 37, Issue 8, Page(s):50 - 56, Aug. 2004 .

[14] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, *and* Deborah Estrin, “Habitat Monitoring With Sensor Networks”, Communications of the ACM, Vol. 47, No. 6, June 2004.

[15] R. Holman, J. Stanley, T. Ozkan-Haller, “Applying Video Sensor Networks to Nearshore Environment Monitoring”, IEEE Pervasive Computing , Volume 2, Issue 4, Page(s): 14 – 21, Oct.-Dec. 2003.

[16] Z. Haas, J. Deng, "Dual Busy Tone Multiple Access (DBTMA)—A Multiple Access Control Scheme for Ad Hoc Networks", IEEE Transactions On Communications, VOL. 50, NO. 6, June 2002.

APPENDIX A. TCL SCRIPT FOR ns-2 SIMULATIONS

```
set val(chan)              Channel/WirelessChannel
set val(prop)              Propagation/TwoRayGround
set val(netif)             Phy/WirelessPhy
set val(mac)                Mac/SMAC              ;# MAC type
set val(ifq)               Queue/DropTail/PriQueue
#set val(ifq)              CMUPriQueue
set val(ll)                LL
set val(ant)         Antenna/OmniAntenna

set val(x)                 600       ;# X dimension of the topography
set val(y)                 600              ;# Y dimension of the topography
set val(cp)                "../mobility/scene/cbr-50-10-4-512"
set val(sc)                "../mobility/scene/scen-670x670-50-600-20-0"

set val(ifqlen)            500              ;# max packet in ifq
set val(nn)                4                ;# number of nodes
set val(seed)              0.0
set val(stop)              10000;#65.0              ;# simulation time
set val(tr)                MyTest.tr        ;# trace file
set val(nam)               MyTest.nam       ;# animation file
set val(lm)                "off"      ;# log movement
set val(energymodel)      EnergyModel    ;
set val(radiomodel)       RadioModel    ;
set val(initialenergy)     100000       ;# Initial energy in Joules

Mac/SMAC set syncFlag_ 1

Mac/SMAC set dutyCycle_ 10
```

```
#POINTERS ARE PASSED TO THE FILES OPENED TO CONTAIN NAM-TRACE AND WIRELESS
TRACE

set ns [new Simulator]
set tracefd [open Ejemplo-para-todas-las-simulaciones.tr w]
set windowVsTime2 [open Ejemplo-para-todas-las-simulaciones.tr w]
set namtrace [open Ejemplo-para-todas-las-simulaciones.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#CREATES AN AREA SPECIFIED BY THE 'load_flatgrid' method
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#PARAMETERS ALREADY DEFINED ARE USED HERE IN ORDER TO CONFIGURE EACH NODE

$ns node-config -adhocRouting AODV \      ;#Routing protocol
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channel [new $val(chan)] \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace ON \
                        -energyModel $val(energymodel) \
                        -idlePower 1.0 \
                    -rxPower 1.0 \
                    -txPower 1.0 \
                    -sleepPower 0.001 \
                    -transitionPower 0.2 \
                    -transitionTime 0.005 \
                    -initialEnergy $val(initialenergy)

        $ns set WirelessNewTrace_ ON

#EACH NODE IS CREATED ACCORDING TO THE ABOVE SPECS AND PASSED A POINTER

for { set i 0 } { $i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
}

#INITIAL POSITION FOR EACH NODE IS SET

$node_(0) set X_ 300.0
$node_(0) set Y_ 10.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 300.0
$node_(1) set Y_ 160.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 310.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 300.0
$node_(3) set Y_ 460.0
$node_(3) set Z_ 0.0

set tcp [new Agent/UDP]
```

```
$tcp set class_ 2
set tcp2 [new Agent/UDP]
$tcp2 set class_ 2
set tcp3 [new Agent/UDP]
$tcp3 set class_ 2

set sink [new Agent/Null]

$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $tcp2
$ns attach-agent $node_(3) $sink



set cbr_(0) [new Application/Traffic/Poisson]
$cbr_(0) set packetSize_ 10
$cbr_(0) set interval_ 300;
$cbr_(0) set random_ 100000;
$cbr_(0) set maxpkts_ 20000
$cbr_(0) attach-agent $tcp

$ns connect $tcp $sink

$ns at 100.00 "$cbr_(0) start"

#INITIAL POSITION FOR NAM: IT IS REALLY THE SIZE OF EACH NODE IN THE SIMULATION

for {set i 0} {$i < $val(nn)} { incr i } {
        $ns initial_node_pos $node_($i) 30
}

#TELL THE NODES WHEN SIMULATION IS GOING TO END:

for {set i 0} {$i < $val(nn) } { incr i } {
        $ns at $val(stop) "$node_($i) reset";
}

set b [$node_(0) set mac_(0)]

set d [Mac/SMAC set syncFlag_]

set c [Mac/SMAC set dutyCycle_]

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

proc stop {} {

        global ns tracefd namtrace
        $ns flush-trace
        close $tracefd
        close $namtrace
}

$ns run
```

APPENDIX B. IMPLEMENTATION OF PROPOSED PROTOCOL

The implementation of the protocol was made in the ns-2 network simulator version 2.8. In the remainder of this section, the most relevant parts of the implementation are explained step by step and after the complete operation of the whole protocol is explained. All descriptions and diagrams are shown as written in the code.

B.1 Construction of packets sent in the Tone Time

In order for the nodes must be able to recognize the packets sent in the Tone Time, which we have called *tone packets*, they must be identified with a number different from those that identify other packets:

```
#define DATA_PKT 0
#define RTS_PKT 1
#define CTS_PKT 2
#define ACK_PKT 3
#define SYNC_PKT 4
//------------------------
#define TONE_PKT 5 //                ←tone packets are 5th type of packets in the network
//-------------------------
```

Furthermore, the size of the packet, header plus payload (3 bytes), must be indicated:

```
#define SIZEOF_SMAC_DATAPKT 50
#define SIZEOF_SMAC_CTRLPKT 10
#define SIZEOF_SMAC_SYNCPKT 9
#define SIZEOF_SMAC_TONEPKT 3; //←Tone packets are 3 bytes long
```

As shown above the tone packets are three bytes long. The tone packet structure is shown below:

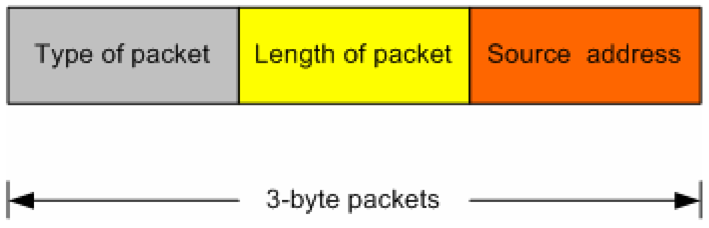


Figure 28. Tone Packets 3-byte structure.

The correspondent block of code to create a structure for tone packets is:

```
struct smac_tone_frame {
  int type;
  int length;
  int srcAddr; //                                   ←Source address
};
```

Tone packets do not have a destination address in it because any node that hears this packets will wake up. It is intended for the future that a node that sends Tone packets have route information and includes a destination address in the packet so neighbor nodes that are not in the route of the packet do not consume energy in overhearing such packets.

B.2 Control variables

The variables shown in bold in code below will be used to control the decisions the code will make. All of them are class variables, so each node will have different versions of them.

Variable declarations are followed by a brief explanation of their major characteristics, also in bold.

```
class SMAC : public Mac {

    int StatusModifiedProtocolTimer        //←Variable used to decide when to
    wake up or not according to the desired schedule for environmental monitoring

    int Periods_Listen_Tone_Packets        //←Variable used in case a Tone packet
    has been received so the node must awake no matter its status

    int NPProtocoloActivado;               //←1 if proposed protocol
    synchronization has already been activated

    int NPActiveStatus;                    //←1 if in Listen Time

    int NPRemainingSleepPeriods;           //←0 if no remaining sleep periods
    exist, !=0 if it still has some periods to sleep

    int NPRemainingListenPeriods;          //←0 if no listen periods exist, !=0 if
    still has to listen

    int Transmission_Control;              //←1 if a tone packet has been received

    int envio_paquetes_urgentes;           //←1 if the node has a triggered
    packets that needs to be sent

}
```

The class *SMAC* shown in the code above is the class that has implemented some functions of the layer 2 of the OSI model. The other functions are common to other layer 2 protocols such as 802.11 and are inherited from superclass *Mac*.

B.3 Timers

Timers in the code control the whole operation of the proposed protocol. Among the many functions are the following:

- Control the schedule: the start or finish of the listen time and the sleep time
- Control the time for back-off procedures
- Maintain the synchronization
- Carrier sense mechanisms

Even though timer classes have different methods, they all have some common methods. These methods are explained below:

- *sched(double number)* method: It is used to initiate a timer with a given number that represents the time that will pass until the timer expires
- *resched(double time)* method: Used when function *sched* fails to schedule a timer
- *expire() method:* Called immediately after the timer has expired in order to schedule the desired operations

The base class to create timers is *Timerhandler*. Class *SmacTimer* inherits from *Timerhandler* part of its functionality and adds some functions as *expire* and *busy* methods. Function *expire* will be called when the timer has expired so some functions must be performed and function *busy* will be invoked to check if the timer is busy or can be invoked again.

```
class SmacTimer : public TimerHandler {          //←Functions inherited from Timerhandler
    public:
    SmacTimer(SMAC *a) : TimerHandler() {a_ = a; }   //←Each timer "belongs" to a specific
    node
    virtual void expire(Event *e) = 0 ;
    int busy() ;
    SMAC *a_;
};
```

The proposed protocol main timer is *ModifiedProtocolTimer* that inherits its functionality from *SmacTimer* class. Also has its own *expire* method.

```
class ModifiedProtocolTimer : public SmacTimer {
      public:
      ModifiedProtocolTimer(SMAC *a) : SmacTimer(a) {}
      void expire(Event *e);
};
```

The most important timer will be an instance of the class SmacCounterTimer (inherited from S-MAC functionality to maintain the listen/sleep schedule); which inherits characteristics from *SmacTimer* class. An instance of this class will control the whole schedule of the protocol with the values of the variables declared in the body of the class.

```
class SmacCounterTimer : public SmacTimer {
       public:
       friend class SMAC;
       SmacCounterTimer(SMAC *a, int i) : SmacTimer(a) {index_ = i;}
       void sched(double t);
       void expire(Event *e);
protected:
double syncTime_;                                    ←SYNC Time
double dataTime_;                                    ←Data Time
double sleepTime_;                                   ←Sleep Time
 double cycleTime_;//                                ←Tf
double toneTime_; //                                  ←Tone Time
};
```

B.4 Relevant methods

Methods of the *SMAC* class are explained in the paragraphs below. Only the most relevant lines of code are show in the method to highlight the purpose and operation of each one.

- The *handleTone* method will be called as soon as a Tone Packet has been received by a node. Once the packet has been received, the node starts another broadcast so the complete route from the originating node to the Base Station is available to disseminate the packet.

```
void SMAC::handleTone(Packet *p)
{
    struct hdr_cmn *ch = HDR_CMN(p);                    // ←Pointers to the packet received
    struct hdr_smac * sh = HDR_SMAC(p);

    Periods_Listen_Tone_Packets = 30;

    if((state_ == SLEEP || state_ == IDLE) && (radioState_ == RADIO_SLP || radioState_ ==
RADIO_IDLE)){                                           //←If radio is not being utilized

        if(state_ == SLEEP){
            wakeup();                                   //←If node is sleeping, wake up
        }
        howToSend_ = BCAST_TONE_PACKET;                 //←Type of packet to transmit
        state_ = CR_SENSE;                              //←Start carrier sense
        double cw = (Random::random() % SYNC_CW) * slotTime_sec_;
        mhCS_.sched(CLKTICK2SEC(difs_) + cw);           //←Wait a random time before sending
    }
}
```

➢ A function must be created that permits to send the tone packets when needed, which is the function *sendTONE*, shown below:

```
bool SMAC::sendTONE(){

    Packet *p = Packet::alloc();
    struct smac_tone_frame *cf = (struct smac_tone_frame *)p->access(hdr_mac::offset_);
    struct hdr_cmn *ch = HDR_CMN(p);

    //THE CONSTRUCTION ON THE LAYER 2 PACKET TAKES PLACE HERE BEFORE
    SENDING PACKET

    cf->size() = SIZEOF_TONEPKT;                //←Size of layer-2 packet
    cf->direction() = hdr_cmn::DOWN;            //←Transmission
    cf->type = TONE_PKT;                        //←Type of packet
    cf->srcAddr = index_;                       //←Layer 2 address of sending node

    if (chkRadio()) {
        transmit(p);                            //←Transmit packet
        return 1;
    } else
     return 0;
}
```

➢ The method handleCsTimer as called as soon as the contention time has finished. Once finished, the node can send a packet because it has not sensed any other transmission.

```
void SMAC::handleCsTimer() {

    // carrier sense successful
    switch(howToSend_) {                        //←Check what kind of packet is
    about to be sent
        case BCAST_TONE_PACKET:                 //←Case a triggered packet
```

```
            if (sendTONE()){
                state_ = IDLE;
            }
            break;
        }
    }
```

- *handleCounterTimer* is the method that controls the schedule of the protocol in each node through the timer called *mhCounte*r_; that is an instance of the *SmacCounterTimer* class. This *mhCounte*r_ is invoked when the schedule is in a new state (e.g. Sleep Time), the timer decreases the value passed and when it is zero it changes the state of the schedule (e.g. Tone Time). In this way the schedule goes from *Sleep Time* to *Tone Time* to *Sync Time* to *Data Time* and again to Sleep Time. Only a small part of code of the method is shown here for simplicity purpose.

```
void SMAC::handleCounterTimer(int id) {

    if (mhCounter_[id]->value_ == sleepTime_) {        //←The timer has the value of
    sleepTime_, so it is coming from sleep are now in Tone Time

    if((state_ == SLEEP || state_ == IDLE) && (radioState_ == RADIO_SLP || radioState_ ==
    RADIO_IDLE)){
        if(Transmission_Control == 1){                  //←There is the need to transmit a
        Tone packet
            if(state_ == SLEEP){
                wakeup();                               //←Wake up is sleeping
            }
            howToSend_ = BCAST_URGENT_PACKET;                  //←Broadcast packet
            state_ = CR_SENSE;                                 //←Start carrier sense
            double cw = (Random::random() % SYNC_CW) * slotTime_sec_;//←Wait a random
            time before sending
```

```
                mhCS_.sched(CLKTICK2SEC(difs_) + cw);                    //
            }
                mhCounter_[id]->sched(CLKTICK2SEC(toneTime_));//←Prepare to go to Sync Time
                after Tone Time
    }}}
```

Until now we have seen methods that entirely deal with deciding when to send a tone packet, transmit the packet or what to do when a node receives a tone packet. In the remainder of this section, methods that change the underlying synchronization are shown.

- The constructor of the SMAC class initializes the variable *StatusModifiedProtocolTimer* with a value of 1. With this value, the schedule on top of the underlying one is the same. Furthermore, the constructor fixes the values of the schedule.

```
SMAC::SMAC() : Mac{
    StatusModifiedProtocolTimer=1;              //←Underlying schedule unchanged
    ListenPeriod = 1;                           //←NLSP
    .
    .
    .
    toneTime_ = 0.6*syncTime_;                  //←Tone Time fixed to have a length of
    0.6 that of the SYNC Time

    listenTime_ = toneTime_ + syncTime_ + dataTime_;//←As seen on Figure 11, the Listen
    Time is comprised of three subintervals

    cycleTime_ = listenTime_ * 100 / dutyCycle_ + 1;    //←CycleTime = Tf or Time Frame

    sleepTime_ = cycleTime_ - listenTime_;      //←Tf = Sleep Time + Listen Time
}
```

- If the proposed protocol schedule, working on top of the underlying S-MAC schedule, remained the same as the underlying one, there would not be energy savings, but as seen on the chunk of code below, the variables *StatusModifiedProtocolTimer* and *ListenPeriod* are changed to other values to configure a different schedule.

  For simulation purposes, the nodes synchronize to a second schedule (proposed protocol schedule) with received acknowledge (ACK) packets after sending data packets. As seen below, Node 0 will set its variable *StatusModifiedProtocolTimer* to *Value_Sleeping*. This variable *StatusModifiedProtocolTimer* is analogous to TCALP shown in Figure 12; however not expressed in time units but in number of cycles (T_f). *StatusModifiedProtocolTimer* is set to a value that is decreased continuously with each T_f. While this value is different than zero the node sleeps, when zero the node awakes again. ListenPeriod variable is analogous to NLSP (see Figure 12). While this variable is different than zero, the node wakes. It is interesting to highlight that the value of *StatusModifiedProtocolTimer* decreases as the number of the node that receives the ACK packets increases. This is because (see Figure 15), node 1 that is the one that generates triggered packets, receives the ACK packet one period before than node 2 and node 2 receives ACK packet one period before than node 3 and so forth. So to have all nodes awake at the same time, a node must sleep one more period than its immediate neighbor that is nearer to the Base Station. The proposed of synchronization schemeis shown in Figure 29.

```
void SMAC::handleACK(Packet *p) {
    if (index_ == 0){
            StatusModifiedProtocolTimer = Value_Sleeping;
            ListenPeriod = NLSP;
    }
    if (index_ == 1){
            StatusModifiedProtocolTimer = Value_Sleeping-1;
```

```
                ListenPeriod = NLSP;
        }
        if (index_ == 2){
                StatusModifiedProtocolTimer = Value_Sleeping-2;
                ListenPeriod = NLSP;
        }
        if (index_ == 3){
                StatusModifiedProtocolTimer = Value_Sleeping-3;
                ListenPeriod = NLSP;
        }
}
```

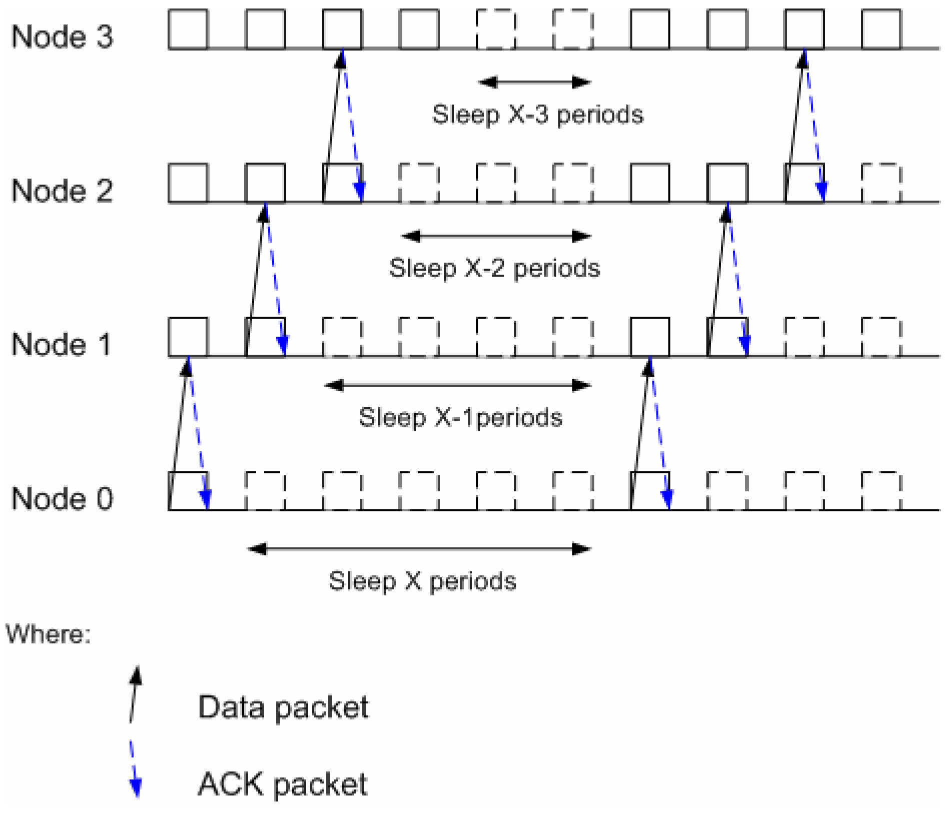


Figure 29. Proposed protocol synchronization

- Some of *handleCounterTimer* method functionalities were explained before. Now other functionalities are explained that pertain to the schedule of the proposed protocol. This method "decides" when a node awakes or goes to sleep to comply with the desired new schedule.

```
void SMAC::handleCounterTimer(int id) {
    if (mhCounter_[id]->value_ == toneTime_) {          //←Coming from tone time, now on
    Sync Time
    StatusModifiedProtocolTimer--;                       //←Variable decreased with each new
    Sync Time
    if(StatusModifiedProtocolTimer == 0  || envio_paquetes_urgentes == 1){


        wakeup();                                        //←Wakeup                    if
        //StatusModifiedProtocolTimer == 0
        StatusModifiedProtocolTimer = 1;                 //←Set  StatusModifiedProtocolTimer
        //to 1 to sleep one period
        ListenPeriod = 0;                                //←NLSP=0
    }

    else if(ListenPeriod !=0){
        wakeup();                                        //←Wake-up if NLSP is different than
        //0
        ListenPeriod--;                                  //←Decrease NLSP
    }
    else if (StatusModifiedProtocolTimer != 0){
        sleep();                                         //←Go      to      sleep      if
        //StatusModifiedProtocolTimer is different than 0
    }
    if(Periods_Listen_Tone_Packets != 0) {
        wakeup();                                        //←Wakeup because the node has
        //received a Tone packet
        Periods_Listen_Tone_Packets--;
        }
    }
```

B.5 Operation of proposed protocol implementation

Figure 30 explains the basic operation of proposed protocol when an interesting event is sensed. The node generates a Tone packet and will broadcast it. The diagram uses the methods, variables and timers already and briefly explained in section 4.1.4.

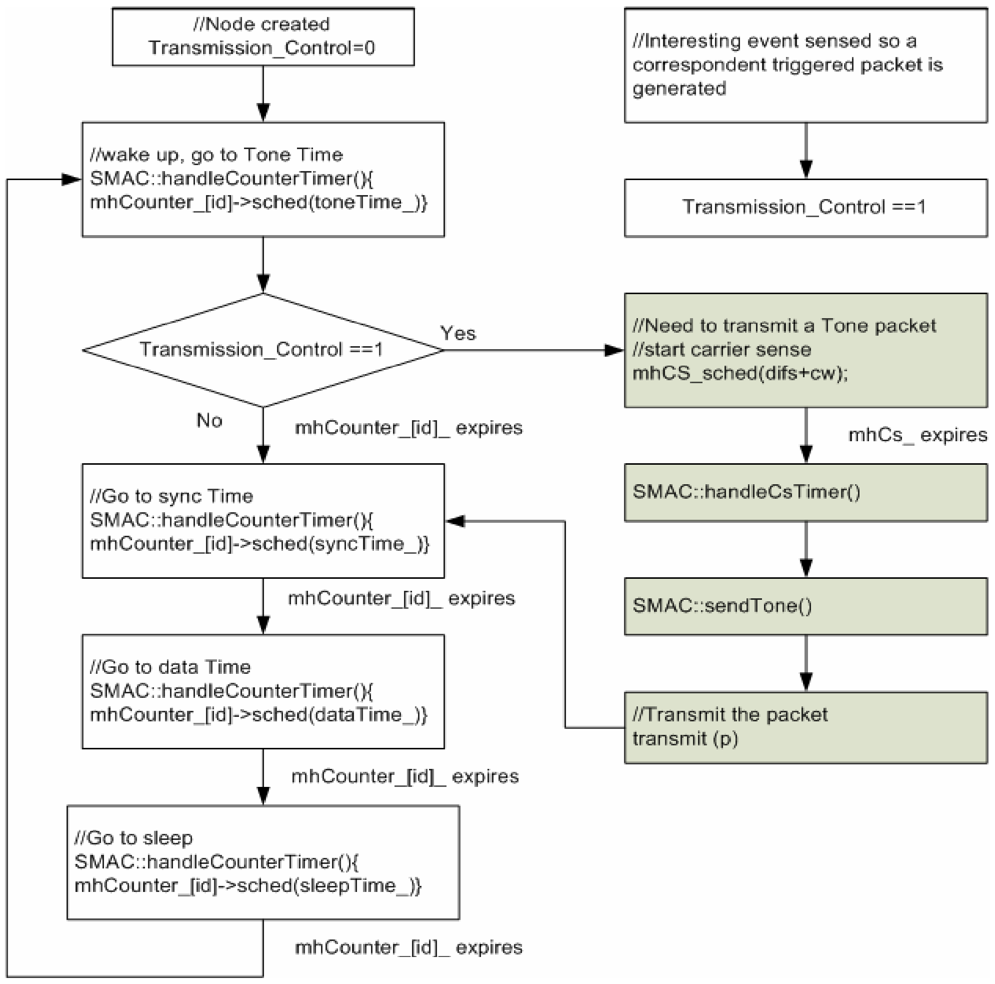

//Node created
Transmission_Control=0
//wake up, go to Tone Time
SMAC::handleCounterTimer(){
mhCounter_[id]->sched(toneTime_)}
Transmission_Control ==1
Yes
No
mhCounter_[id]_ expires
//Go to sync Time
SMAC::handleCounterTimer(){
mhCounter_[id]->sched(syncTime_)}
mhCounter_[id]_ expires
//Go to data Time
SMAC::handleCounterTimer(){
mhCounter_[id]->sched(dataTime_)}
mhCounter_[id]_ expires
//Go to sleep
SMAC::handleCounterTimer(){
mhCounter_[id]->sched(sleepTime_)}
mhCounter_[id]_ expires
//Interesting event sensed so a correspondent triggered packet is generated
Transmission_Control ==1
//Need to transmit a Tone packet
//start carrier sense
mhCS_sched(difs+cw);
mhCs_ expires
SMAC::handleCsTimer()
SMAC::sendTone()
//Transmit the packet
transmit (p)

Figure 30. Code for proposed protocol operation when an interesting event is sensed

Figure 31 below shows the proposed protocol operation when a Tone packet is received. As explained in chapter 3, the node that receives a Tone packet will broadcast another Tone packet to make other nodes in the route to wake up.

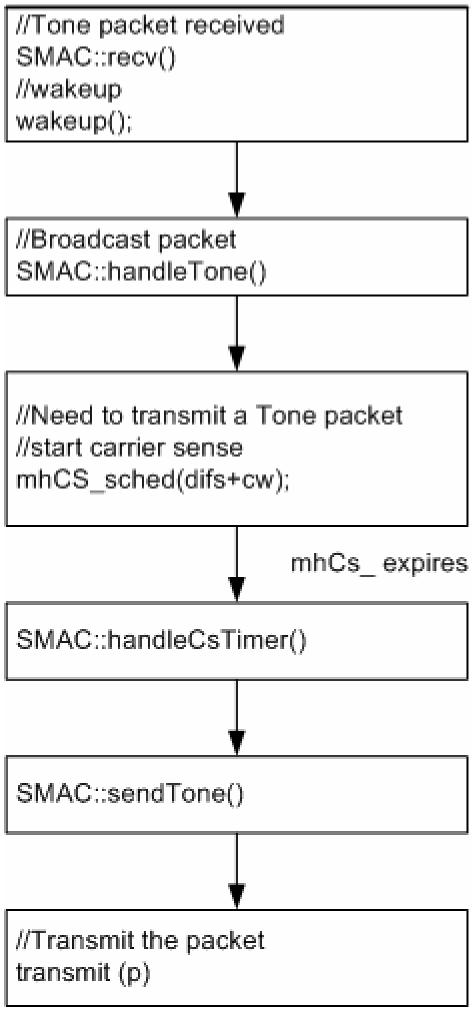
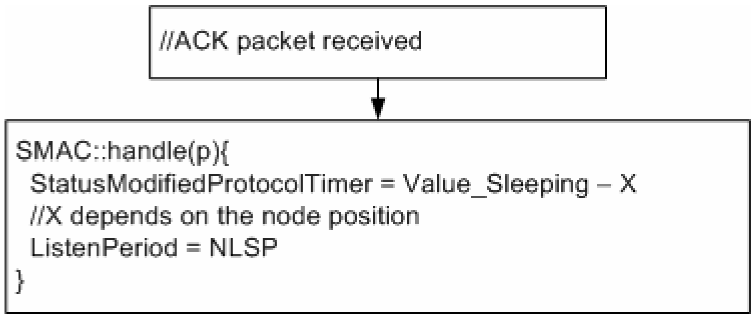


Figure 31. Code for proposed protocol operation when a Tone packet is received

Figure 32 shows the operation correspondent for ACK synchronization explained in previous paragraphs. The procedure to create a second schedule on top of existing one in shown in Figure 33.



```
//ACK packet received
```

```
SMAC::handle(p){
  StatusModifiedProtocolTimer = Value_Sleeping – X
  //X depends on the node position
  ListenPeriod = NLSP
}
```

Figure 32. Code for synchronization with ACK packets

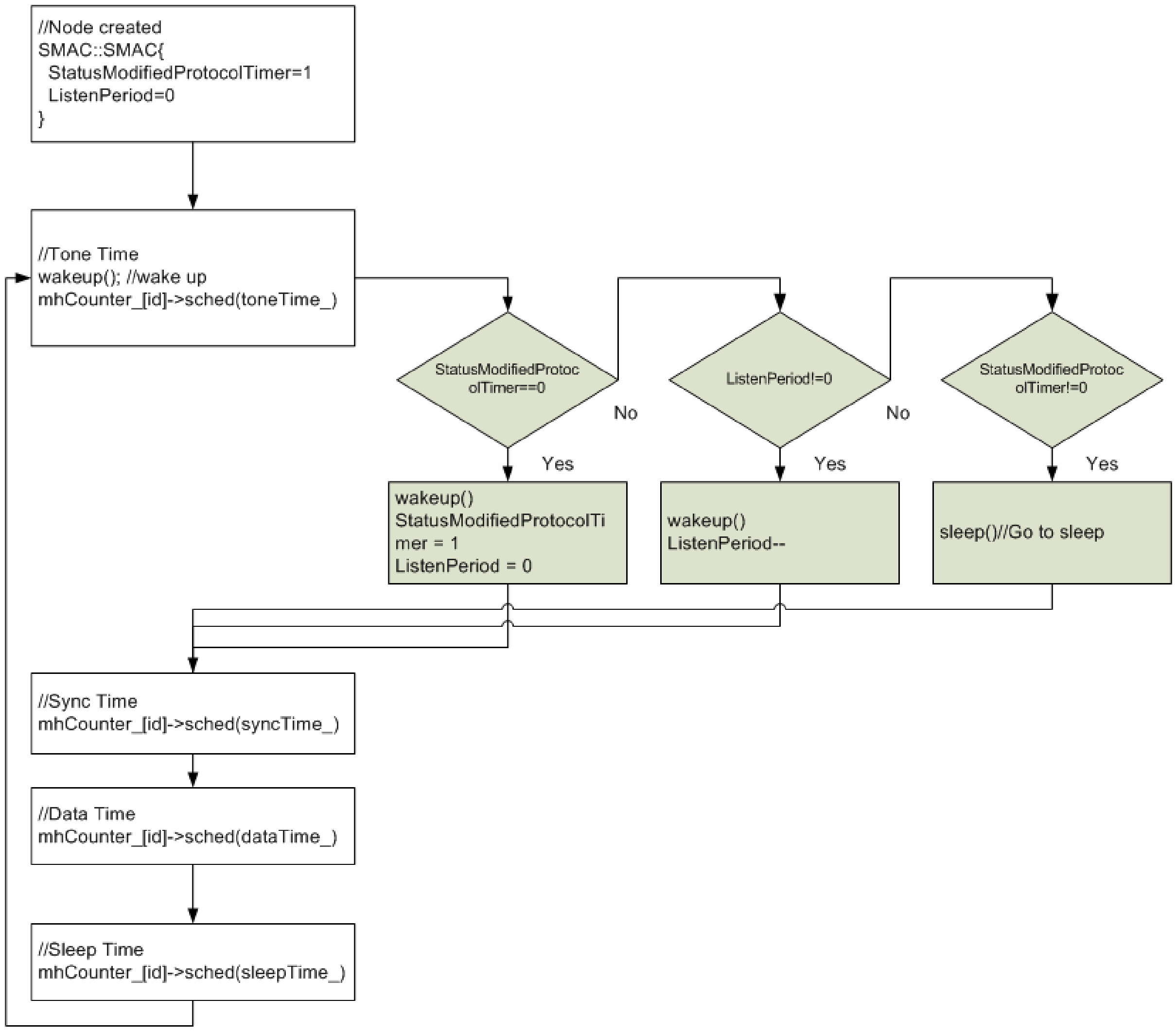

//Node created
SMAC::SMAC{
StatusModifiedProtocolTimer=1
ListenPeriod=0
}
//Tone Time
wakeup(); //wake up
mhCounter_[id]->sched(toneTime_)
StatusModifiedProtocolTimer==0
No
Yes
ListenPeriod!=0
No
Yes
StatusModifiedProtocolTimer!=0
Yes
wakeup()
StatusModifiedProtocolTimer = 1
ListenPeriod = 0
wakeup()
ListenPeriod--
sleep()//Go to sleep
//Sync Time
mhCounter_[id]->sched(syncTime_)
//Data Time
mhCounter_[id]->sched(dataTime_)
//Sleep Time
mhCounter_[id]->sched(sleepTime_)

Figure 33. Code for proposed protocol procedure to create and use a second schedule on top of existing one