Exploiting Redundancy for Secure Data Dissemination in Wireless Sensor Networks

Explotando Redundancia para Diseminación Segura de Datos en Redes Inalámbricas de Sensores

Luis E. Palafox and J. Antonio García Macías
CICESE Research Center
Computer Science Department
Km. 107 Carretera Tijuana-Ensenada C.P. 22860
Ensenada, B.C., México
palafox@cicese.mx, jagm@cicese.mx

Article received on June 18, 2007; accepted on October 24, 2007

Abstract.
Monitoring the environment is one of the main applications of wireless sensor networks. Given that these networks are densely populated and that local variations in the environmental variables are small, a large amount of redundant data is generated by sensor nodes. In this paper we present a protocol for data dissemination in hierarchical clustered sensor networks that integrates security and reduces communication overhead by removing data redundancy from the network. Furthermore, we show that by using our protocol we can provide security to the network without spending additional energy resources, particularly when we are dealing with high levels of redundancy. Additionally, a military monitoring application is introduced as a case study to evaluate the functionality of our protocol.
Keywords: Secure data dissemination, secure data aggregation, redundancy in WSN.

Resumen.
El monitoreo del medio ambiente es una de las muchas aplicaciones de las redes inalámbricas de sensores. Dado que estas redes están densamente pobladas y que las variaciones locales de variables ambientales son pequeñas, los nodos de sensado generan una gran cantidad de datos redundantes. En este artículo se presenta un protocolo de diseminación de datos en una red de sensores jerárquica con clusters, el cual integra seguridad y reduce el sobrecosto de comunicación al remover datos redundantes de la red. Además, se muestra que al utilizar el protocolo se puede proveer de seguridad a la red sin consumir recursos energéticos adicionales, esto particularmente cuando se trata con altos niveles de redundancia. Adicionalmente, se introduce una aplicación de monitoreo militar como caso de estudio para evaluar la funcionalidad de nuestro protocolo.
Palabras clave: Diseminación segura de datos, agregación segura de datos, redundancia en redes inalámbricas de sensores.

1 Introduction

Recently, many potential applications for wireless sensor networks (WSN) that require hundreds or maybe thousands of nodes have been published; thus, high scalability is a fundamental requirement. Mesh-type networks do not fulfill this requirement, as every node plays the same role and has the responsibility of sensing the environment and routing traffic directed to other nodes and/or to the base station. In contrast, clustering is a standard approach for achieving energy efficiency and scalability in wireless sensor networks [Banerjee and Khuller, 2001]. Clustering facilitates the distribution of control over the network and, hence, enables the locality of communication. The nodes that belong to a specific cluster can only interact directly with a predetermined node that plays the role of the cluster leader, also known as cluster-head. Only cluster-heads need to communicate through larger distances in order to reach the base station, however, this drawback can be alleviated by using hierarchical clustering (i.e. by applying clustering recursively over the cluster-heads of a lower level). Additionally, the tasks related to these hierarchical networks are more distributed as opposed to networks that have a flat structure (i.e. mesh networks). For instance, sensing the environment, and communicating the sensor readings to the cluster-head are tasks corresponding to the sensor nodes;

meanwhile, the cluster-head has the responsibility of processing the data read from the sensor nodes in its cluster (i.e. through data aggregation) and sending the results directly to the base station or to a higher hierarchy level node in the network.

In this paper we focus particularly on the problem of transmitting the sensed data from the sensor nodes to the cluster-head. This problem may seem trivial at first, but in fact, as we mentioned earlier, many applications require densely populated networks that monitor changes in the environment. This requirement is mainly because the nodes involved are prone to failure if the environment on which they are deployed is very hostile. Furthermore, in order to reduce node cost there is no added physical protection to the sensor nodes. Due to the fact that placing many sensor nodes on the same area is common practice, the network frequently generates high levels of redundant data; this results in excessive amount of power consumed for communicating the data from the sensor nodes to the cluster-head.

It should be noted that although many documented applications require data integrity and security services such as authentication and confidentiality, security is an issue that has been frequently overlooked, in communication networks in general and in WSNs in particular. Additionally, because of the well documented extreme resource limitations of this type of networks, the question about the feasibility of integrating security mechanisms into WSNs has often been raised. Furthermore, being still an emerging research area, efforts in WSN have been directed toward issues such as energy-efficient protocol design, prototype application implementations and others, while security has not been the most pressing aspect.

In this paper we introduce a novel node-to-cluster-head data transmission mechanism that takes full advantage of the high levels of data redundancy for reducing data consumption while also providing data integrity/security services. As we show at the end of the paper, the integration of our security oriented protocol does not incur additionally energy costs to the network; in fact, in some cases it results on significant power savings, which eventually could extend the life of the network. This paper is organized as follows: in Section II we present some work related to security and data transmission in hierarchical sensor networks, in Section III we present our proposal for secure data dissemination in hierarchical wireless sensor networks, in Section IV we present a series of experiments that we performed with our protocol; finally, in Section V we give concluding remarks and outline future work.

2 Related work

We now briefly summarize some of the most relevant related work. Traditionally, protecting against most network attacks relies heavily on integrating an efficient authentication mechanism; in the context of data dissemination in hierarchical sensor networks, sending a Message Authentication Code (MAC) attached to the sensor reading has been used for this purpose. The MAC code is generated through a well known function (i.e. HMAC [Bellare *et al.*, 1996], SHA-1 [NIST, 1995]) that uses the sensed data and a secret key (K) as inputs. The secret key must be shared by all the sensor nodes and the cluster-head. The receiving node (in our case, the cluster-head) must try to duplicate the MAC through the received data and the secret key that it shares with the sensor nodes to verify the message authenticity. The authentication process is shown in Fig. 1.

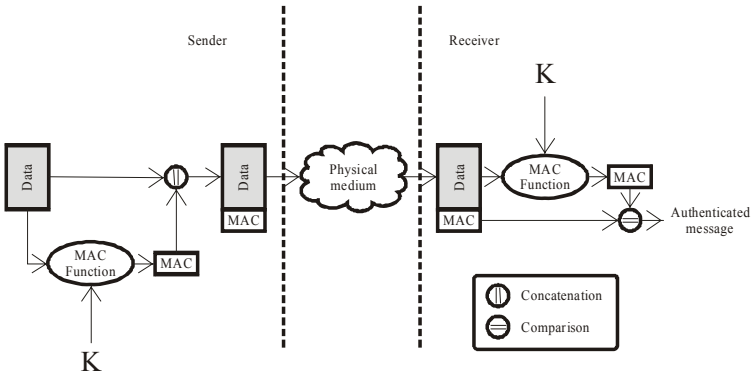

Fig. 1. Traditional authentication process [Schneier, 1996]

Perrig and Tygar (2003) show that integrating the MAC mechanism to a sensor network through the SPINS (Security Protocols In Networked Sensors) family of security protocols [Perrig *et al.*, 2002] only adds 6 bytes of overhead to the data packet. In terms of consumed energy this accounts for only 20% of the total amount of consumed energy. In Table 1 we show the cost of adding security to a sensor network through the SPINS security protocols. As it can be observed, most of the energy cost is due to data transmission and not because of the computational cost added by the security algorithms.

Table 1. Energy costs of adding security protocols to a sensor network [Perrig and Tygar, 2003]

71%	Data Transmission
20%	MAC Transmission
7%	Nonce Transmission (for freshness)
2%	MAC and encryption computation

ESPDA [Çam *et al.*, 2006] (Energy-Efficient Secure Pattern-based Data Aggregation) is another node-to-cluster-head protocol based on pattern generation for hierarchical sensor networks that takes advantage of redundant data. However, their main goal is not security; but instead, a mechanism for selecting the nodes whose sensed region overlap, in order to know which nodes can be turned off. Additionally, redundancy detection is based on pattern generation through a specific algorithm that in order to be efficient has to yield some certain precision. In another publication, the same authors propose an energy-efficient protocol for wireless sensor networks [Çam *et al.*, 2003] that works in conjunction with ESPDA. This protocol defines a simple key distribution scheme for hierarchical sensor networks and uses the traditional approach to provide authentication (shown in Fig. 1).

Our approach is similar to ESPDA in the fact that it relies on redundant data to achieve power efficiency. However, as it will be shown in the next section, our protocol defines a novel way for detecting redundant data and uses a two-phase mechanism to provide authentication. This mechanism differs substantially from the one shown in Fig. 1 mainly because in the traditional approach the MAC code is appended to the data packet and in our approach, as we will show later, a special packet containing the MAC code is sent prior to the data packet.

3 Our approach for node to cluster-head data dissemination

3.1 Design

The proposed protocol is based on a novel two-phase authentication process, this process detects redundant data while achieving authentication. Additionally, other security services such as confidentiality can be provided trivially by means of the same secret key shared by the cluster. Basically, our protocol is based on the following premise, if:

$$A = MAC_f(K_1, M_1) \tag{1}$$

and

$$B = MAC_f(K_2, M_2) \tag{2}$$

then

$$(A = B) \forall (K_1, M_1) = (K_2, M_2),\ f \tag{3}$$

In equations 1, 2 and 3 we express formally that in our protocol, redundancy detection is based on the idea that if we have two equal MAC codes (A, B) generated by the same MAC function f, we would expect that the keys K_1 and K_2 and the messages M_1 and M_2 used for generating A and B to be equal. Therefore, if the same key is shared by all cluster nodes, the cluster-head should be able to detect redundant data after receiving just the MAC codes for all of them.

As we will later show, we selected CBC-MAC [Bellare et al., 2000; ISO, 1989] (Cipher Block Chaining Message Authentication Code) algorithm for MAC computation. One important aspect to consider in CBC-MAC as well as in other MAC functions is the fact that collisions are a possibility, this is, two different messages can produce the same MAC code while using the same key, and the presence of collisions would negatively affect our protocol by detecting redundant data while this would not be necessarily true (i.e. false positives). However, the CBC-MAC collision probability for two messages with length *m* using a *n*-lenght MAC code and the same key is upper bounded by [Black and Rogaway, 2005]:

$$P_n(m) \leq \frac{(2m/n)^2}{2^n} \tag{4}$$

Thus, the probability of collision for our implementation (96-bit messages and 32-bit MAC codes) is very small:

$$P_{32}(96) \leq 8.3819 X 10^{-9} \tag{5}$$

Nevertheless, we considered the possibility of integrating a mechanism based on spatial correlation to further reduce collision probability, but decided that the additional memory and computational costs involved would not justify taking provisions for such an improbable scenario[1].

3.1.1 Message exchange

Regarding message exchange in our protocol, we considered five types of messages, in Table 2 we list those messages along with a brief description of its payload and payload length. We assigned an ID value for each message type in our protocol for later use in our implementation.

[1] The worst that could happen is that roughly once in 12.5 million times a false positive would appear, but the application should provide for means to verify the actual occurrence of the event anyway.

Table 2. Types of messages defined for our protocol

ID value	Description	Payload (Length in bytes)
0x4a	MAC code sent to cluster-head	MAC code (4)
0x4b	Data request from cluster-head to sensor nodes	Node list (1)
0x4c	Sensor data sent to cluster-head	Sensor data (20)
0x4d	Node-specific retransmission request	Empty (0)
0x4e	Counter update beacon	CNTR || MAC (6)

For our test implementation we considered packets with a 10 byte header as in IEEE 802.15.4 [IEEE, 2003] with a maximum packet size of 30 bytes (20 bytes of payload) as in TinyOS [Hill *et al.*, 2000].

The five defined messages are used during the different stages of the protocol. We describe the operation of the protocol as follows:

1. Nodes in a cluster sense data, cluster-head nodes are pre-defined through a cluster-head selection algorithm such as the one used by LEACH [Heinzelman *et al.*, 2002] (Low-Energy Adaptive Clustering Hierarchy).
2. Using their sensor readings $(DATA)$ and a counter $(CNTR)$ that is synchronized with the rest of the cluster, each node computes its own 4 byte MAC code using a shared cluster key $(K_{cluster})$, this code is attached to a packet with type *0x4a* and sent to the cluster-head. By using a secret counter that is shared by cluster members we provide implicit authentication (i.e. we do not ensure that a packet is sent by a specific node, but we ensure that the packet has been sent by an authorized node):

$$Node \rightarrow ClusterHead : MAC(K_{cluster}, DATA \,||\, CNTR) \tag{6}$$

3. Once that all MAC codes are received (or after a predefined time-out period), the cluster-head stores them on a memory buffer classifying them according to the node from which each one came from (source node).
4. Afterwards, the cluster-head detects which MAC codes are duplicated in the buffer, and can trivially infer which nodes have redundant data.
5. Once that the nodes with redundant data have been identified, the cluster-head constructs a list of nodes $(LIST)$ for which explicit data requests are necessary. Additional criteria, such as the remaining power on each node, can be used to include a node in the list.
6. Once that the list has been constructed, it is appended to one or more type *0x4b* packets (depending on the size of the list). These packets should be broadcast from the cluster-head to the sensor nodes. The packets are authenticated with the counter we mentioned before to assure freshness and avoid "store and replay" attacks. As we will show ahead, this method is the simplest one because it does not involve protocols from any other layer, but it is not the most efficient one. A more efficient way to do this is to use some header bits in the ACK signals from the MAC layer to indicate to each node if its data is required by the cluster-head.

$$ClusterHead \Rightarrow Nodes : LIST \,||\, MAC(K_{cluster}, LIST \,||\, CNTR) \tag{7}$$

7. The sensor nodes on which a specific data request was made, must integrate a packet type *0x4c* with their sensor readings, this packet must be sent to the cluster-head:

$$Node \rightarrow ClusterHead : DATA \tag{8}$$

8. Finally, the cluster-head knows the sensor readings from the nodes that received explicit requests, as well as from those that presented duplicated MAC codes; thus, verification of message integrity/authenticity can be performed by the cluster-head. If the verification fails, the cluster-head can request data to such nodes. Alternatively, an estimation algorithm may be used instead of asking for retransmission.

3.1.2 Counter Synchronization

The security of our protocol relies heavily on the secrecy of the counter being shared by all cluster members. Furthermore, for the protocol to work properly, the counter must be synchronized within the cluster (i.e. the internal counter value must be the same in all cluster members). Due to clock drift, it is possible that synchronization breaks down among cluster nodes during the lifetime of the network. For this reason, we are proposing a fairly simple counter synchronization technique in order to alleviate counter desynchronization. This technique is coordinated by the cluster-head, and consists on the cluster-head sending periodical counter update beacons (message type *0x4e*). These beacons contain a new counter value randomly generated by the cluster-head, when the member nodes receive an authenticated counter update beacon they would have to reset their own counter value to the one they just received.

Resetting the counter periodically to a new randomly generated value would also difficult counter estimation for an attacker producing a similar effect to that of traditional key renewal techniques. But in our case we are coping with synchronization along with counter updates[2] with the same simple technique.

For security reasons, encryption of the counter update messages is mandatory, this due to the fact that if an intruder knows the current counter value, he may trivially forge packets and deceive the cluster-head into accepting fake data / MAC messages.

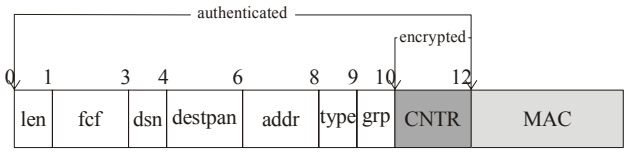

Fig. 2. Counter update packet format

To optimize code memory, for encryption, we are using the same block cipher that we use for the MAC generation algorithm (based on the RC5 cipher). In Fig. 2 we show the packet format of the counter update beacons. As we show in expression 9, the cluster-head broadcasts a message containing the encrypted counter (*CNTR*) and the MAC code computed with the message header and the encrypted *CNTR*. Note that in this case, the MAC code is sent along with the rest of the packet as in the traditional approach.

$$ClusterHead \Rightarrow Nodes : E_{K_{cluster}}(CNTR) \,||\, MAC(K_{cluster}, HDR \,||\, CNTR) \tag{9}$$

This counter synchronization technique also introduces additional resource requirements for our sensor network, but we are foreseen that we would not need to send counter updates very frequently. The counter update beacons frequency depends directly on the length of the sensing period[3] per beacon as well as the hardware platform on which we are implementing it. In the experiments section we will go into more details surrounding counter update frequency for our particular implementation.

[2] In our context, the counter is secret, and its value directly influences the value of the MAC code. Thus, we can see counter update as alternative technique to key renewal.

[3] We define sensing period as the time between two received MAC code packets from the same node.

3.2 Implementation

For our base implementation, we considered a military warehouse monitoring system prototype. The main objective of our application is to provide a controlled environment for the storage of chemical substances and others sensitive materials. Under this deployment scenario, security is a must, because we cannot afford attacks such as an intruder placing rogue nodes that potentially inject forged sensor readings packets into our monitoring network.

As we show in Fig. 3, nodes were placed inside a relatively small "L" shaped building (464 sq. ft.) very close to each other (average distance between neighbor nodes was about 7 ft.), and since the cluster-head (CH) was placed in the middle of the building, the maximum distance between a node and the cluster-head was approximately 20 ft. Thus, single-hop communication between every member node and the cluster-head was possible.

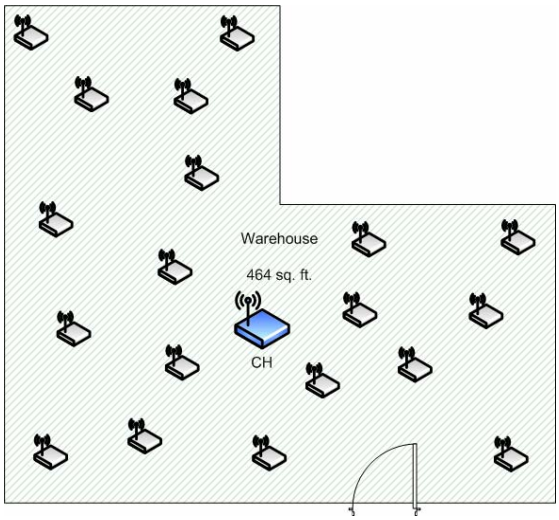

Fig. 3. Node deployment for a military warehouse monitoring application

A crucial aspect in the implementation of our protocol is deciding which cryptographic primitives to use. The AES [Daemen and Rijmen, 2000] (Advanced Encryption Standard) algorithm is commonly used in traditional network security; we have considered the possibility of using it in our protocol as well. However, this algorithm consumes 800 bytes in lookup tables, a requirement that is not feasible for our implementation platform. Other alternatives were also analyzed, such as the DES [NIST, 1999] (Data Encryption Standard) algorithm, but it requires too much memory for storing permutation tables. Finally, we decided to use the RC5 [Rivest, 1994] algorithm because it has many features that makes it attractive for our platform. For instance, it has a very low memory requirement, it is a fast algorithm and it is very flexible in terms of block and key size.

Regarding block and key size, we selected RC5-32/12/16, because according to the literature [Rivest, 1994], it offers a good balance between efficiency and security level. Under this scheme, the cipher would process 32 bit blocks at a time through 12 rounds using a 16-byte cryptographic key.

There are many ways to compute the MAC code for a given message, like HMAC and SHA-1 among others. However, to implement any of these MAC functions we need to spend additional data and code memory. For this reason, as we mentioned earlier the CBC-MAC algorithm was selected for MAC computation, this algorithm uses a conventional block cipher like RC5 iteratively to compute the MAC code. Thus, by selecting this algorithm we use the same block cipher to provide both confidentiality and authentication services, which obviously results in saved memory (data and code).

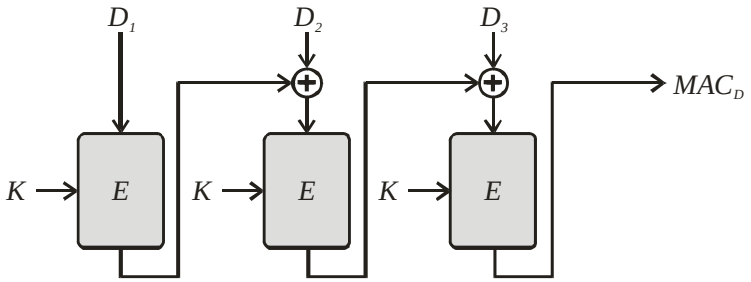

Fig. 4. Operation of the CBC-MAC algorithm [ISO, 1989]

In Fig. 4 we show the operation of the CBC-MAC algorithm, where each data block is processed by the cipher and its output is XORed with the next data block. The output of the last cipher block is the MAC code of the input message *D*.

The protocol was implemented on TMote Sky motes and TinyOS 1.1.15, the implementation was done based on an application that periodically sends readings from all the on-board sensors. Twenty nodes were placed inside the warehouse which dimensions

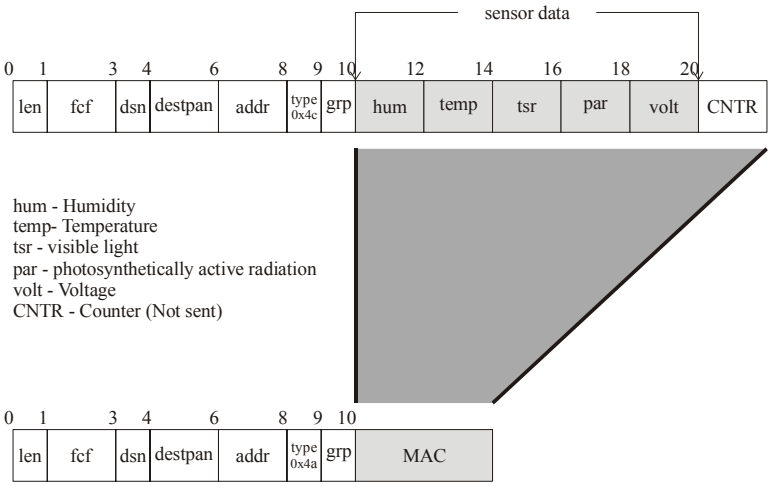

Fig. 5. Packet format for messages containing sensor data and MAC codes

In Fig. 5 we show the packet format for packets *0x4a* and *0x4c* (refer to Table 2), the first 10 bytes represent the header of a TinyOS standard message. Additionally, in our implementation, sensor data contains readings from the humidity, temperature, visible light, par light and voltage sensor aboard the TMote Sky motes. As shown in this figure, the sensor data length is 10 bytes and the MAC code is 4 bytes long, the sensor data as well as the synchronized counter (*CNTR*) are used to compute the MAC code. However, the counter is not sent in the message.

In order to deal with small discrepancies in sensor readings due to sensor calibration we are using discrete intervals to group sensor reading values, each interval is represented by its central value, so when a reading falls into a given interval, the node transmits the MAC code computed with the central value of that particular interval. The drawback of this technique is that we are sacrificing precision but in exchange we are increasing the probability of obtaining redundant data which would obviously benefit power consumption as we will show in the next section. For

instance, the TMote Sky has an on-board SHT11 humidity/temperature sensor from Sensirion[4], this sensor has a 14-bit resolution with an accuracy of ±0.4° C @ 25° C. According to technical specifications, the temperature (in Celsius degrees) can be calculated from the 14-bit digital output (SO_T) through the following equation[5]:

$$T = -39.60 + 0.01 \bullet SO_T \tag{10}$$

From equation 10 we can observe that each digital unit from the 14-bit output only represents a hundred of a Celsius degree. Therefore, if we relax our temperature accuracy requirements[6] we could use our proposed intervals scheme in order to cope with calibration issues and small local variations among nodes in the same cluster. Particularly, in our implementation we considered intervals of 0.32° C, we did so by using simple bit manipulation: masking the first four least significant bits and setting the fifth to one. Thus, we avoided any kind of floating point operation in the mote, which is highly recommended due to their computational cost.

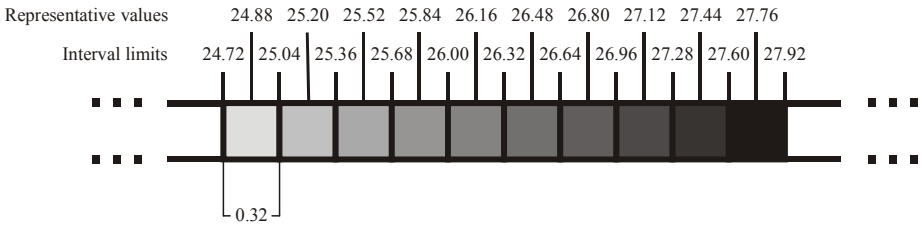

Fig. 6. Temperature intervals used for our implementation (units in Celsius degrees)

In Fig. 6 we show the intervals generated by using the described scheme, using 0.32° C degrees obviously yields to an error of ±0.16° C which is admissible for our application. Readings coming from other sensors were handled similarly; we only varied the interval width according to accuracy requirements for each sensed variable.

4 Experiments and simulations

A series of experiments were conducted, our first goal was to quantify the amount of additional power consumption that the proposed protocol introduces. Using Matlab, we modeled a cluster containing 20 sensor nodes with a 10% duty cycle and a 5 second sensing period transmitting at 0 dBm. Previously, we defined the messages and their respective lengths, thus, we can estimate the amount of traffic involved in the operation of our protocol.

We compared our protocol to the traditional node-to-cluster-head data transmission approach (with no security) where every sensor node transmits their raw sensor reading to the cluster-head on each sensing period.

For our traffic experiments in our 20-node cluster, we noticed that the amount of transmitted packets is always greater in our protocol than in the traditional approach because of the two-phase process as we can see in Fig. 7. However, as we show in Fig. 8, in our approach we transmit a much smaller total number of bytes; this is due to the fact that the packets containing the MAC code are considerably shorter in size as opposed to the packets containing the raw sensor readings. Thus, we incur less total traffic than the traditional approach. In Fig. 7 and Fig. 8 we show plots of number of sent packets and sent bytes respectively vs. number or redundant packets per period for our 20-node cluster experiment. For instance, if 12 redundant packets are generated by the cluster on a certain period, the total amount of traffic is approximately 540 bytes as opposed of 600 bytes of traffic generated by the traditional

[4] Website: http://www.sensirion.com/
[5] When using a 3V source.
[6] Given the fact that we are using a sensor with a ±0.4° C accuracy, we can state that we already relaxed accuracy requirements for our application.

approach. This yields a 10% in bandwidth savings. The saved bandwidth increases even more as the number of redundant packets also increases. It is important to note that our protocol is intended for applications that require densely populated networks (i.e. nodes placed very close to each other). Thus, a high number of redundant packets is expected.

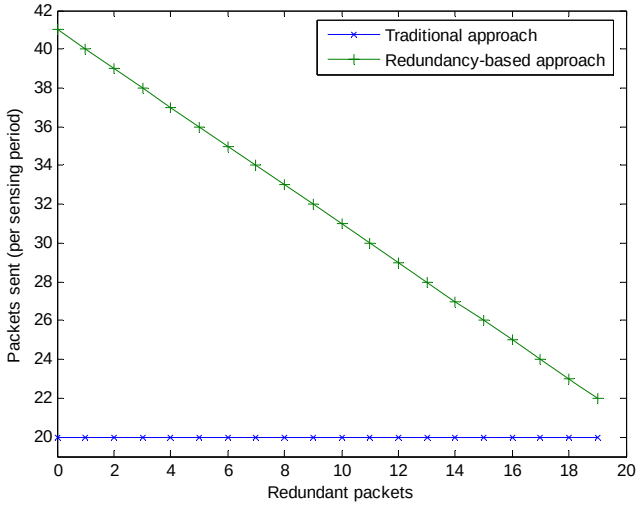

Fig. 7. Plot of total number of sent packets per period vs. redundant number of packets

Also, considering we use TMote Sky motes for our implementation, and knowing that those motes use AA batteries for their operation, we estimated the expected lifetime for each mote considering different levels of data redundancy for our cluster. We introduced the technical specifications of the motes to our simulation, particularly, those specifications related to power consumption for their different operation modes. A pair of AA batteries provide a current of approximately 2.5 Ah, but we also considered that it is impossible to consume all of the available current because at a certain point the provided voltage falls under the operating threshold for the motes; however, it is safe to assume 2200 mAh of battery charge [Mainwaring *et al.*, 2002].

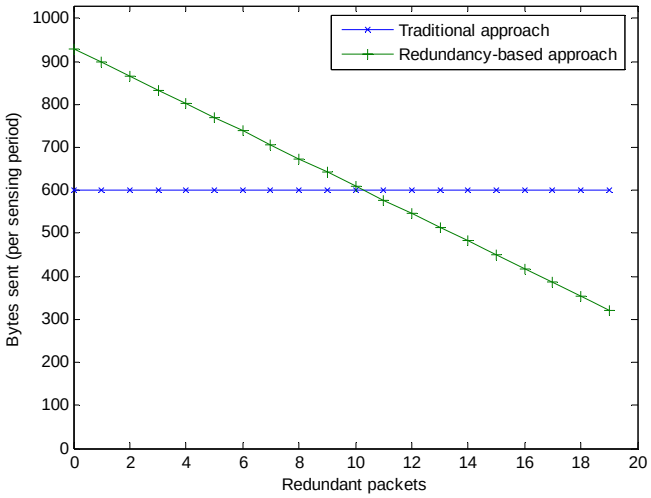

Fig. 8. Plot of generated traffic vs. redundant data

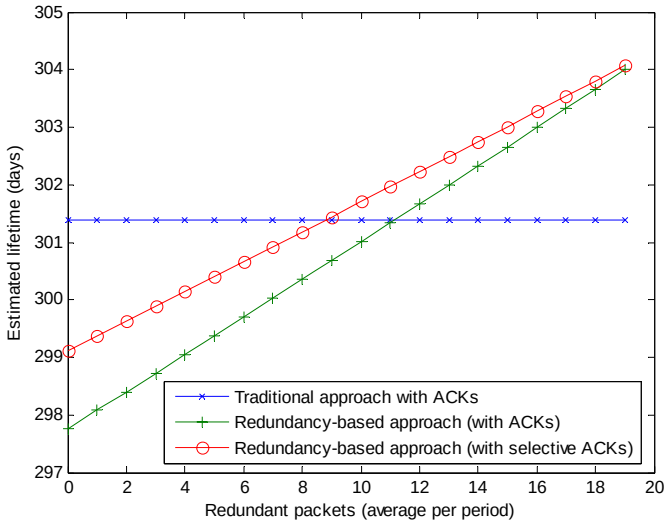

Fig. 9. Plot of cluster nodes estimated lifetime vs. average redundant data packets in the cluster-head using the ACK signals from the MAC layer

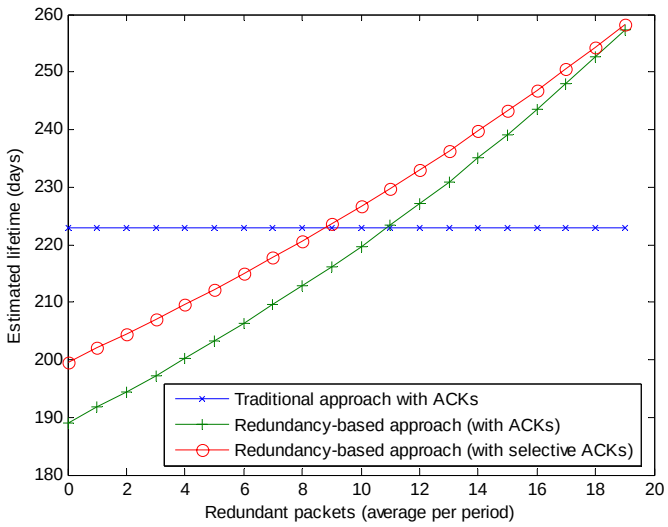

Fig. 10. Plot of cluster-head estimated lifetime vs. average redundant data packets in the cluster-head using the ACK signals from the MAC layer

For our energy consumption experiments we noted that our protocol did not extend the lifetime of the nodes considerably (about 3% lifetime increase at most, refer to Fig. 9) this is mainly because even though we send shorter packets we spend additional energy receiving data requests from the cluster-head, and surprisingly, receiving consumes more energy than transmitting data in this particular platform[7]. However, in Fig. 10 we show that we

[7] According to the TMote Sky datasheet: See http://www.moteiv.com

extended the life of the cluster-head up to 20% depending on the redundancy. In this graph, we show the traditional approach along with our protocol in which we use the ACK messages from the MAC layer, and a third approach in which we only send ACK messages to the nodes where their data is required (called selective ACK approach). By using the latter, we enhance the performance of our protocol for power consumption; this is because we save energy by reducing the amount of ACK messages sent from the cluster-head to the nodes.

Regarding counter updates frequency for synchronization, we considered that TMote Sky motes use a 32 kHz crystal from CITIZEN[8] with a tolerance of ±20 ppm according to technical specifications. Thus, in a worst case scenario this yields to a 1 second drift every 49,153 seconds of operation (≈ one second drift every 13.5 hours at worse). If we are considering a 5 second sensing period, sending a counter update beacon every 24 hours would be a good measure[9]. As shown in Fig. 11 and Fig. 12, the energy cost of inserting one extra packet daily is negligible in either the cluster nodes (Fig. 11) and in the cluster-head (Fig. 12).

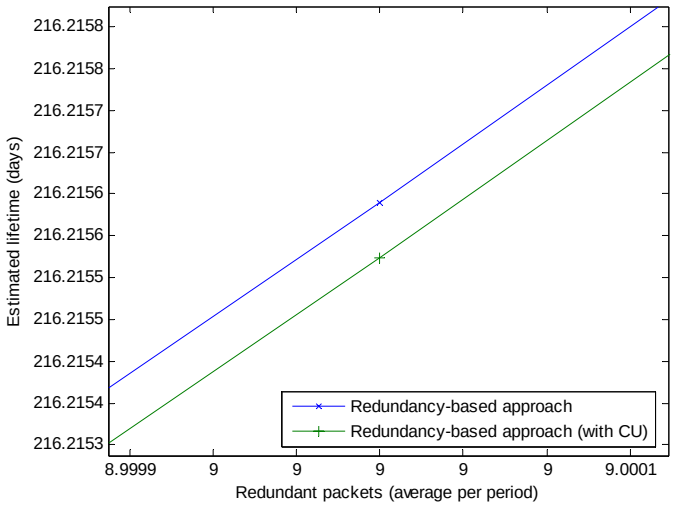

Fig. 11. Effect on the cluster nodes expected lifetime of the added cost of daily counter updates (CU)

[8] Website: http://www.citizencrystal.com
[9] This would imply that we are allowing a maximum drift of about 2 seconds.

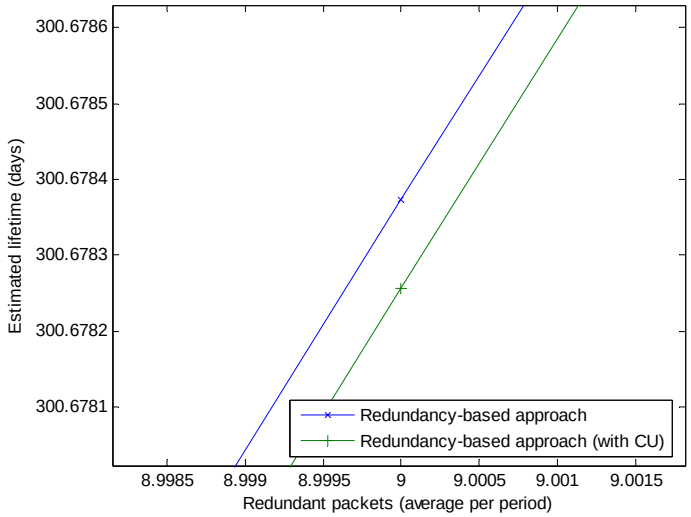

Fig. 12. Effect on the cluster-head expected lifetime of the added cost of daily counter updates (CU)

Concluding Remarks and Future Work

In this paper we showed that by integrating our security oriented node-to-cluster-head protocol to a hierarchical sensor network, we can provide integrity and authentication services without increasing energy consumption requirement. We also showed that with an increased redundancy rate we consume less energy than the traditional approach where no security services are provided. As it was shown, by saving energy we obviously extended network lifetime by a factor of up to 3% in the member nodes and up to 20% in the cluster-head. Thus, the main contribution of this paper is to provide security services to hierarchical cluster-based wireless sensor networks by using the resources saved through our redundancy-driven protocol, specifically, bandwidth and energy.

The integration of a sensor reading estimation algorithm for adding robustness to our protocol is ongoing work, as well as measuring the performance of the proposed protocol on more populated networks.

References

1. **Banerjee, S. and S. Khuller,** "A clustering scheme for hierarchical control in multi-hop wireless networks". In INFOCOM, 2001, pp. 1028-1037.
2. **Bellare, M., R. Canetti and H. Krawczyk**, "Keying hash functions for message Authentication". In CRYPTO, edited by N. Koblitz, Vol. 1109 of Lecture Notes in Computer Science, Springer, 1996, pp. 1-15.
3. **Bellare, M., J. Kilian and P. Rogaway,** "The security of the cipher block chaining message authentication code". Journal of Computer and System Sciences 61 (3) (2000) 362–399.
4. **Black, J. and P. Rogaway**, "CBC MACs for arbitrary-length messages: The three-key constructions". Journal of Cryptology 18 (2) (2005) 111–131.
5. **Çam, H., D. Muthuavinashiappan and P. Nair**, "Energy efficient security protocol for wireless sensor networks". In Proceedings of the IEEE VTC Conference, IEEE Computer Society, 2003, pp. 2981–2984.
6. **Çam, H., S. Özdemir, P. Nair, D. Muthuavinashiappan and H. O. Sanli**, "Energy-efficient secure pattern based data aggregation for wireless sensor networks". Computer Communications 29 (4) (2006) 446–455.
7. **Daemen, J. and V. Rijmen,** "Rijndael for AES". In AES Candidate Conference, 2000, pp. 343–348.

8. **Hill, J., R. Szewczyk, A. Woo, S. Hollar, D. E. Culler and K. S. J. Pister,** "System architecture directions for networked sensors". In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, 2000, pp. 93–104.
9. **IEEE**, "802.15.4 - 2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)", IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2003. URL http://www.ansi.org/
10. **ISO/IEC 9797,** "Data cryptographic techniques - Data integrity mechanism using a cryptographic check function employing a block cipher algorithm". International Standards Organization (ISO), 1989. URL http://www.ansi.org/
11. **Mainwaring, A.M., D. E. Culler, J. Polastre, R. Szewczyk and J. Anderson,** "Wireless sensor networks for habitat monitoring". In WSNA edited by C. S. Raghavendra and K. M. Sivalingam, ACM, 2002, pp. 88–97.
12. **NIST (National Institute of Standards and Technology)**, "FIPS PUB 180-1: Secure Hash Standard", National Institute for Standards and Technology, Gaithersburg, MD, USA, 1995, supersedes FIPS PUB 180 1993 May 11. URL http://www.itl.nist.gov/fipspubs/fip180-1.htm.
13. **NIST (National Institute of Standards and Technology)**, "FIPS PUB 46-3: Data Encryption Standard (DES)". National Institute for Standards and Technology, Gaithersburg, MD, USA, 1999, supersedes FIPS 46-2. URL http://www.itl.nist.gov/fipspubs/fip186-2.pdf
14. **Perrig, A., R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler**, "SPINS: Security protocols for sensor networks". Wireless Networks 8 (5) (2002) 521–534.
15. **Perrig, A. and J. Tygar**, "Secure Broadcast Communication in Wired and Wireless Networks". Kluwer Academic Publishers, 2003.
16. **Rivest, R.L.,** "The RC5 Encryption Algorithm". In Fast Software Encryption edited by B. Preneel, Vol. 1008 of Lecture Notes in Computer Science, Springer, 1994, pp. 86–96.
17. **Schneier, B.**, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 2nd Edition, John Wiley, 1996.

Luis E. Palafox *received his M.S. degree in digital systems from the National Polytechnic Institute in Mexico (IPN) in 2002. He is a Ph.D. Candidate in the Computer Science Program at CICESE and a member of the Faculty of Chemical Science and Engineering at the University of Baja California (UABC) since 1999. His areas of interest are: computer networking, embedded systems, wireless sensor networks and digital signal processing.*

J. Antonio García Macías *obtained his Ph.D. in Computer Science from the Institut National Polytechnique de Grenoble (INPG), France. He is currently a researcher at CICESE, and a member of the National Researchers System (SNI). His current research interests are ubiquitous computing, wireless adhoc and sensor networks, as well as next-generation Internet services and protocols.*