# Cross-Layer Interference Avoidance MAC Protocol for Dense Wireless Sensor Networks

Behnam Dezfouli, Marjan Radi, Mohammad Ali Nematbakhsh
{dezfouli, radi}@ieee.org

***Abstract*—In this paper, we propose IAMAC, an Interference Avoidance MAC protocol to avoid inter-node interference in dense wireless sensor networks. IAMAC interacts with routing protocol via cross-layer information sharing between the MAC and network layer. By providing information from network layer, we enable the MAC protocol to make proper decisions which result in fewer inter-node interference and lower delay. Through interference avoidance, IAMAC reduces energy consumption per node and leads to higher network lifetime compared with S-MAC and Adaptive S-MAC. In addition, IAMAC has lower delay than S-MAC. In our evaluations, we considered IAMAC in conjunction with two error recovery methods (ARQ and Seda). Our simulation results show that our protocol is highly compatible with Seda and this integration achieves higher network throughput and lifetime.**

***Keywords-Sensor Networks; Medium Access Control; IAMAC; Lifetime; Cross-Layer Optimization***

## I. INTRODUCTION

There have been a lot of efforts [1-3] to investigate wireless communication characteristics in wireless sensor networks. These works revealed the irregularity and unreliability of low power wireless links. Accordingly, three distinct reception regions can be identified in a wireless link [1,2]: connected, transitional, and disconnected. In sensor networks, many links fall into transitional region. As far as these links exhibit high variations (due to noise, inter-node interference, etc.), there are many non-perfect links that can be used in route selection. Neither the routing algorithms [3,4] nor the MAC protocol collision avoidance methods [5-8], cannot handle the effects of inter-node interference completely.

Even though link estimation can help to select the best next hop neighbour along the path to the sink, but this is not enough. Concurrent transmissions of neighbor nodes lead to inter-node interference and increase packet corruption rate. This issue results in more energy consumption per node which is in contrast with long lifetime of tiny sensor nodes.

On the other hand, recent studies in designing protocols for sensor networks reveal the fact that reducing power consumption cannot be handled completely in one layer of the protocol stack and without any interaction with other layers [9-11]. For example, to avoid collision in S-MAC [5], all the nodes which overhear control packets (RTS and CTS) are prevented from packet transmission. This mechanism results in very high end-to-end delay of this protocol. To improve the latency of S-MAC, Adaptive S-MAC [6] is proposed and uses adaptive node activation based upon the estimated transmission duration. This method has two main drawbacks: (1) due to the variations in wireless channel, exact time of activation cannot be calculated and many nodes may suffer from idle listening or overhearing, (2) as the network density increases, the number of nodes which will try to adaptively wake up increases and results in higher energy consumption of the network. According to these two disadvantages, Adaptive S-MAC has a very low lifetime that limits its applicability in many applications.

In this paper, we propose an Interference Avoidance MAC protocol (IAMAC) that its main objective is to provide higher network lifetime through avoiding inter-node interference. In addition, we do not compromise delay as S-MAC does and IAMAC provides lower latency than S-MAC. By information sharing between IAMAC and network layer, we make proper decisions in each node to avoid inter-node interference and reducing delay. Reduction in interference, leads to fewer data corruptions, which means fewer active time of each node. The result of this reduction is significant increase in network lifetime in comparison with S-MAC and Adaptive S-MAC.

Monitoring and Surveillance are the main applications of IAMAC. Nodes sample their environment periodically and send their results to the sink node. Primary demands of these applications are long network lifetime and transmission reliability. Usually the delay of several minutes can be tolerated. It should be noted that other applications can also be envisioned for this protocol.

## II. INTERFERENCE AVOIDANCE PROTOCOL

In this section we introduce our cross-layer MAC protocol. Although the routing algorithm is not claimed to be completely new, but it is necessary to be introduced before we proceed to the MAC protocol description.

Proceedings of the International Conference on Networks & Communications (NetCom 2009)
**IEEE Computer Society**
doi:10.1109/NetCoM.2009.44

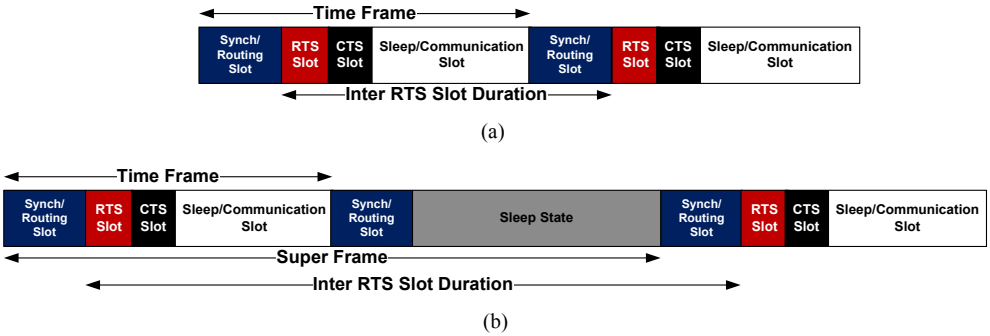


Figure 1. Time Frame and Super Frame structures. (a): If the duration between two consecutive RTS Slots is less than 12 seconds, simple Time Frame structure is used. (b): If the duration between two consecutive RTS Slots is more than 12 seconds, Super Frame structure is applied. In this condition, Time Frame duration must not exceed 12 seconds.

### *A. Routing Protocol*

We use spanning tree optimization as the routing algorithm. In the first step, each node broadcasts a fixed number of control packets and records the number of successfully received packets from its neighbors. After this step, a preliminary neighbor table containing link qualities is formed. Each pair of nodes has a link cost corresponding to forward and backward packet reception rates. We use ETX [4] as our link cost function. In the second step, sink node sets its cost to zero and broadcasts this cost to its neighbors. This broadcast is performed by transmitting Synch/Routing packet. A lightweight time synchronization protocol [12] is also used to synchronize sleep and wake schedules among the nodes. Thereby, a Synch/Routing packet contains time synchronization data along with the current cost of each node to the sink. Upon receiving such a broadcast packet, each node adds the received cost to the link cost of the node from which this packet has been received. For example, consider that node A receives a Synch/Routing packet from node B. Then it adds the cost contained in Synch/Routing packet to the cost of link A-B. If the resulting cost is less than the current cost of A to the sink, B is selected as A's parent. Once the network reaches to a stable condition, each node follows its sleep/wake schedule. Broadcast interval of Synch/Routing packets during normal operation of the network depends on the time synchronization accuracy and the route change frequency. If the route cost of one node has been changed, it must be broadcasted to notify its neighbors.

### *B. MAC Protocol*

IAMAC is a sleep/wake MAC protocol. Fig.1 shows the structure of Time Frame in IAMAC. In order to separate synchronization from Time Frame duration, we proposed Time Frame and Super Frame structures. For applications in which network lifetime and delay are equally important, we use simple Time Frames, and in applications in which lifetime is critical, Super Frame structure can be used.

The first slot is devoted to transmission and reception of the Synch/Routing packets (described in the previous section). The main interference avoidance algorithm runs in the RTS Slot. Fig. 2 shows its pseudo code. We have provided enough comments in the algorithm to be self-explanatory, but some more comments are worth mentioning here.

If a node receives a RTS packet that is destined for it, this RTS packet is added to the received RTSs queue. In this state, if the RTS transmission from this node is scheduled, it must be canceled. It should be noted that this node can still receive more RTS packets. By applying such a mechanism, a priority scheme can be used: a node with higher priority (e.g. longer packet queue), schedules its RTS packet transmission earlier.

If the received RTSs queue is not empty and no RTS packet is overheard by this node, except the ones destined for this node's parent, this node can change its role. It means that if its data queue is not empty, it will act as a sender and reschedules its RTS packet transmission. This results in sequential data transmissions from multiple nodes to a common parent in one Sleep/Communication Slot.

The proposed algorithm for the RTS Slot is not a complete solution to prevent inter-node interference. Therefore, we propose a complementary algorithm to be run in the CTS Slot. Fig. 3 provides its pseudo code. When the RTS packets cannot be heard by neighboring nodes, while CTS packets can be heard, this algorithm prevents inter-node interference.

By applying a priority scheme in choosing transmission time of the CTS packet, nodes with higher priority will have higher chance to send their data packets. Inserting transmission priority in the RTS packet, informs the receiver to choose an appropriate reply time according to the provided priority.

Figure 2. RTS Slot algorithm.

```
1. ParentAddress denotes the address of this node's parent;
2. MyAdress denotes the address of this node;
3. CancelRTSTrans=FALSE; //a variable
4. CancelCTSTrans=FALSE; //a variable

5. /*if this node has some data packets to be sent*/
6. If (PacketQueue.Length!=0)
7.       Choose a random time for RTS transmission;

8. While (we are in RTS Slot){
9.   If (new packet arrives)
```

```
10.    Pkt=Arrived Packet; // an RTS packet arrived;
11. /*transmit RTS packet; set the appropriate variable*/
12. If ( (RTS timer is reached) && (channel is empty) ){
13.    Send RTS to Parent;
14.    CancelCTSTrans=TRUE; }

15. /*if the channel was not empty*/
16. If ( (RTS timer is reached) && (channel is not empty) ) {
17.    /*an RTS is received, if this packet is destined for this node's
       parent, reschedule RTS transmission*/
18.    If (Pkt.RecAdress==ParentAddress)
19.       Choose a random time for RTS transmission;
20.    Else Deactivate=TRUE; }

21. /*if no previous RTS is transmitted from this node*/
22. If ( (A RTS packet arrived) && (CancelCTSTrans==FALSE) ) {
23.    /*if the received RTS packet is not destined for this node and
       this node's parent*/
24.    If ( (Pkt.RecAddrees!=MyAdress) &&
         (Pkt.RecAdress!=ParentAddress) ) {
25.       Clear received RTSs queue;
26.       Cancel RTS transmission;
27.       CancelRTSTrans=TRUE;
28.       /* cancel CTS transmission from this node*/
29.       CancelCTSTrans=TRUE;
30.       /*This node cannot be the part of any communication in this
          Time Frame; deactivate the node (sleep until the next Time
          Frame)*/
31.    Deactivate=TRUE; }

32. /*if the RTS packet is destined for this node*/
33. Else if (Pkt.RecAddrees==MyAdress) {
34.    Add Pkt to the received RTSs queue;
35.    Cancel RTS transmission from this node;
36.    CancelRTSTrans=TRUE;  }

37. /*if this node receives an RTS packet destined for its parent and it
       has not overheard any RTS*/
38. Else if (Pkt.RecAdress==ParentAddress) &&
       (CancelRTSTrans==FALSE) {
39.       /*this node can be a sender*/
40.       /*cancel CTS transmission*/
41.       CancelCTSTrans=TRUE;
42.       If (PacketQueue.Length==0)
43.       Deactivate=TRUE; }

44. /*if this node has received an RTS packet that is destined for its
       parent and it has received another RTS packet destined for it, this
       node can be a sender*/
45. Else if ( (Pkt.RecAdress==ParentAddress) &&
       (CancelRTSTrans==TRUE) ) {
46.       /*do not respond to previously received RTSs (clear the
          received RTSs queue)*/
47.       Clear received RTSs queue;
48.       CancelRTSTrans=FALSE;
49.       CancelCTSTrans=TRUE;
50.       /*if this node has something to send, reschedule RTS
          transmission*/
51.       If (PacketQueue.Length!=0)
52.       Choose a random time for RTS transmission;
53.       Else Deactivate=TRUE; }
54. }}
```

Figure 3. CTS Slot algorithm.

```
1.  /*cancelRTSTrans and CancelCTSTrans variables are defined in
       RTS Slot algorithm*/
2.  /*if received RTSs queue is not empty and this node is allowed to
       send CTS packet*/
3.  If ( (RTSQueue.Length!=0) && (CancelCTSTrans==FALSE))
4.     Choose a random time for CTS transmission;
5.  While (we are in CTS Slot) {
6.   If (new packet arrives)
7.     Pkt=Arrived Packet; // a CTS packet arrived
8.   /*if it is time to send CTS packet*/
9.   If ( (CTS timer is reached) && (channel is empty) )
10.    Send CTS packet;

11. /*overhearing a CTS packet, cancel CTS transmission*/
12. If (Pkt.RecAddress!=MyAddress) {
13.    Cancel CTS transmission timer;
14.    /*due to link asymmetry, RTS packet may not have been
          received correctly, by overhearing a CTS packet which is not
          destined for this node, we deactivate this node*/
15.    Deactivate=TRUE; }
16. If (Pkt.RecAddress==MyAddress){
17. /*if this node receives a CTS packet, it is allowed to transfer its
       data in Sleep/Communication Slot*/
18. } }
```

### C. Durations and Access Methods

The RTS Slot is divided into smaller contention slots which are named RTS Contention Slot. Each node selects a random RTS Contention Slot to transmit its RTS packet. In the rest of the RTS Contention Slots, the node listens to the channel to receive probable RTS packets. When a node arrives at its randomly selected RTS Contention Slot, a small random time is selected and the node continues listening to the channel. If nothing is sensed during this time, it can send its RTS packet. Otherwise, if the node receives a RTS packet destined for its parent, it selects another RTS Contention Slot among the remaining RTS Contention Slots and repeats these steps. If the received RTS packet is not destined for this node or its parent, the node becomes inactive.

The required number of RTS transmissions in each RTS Slot is the number of nodes which can transmit their data packets sequentially at one Time Frame. This is dependent on the maximum number of the nodes with a common parent. If the children of a node compete to grasp the channel and their RTS transmissions collide at the parent, they will suffer more delay, since they cannot transmit their data packets at the same Time Frame. This condition happens when the children cannot hear each other's transmissions. Considering $n$ nodes with this condition, and with a common parent, the probability of correct reception of RTS packets from these nodes (i.e. no collision) is as follows ($w$ is the number of RTS Contention Slots):

$$p_{0=}\binom{w}{n}.n!\left(\frac{1}{w}\right)^{n} \qquad (1)$$

Accordingly, the RTS Slot duration depends on the number of children per node. Since the RTS Slot duration should be equal for all the nodes in the network, scalability problems may be arisen. In order to remedy this problem, we can limit the maximum number of children per node. It means that, when a node decides to select its parent, it also considers the current number of children whose neighboring nodes currently have. Each node looks for a qualified node

(in terms of cost, and number of children) and then selects its parent.

When the sender nodes can hear each other while the receivers cannot, random listening time at the start of the selected RTS Contention Slot plays an important role. In this situation, if two or more sender nodes select the same RTS Contention Slot and receivers receive their corresponding RTS packets, concurrent transmission of sender nodes may be resulted in inter-node interference. Selection of random listening time at the start of the RTS Contention Slot solves this problem.

In the CTS Slot, when two or more nodes send their CTS packets simultaneously, no one can be aware of the other's transmission. This can be resulted in severe interference at the receiver. Therefore, it is essential to avoid concurrent transmission of CTS packets. Due to fewer contentions in transmitting CTS packets, we can consider a CTS Slot that is not divided into any more contention slots. Each node chooses a random time in the CTS Slot to transmit its CTS packet. When the CTS Slot starts, the node begins listening to the channel until the randomly selected time reaches. During this time, any overhearing of CTS packet prohibits this node from CTS transmission. In this way, the probability that two nodes concurrently transmit their CTS packets is so small, because the signal propagation delay in wireless sensor networks is insignificant. According to these points and our simulation results, CTS Slot duration can be equivalent to the duration of transmitting three CTS packets.

## III. EVALUATION

The simulation application is programmed in OMNeT++ framework. Table 1 represents our general simulation settings (similar to the characteristics of MICA2 motes). In evaluating our proposed protocol, we may change some of these parameters.

### A. *Interferer Nodes per Time Frame*

In this section, we evaluate the proposed protocol in the context of interference avoidance. In order to measure the interference avoidance level, we define $CS_{Ni}$ as the colliding set of node $N_i$. $CS_{Ni}$ is the number of nodes in the neighborhood of node $N_i$ which send their data packets concurrently with $N_i$ reception in the same Time Frame. It should be noted that $CS_{Ni}$ excludes the node that is currently sending to $N_i$. When a node is receiving data packets while its $CS_{Ni}$ is not zero, inter-node interference is possible.

TABLE 1. DEFAULT SIMULATION SETTINGS

| **Radio** | | | |
|---|---|---|---|
| Modulation | FSK | Encoding | NRZ |
| Output Power | 0 dBm | Frame | 45 bytes |
| **Transmission Medium** | | | |
| Path Loss Exponent | 4 | $PL_{D\,0}$ | 55 dBm |
| Noise Floor | -105 dBm | $D_0$ | 1 m |
| **Other Parameters** | | | |
| Number of Nodes | 200 | Area | 100×100 $m^2$ |

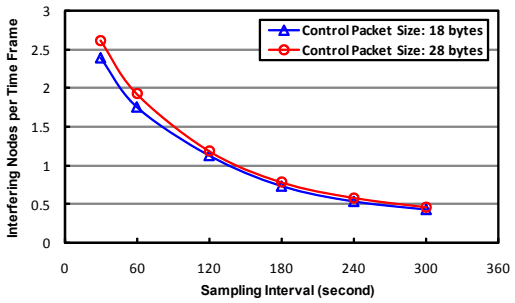


Figure 4. Summation of each node's $CS_{Ni}$ in each Time Frame. As the sampling interval increases, contention for packet transmission reduces. Therefore, the number of interfering nodes per Time Frame reduces. (Time Frame duration=1 sec.)

Therefore, we sum $CS_{Ni}$ of each node over the entire network in each Time Frame. Fig. 4 depicts this sum at two sizes of control packet (18 and 28 bytes, except the physical and MAC headers). The only occasion in which $CS_{Ni}$ is not zero is when RTS or CTS packets are not received correctly. This situation can happen due to unreliable wireless communication and packet corruption. According to Fig. 4, the protocol has been able to avoid inter-node interference in the network. The average of each node's $CS_{Ni}$ throughout a network composed of 200 nodes is about 2.5 nodes per Time Frame, for sampling interval of 30 seconds. As the sampling interval increases, the number of interfering nodes per Time Frame decreases. This is due to the lower contention for packet transmission.

Although we cannot expect the protocol to eliminate inter-node interference completely (due to unreliable wireless links and control packet corruption), these interferences have no severe effect on the packet reception rate. Simulation results have demonstrated that for transitional region radius of 20 meters and control packet size of 18 bytes, about 70% of the interferer nodes reside 18 meters away from the receiver and about 97% of them reside 16 meters away from the receiver.

### B. *Throughput*

One of the effective factors on the network throughput is the number of concurrently transmitting nodes in each Time Frame. As the radius of the transitional region grows, IAMAC prevents more nodes from transmission and it reduces the number of nodes concurrently transmitting in each Time Frame. The same effect happens by increasing the output power level. Fig. 5 shows the average number of sender nodes in each Time Frame. Starting from 0 dBm, as the output power level reduces, the number of sender nodes per Time Frame increases. This is caused by less interference among the contending nodes. For each network density, this increment stops at a certain output power level. When the output power level goes below this threshold level, the number of nodes with a common parent decreases. Therefore, the number of sender nodes in each Time Frame reduces. The optimal output power level is inherently a cross-layer parameter which mainly depends on the network

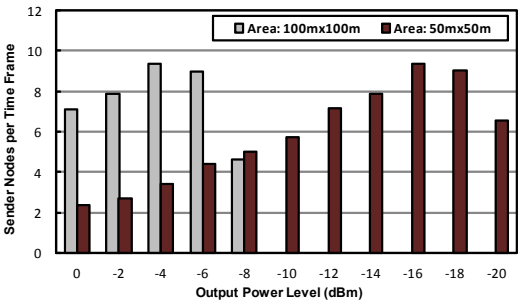


Figure 5. Average number of sender nodes in each Time Frame. Each network density corresponds to an optimal output power level which trades off between the radio interference level and the number of nodes with common parent. (Time Frame duration=1 sec, Sampling interval=60 sec.)

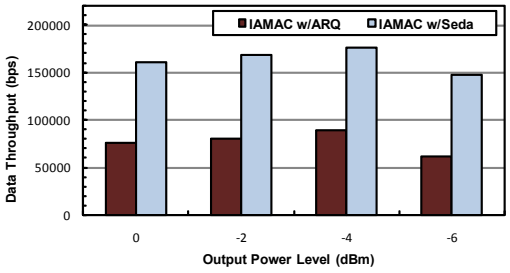


Figure 6. Effect of output power level on network throughput. (Time Frame duration=1 sec, Sampling interval=1.1 sec.)

density, output power level, routing protocol, and sampling rate. Notice that the output power levels less than -8 dBm caused the 100×100 $m^2$ network to be disjointed and therefore there is no result for this condition.

In order to measure the maximum network throughput, we forced each node to sample the environment as fast as it can and transmit its data packets with maximum capacity. We used two error recovery methods (ARQ and Seda [13]) in conjunction with IAMAC. Fig. 6 shows the results. As it can be observed, combination of IAMAC with Seda achieves higher throughput. This is due to less packet corruption and more efficient error recovery of Seda. In Fig. 6, notice the rise and fall of the network throughput that is similar to Fig. 5.

### C. *Lifetime*

In this section, we evaluate energy consumption of IAMAC. Exact energy consumption of each operation (radio operations and sampling) is provided in [8].

Fig. 7 demonstrates the lifetime of IAMAC against S-MAC and Adaptive S-MAC. It is evident that IAMAC provides higher lifetime than Adaptive S-MAC. Lower lifetime of Adaptive S-MAC is due to its adaptive listening mechanism, which is provided to reduce delay. This issue restricts its applicability in many scenarios.

With equal Time Frame durations, IAMAC provides lower lifetime compared with S-MAC. Although this is true for equal frame durations, but generally IAMAC has higher performance than S-MAC in terms of lifetime and delay. This will be discussed later.

As the sampling interval increases, the number of generated packets in each node reduces. This results in lower active time (i.e. lower duty cycle) and higher lifetime of the network. Notice that by increasing the sampling interval, we cannot increase the lifetime indefinitely, since: (1) each node has a limited initial energy (we have considered a 2400 mAh battery per node), (2) synchronization overhead limits the maximum network lifetime, (3) by increasing the Time Frame duration, the number of queued packets in each node is also increased and this results in shorter sleep time.

An interesting behavior in Fig. 7 is the slight rise and fall of the IAMAC's lifetime around a particular sampling interval. At this point, a trade off is established between the node active time, number of serial transmissions per Time Frame, and the number of deactivated nodes. This condition results in maximum lifetime. At this point, in addition to the large number of transmissions per Time Frame, many nodes are deactivated by control packet overhearing.

According to Fig. 7, Seda can improve the lifetime of IAMAC and this improvement is more evident with low sampling intervals and longer Time Frame durations. When the number of data packets to be transmitted in each Time Frame is high, Seda can benefit from its low packet corruption rate and its efficient error recovery.

### D. *Latency*

In Fig. 8, the latency of IAMAC is evaluated and compared with S-MAC and Adaptive S-MAC.

The interference avoidance capability of IAMAC comes at a cost. When IAMAC senses probable inter-node interference and prevents some nodes from data transmission, data packets experience a delay equal to the Time Frame duration. With low sampling intervals, there will be a high contention between neighboring nodes. Therefore, many nodes are prohibited from communication and delay increases. When the sampling interval increases, channel contention is decreased and this results in lower delay. In addition, periodic sleep and wake schedules avoid the nodes from being active all the times. Therefore, each node must wait to arrive at a new Time Frame to start its contention for packet transmission.

Comparing to S-MAC, IAMAC provides smaller delay. Multiple transmissions to a common parent in each Time Frame, and lower contention period of IAMAC, caused its lower delay. In addition, as we have mentioned before, IAMAC completely surpasses S-MAC in terms of lifetime and delay. For example consider IAMAC (10 sec) versus S-MAC (5 sec) in Fig. 7. It can be seen that IAMAC provides higher lifetime than S-MAC. Also, according to Fig. 8, IAMAC (10 sec) has lower delay than S-MAC (5 sec). Therefore, IAMAC provides higher lifetime and lower delay compared with S-MAC.

Even though Adaptive S-MAC demonstrates lower delay than IAMAC (Fig. 8), but according to Fig. 7, Adaptive S-MAC has a very low lifetime.

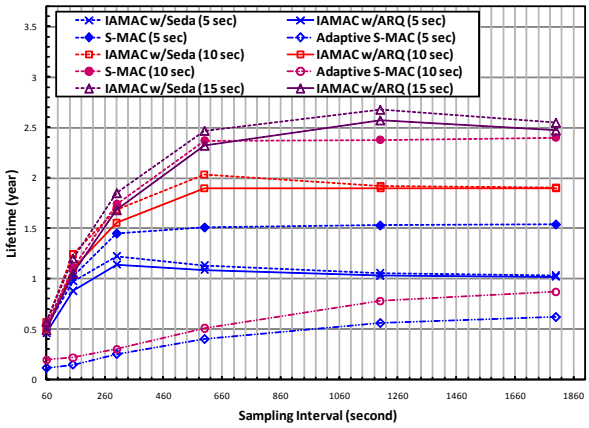


Figure 7: Lifetime of IAMAC vs S-MAC and Adaptive S-MAC. According to this figure and Fig. 8, IAMAC has higher performance than S-MAC in terms of lifetime and delay. In addition, Adaptive S-MAC has a very low lifetime that limits its applicability in many situations.

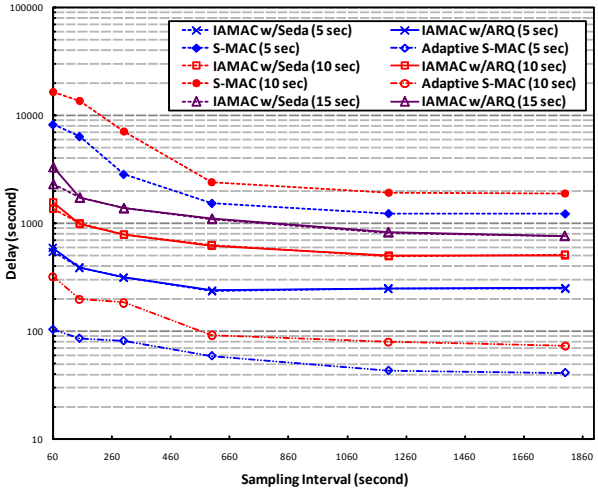


Figure 8: End-to-end delay of IAMAC vs S-MAC and Adaptive S-MAC. It is evident that IAMAC provides much lower delay than S-MAC. Also, in spite of the lower delay of Adaptive S-MAC compared with IAMAC, but Adaptive S-MAC highly trades off lifetime for lower delay.

## IV. Conclusion

In this paper, we proposed IAMAC, an Interference Avoidance MAC protocol to avoid inter-node interference and increasing network lifetime in WSNs. By conducting extensive simulations, we evaluated IAMAC in real network conditions. According to the results, IAMAC has higher lifetime compared with S-MAC and Adaptive S-MAC, while it has lower delay than S-MAC. Therefore, in lifetime critical applications, IAMAC is a good solution. In addition, unlike S-MAC and Adaptive S-MAC, IAMAC separates synchronization from Time Frame duration. This results in higher flexibility of IAMAC and allows the user to trade off between delay and lifetime, depending on the application. Furthermore, since IAMAC tries to reduce packet corruptions due to inter-node interference, ARQ and Seda are considered as its error recovery mechanisms and higher performance of Seda is demonstrated.

## References


[1] M.Z. Zamalloa and B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links," ACM Trans. Sen. Netw., vol. 3, 2007, p. 7.

[2] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA: ACM, 2003, pp. 14-27.

[3] Q. Cao, T. He, L. Fang, T. Abdelzaher, J. Stankovic, and S. Son, "Efficiency centric communication model for wireless sensor networks," Infocom 2006. 25th IEEE International Conference on Computer Communications. Proceedings, 2006, pp. 1-12.

[4] D.S.J.D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," Wirel. Netw., vol. 11, 2005, pp. 419-434.

[5] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," Infocom 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2002, pp. 1567-1576 vol.3.

[6] T.V. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA: ACM, 2003, pp. 171-180.

[7] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," IEEE/ACM Trans. Netw., vol. 12, 2004, pp. 493-506.

[8] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA: ACM, 2004, pp. 95-107.

[9] I. Akyildiz, M. Vuran, and O. Akan, "A cross-layer protocol for wireless sensor networks," 40th Annual Conference on Information Sciences and Systems, 2006, pp. 1102-1107.

[10] M. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," Infocom 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, 2004, pp. 1740-1750 vol.3.

[11] V. Srivastava and M. Motani, "Cross-layer design: A survey and the road ahead," Communications Magazine, IEEE, vol. 43, 2005, pp. 119, 112.

[12] J.V. Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, San Diego, CA, USA: ACM, 2003, pp. 11-19.

[13] R.K. Ganti, P. Jayachandran, H. Luo, and T.F. Abdelzaher, "Datalink streaming in wireless sensor networks," Proceedings of the 4th international conference on Embedded networked sensor systems, Boulder, Colorado, USA: ACM, 2006, pp. 209-222.