# Pushing the Throughput Limit of Low-Complexity Wireless Embedded Sensing Systems

Vahid Salmani and Pai H. Chou
*Center for Embedded Computer Systems*
*University of California, Irvine*

***Abstract*—To maximize the communication throughput for wireless sensing systems, designers have attempted various combinations of protocol design and manual code optimization. Although the theoretical bandwidth limit is easy to determine loosely, there have been no systematic ways to arrive at a tight upper-bound. One contribution of this paper is a formula for deriving a tight upper-bound on the throughput of low-complexity wireless interfaces transmitting packets of a fixed size. It takes into account not only the software execution times on the nodes but also other communication protocols that must be bridged by the base station. The proposed upper-bound, which we believe is the tightest, represents the maximum amount of bandwidth utilization that can be achieved in practice. It can also serve as a means of comparing protocols built on different platforms. Another contribution is a streamlined schedule-based protocol, called RIPE-MAC, which achieves at least 83% of the upper-bound, significantly higher than previously achieved throughput. The proposed protocol needs no clock synchronization and incurs no further complexity on sensor nodes. In the proposed protocol, synchronization and schedule updates are reduced to a single pull message.**

***Keywords*—*wireless sensor networks, medium access control protocols, communication scheduling, high-data-rate monitoring, analytical upper-bound on throughput, time division multiple access***

## I. Introduction

A class of sensor network applications to which more attention has been recently paid is high-data-rate monitoring. It involves relatively high data rates and precise timing of the captured signals. High-data-rate applications continuously require turn-wise and periodic transmission of data from sensors in real-time. They also demand large data size and a relatively high duty cycle. In this class of applications, what matters is high fidelity of data, aggressive sampling and collection, and short latency; low power consumption, while interesting, is of secondary concern.

Some high-data-rate applications may reach sensing rate of $10^2$ to $10^5$ Hz and consume up to 10 or 100 Mbps bandwidth. Therefore, they require five to ten times improvement on the channel utilization of existing sensor networks [1]. In some applications, the system is capable of acquiring more data than can be delivered to the base station, so they focus on reliable collection of high-resolution signals and maximizing the value of the collected data, subject to resource constraints [2]. However, it should not prevent system designers from maximizing the bandwidth utilization.

On the other hand, sensor networks are exposed to several technical limitations including memory and energy constraints, available processing power, transmission rate, synchronization difficulties and robustness in operation. Many of these technical limitations can be overcome by an optimized design of sensor network protocols and operations.

In order to design efficient protocols for wireless sensor networks, it is important to recognize the parameters that are relevant to the applications [3] because the design of a given protocol or algorithm has a deep dependence on the nature of the application [4]. Continuous sensing, periodic data transmission, scalability, fault tolerance and reliability, and energy efficiency are some design considerations that are common among typical high-data-rate applications.

Before designing or optimizing a communication protocol, it is important to measure the throughput limit of the system because it can be helpful in tuning the protocol as the amount of possible improvement is known. The throughput upper-bound can also serve as a means of comparing protocols built on different platforms.

This paper presents a tight upper-bound on throughput of single-hop star-topology wireless embedded sensing platforms and a MAC protocol for approaching it. The idea is to keep the base station always busy by thoroughly eliminating idle listening. In addition, control packets are reduced to a single pull message thereby devoting most of the bandwidth to upstream data packets from sensor nodes.

To make the ideas more concrete, we studied a real-world wireless sensing platform called Eco [5] whose details are presented in Section V. To evaluate our work, the proposed approach as well as some other protocols are implemented on the Eco platform and compared. The reason for choosing Eco is its simplicity and low cost compared to the platforms using ZigBee, Z-Wave or Bluetooth. Moreover, the application-centric work has shown that simple protocols have worked and sufficed in practice [4].

We concentrate on single-hop star-topology networks because they achieve much higher throughputs compared to multi-hop networks. For instance, Flush [6] and Straw [7] achieve only 8 Kbps and 3.5 Kbps for a reliable transfer

over multiple hops, respectively. These gains are less than 1% of the available bandwidth of the state-of-the-art 1 Mbps transceivers. Moreover, multi-hop protocols do not guarantee real-time operation and their delay may prove unacceptable for some applications.

The rest of the paper is organized as follows. In Section II, we review some MAC protocols designed for high-data-rate applications. Section III is dedicated to the proposed upper-bound and a parameterized system configuration for approaching it. The implementation of the RIPE-MAC protocol on Eco is discussed in Section IV. In Section V, we study the performance of our proposed protocol and compare it with that of the protocols reviewed in related works. Finally, the last Section concludes the paper.

## II. Related Work

A communication protocol must be designed in such a way that all nodes utilize the available bandwidth effectively. Existing solutions are often divided into schedule-based (contention-free), contention-based, and hybrid schemes.

Contention-based schemes resort to some type of random access (unscheduled) mechanism. As assumed by the popular IEEE 802.15.4, *Carrier Sense Multiple Access* (CSMA) is the most widely used contention-based protocol, which is popular due to its simplicity, flexibility and lack of the need for clock synchronization and global topology information. Nodes can dynamically join or leave without extra operations. However, the chief disadvantage of CSMA is *collision*, which causes energy waste. The CSMA protocol is efficient when the utilization is low, but the probability of collision increases rapidly as utilization increases [8]. Therefore, CSMA might not be a good option for high-data-rate applications. In general, contention-based MACs behave quite poorly under high query rate conditions and are not acceptable in real-time or time-sensitive applications.

On the other hand, schedule-based approaches synchronize nodes in order to align their active or sleeping periods. *Time-Division Multiple Access* (TDMA) divides time into slots. Each sensor node only transmits during its own time-slots. This approach greatly reduces idle-listening time, but the required time synchronization introduces extra overhead and complexity. TDMA is a better alternative for high data rates than CSMA.

Hybrid schemes attempt to combine the best features of both approaches while offsetting their weaknesses. They try to adapt to different bandwidth conditions depending on demand. A category of hybrid protocols such as GMAC [9] divides nodes into subsets and assigns to each subset its own contention period. The *scheduled contention* scheme is no longer collision-free and thus it is not suited for high data rates as the probability of collision is high. Because node belonging to the same have to contend for the medium in their common slot as in CSMA.

On the other hand, some hybrid schemes such as R-MAC [10] alternate between a contention period and a scheduled period. They perform control operations such as time-slot assignment during the contention period, and the schedule-based period is used for collision-free data transmission. This hybrid scheme is more energy efficient under high rates than the scheduled contention scheme and thus it is a better option for high-data-rate monitoring applications.

The User Configured TDMA (UC-TDMA) [11] is a hybrid MAC protocol of a WSN designed for machinery condition-based maintenance in small machinery spaces. It combines scheduling and contention with deterministic slot allocation to overcome clock inaccuracies. Its modified RTS-CTS scheme uses a virtual-RTS signal generated by the base station on behalf of the node that is scheduled to transmit data.

The above work, however, has the following drawbacks that prevent it from being a general approach. First, the transmission times are ignored in UC-TDMA, possibly because they are negligible compared to the frame length. Secondly, UC-TDMA has to periodically update the schedule for all nodes one-by-one to avoid the clock drift between nodes. The essence of control packets, together with the RTS-CTS mechanism, could incur considerable overhead on the protocol. Although UC-TDMA is designed for a star-topology WSN, it seems to incur high overhead in terms of control packets. Instead, low-complexity wireless sensor networks such as Eco require much simpler and more efficient solutions.

Based on the above discussion, we believe that TDMA plays an important role in designing high-data-rate wireless sensing applications. Therefore, an optimized TDMA scheme, as an integral part of MAC protocols, can significantly improve performance of such systems. Nevertheless, schedule-based schemes require synchronization, which may impose high overhead on the protocol. Moreover, since sensor nodes do not usually have very accurate clocks, precise time synchronization is hard to achieve. For instance, a coordinator must broadcast SYNC packets frequently to minimize clock drifts and maintain tight synchronization, which may result in unacceptable overhead.

Another downside with tight clock synchronization is the added complexity to the protocol stack, which results in larger firmware footprint of nodes. Reference [12] reports that a pure TDMA protocol designed for a single-hop star-topology network requires 18-21 KB of code. This is five times the size of Eco's total EEPROM capacity. In the following, we review two *lightweight* schedule-based protocols designed for low-complexity WSN platforms to address high-data-rate applications and state essential characteristics and drawbacks of each.

Starting with the simplest, EcoDAQ [13] is a collision-free protocol which follows the pulling style. The following four steps are performed in this protocol as illustrated in Fig.

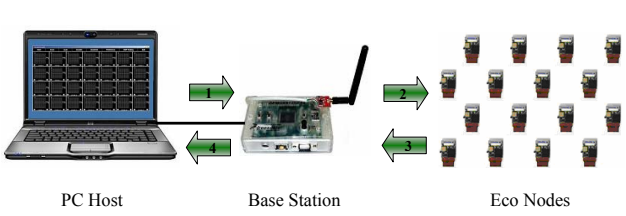


Figure 1. Steps of the EcoDAQ Protocol

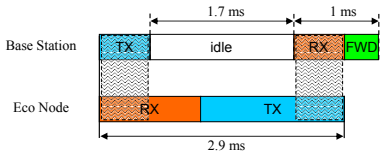


Figure 2. Communication timing cycle of the EcoDAQ protocol. Throughout the paper, the hatched areas in the timing diagrams represent the actual transmission times.

1. The PC host issues a command packet to the base station, containing the ID of the node whose data is desired. The base station broadcasts the command packet to all sensor nodes. The sensor node whose ID is equal to the ID field of the command packet will respond to the packet by transmitting its sensing data back to the base station. Finally, the base station forwards the response packet it receives from the sensor node to the host. Based on the collected data from sensors, the host can decide which nodes must be assigned more time-slots.

The pulling or receiver-initiated style has the following advantages: simplicity on the node side, eliminating the need for time synchronization, and achieving the equivalent of NACK messages by re-pulling lost packets. However, it suffers from some disadvantages. Fig. 2 depicts the communication timing cycle of EcoDAQ in which the base station spends about 50% of the cycle on idle listening prior to receiving a reply from a sensor node. This is an important disadvantage because it tends to waste the wireless bandwidth, thereby hindering the throughput. Another downside with EcoDAQ is the overhead of the pulling message, which considerably holds back the throughput. This overhead may be amortized by increasing the number of reply messages in response to each pull [13]. As shown in Fig. 3, upon receipt of the command packet, the corresponding sensor node transmits a predefined number of sensing data packets back to the base station one after another. Although this modification considerably increases the throughput due to the reduced idle-listening, it has its own disadvantages. Sending multiple replies incurs a significant latency, which may prove unacceptable for some applications. Besides, it does not thoroughly eliminate the idle-listening on the base station side.

Sensemble [14] is designed for processing multi-point

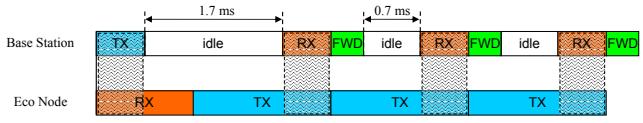


Figure 3. Timing diagram of EcoDAQ with 3 reply packets

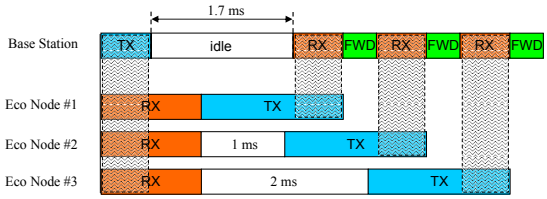


Figure 4. Timing diagram of Sensemble implemented on the Eco platform

human motion with low latency and high resolution. The communication protocol of the Sensemble platform works in a TDMA style as depicted in Fig. 4. For brevity, we use Sensemble to refer to the MAC protocol of the Sensemble platform throughout the paper. Synchronization and schedule updates are reduced to a single pull message in Sensemble, considerably reducing overhead. Upon receiving the broadcast message, the sensors are sampled, and after a preprogrammed time interval determined by each node's ID transmit data back to the base station. By imposing a strict synchronization, the operations of the nodes are overlapped to pack received messages at the base station as tightly as possible. However, the latency of the operation of the first Sensemble node delays the operation of the whole nodes in the cycle. Moreover, Sensemble entails idle-listening prior to the arrival of the first node's packet, holding back the throughput.

## III. Analytical Upper-bound on Throughput

In this section, we define a tight upper-bound on throughput for star-topology wireless sensor networks and propose a schedule-based mechanism for approaching the upper-bound. The proposed mechanism is general and parameterized so that it can be applied to any WSN with star topology.

### A. Upper-bound Formula

In an embedded wireless sensing platform, there are two types of communication packets, namely *control packets* and *data packets*. Control packets are used for management purposes including signaling, scheduling, synchronization, routing updates and collision avoidance, and their transmission will consume extra energy. Since the network bandwidth in WSNs is limited and the data packet size is small, control packets cannot be ignored as small overhead.

While control packets may be sent in both directions between the base station and sensor nodes, data packets are sent only in the upstream direction, i.e., from sensor nodes

to the sink. To maximize the bandwidth utilization for high-data-rate applications we should maximize the number of data packets and minimize the number of control packets. In other words, the system's *goodput* (data throughput) should be maximized while the overal throughput might remain unchanged. In this scenario, the base station spends most of its time receiving packets and the sensor nodes spend their time mostly on sampling and sending data packets.

In an ideal case where sensor nodes work in perfect synchrony, the base station works only in the receive mode and never transmits. Therefore, we define a tight upper-bound for goodput as follows:

$$\text{Upper-bound} = \frac{B}{PPT} \tag{1}$$

where $B$ is the number of bits of data payload per packet and $PPT$ represents the packet processing time at the base station in RX mode. The above formula holds when nodes transmit packets of the fixed size $B$.

We believe the above formula gives the tightest upper-bound as long as data packets are processed individually. However, if multiple packets are bundled in sensor nodes and sent to the base station, higher throughputs can be achieved.

### B. Parameterized System Configuration

In the following, we propose a receiver-initiated mechanism to approach the proposed upper-bound. The idea is based on the TDMA style in Fig. 4 but in such a way that completely eliminates the idle listening at the base station. For that reason, some nodes must be pulled in advance (i.e., *pre-pulled*) so that they are ready to transmit their sensing data right after the next pull message is sent by the base station. The implementation details are presented in Section IV.

The following is a semi-automatic method for configuring a system to which the proposed mechanism is applied. Suppose an embedded wireless sensing system with a communication timing cycle as depicted in Fig. 2. The durations of the system operations are represented as parameters. On the base station side, we have:

$TX_{BS}$ : pull message transmission time of the base station
$idle_{BS}$ : idle listening time of the base station
$PPT_{BS}$ : data packet processing time of the base station (including data packet receive time (RX) and data packet forward time to the host (FWD))

Note that some details are not shown for brevity. For example, $TX_{BS}$ might include the switching time between send and receive modes, SPI communication time and the actual RF transmission time, which is depicted as a hatched area in Fig. 2. Having known the timing of system operations in the pulling style with a single reply as parameterized above, the minimum number of the nodes to be pulled in each frame (i.e., $n$) can be determined:

$$n = \left\lceil \frac{idle_{BS}}{PPT_{BS}} \right\rceil + 1 \tag{2}$$

where $\lceil idle_{BS}/PPT_{BS} \rceil$ represents the number of pre-pulled nodes per pull message. We want to emphasize that the above formula gives the minimum number of nodes per frame. However, this number can be larger to further amortize the overhead of the pulling message.

The base station must perform $n$ PPT operations one after another. Having known this number, we can determine the frame length, which is the duration between two consecutive pull messages. This interval, which we call RTI, can be emulated as the duration between two successive real-time interrupts. It can also be achieved using a real-time scheduler if the base station runs an RTOS.

$$RTI = TX_{BS} + n \cdot PPT_{BS} \tag{3}$$

Similarly, the operations of a sensor node are parameterized as follows:

$RX_{Node}$ : pull message receive time of a node
$TX_{Node}$ : sensed data transmission time of a node

For simplicity, the sampling time is also included in $TX_{Node}$. Now we can determine the pre-transmission waiting time for each sensor node $i$ (i.e., $wait_i$). A sensor node may be pre-pulled to deliver its data after the next pull is sent (i.e. during the next frame), or it is not to be pre-pulled and is supposed to respond before the next pull message (i.e. during the same frame in which it is pulled). The waiting time is determined as follows: for $1 \le i \le n$:

$$wait_i = TX_{BS} + i \cdot PPT_{BS} - RX_{Node} - TX_{Node} + \begin{cases} RTI & \text{pre-pulled} \\ 0 & \text{not pre-pulled} \end{cases} \tag{4}$$

Note that the waiting times must be positive. However, if $wait_i$ can be determined using both formulas, the minimum non-negative value should be chosen to minimize latency and response time. If the minimum value for $n$, as presented in (2), is considered, only the last (i.e. $n^{\text{th}}$) node is not pre-pulled, and the first $n-1$ nodes must be pre-pulled. However, if a larger number is chosen for $n$ to further amortize the overhead of the pulling message, only $\lceil idle_{BS}/PPT_{BS} \rceil$ nodes should be pre-pulled to minimize latency, and the remaining $n - \lceil idle_{BS}/PPT_{BS} \rceil$ nodes should deliver their data within the same frame in which they are queried.

Based on the number of the pulled nodes in each frame the throughput can be measured using the following formula:

$$\text{Throughput} = \frac{n \cdot B}{TX_{BS} + n \cdot PPT_{BS}} \tag{5}$$

where $B$ represents the number of bytes of data payload per packet in each transmission. As mentioned before, the number of pulled nodes per frame can be larger to

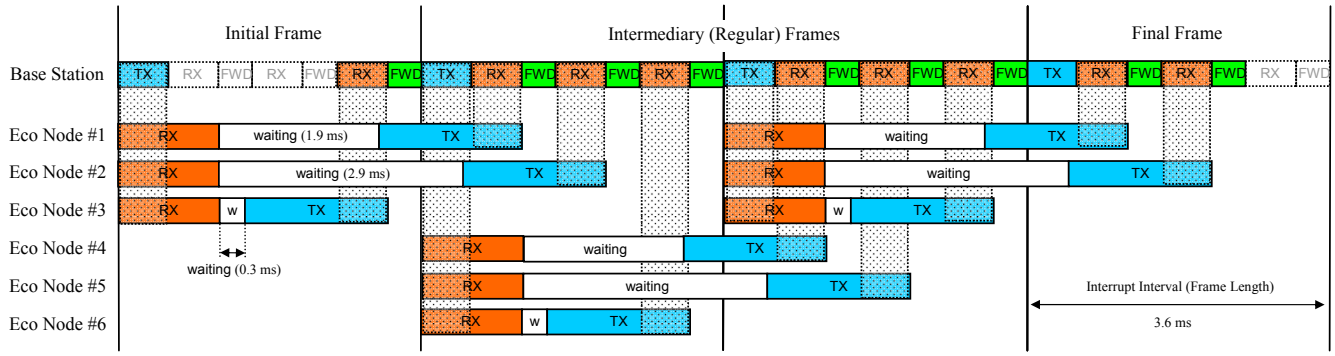


Figure 5. Timing diagram of the RIPE-MAC protocol with 3 time-slots per frame

further amortize the pulling overhead. Based on this fact, we can measure an upper-bound for the maximum practical throughput using the above formula.

$$\text{Upper-bound} = \lim_{n\to\infty}\left[\frac{n\cdot B}{TX_{BS}+n\cdot PPT_{BS}}\right] = \frac{B}{PPT_{BS}} \quad (6)$$

The above formula is the same as (1), showing that the proposed mechanism is able to approach the maximum practical throughput.

## IV. RIPE-MAC Protocol

In this section, we describe our proposed *Receiver Initiated, Pre-pull Enabled Medium Access Control* (RIPE-MAC) protocol based on the mechanism proposed in Section III, targeting the upper-bound defined in (1). As mentioned before, RIPE-MAC is built on top of the Eco platform. In the following we present a detailed implementation of the proposed protocol.

In order to send pull messages on a regular basis, it is essential that they are initiated from the base station rather than the PC host. Therefore, in our implementation the base station works independently from the host in the sense that it initiates the pull command. In this way, the host takes over only the high-level and supervisory tasks such as starting/stopping the system operation or determining the number and order of the nodes to be inquired. Since the RIPE-MAC protocol requires a predictable behavior, the base station sends the pull command based on a real-time interrupt which operates from an internal oscillator. This provides the base station operation with predictability which is a key factor in scheduling.

As shown in Fig. 2, with EcoDAQ, the base station spends around 1.7 ms on idle listening. This time is nearly twice as much time as needed for receiving a node's data, having known that PPT is about 1 ms for each packet. Therefore, at least two nodes must be pre-pulled. This means, according to (2), the base station must process data packets from at least three nodes between consecutive interrupts. In other words, each frame must consist of at least three time-slots.

**Algorithm 1** Pseudo-code of the RIPE-MAC protocol for the base station

PULL(n)

```
1: pullMessage[0] ← n
2: for i = 1 to n do
3:     pullMessage[i] ← Dequeue()
4:     Enqueue(pullMessage[i])
5: end for
6: Broadcast(pullMessage)
7: Listen()
8: for i = 1 to n do
9:     packet ← ReceiveReply()
10:    ForwardToHost(packet)
11: end for
```

Fig. 5 depicts the RIPE-MAC's timing diagram in the simplest form, i.e., pulling data from three nodes in each frame. Transmission times are scheduled in such a way that the base station does three pairs of receive (RX) and forward-to-host (FWD) operations one after another. The pull message contains three node IDs, two of which are actually pre-pulled for the next frame and the last one is pulled for the same frame. Upon receipt of the pull command and based on its order, each node has to wait for a certain amount of time as in (4) and then transmit its sensed data. Note that sensor nodes do sampling while waiting and transmit the most recent samples.

Algorithm 1 shows the pseudo-code of the RIPE-MAC protocol running on the base station. The *PULL* procedure is called at each real-time interrupt. The pull message is filled with the number of nodes ($n$) and the first $n$ node IDs in the queue. Having added to the pull message, each ID is enqueued to form a circular queue. The base station broadcasts the pull message and immediately switches to RX mode. It then performs $n$ pairs of RX/FWD prior to the next interrupt.

RIPE-MAC incurs no further complexity on nodes. Algorithm 2 depicts the pseudo-code of the RIPE-MAC protocol

running on Eco nodes. Upon receiving a pull message, a node checks if it is being pulled. If so, the node determines its waiting time based on its order and the total number of the queried nodes in each pull message as in (4). Having spent the waiting time, the node transmits the sensed data to the base station. If a node is not pulled, it goes to the sleep mode and wakes up just before the next pull message is sent.

**Algorithm 2** Pseudo-code of the RIPE-MAC protocol for Eco nodes

```
loop
    Listen()
    pullMessage ← ReceivePacket()
    pulled ← false
    n ← pullMessage[0]
    for i = 1 to n do
        if pullMessage[i] is MyID then
            waitTime ← GetWaitingTime(i, n)
            pulled ← true
            break
        end if
    end for
    if pulled then
        Wait(waitTime)
        TransmitSensedData()
    else
        Sleep()
        pullTime ← GetNextPullTime(n)
        WakeUpAt(pullTime)
    end if
end loop
```

If a node fails to send data or its packet is somehow lost, the operation of other nodes will not be affected; however, the node does not try to retransmit data, because it would breach the periodic data collection, but the equivalent of NACK messages can be easily achieved by re-pulling lost packets. Owing to the timely and predictable behavior of RIPE-MAC, the best periodic sleep time for each node can be easily measured in order to minimize idle-listening and save power by turning off the transceiver.

It is worth mentioning that RIPE-MAC does not need to perform time synchronization because the nodes respond relatively to the pull messages with different time offsets. In fact, synchronization and schedule propagation are reduced to a single pull message. The number of involved nodes in each frame can be more than three, and this approach is quite scalable. However, we have considered a small gap between successive pairs of RX/FWD to avoid probable time inaccuracies caused by interrupt jitter, hardware jitter or clock drift.

Similar to any other protocol, RIPE-MAC has its own limitations. For the most part, it assumes all nodes have the same bandwidth demand that is determined statically. As mentioned in Section II, our aim is to design an optimized TDMA-style MAC protocol capable of achieving the maximum practical throughput with minimum latency. To handle variable demands the RIPE-MAC protocol can be used in the scheduled-based period of existing hybrid protocols. As an alternative and based on the received data, the host can decide to assign more time-slots to the nodes with high-resolution data and less to those with no data or less meaningful data.

Finally, the upper-bound might be pushed even further by having RX and FWD performed in parallel at the base station. In that case, $PPT_{BS}$ would be $\max$(RX , FWD) because the FWD operation of the current packet could be overlapped with the next RX operation. However, the resulting upper-bound would still follows (1).

## V. Performance Evaluation

In this section, we study the performance of the RIPE-MAC protocol and compare it with that of CSMA/CA and the lightweight protocols reviewed in Section II. The performance metrics used for comparison are throughput and latency. In the presented experiments, $n$ is equal to 3, unless mentioned otherwise.

### A. Experimental Setup

In order to compare the performance of the protocols being investigated, we use Eco [5], [15], an ultra-compact, self-contained, expandable sensor node. It contains 4 KB RAM and 4 KB EEPROM and comes with a triaxial accelerometer with a temperature sensor, and an IR light sensor. Eco uses the Nordic nRF24E1, which includes an 8052 MCU core and an nRF2401 radio transceiver capable of 1 Mbps wireless speed in the 2.4 GHz band.

We performed the experiments using a Fast Ethernet base station, which was built by connecting a Freescale DEMO9SNE64 evaluation board [16] to a Nordic nRF24L01 [17] transceiver module via SPI bus. The Nordic nRF24L01 transceiver is compatible with Eco at 1Mbps speed mode. The base station is connected to the host computer over Fast Ethernet.

As shown in Fig. 1, the wireless network topology is single-hop, consisting of a number of Eco nodes in a star topology. The base station and all nodes communicate wirelessly over the same frequency channel. Each sensor node transmits packets with data payload of 27 bytes to the base station.

### B. Experimental Results

Since $B$ is 27 bytes and $PPT_{BS}$ is around 1 ms, according to (1), the upper-bound of our system is 211 Kbps, which is about 21% of the total wireless bandwidth. As shown in Fig. 6, RIPE-MAC achieves about 83% of the upper-bound, while EcoDAQ achieves about 30% of the upper-bound.

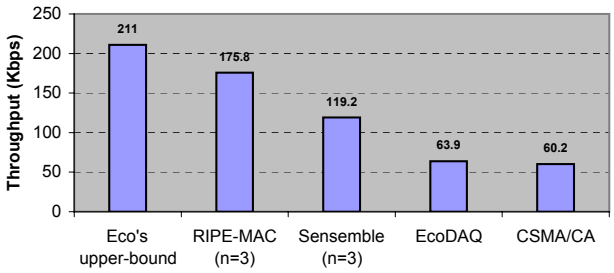


Figure 6. Comparison of throughput

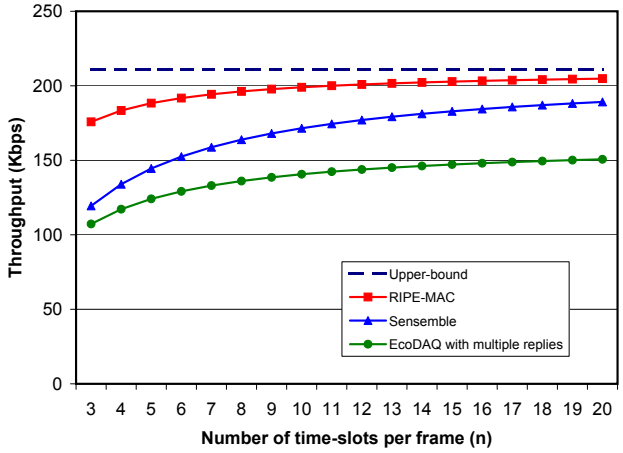


Figure 7. Comparison of throughput based on the number of time-slots per frame

In order to show the competency of RIPE-MAC, we also compare it with Sensemble. Fig. 4 shows the timing of Sensemble with three time-slots per frame, where the length of each frame is 5.3 ms. Sensemble does not hide the idle-listening time before the first reply arrives at the base station, resulting in an aggregate throughput of 119.2 Kbps (56% of the upper-bound).

We have also implemented the binary exponential backoff based CSMA/CA and measured its performance. It failed to scale to a comparable number of nodes while attempting to saturate the bandwidth, resulting in a throughput of 60.2 Kbps (about 29% of Eco's upper-bound). We observed in our experiments that with CSMA/CA more than half of the transmissions were unsuccessful mostly due to collisions.

For a better assessment, we compare RIPE-MAC with Sensemble and EcoDAQ with multiple replies based on the number of time-slots per frame ($n$). In other words, $n$ denotes the number of reply packets received by the base station between successive pull messages. In the case of EcoDAQ with multiple replies, $n$ represents the number of reply packets. Fig. 7 shows the achieved throughput with negligible packet loss and clock drift. RIPE-MAC shows a

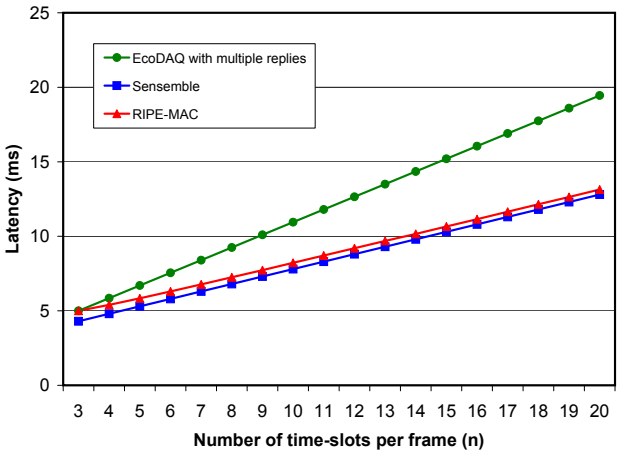


Figure 8. Comparison of latency based on the number of time-slots per frame

better performance and is able to approach the upper-bound much faster than the other protocols. In the case that $n$ is 20, RIPE-MAC achieves the throughput of 204.8 Kbps, which is about 97% of the upper-bound. In that case, Sensemble and EcoDAQ achieve 189.2 Kbps (90%) and 150.7 Kpbs (71%), respectively.

Another important performance metric is latency. Fig. 8 depicts the latency of the protocols being compared based on the number of time-slots per frame. The latency of RIPE-MAC is a little more than Sensemble due to the fact that the pre-pulled nodes must wait longer than the average latency. The diagram shows that our approach achieves a high throughput without considerably increasing latency.

The significance of RIPE-MAC is that it quickly approaches the upper-bound even with a few number of time-slots per frame. As a result, it is less prone to time inaccuracies than the other protocols. Our proposed protocol achieves higher throughputs with shorter latency compared to Sensemble and EcoDAQ.

### C. Comparison Based on Upper-bounds

The Sensemble platform has achieved 320 Kbps using 25 nodes in each cycle [14]. This higher gain, compared to our approach, is due to using faster hardware. The base station, which is connected via USB to the host computer, spends around 330 $\mu$s to process each data packet while in our case it is around 1 ms. Sensemble use a dedicated MCU on their nodes and it takes about 1.5 ms for each node to respond; however, it takes 2.9 ms in the case of Eco.

As mentioned before, the upper-bound presented in (1) can serve as a means of comparing protocols built on different platforms. Therefore, to do a better assessment we have estimated the upper-bound for the Sensemble's platform based on the information provided in [14] and [18]. Since $PPT_{BS}$ is 330 $\mu$s and $B$ is 0.128 Kb per packet,

according to (1) the Sensemble's upper-bound is around 387.9 Kbps. This means Sensemble achieves about 82% of its upper-bound pulling from 25 nodes in each cycle. In comparison, RIPE-MAC achieves 97.6% of the upper-bound pulling from 25 nodes per frame.

Finally, to show the superacy of RIPE-MAC we conjecture its projected performance if implemented on the Sensemble platform. In order to implement the RIPE-MAC protocol on the Sensemble platform, according to (2), at least 5 nodes must be pre-pulled because the base station spends 1480 $\mu$s on idle-listening prior to the first node's reply arrives. If RIPE-MAC is implemented on the Sensemble platform with six time-slots per frame, it will achieve about 384.6 Kbps which is 99% of Sensemble's upper-bound.

## VI. Conclusions

High-bandwidth, multiple-access communication is a challenge to designers of wireless sensing applications. This is because for resource-constrained platforms, one cannot afford to rely on localized abstractions provided by protocol stacks or operating systems. Instead, one must consider the timing globally. We present a formula for deriving the practical upper-bound on throughput. It shows the maximum possible throughput achievement regardless of the MAC protocol being used. We show how this upper-bound can be used to compare performance of protocols built on different platforms. We also propose a streamlined protocol that achieves over 98% of this upper-bound. It is able to approach the presented upper-bound very fast even with limited number of nodes per frame. Our approach is general and can be applied to various combinations of microcontrollers and radio transceivers. To apply the proposed protocol to similar WSNs, a parameterized system configuration is presented.

## Acknowledgment

The authors would like to thank Chulsung Park for design and implementation of the Eco platform, Chong-Jing Chen for development of the EcoDAQ protocol, and Sehwan Kim for his assistance in conducting the experiments. This work was supported in part by the U.S. National Science Foundation CAREER grant CNS-0448668.

## References

[1] H. Balakrishnan, "Opportunities in high-rate wireless sensor networking," in *NSF Networking of Sensor Systems (NOSS) Principal Investigator and Informational Meetings*, October 2004.

[2] G. Werner-Allen, S. Dawson-Haggerty, and M. Welsh, "Lance: optimizing high-resolution signal collection in wireless sensor networks," in *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. Raleigh, NC: ACM, November 2008, pp. 169–182.

[3] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–669, October 2002.

[4] B. Raman and K. Chebrolu, "Censor networks: a critique of "sensor networks" from a systems perspective," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 75–78, July 2008.

[5] C. Park and P. H. Chou, "Eco: Ultra-wearable and expandable wireless sensor platform," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE Computer Society, April 2006, pp. 162–165.

[6] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: a reliable bulk transport protocol for multihop wireless networks," in *Proceedings of the 5th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*. Sydney, Australia: ACM, November 2007, pp. 351–365.

[7] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*. Cambridge, MA: ACM, April 2007, pp. 254–263.

[8] J. V. Misic, S. Shafi, and V. B. Misic, "The impact of MAC parameters on the performance of 802.15.4 PAN," *Ad Hoc Networks*, vol. 3, no. 5, pp. 509–528, January 2005.

[9] C.-W. Chen, C.-C. Weng, and C.-J. Ku, "Design of a low power and low latency MAC protocol with node grouping and transmission pipelining in wireless sensor networks," *Computer Communications*, vol. 31, no. 15, pp. 3725–3738, September 2008.

[10] S. Yessad, F. Nait-Abdesselam, T. Taleb, and B. Bensaou, "R-MAC: Reservation medium access control protocol for wireless sensor networks," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN)*. Dublin, Ireland: IEEE Computer Society, October 2007, pp. 719–724.

[11] A. Tiwari, P. Ballal, and F. L. Lewis, "Energy-efficient wireless sensor network design and implementation for condition-based maintenance," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 1, pp. 1–23, March 2007.

[12] K. Klues, G. Hackmann, O. Chipara, and C. Lu, "A component-based architecture for power-efficient media access control in wireless sensor networks," in *Proceedings of the 5th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*. Sydney, Australia: ACM, November 2007, pp. 59–72.

[13] C.-J. Chen and P. H. Chou, "EcoDAQ: A case study of a densely distributed real-time system for high data rate wireless data acquisition," in *Proceedings of the 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2008, pp. 427–432.

[14] R. Aylward and J. A. Paradiso, "A compact, high-speed, wearable sensor network for biomotion capture and interactive media," in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*. Cambridge, MA: ACM, April 2007, pp. 380–389.

[15] "Ecomote," http://www.ecomote.net/.

[16] "Freescale Semiconductor," http://www.freescale.com/.

[17] "Nordic Semiconductor Inc." http://www.nordicsemi.no/.

[18] R. Aylward, "Sensemble: A wireless inertial sensor system for interactive dance and collective motion analysis," Master's thesis, MIT Media Laboratory, Cambridge, MA, August 2006.