12-2009

Press the Cancel Button! A Performance Evaluation of Scalable In-Network Data Aggregation

Jamie Macbeth
University of California, Los Angeles, jmacbeth@smith.edu

Majid Sarrafzadeh
University of California, Los Angeles

Press the Cancel Button!
A Performance Evaluation of Scalable In-Network Data Aggregation

Jamie Macbeth
Department of
Computer Science
University of California
Los Angeles, California 90095
Email: macbeth@cs.ucla.edu

Majid Sarrafzadeh
Department of
Computer Science
University of California
Los Angeles, California 90095
Email: majid@cs.ucla.edu

***Abstract*—We perform real-world tests of the performance of medium access control protocol schemes for scalable data aggregation in sensor networks. Specifically, we evaluate the performance of a Listen-and-Suppress Carrier Sense Multiple Access (LAS-CSMA) scheme on the *duplicate-insensitive exemplary monotonic* aggregates MAX and MIN. These schemes reduce power consumption, network bandwidth usage and delays by suppressing node packet transmissions that are proven to be unnecessary in the query response. This is possible when nodes listen to the transmissions of other nodes as they respond.**

Scalability tests were performed for 8 networks of various sizes, the largest having 24 Crossbow IRIS wireless mote modules as data nodes. We found that for the largest networks, packet transmissions, receptions, and response delays were reduced 44%, 65%, and 38% respectively. Packet transmissions and response delays were found to scale nearly logarithmically with respect to the size of the network, which resembles predictions by recent theoretical results. However, the consumption of energy and network resources is further reduced in platforms with better software implementations and hardware support for the TinyOS Active Messages interface. It is found that the scalability and performance of LAS-CSMA depends strongly on how quickly nodes can cancel a pending packet transmission based on a calculation with data from an incoming packet.

***Keywords*-In-Network Data Aggregation; Medium Access Control; Scalability; Wireless Sensor Networks.**

I. Introduction

As networked embedded mote modules continue to diminish in size and cost, the interest increases in the design of protocols, hardware and software components, and applications for networked embedded systems. The use of wireless communication links and the addition of sensing and actuation devices has encouraged their use in many domains such as environmental and medical monitoring, habitat monitoring, and participatory urban sensing.

These networked embedded systems are usually battery powered, and may be limited in other respects. The radio transceiver often dominates the power dissipation characteristics of the system, and many of these systems are constrained in their available communication bandwidth. Overall, the system must use the radio transceiver carefully to achieve good performance and lifetime from the system.

Wireless sensor networks nearly always have only a small subset of base station nodes that are connected to a host computer or wider area network. Therefore, for operations in which all nodes take part, for example, sensor data collection or actuation command distribution, the broadcast and "convergecast" patterns of communication are most common. Data aggregation techniques seek to optimize the system for these patterns of communication.

One objective in data aggregation system design is scalability–assuring that nodes can be added to the network without significantly degrading performance or without straining scarce resources. When monitoring or sensing networks collect data, often only a summary or cross-section of data is required. Data aggregation techniques can use the fact that not all of the data is needed and try to save resources.

A. MAC Protocols

Constraints important to sensor and actuation networks present challenges at all layers of network design. However, when issues such as transceiver power consumption and network fairness, delay, and throughput are of import, controlling access to the communication medium is most essential. The traffic created by these networks exhibits patterns that are aren't very well addressed by general purpose medium access control protocols. The exploitation of predictable network traffic patterns to extend network lifetime and increase reliability and efficiency is the aim of research of MAC protocols for sensor networks.

In this paper, we evaluate the performance of contention based medium access control schemes for data collection in sensor networks. In particular, we study Listen-And-Suppress (LAS) CSMA as introduced in [1]. We choose the LAS-CSMA scheme because of its ease of implementation and generality. Under LAS-CSMA nodes listen to the channel and avoid transmitting and sleep if their data is not needed.

We compare the performance of data aggregation with and without the LAS-CSMA protocol concept. We measure the query completion delay, number of transmissions, and number of receptions for networks of various sizes between 1 and 24 data nodes. The power dissipation of executing queries can be examined by considering the average total number of transmissions made or attempted by all data nodes in the network. For many types of networks listening time is more strongly correlated to node power consumption, so we measure the collective average time that nodes spend listening to the network during the query process.

II. Previous Work

The TinyDB query processing framework was introduced in [2] and measurements of the power consumption profile of the query execution and processing framework are given in [3]. The TAG system's snooping based techniques are reminiscent of LAS-CSMA, but are not approached at the MAC level.

Cougar is another query processing framework that, similar to TinyDB, has a declarative syntax language based interface. In [4] the interaction between Cougar's query and network layers is simulated using the ns-2 network simulator. The simulation includes 802.11 MAC layer collisions and gives the per-node energy dissipation for different network topologies and query operator selectivities.

Most evaluations of the MAC performance of data acquisition systems are performed in network simulation. In [5] Q-MAC a new sleep schedule for query based sensor networks is proposed that provides minimum end-to-end latency with energy efficient data transmission. Its performance is evaluated with the Glomosim simulator.

DMAC [6] is a low-latency energy efficient MAC designed for data aggregation in unidirectional trees of nodes. In networks with sleep/wake cycles it is designed to prevent the increased latency of data forwarding due to sleep delay. Its performance was evaluated with ns-2 using the CMU wireless extension. However, ns-2 may not account for differing transceiver chips, their MAC logic and the associated drivers.

Occasionally sensor database query processing systems are evaluated by performing simulations using real data sets acquired without the query processing system (in [7] for instance). In this paper we focus on a generalized protocol paradigm which was simple to implement at the application level of most wireless sensor networks, and evaluate its scalability and performance experimentally.

III. Background

A. Scalable Data Aggregation

In our experiments we perform a real-world test of the performance of medium access control protocol schemes for data aggregation. Specifically we test the performance of a Listen-and-Suppress Carrier Sense Multiple Access (LAS-CSMA) scheme on the *duplicate-insensitive exemplary monotonic* aggregates MAX and MIN. LAS-CSMA [1] reduces power consumption, network bandwidth usage, and delays by suppressing node transmissions that are proven to be excluded from the query response.

In a real data aggregation network, a node (usually the *base station* node) issues a query by sending an appropriate query packet on the network. The query reaches all associated *data nodes* either by broadcast or through forwarding. Each data node responds by sending its records needed to satisfy the query, but while attempting to send, each data node listens to the other records that appear on the network. Because *duplicate-insensitive exemplary monotonic* queries request only the extreme (maximum or minimum) record of all records existing on the data nodes, a data node can suppress its transmission if it hears a record that is more extreme from its own.

For example, assume that each of 6 stations has a single integer in storage, and that the query requests the single greatest integer value. If the values stored among the data nodes are the integers 2, 5, 7, 8, 11, 13 and the integer 8 is successfully transmitted before the data nodes storing 2 and 7 are able to transmit, these data nodes can cancel their transmissions, even if one of their attempts to transmit resulted in a collision. This reduces the contention for the network, making it easier for the data nodes that store 11 and 13 to transmit.

Macbeth and Sarrafzadeh [1] study the characteristics of medium access control for scalable data aggregation. They characterize the scheduling of packets sent in response to a data aggregation query as a Poisson process with rate g. They analyze CSMA/CD, which differs only slightly from the CSMA/CA used in IEEE 802.15.4.

They find that for a fully-connected network of n data nodes, the expected delay, transmissions, and slots spent listening are, respectively:

$$\begin{aligned} E[\text{Delay}(n)] &\simeq \frac{TH_n}{S_{\text{CSMA}}} \\ E[\text{Trans}(n)] &\simeq TH_n e^{g\tau} \\ E[\text{Listens}(n)] &\simeq \frac{T}{S_{\text{CSMA}}}\ O(n). \end{aligned} \tag{1}$$

where S_{CSMA} is the throughput, T is the response packet length, and τ is the maximum propagation delay between any two nodes. H_n denotes the nth Harmonic Number, and it is $O(\lg n)$.

IV. Experimental Platform

A. Hardware

Two sets of experiments were performed using two types of mote modules. The first uses the Crossbow IRIS mote

module, also called the XM2110CA [8]. The IRIS platform is intended to enable wireless low-power sensor networks and is based on an Atmel ATmega1281 8 bit microcontroller with a clock speed of 7.37 MHz and 128Kb of program memory. Its RF radio transceiver is the IEEE 802.15.4 compliant Atmel RF230 [9], with a data rate of 250 Kbps in the ISM band, 2.4 to 2.48 GHz.

The second set of experiments uses the Crossbow MICAz, also known as the MPR2400 [8]. The MICAz is based on an Atmel ATmega128L microcontroller, also at 7.37 MHz with 128Kb of program memory. However, differing from the IRIS, MICAz uses the Chipcon CC2420 802.15.4 compliant RF transceiver [10]. Like the RF230, the CC2420 operates in the 2.4 to 2.48 GHz band with a data rate of 250 Kbps. We chose the IRIS and MICAz motes because of their wide popularity and extensive use in the sensor network community.

For the IRIS mote experiments a variable number of Crossbow IRIS Motes were arranged in close proximity as data nodes, and one more IRIS Mote acts as a base station while connected to the Crossbow MIB520 USB interface board. The data nodes and base station were all placed side by side on a small workbench so that each pair of nodes would be well in transmission range. This fully-connected broadcast network topology can be represented by a complete graph, which is most appropriate for a performance evaluation of medium access control.

A similar arrangement was used for the MICAz experiments. Both the IRIS and MICAz are designed for battery power (2 AA batteries) and we use battery power for all motes except the base station. The base station is powered through the MIB520.

B. Software

The motes in our experiment run TinyOS [11], an open-source component-based operating system designed for wireless embedded sensor networks. At the time of this writing only the CVS version of TinyOS 2.x (version 2.0.2) has compiler and programming support for the Crossbow IRIS. TinyOS and applications running under TinyOS are written in a special programming language for networked embedded systems called NesC [12].

Typically, programming the motes is performed by connecting directly via the mote's 51 pin port to the MIB520 USB gateway. Alternatively, motes can be reprogrammed over the network using Deluge [13], a dissemination protocol for large data objects. Motes are individually programmed with a Deluge bootloader image that can receive a program binary wirelessly. The program binary image is broadcast by the base station mote, and the 24 other nodes retrieve the application binary and reboot with that image. We chose this method because it saves us from having to program each mote by hand in each phase of development and experimentation.

C. TinyOS Active Messages

TinyOS has a simple event-based paradigm for wireless communication between nodes called *Active Messages* [14] [15]. The Active Message (AM) model matches its communication primitives with the constrained hardware environment of embedded networked systems. Its programming interface allows for the overlap and integration of communication and computation, which is indispensable for efficient in-network data aggregation in sensor networks. It also allows multiple applications to simultaneously use communication resources. TinyOS's Active Message model presents the API given in Figure 1.

Using a `message_t` object, the API allows one get a pointer to the message payload, and, subsequently, to send a message containing that payload to another node, or to the broadcast address. The `send` method returns a `SUCCESS` error code if a send operation was successfully initiated. The API is asynchronous and event-based, and once the send operation is completed, the component will receive the `sendDone` event with an error code indicating success or failure.

In between the return of a call to `send` and the reception of a `sendDone` event, the application can call `cancel` and attempt to cancel the last initiated send operation. The `cancel` method returns a `SUCCESS` error code if the transmission was canceled properly (i.e. the message was not sent in its entirety). The `cancel` method returns `FAIL` if the attempt to cancel was not successful. This means that the send will continue, but does not guarantee that it won't fail. A successful call to `cancel` should always result in a `sendDone` event with a `FAIL` error code indicating send failure.

The performance of scalable in-network data aggregation weighs heavily on the ability of a station to cancel a transmission. After hearing the query packet, a station will attempt to transmit while listening to the channel for other stations transmissions. If the channel is busy and the transmission is rescheduled, a packet may arrive that allows the station to suppress its transmission through cancellation. Alternately, if there is an attempt to transmit and the `sendDone` event occurs with a `FAIL` error code, the code suppresses transmission simply by not retrying the send method.

V. Experimental Procedure

We perform experiments on a simplified scalable data aggregation system similar to that listed above. Each iteration of the experiment is performed with the following steps:

1) The base station node sends a packet to the broadcast address with a bit pattern indicating that it is a query packet.
2) Each of the data nodes, upon receiving the query packet, randomly generates an 8 bit unsigned integer.

```
interface AMSend {
  command error_t send(am_addr_t addr, message_t* msg, uint8_t len);
  command error_t cancel(message_t* msg);
  event void sendDone(message_t* msg, error_t error);
  command uint8_t maxPayloadLength();
  command void* getPayload(message_t* msg, uint8_t len);
}
interface Receive {
  event message_t* receive(message_t* msg, void* payload, uint8_t len);
}
```

Figure 1. The TinyOS Active Message Interface

3) Each data node then attempts to send a packet with this integer in the payload to the broadcast address along with a pattern to indicate that it is part of the query response. This transmission may not be successful depending on channel contention.
4) Concurrent with the attempts to send the packet, each data node also listens to successful transmissions performed on the channel.
5) If a data node receives a query response packet from another data node with a greater integer payload and has not yet completed its own transmission, it attempts to cancel its transmission.
6) After transmitting the query, the base station node waits for 3 seconds to receive query response transmissions from the data nodes.
7) The base station records the time elapsed between the transmission of the query packet and its receipt of the last query response packet. This will be called the *query response delay*.

A simplified representation of the code for the TinyOS scalable data aggregation test application (without the data collection framework) is given in Figure 2.

After this first phase is completed, the base station node executes a second query to collect individual information from the data nodes about the query execution over the network. This second query does not use a scalable data aggregation medium access scheme. It consists of the following steps:

1) The base station node prepares a packet with a special data pattern to indicate that it is a "meta-query" for data about the execution of the previous query
2) The base station iterates over the set of data nodes performing the following two steps
   a) It sends the "meta-query" packet to the data node individually.
   b) It sets a short timer to wait for the response.
3) Each data node responds with the following:
   a) Its node ID
   b) The stored integer from the query-response experiment just performed,
   c) A bit indicate whether or not the data node performed a successful transmission,
   d) An 8 bit integer indicating how many records it received from other data nodes before making a successful transmission or suppressing it,
   e) A bit indicating whether or not the transmission was successfully canceled, and
   f) A bit indicating whether or not it heard a better record value from another data node.

To create conditions that might be present in a typical sensor network data aggregation system, all payloads in our performance evaluation were 28 bytes.

A. Data Collection

Data is collected using the base station mote and sent to the host PC using TinyOS's printf facility. The TinyOS printf library provides terminal printing functionality to TinyOS applications through the base station mote connected to a the PC via USB. The base station TinyOS application's calls to printf are sent via USB to a java client application on the host PC.

The space-delimited data collected on the host PC includes the query response delay, the final query result, and the number of packets received by the base station node during the query response. Also, a line is logged for each packet received by the base station from the data nodes in the second phase. These include the data node's record, a bit indicating if it made a successful cancel, a bit indicating if it made a successful transmission, the number of packets it received from other data nodes before it transmitted or canceled, and a bit indicating if it heard a better record in a packet that it received. All lines in the log include a unique experiment ID.

B. Retries

The TinyOS chip specific code for the Atmel RF230 automatically retries transmissions at the message buffer layer level. By default it tries to retransmit 4 times if

```
module ScalableDataAggTest {
  uses interface AMSend as AMSender;
  uses interface Receive as AMReceiver;
  uses interface Packet;
  uses interface Random;
}
implementation {
  message_t packet;
  task void doSend() {send();}
  void send() {
    if (!heardBetterResults &&
        call AMSender.send(BROADCAST_ADDR,
                           &packet,
                           PACKET_LENGTH) != SUCCESS) {
          post doSend();
      }
  }
  event void AMSender.sendDone(message_t*msg, error_t err) {
    if(err != ECANCEL && err != SUCCESS)
      post doSend();
  }
  event message_t * AMReceiver.receive(message_t*msg, void*payload, uint8_t len) {
    uint8_t * rx_payload_ptr = (uint8_t*) payload;

    if (rx_payload_ptr[PACKET_TYPE] == QUERY_PACKET) {
        // query packet
        recordValue = call Random.rand16();
        tx_payload_ptr = call Packet.getPayload(&packet, PACKET_LENGTH);
        tx_payload_ptr[PACKET_TYPE] = RESPONSE_PACKET;
        tx_payload_ptr[RECORD_VALUE] = recordValue;
        post doSend();
    } else if (rx_payload_ptr[PACKET_TYPE] == RESPONSE_PACKET &&
              heardBetterResults == 0 &&
              rx_payload_ptr[RECORD_VALUE] >= recordValue) {
        // packet from another data node
        heardBetterResults++;
        call AMSender.cancel(&packet);
    }
    return msg;
  }
}
```

Figure 2. TinyOS Scalable In-network Data Aggregation Test Application

the channel is sensed to be busy, or if there is another condition to prevent immediate transmission. Because this occurs at the lower level and the listen and suppress scheme is implemented at the application level, attempts to cancel or suppress an attempted transmission upon hearing a transmission from another data node may be difficult depending on the semantics of cancel and its interaction with the retry mechanism.

We found that for non-LAS networks, reducing the number of retries had no significant effects because the higher level application keeps trying to transmit until there is success; changing retries at the lower level should make no difference.

We found that for LAS networks of 24 data nodes, there was an average of 23.6 transmissions with a standard deviation of 0.56 when the driver was set to perform 4 retries. We assume that data nodes hear transmissions from other data nodes but their attempts to cancel transmissions

are unsuccessful, and the driver goes on to retry and perform an unnecessary transmission. In order to test under favorable conditions, we chose to set the retries to 0 to make transmission suppression most likely.

C. Broadcasts and Acknowledgement

In our experiments both the data nodes and the base station node transmit using the broadcast address, which doesn't allow for acknowledgment packets. If LAS is used in a setting where the system can account for a slight error in the query response, then this is acceptable. Otherwise in settings where exactness is required, the base station node can resend the query along with the final received query result. Then any data nodes could resend their records if they can improve the result. In this case the base station's echo of the query response can act as an acknowledgement.

For LAS networks, data nodes are required to send their query response packets on the broadcast address in order for other data nodes to listen. For the non-LAS data nodes don't snoop on each other's transmissions, and this gives us the choice of using the broadcast address or the base station address when they send their query response packets.

In evaluating the performance a system without LAS we chose for data nodes to use the broadcast address in sending their query response to the base station node. This would make for a better comparison with LAS without the network contention due to the base station's acknowledgement and retransmission of data node messages that are addressed only to the base station.

VI. Results

The experimental procedure described above was performed for 60 iterations for each mote network configuration. We performed the experiments for 8 network configurations consisting of 1, 2, 4, 8, 12, 16, 20 and 24 data nodes and a single base station node. Experiments for each of the 8 network configurations were performed with the listen and suppress scheme and without.

Results from our tests with IRIS motes are given in Figures 3, 4, and 5. The average transmissions, receptions, and delay, all with standard deviations, are given in Figure 3. Figure 4 shows the average packets received by the base station, the average lost transmissions, and the average cancels, all with standard deviations. Figures 3 and 4 show scaling plots for 8 network configurations consisting of 1, 2, 4, 8, 12, 16, 20 and 24 data nodes. Figure 5 shows scatter plots that compare data node transmission and receives and base station receives for all tests of the 24 data node network.

Figure 3a shows the average total transmissions for 8 networks of different sizes with and without LAS. Without LAS, all stations successfully transmitted records for all experiments. LAS is seen to significantly reduce transmissions for the larger networks, and is nearly logarithmic in the size of the network, in agreement with [1]. For the 24 data node network, the average transmissions are reduced by 44%.

In Figure 3b the total number of receptions (summed over all data nodes) is seen to be reduced significantly as well–for the 24 node network, the reduction is 65%. This is important because for many embedded systems and devices, as much or even more energy is expended in receiving messages than in sending them.

In our experiments, all receptions by a data node are counted until either the station completes a transmission or the station hears a result that allows it to suppress its transmission. This is meant to simulate how in a real system the data nodes save power by completely shutting down the radio transceiver when no more transmitting or receiving is needed for the query. The growth with network size is linear in agreement with [1].

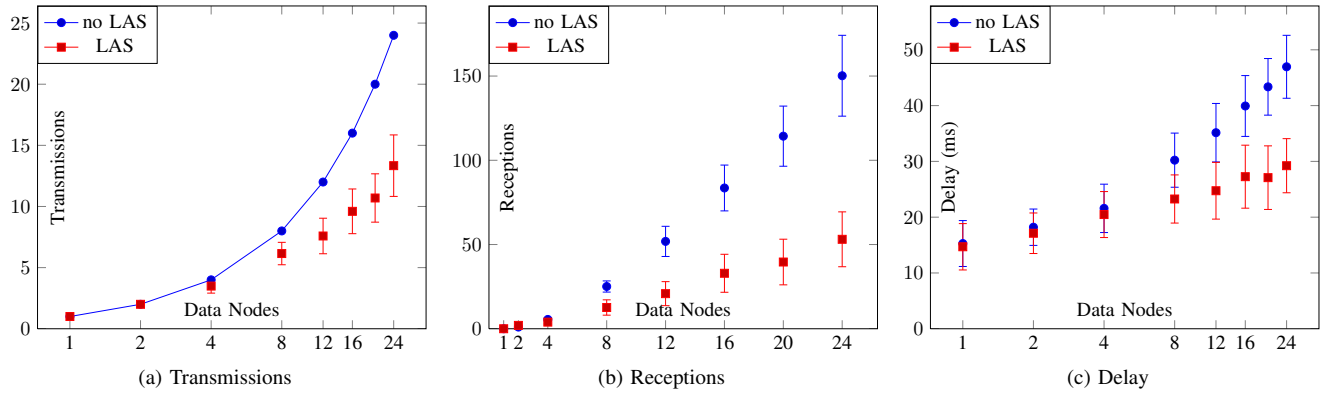
Before discussing the delay measurements we calculate the transmission time of data frames in our experimental arrangement. This calculation doesn't include other factors that may contribute to the throughput, like initial and subsequent back-off periods.

Both the CC2420 and the Atmel RF230 support a maximum data rate of 250 Kbps. Pessimistically, if we assume that the maximum number of addressing bits are used in the IEEE 802.15.4 frame, the length of PPDU frames in our experimental setup (with a data payload of 28 bytes) is 59 bytes. This gives us a frame transmission time of 1.89 ms.

We observe in Figure 3c that the use of LAS significantly reduces delay in receiving all packets for the query response. Delay is measured by the base station (in milliseconds) and is timed starting just before the transmission of the query packet and ending at the receipt of the last response packet. Even without considering the overhead of the query packet transmission and reception, the delay for the 24 node LAS network is reduced by 38%. For the LAS paradigm, the delay appears to be logarithmic with respect to the network size.

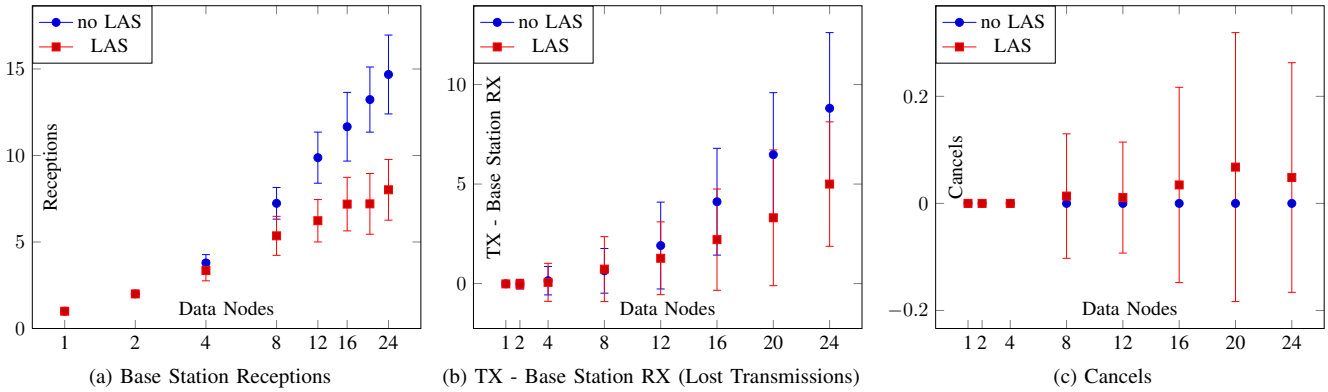
The base station also measures the total number of packets it receives during the query response phase. This will also help give an impression about the base station's energy consumption due to reception. Many transmissions that appear to be successful at the sender don't appear intact at the receiver. This is presumably due to congestion and collisions which are very likely just as the query response begins. Recording the number of transmissions received and comparing it to the number sent gives us an idea of the likely extra network traffic due to retries, and of the likely errors incurred if the system doesn't use acknowledgement or retransmission.

In Figure 4a it is seen that LAS reduces the number of receptions by the base station significantly. But what is also noticable is the increased number of receptions that are missed when the system is not using LAS. In Figure 4b a plot of the difference between data node transmissions and



(a) Transmissions (b) Receptions (c) Delay

Figure 3. Average Transmissions, Receptions, and Delay



(a) Base Station Receptions (b) TX - Base Station RX (Lost Transmissions) (c) Cancels

Figure 4. Base Station Receptions, Lost Transmissions, and Cancels

base station receptions is given. This represents the packets that were likely lost due to collisions and may need to be retransmitted. Very few cancels were successful in our test with the IRIS motes (Figure 4c). It would seem that most transmissions were avoided before `send` was called or after a failed call to `send`. This is discussed further below in our comparison of the IRIS and MICAz mote platforms.

Figures 5a and 5b are scatter plots of delay versus receptions and delay versus transmissions for a network with 24 data nodes. The results appear to be linear for both LAS and non-LAS networks, with significantly less transmissions, receptions and delay for LAS. A similar plot is given for the receives by the base station. For the non-LAS networks, all 24 data nodes always transmit, accounting for a standard deviation of 0 for the transmissions.

A. *Initial Response Bursts*

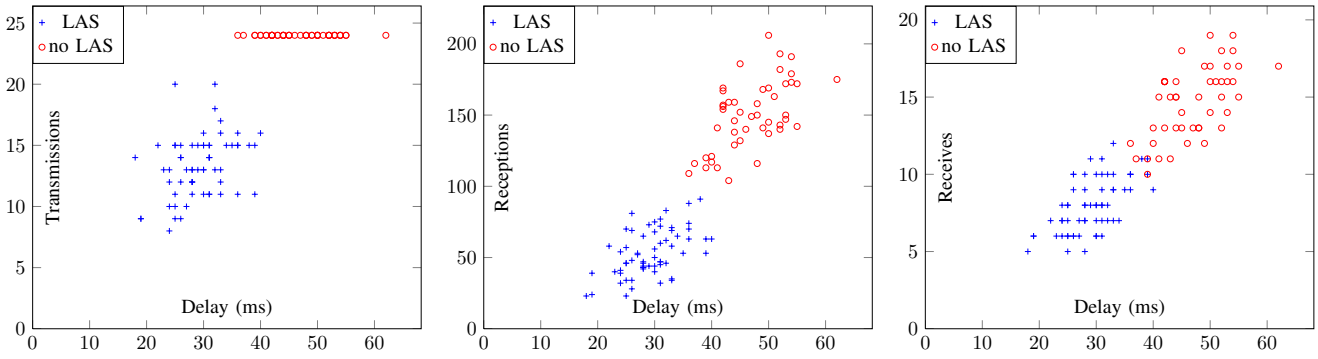
One phenomenon considered in [1] is the high number of collisions that occur just following the transmission of the query packet. In this experiment there is no routing, and all data nodes are in close proximity to the base station node. The data nodes have no other processing tasks and identical hardware, so we expect them all to respond to the query at the same time, causing many packets in the network to become ready to transmit at the same instant. This would cause many or all data nodes to participate in a collision right at the start of the query response.

IEEE 802.15.4 requires that nodes schedule a packet for transmission a random time in the future for the first attempt. This feature allows the system to avoid collisions due to the initial burst of available packets just after the query packet is received by all data nodes. As a result we did not observe major packet collisions due to an initial burst in data node responses.

B. *Comparing IRIS and MICAz*

We repeated the LAS versus non-LAS experiments for a network consisting of 4 MICAz motes. In comparing the 4 data node MICAz network with the 4 data node iris network we found some dramatic differences. For networks without LAS, we found again that all data nodes were able to make a complete transmission in each experiment.

We found that, with the LAS paradigm, there were 39% less transmissions on average with the MICAz motes than with the IRIS motes. There were also somewhat less packets received by the base station node in the MICAz case.



(a) Delay/Transmission Scatter Plot (b) Delay/Reception Scatter Plot (c) Base Station Receive Scatter Plot

Figure 5. Transmission and Reception Scatter Plots

Table I
COMPARING IRIS TO MICAZ

	Data Nodes	TX	RX	Delay (ms)	Base Station RX	lost TX	Cancels
iris LAS	4	3.49 ± 0.58	3.9 ± 1.0	20 ± 4	3.3 ± 0.6	0.1 ± 0.9	0 ± 0
iris no LAS	4	4 ± 0	5.5 ± 1.0	21.5 ± 4.3	3.79 ± 0.48	0.1 ± 0.7	NA
MICAz LAS	4	2.14 ± 0.82	3.3 ± 0.82	20.3 ± 4.9	1.92 ± 0.66	0.2 ± 1.1	1.82 ± 0.81
MICAz no LAS	4	4 ± 0	5.37 ± 0.90	24.9 ± 3.4	3.58 ± 0.58	0.36 ± 0.79	NA

Without LAS, the results for the MICAz and IRIS were very similar.

We also note the lack of success data nodes had in canceling transmissions when records were received from other data nodes. For networks of 4 data nodes, there were no successful cancels among the IRIS data nodes in our experiments. The MICAz data nodes had, collectively, 1.82 cancels per experiment, so that, on average, nearly half of the data nodes avoided making complete transmissions. Comparing the lost transmission results to the completed transmission results, on average only 1% of the transmissions that MICAz data nodes completed were lost presumably due to collisions.

To learn more about this discrepancy, we studied the TinyOS Active Message implementations for both the Atmel RF230 and the Chipcon CC2420, the respective transceiver chips for the IRIS and MICAz mote platforms. IEEE 802.15.4 requires that nodes perform a clear channel assessment (CCA) operation before attempting to transmit. Both the RF230 and the CC2420 offer an accelerated atomic-CCA-and-transmit operation. On the CC2420, this is the default mode of operation; the Active Message implementation transfers the data frame to the CC2420 and issues a transmit command which is ignored if the channel is sensed to be busy. The transmit command is issued from a state where it can receive incoming packets and avoid the transmission if the initial attempt is unsuccessful.

However, the TinyOS RF230 driver uses software CCA when transmitting data frames. the RF230 driver must, in software, deal with assessing that there is a clear channel before making a transmission. This requires setting a sub register to initiate the clear channel assessment request, reading two other sub registers to determine that the assessment completed, and the outcome. At this time, if the channel is sensed to be free, the transmit sequence is begun. The reception-cancellation process that would help to suppress the transmission appears to be disabled in this mode of operation. This would help in explaining how the MICAz outperforms the IRIS in our tests.

VII. DISCUSSION AND FUTURE WORK

In this work we have demonstrated experimentally the viability of the listen and suppress paradigm of MAC protocols for scalable in-network data aggregation. This is demonstrated for data queries that exhibit a high level of redundancy. These protocols are shown to have significant advantages toward scalability when it is gauged by the total expected number of transmissions and receptions in the network. The use of these paradigms also reduces the time needed for query response. All of these characteristics can be shown to reduce overall power dissipation and bandwidth usage in the network.

Flexible and efficient implementations of the Active Messages API reduce transmissions, overhearing and delays in the query response by receiving other data nodes transmissions and canceling their own if they are determined to be unneeded. We found that an effective cancel operation can cause a dramatic improvement in the performance of in-network data aggregation through the receive-and-cancel process. It is convenient that Active Messages allow the network communication and application to overlap and collaborate so well for the goal of resource conservation in resource-constrained systems.

In the future we are drawn towards the performance evaluation and comparison of contention based medium access with contentionless TDMA-like schemes. A TDMA-like scheme may be more difficult to use in ad hoc networks, but theoretically [1] can reduce packet transmissions and receptions even more dramatically.

Future implementations and experiments will consider the same problem for multi-hop networks. This raises many issues for the LAS scheme– most importantly the fact that many data nodes will be forced to transmit in order for the needed record to reach the base station node. Effective routing will be necessary, but in that context we would still like to consider scalability.

REFERENCES

[1] J. Macbeth and M. Sarrafzadeh, "Scalable Medium Access Control for In-Network Data Aggregation," in *DIAL M-POMC '08: Proceedings of the Fifth International Workshop on Foundations of Mobile Computing*. New York, NY, USA: ACM, 2008, pp. 13–22.

[2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-hoc Sensor Networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, 2002.

[3] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an Acquisitional Query Processing System for Sensor Networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[4] Y. Yao and J. Gehrke, "Query Processing in Sensor Networks," in *CIDR*, 2003.

[5] N. A. Vasanthi and S. Annadurai, "Energy Efficient Sleep Schedule for Achieving Minimum Latency in Query Based Sensor Networks," in *SUTC '06: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing - Vol 2 - Workshops*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 214–219.

[6] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks," *IPDPS*, vol. 13, p. 224a, 2004.

[7] A. Deshpande, C. Guestrin, W. Hong, and S. Madden, "Exploiting Correlated Attributes in Acquisitional Query Processing," *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pp. 143–154, April 2005.

[8] *MPR-MIB Users Manual*, Crossbow Technology, San Jose, CA, 2007, www.xbow.com.

[9] *AT86RF230 Data Sheet*, Atmel Corporation, San Jose, CA, 2009, www.atmel.com.

[10] *CC2420 Data Sheet*, Texas Instruments, Inc., Dallas, TX, 2009, www.ti.com.

[11] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," *SIGPLAN Not.*, vol. 35, no. 11, pp. 93–104, 2000.

[12] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The NesC Language: A Holistic Approach to Networked Embedded Systems," in *PLDI '03: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*. New York, NY, USA: ACM, 2003, pp. 1–11.

[13] J. W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," in *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2004, pp. 81–94.

[14] D. E. Culler, J. Hill, P. Buonadonna, R. Szewczyk, and A. Woo, "A Network-Centric Approach to Embedded Software for Tiny Devices," in *EMSOFT '01: Proceedings of the First International Workshop on Embedded Software*. London, UK: Springer-Verlag, 2001, pp. 114–130.

[15] T. Eicken, D. Culler, S. Goldstein, and K. Schauser, "Active Messages A Mechanism for Integrated Communication and Computation," 1992, pp. 256–266.