AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks

Tian He Brian M. Blum John A Stankovic Tarek Abdelzaher
Department of Computer Science
University of Virginia

Abstract

Sensor networks, a novel paradigm in distributed wireless communication technology, have been proposed for use in various applications including military surveillance and environmental monitoring. These systems could deploy heterogeneous collections of sensors capable of observing and reporting on various dynamic properties of their surroundings in a time sensitive manner. Such systems suffer bandwidth, energy, and throughput constraints that limit the quantity of information transferred from end to end. These factors coupled with unpredictable traffic patterns and dynamic network topologies make the task of designing optimal protocols for such networks difficult. Mechanisms to perform data centric aggregation utilizing application specific knowledge provide a means to augmenting throughput, but have limitations due to their lack of adaptation and reliance on application specific decisions. We therefore propose a novel aggregation scheme that adaptively performs application independent data aggregation in a time sensitive manner. Our work isolates aggregation decisions into a module that resides between the network and the data link layer and does not require any modifications to the currently existing MAC and network layer protocols. We take advantage of queuing delay and the broadcast nature of wireless communication to concatenate network units into an aggregate using a novel adaptive feedback scheme to schedule the delivery of this aggregate to the MAC layer for transmission. In our evaluation we show that end-to-end transmission delay is reduced by as much as 80% under heavy traffic loads. Additionally, we show as much as a 50% reduction in transmission energy consumption with the addition of only 2 bytes of header overhead per network unit. Theoretical analysis, simulation, and a test-bed implementation on Berkeley's MICA motes are provided to validate our claims.

1. Introduction

Wireless Sensor Networks have emerged as a new information-gathering paradigm based on the collaborative effort of a large number of sensing nodes. In such networks, nodes deployed in a remote environment must self-configure without any *a priori* information about the network topology or global view. Nodes will act in response to environmental events and relay collected and possibly aggregated information through the formed multi-hop wireless network in accordance with desired system functionality. The inherently dynamic and distributed behavior of these networks, coupled with inherent physical limitations such as small instruction and data memory, constrained energy resources, short communication radii, and a low bandwidth medium in which to communicate, make developing communication protocols difficult.

Research on hardware for such devices has taken place at Berkeley [14][30][32] and various other research institutions [24] throughout the world. Using such hardware as a basis for development, the software architecture and communication stack residing on these devices has prompted prolific research in the areas of ad-hoc networking [10][15][17][20], data aggregation [16][21][26], cluster formation [25], distributed services [22], group formation [6], channel contention [3][5][7][19], power conservation [4][12], and much more. Work on the utility of such an innovative technology has unearthed potential applications including environmental monitoring, event tracking [1], disaster relief, and search and rescue.

In this work we address the problems of low bandwidth and energy limitations inherent to sensor devices. These networks' ever-changing and unpredictable state demands a self-configuring, adaptive elixir. Our solution is a novel adaptive application independent data aggregation (AIDA) component that fits seamlessly into the current sensor network communication stack. Our goal is to maximize utilization of the communication channel (single frequency) with energy savings coming as an ancillary benefit. With significant costs incurred from channel contention, packet header overhead, and data padding for fixed sized packets, this work abates

such costs by employing varying degrees of data aggregation at forwarding nodes in accordance with current local traffic patterns.

Data aggregation techniques have been extensively investigated in recent literature. Our work, as a novel data aggregation approach, distinguishes itself from current state of the art solutions in three respects. First, all prior Application Dependent Data Aggregation (ADDA shown in Figure 1b) relies on application layer information and must have a bi-directional interface, and therefore dependence with, the data centric routing protocol implemented. AIDA isolates aggregation decisions from application specifics by performing adaptive aggregation in an intermediate layer that resides between the traditional data-link and network layer protocols (Figure 1.a). AIDA is totally transparent to other layers and can be swapped into the network stack without modifying any existing interface. Second, no prior work in data aggregation adapts itself to the traffic situation in a time sensitive manner. AIDA takes the timely delivery of messages as well as protocol overhead into account to adaptively adjust aggregation strategies in accordance with assessed traffic conditions and expected sensor network requirements. Simulation results show that AIDA can adapt to varying traffic situations and dramatically reduce network congestion and transmission energy consumption**.** Third, previous data aggregation schemes (e.g., data centric routing [16]) perform in-network processing to reduce the amount of application data transmitted. These in-network processes (e.g. averaging) can achieve higher degrees of aggregation; however data are less available to the application (e.g. standard deviation of the data set can not obtained from the average). In contrast, AIDA performs loss-less aggregation allowing the upper layer to decide whether information compression is appropriate at the time. Very important, our design enables AIDA to remain complementary to other data aggregation strategies (Figure 1.c) while providing significant timesaving benefits in the lower layers of the communication stack.

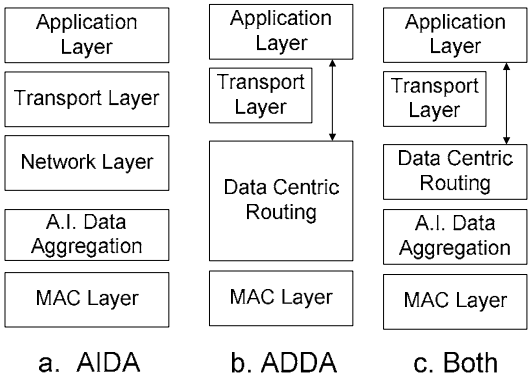

Figure 1: Architectural Designs

This paper attempts to address the aforementioned problems through a novel adaptive time sensitive data aggregation component. As an introduction to sensor networks, and to provide a more in depth discussion of the type of research taking place within this field, we begin section 2 with a discussion of related and ongoing work. Section 3 addresses the need for adaptation, data aggregation, and real-time data delivery. Section 4 then presents specific details about our protocol. Sections 5 and 6 describe our simulation environment, the type of experiments run, and a discussion of the results we obtain in both simulation and in the Berkeley MICA test-bed. Finally, we conclude in Section 7.

2. Leveraging Previous Work

Efforts to maximize channel utilization have been spread across various layers of the sensor network communication stack. Starting at the MAC layer, these include attempts to minimize collisions through contention-based mechanisms designed for a lossy wireless medium. Such work includes 802.11 [3], MACA [19], MACAW [5], FAMA[7], S-MAC[29], and Multi-Hop Scheduling [18], to name a few. All of these solutions reside within the data-link layer of the communication stack and, therefore, can coexist with the higher layer aggregation component we provide.

Similar to the data link layer the network layer, and more specifically the routing component, has brought about significant efforts to avoid congestion and maximize use of the communication medium. Such schemes

include distributing the traffic load to route around congestion [10] and using a minimal hop path to reduce the total number of transmissions [27]. Beyond the routing layer the communication stack in sensor networks becomes more amorphous. Clustering [25], group formation [6], and other higher layer hierarchical components serve to combine node responsibilities and come to consensus on what data to send. Often such information is application specific and must rely on a general understanding of exactly what the network is tasked to do. Additionally, the hierarchical and grouping components often utilize various forms of data aggregation through consensus algorithms or other forms of local processing.

Basic schemes [16] for the aggregation of data include the Center at the Nearest Source (CNS), where data is aggregated at the source nearest to the destination; Shortest Path Trees (SPT), where data is sent along the shortest path from source to sink and aggregated at common intermediate hops along the way; and Greedy Incremental Trees (GIT), which builds an aggregation tree sequentially to merge paths and provide more aggregation opportunities.

An extremely popular data aggregation scheme for sensor networks, Directed Diffusion [15][11], is a data-centric architecture where named (application specific) data gets propagated along paths back to the requestor. Effective paths are reinforced as they are used to optimize communication from point to point. Specifically designed for sensor networks, Directed Diffusion aggregates data along these reinforced paths to reduce the quantity of data transmitted across the network. Similarly Data Placement [26] is designed for applications where multiple sinks coexist and use in-network caching to update and distribute data to leaf nodes at the minimally requested rate. LEACH [12] is a high layer protocol that provides clustering and local processing to aggregate sensor data and reduce global communication. Many other data aggregation schemes exist that also provide network, transport, and application level mechanisms taking advantage of application specific knowledge about the data in question. All of these schemes reside either at or above the network layer and are orthogonal and can coexist with our work.

Aggregation scheme comparison studies have demonstrated the effect of network parameters and the utility of aggregation mechanisms in a wide variety of applications [16][21]. These studies discuss potential savings that aggregation can provide and are noted to explicate the potential for such work to improve network throughput.

To date, very few sensor network papers have addressed the need for incorporating adaptive behavior into their protocols. Sensor networks exhibit complex distributed behavior rendering static pre-configuration utterly useless as network traffic, often initiated by environmental events of interest, transitions from one extreme to another. Several protocols have taken a first stab at addressing the need for adaptive behavior in such dynamic networks. RAP [23] and SPEED [10] utilize locally available information to adjust priority levels or make more informed routing decisions in response to network congestion and changing traffic patterns. SPIN [13] makes adaptive decisions to participate in data dissemination based on current energy levels and the cost of communication. [30] uses adaptive rate control at the data-link layer to fine tune contention parameters in response to local traffic conditions. GAF [28] monitors network connectivity and turns nodes on/off to adapt network density for energy-conservation. While many more examples of online adaptation exist, these solutions provide relevant examples of how adaptation is beneficial in dynamic and unpredictable sensor networks and serve as a starting point to introduce adaptive behavior into these complex systems.

In addition to maximizing channel utilization and adapting to dynamic network conditions, energy conservation has become a central focus in sensor network research. Similar to data aggregation, work in energy conservation for sensor networks has been considered at various levels of the communication stack. Aside from minimizing power consumption at the hardware level [24], MAC layer protocols developed for energy savings mostly take advantage of overhearing and scheduling to allow nodes to sleep while they are not transmitting or receiving messages [8][12][27]. At the network and routing layer, schemes work to minimize power along the transmission path [26], set routes according to the energy remaining at nodes along that path [31], and use mechanisms to save power through the distribution of messages among various paths from source to destination [10]. Finally, higher layer protocols that often incorporate routing semantics exist to form groups and rotate leadership responsibilities allowing non-leader nodes to sleep and conserve their energy [4]. Again all of these protocols involve layered decisions that should adhere to strict modular programming interfaces allowing our work to coexist with them.

3. Analysis of the Problem

Various studies of throughput and channel utilization for wireless ad hoc networks have identified the limits of sensor networks with respect to asymmetric channel, multi-hop interference, high traffic density, and unpredictable communication patterns. To minimize such problems, mechanisms for contention have been introduced to notify neighbors of a node's intention to send a message. While such mechanisms have proven effective in minimizing collisions and, therefore, make better use of the channel, the overhead involved in sending control messages remains significant. Aside from control overhead incurred during handshaking, additional idle time is spent listening to the channel and backing off to determine when it is appropriate to initiate channel contention. Such properties create ample opportunity for improvement.

If it is possible to reduce the number of control messages sent while still distributing information about a node's communication intentions, it would save significant time and energy by reducing the total number of messages and time spent contending for the channel. One mechanism for achieving such a feat is through application dependent data aggregation (ADDA). The merging of data that maintain common properties (semantics) and are destined for the same node has been a common approach to reducing traffic. While such mechanisms have proven effective in reducing traffic and easing congestion, several issues that limit the extent to which they are applicable provide us with insight into developing an application independent aggregation (AIDA) mechanism.

- Due to the nature of application specific aggregation, such mechanisms require the appropriate naming of data and require that lower level protocols performing such aggregation have knowledge of these naming semantics. We therefore seek a solution without such cross-layer dependencies.

- By placing a naming restriction on aggregated data you limit the types of data that can be aggregated. For example, if certain messages contain temperature information while others contain acoustic data, there is no way to combine such data when aggregation is performed in an application aware manner. We desire a solution that allows us to aggregate traffic originating at various application protocols without any knowledge of the application that generated this data.

- Pervious aggregation schemes combine application specific data through consensus algorithms, averaging functions, or by some other mathematical manipulation of data, resulting in a loss of information. Because such schemes bind algorithms to the application and make it difficult to control the degree of information loss we seek a solution that performs lossless aggregation in a more general context.

- To properly aggregate named data from a common source, one must associate both location and time to that data to ensure that information is not lost or inappropriately merged. For example, reports on temperature from the northeast corner of a network should not be combined with temperature reports from the southwest corner just because they share a common type. Any aggregation performed must therefore be time and direction sensitive to ensure that data received at the requester remains meaningful.

- Current aggregation schemes assume that more aggregation is always better. As sensor network traffic changes, there exist times when varying degrees of aggregation are necessary to optimize communication and augment throughput. However at other times aggregation simply acts to delay data transmission and can be problematic. We therefore require a solution that takes current network traffic conditions into consideration when making aggregation decisions to adaptively optimize bandwidth while minimizing system energy consumption.

Application dependent data aggregation (ADDA) schemes have proven to be effective solutions for sensor networks. However, given the limitations inherent to such schemes, we seek a value-added solution that adapts to changing network conditions, improves the networks use of bandwidth, is simple and fast, has limited over-

head, performs aggregation without loss of information, and considers the timeliness of end-to-end traffic. In addition, we require a solution that performs aggregation transparent to other components. This will allow AIDA to work with, or exist independently of, other communication protocols so that AIDA can leverage the performance and overcome the limitations inherent to existing ADDA schemes.

4. Protocol Design

Our solution is an aggregation layer module that resides between the data-link and networking layer to aggregate packets through network unit concatenation. The aggregation component combines network units into a single outgoing AIDA payload to reduce the overhead incurred during channel contention and acknowledgment. No semantics of the data in the network units is used. Aggregation decisions are made in accordance with an adaptive feedback-based packet-scheduling scheme that dynamically controls the degree of aggregation in accordance with changing traffic conditions.

4.1. AIDA Architecture Design

The basic design of AIDA is shown in Figure 2. We separate AIDA functionality into two components. One is the functional unit that aggregates and de-aggregates network packets (units). The other is the AIDA Aggregation Control Unit, employed to adaptively control timer settings and fine-tune the desired degree of aggregation.

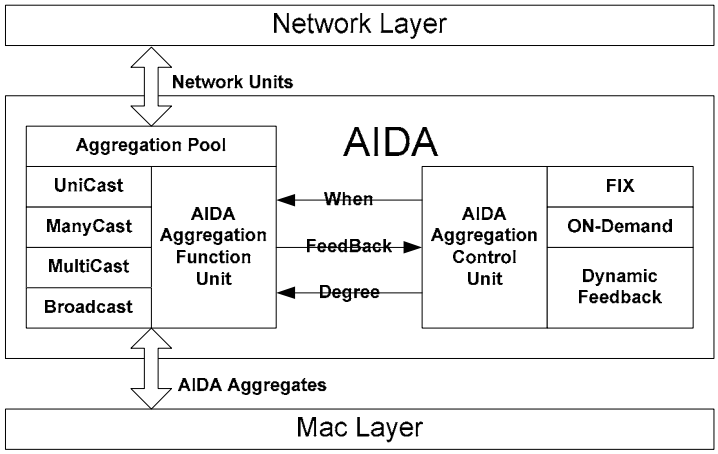

Figure 2: AIDA Components

The protocol works as follows: Packets from the network layer are placed into an aggregation pool. According to the number of packets to be concatenated in one aggregate and the next-hop destinations of those packets, AIDA's Aggregation Function Unit chooses one of four AIDA packet formats (Described in depth in section 4.3) to build an aggregate and passes this aggregate down to the MAC layer for transmission. The decision of how many packets to aggregate and when to invoke such aggregation is left up to the AIDA Aggregation Control Unit, a feedback based adaptive component which makes on-line decisions based on local current network conditions.

Similar to outgoing traffic, incoming traffic is received at the MAC layer and passed up to AIDA. Within AIDA the incoming aggregates are re-fragmented into their original network units of which each piece of the aggregate is passed up to the network layer for re-routing or application de-multiplexing and delivery. Although we acknowledge that many aggregates may be bound for the same ultimate destination (it could be more efficient not to de-aggregate and re-aggregate at every intermediate node), we perform such de-aggregation to ensure the modularity of layers and allow the networking component to determinate routes independently for each network unit.

The aggregation of multiple network units into a single AIDA aggregate for transmission reduces the overhead of channel contention (wait/backoff) and the transmission overhead of control packets (such as

RTS/CTS/ACK in 802.11 [3], RTS/CTS in MACAW [5], ACK in regular reliable MAC) so that these costs are incurred once per aggregate. By increasing the number of network units combined into a single AIDA aggregate (referred to as the degree of aggregation [DOA]), we are able to save [DOA – 1] * [contention time] msec on each transmission.

While the aforementioned AIDA function unit is straightforward, it is an intricate research problem to design an adaptive AIDA control unit to set appropriate timing and DOA parameters online. As we show in our evaluation section, different control schemes do have a huge impact on system performance. More detail on these control schemes are provided and discussed in section 4.2.

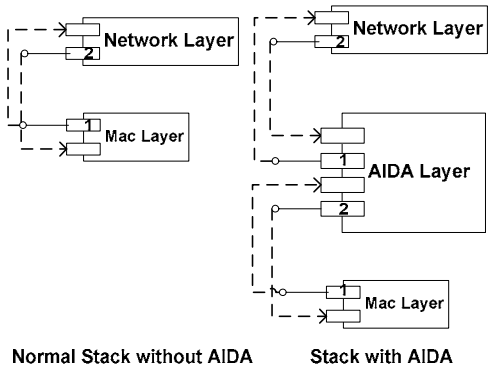

Figure 3: AIDA Implementation Design

To keep AIDA transparent from other protocol layers, we use a *delegation* approach to intercept all function calls between the MAC and Network layer. The networking component assumes it is talking directly to the MAC layer and vice versa. Using this method, our data aggregation layer imitates the interfaces exposed by both the MAC and Networking layer. We then turn off delegation to acquire baseline data in our tests. The stack resulting from this technique appears in Figure 3.

4.2. Aggregation Schemes in AIDA control Unit

To better understand the effect of aggregation and our success in building an adaptive solution, we design, implement, test, and compare several versions of AIDA. Versions of our architecture include FIX, On-Demand and Dynamic Feedback schemes. These schemes range from aggregation decisions based on static thresholds to our ultimate solution that incorporates a dynamic online feedback control mechanism into our protocol. A baseline without aggregation is also provided for comparison. Details of these implementations are provided in this section.

4.2.1. No Aggregation

With no aggregation (the baseline scheme), we simply employ the normal network stack without modification passing packets directly from the network protocol to the MAC protocol and vice versa.

4.2.2. Fixed Scheme

In the fixed scheme (FIX), AIDA aggregates a fixed number of network units into each AIDA payload ($DOA = N_{fixed}$). When this fixed number of network units has been aggregated, the AIDA payload is passed down to the MAC layer for transmission. To ensure that network units don't wait an indefinite amount of time before being sent, we also incorporate a timeout value (T_{fixed}) into this scheme to ensure that aggregation is performed, regardless of the number of network units, within some time threshold. The design of the FIX scheme is shown in Figure 4.

4.2.3. On-Demand Scheme

To prevent unnecessary per hop delay, our On-Demand scheme monitors the AIDA output queue to ensure that there is always an AIDA payload resident for MAC layer dequeing and transmission. When the MAC is available for transmission, no network units will be held back by the AIDA layer in an attempt to achieve a higher DOA. AIDA layer data aggregation only takes place when time is available (the outbound message queue has built up or the medium is busy preventing the MAC layer from accessing the channel). This scheme provides virtually transparent aggregation without incurring message delay costs. The inner works of the On-Demand scheme is shown in Figure 5. It is worth noting that the On-Demand Scheme is a reactive solution, where passive measures allow the DOA to dynamically change with varying traffic patterns. When there is little traffic, the outbound message queue rarely builds up and no aggregation is performed. As traffic increases, the length of the outbound message queue increases resulting in a proportional increase in the DOA.

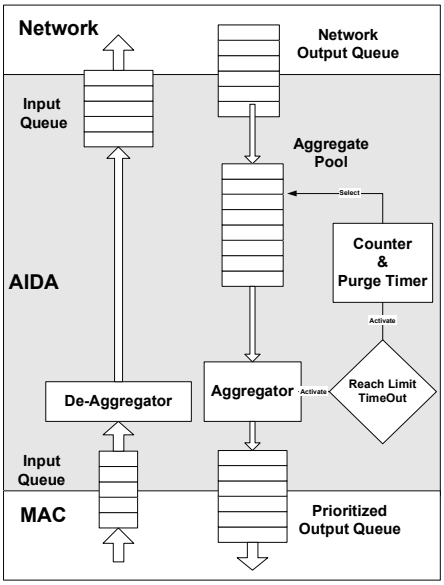

Figure 4: AIDA FIX scheme

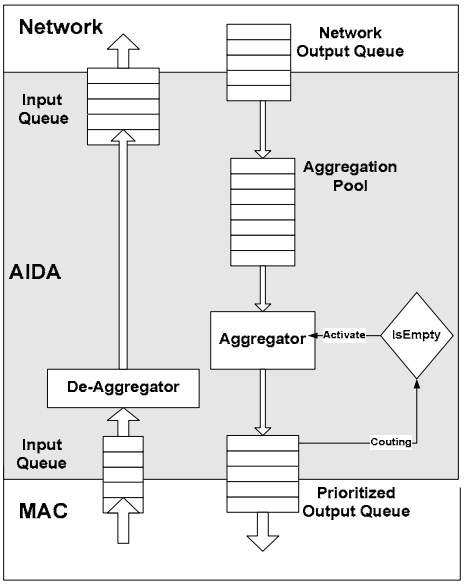

Figure 5: AIDA On-Demand scheme

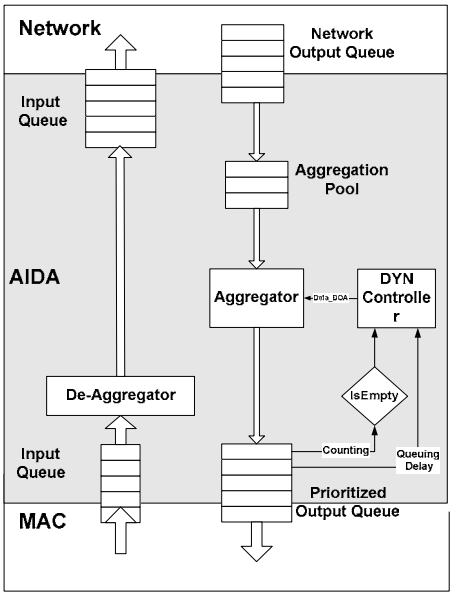

Figure 6: AIDA Dynamic Feedback scheme

As shown in Figure 5, the On-Demand scheme only requires simple monitoring logic to test whether the outbound message queue is empty or not. This simplicity of code is preferable for a constrained sensor node. Moreover, with such a simple design, the On-Demand scheme can provide much better performance during changing traffic patterns in comparison to the FIX scheme. We demonstrate this in our forthcoming evaluation.

4.2.4. Dynamic Feedback Scheme (DYN)

Our ultimate solution, the Dynamic Feedback scheme (DYN), implements a combination of on-demand and fixed aggregation where the DOA threshold (N_{DYN}) is adjusted dynamically. As shown in Figure 6, the scheme works by monitoring the AIDA output queue to determine its availability while also collecting data on the queuing delay imposed on AIDA payloads awaiting transmission. Using this information and operating under the basic premise of control theory, our aggregation mechanism dynamically adjusts the degree of aggregation ($DOA=N_{DYN}$). This scheme begins with N_{DYN} set to one. In the case of low network traffic, DYN will default to the On-Demand mechanism delivering packets to the MAC transmission queue as soon as they are ready. As network traffic builds and time contending for the channel delays transmission, our feedback loop adjusts our admission threshold (N_{DYN}) to allow a greater degree of aggregation prior to sending.

Feedback Control Design:

Intuitively, an algorithm based on heuristics rather than theoretical foundations can be used to adjust the DOA values to affect the MAC layer delay a packet experiences. When the MAC delay increases, the DOA threshold increases to lower the feeding rate to the MAC layer. As a result, fewer nodes participate in channel contention leading to a lower MAC delay. However, since heuristic feedback control lacks knowledge of system dynamics, it is subject to over or under reaction and cannot adapt to the system well. This warrants the development of an analytical model to reveal the dynamics between DOA values and the MAC layer. Such a model serves as a guide for developing an appropriate feedback controller.

A general form for calculating MAC delay can be defined as

$$D_{mac}(k) = D_{minium} + \#collisions(k) * D_{reslove} \quad (1)$$

where $D_{mac}(k)$ is the MAC delay packets experience during time period $[k,k+1]$, $D_{minimum}$ is the MAC delay when no back-off or collision is experienced and it is regarded as constant, $\#collisions(k)$ is the number of collisions a successful transmission will encounter at time interval $[k,k+1]$, and $D_{reslove}$ is the collision delay plus the time to resolve a single collision, also considered to be a constant.

Assume at a certain time interval $N(k)$ packets from different sensor nodes are ready for transmission. Statistically, AIDA will pass down only an average of $N(k)/DOA(k)$ packets to actively compete for the channel. $DOA(k)$ here is the average DOA values of the all nodes who compete for the channel. We denote the probability of a packet being transmitted at this time slot by the symbol $\tau(k)$. This $\tau(k)$ value is a function of the type of MAC protocol and current traffic scenario. A transmitting packet encounters a collision when at least one other packet from the remaining $N(k)/DOA(k)$-1 packets chooses the same time slot. Accordingly, the average collision probability P can be calculated as

$$p = 1 - (1 - \tau(k))^{N(k)/DOA(k)-1} \quad N/DOA \geq 1 \quad (2)$$

Naturally, the average number of transmissions required for each successful transmission is

$$E(\#collsions + 1) = \frac{1}{(1-p)} \quad (3)$$

Substituting (2) into (3) gives us the expected number of collisions each successful transmission will encounter.

$$E(\#collsions) = \frac{1}{(1-\tau(k))^{N(k)/DOA(k)-1}} - 1 \quad N/DOA \geq 1 \tag{4}$$

Combining (1) and (4) then gives us the approximate correlation between the DOA values and the MAC layer delay

$$D_{mac}(k) = \left[D_{\min imum} - D_{reslove}\right] + D_{reslove}(1-\tau(k))^{1-\frac{N(k)}{DOA(k)}} \tag{5}$$

Since D_{minium}, $D_{reslove}$ and $\tau(k)$ are independent of DOA values, we calculate the differential of equation (5) and get the small-signal model for the system:

$$\begin{cases} D_{mac}(k+1) = D_{mac}(k) + \dfrac{\lambda_1}{DOA(k)^2}\lambda_2^{\frac{1}{DOA(k)}}\Delta DOA(k) \\ DOA(k+1) = DOA(k) + \Delta DOA(k) \end{cases} \tag{6}$$

$$where \quad \lambda_1 = D_{reslove} * (1-\tau(k))N(k)(Ln(1-\tau(k)), \quad \lambda_2 = (1-\tau(k))^{-N(k)}$$

Because λ_1 and λ_2 are independent of the DOA, they can be considered constant in the vicinity of a small signal control model. Note that the goal of this approximate model (6) is not used to precisely calculate MAC delay under different DOA settings, but is used to design our controller. A tailored model can be established by derivation on the values of λ_1 and λ_2 based on particular properties of the chosen MAC protocol. However, for the sake of MAC-independence, we design a general form for our controller in accordance with equation (6) as follows.

$$\Delta DOA(k) = G(k) * e(k) \tag{7}$$

$$where \quad G(k) = P_{DOA} * DOA(k)^2 \quad e(k) = (D_{mac}(k) - D_{\min imum})$$

In equation (7), P_{DOA} is an implementation parameter to set the gain between the changes of DOA and the error in MAC delay control.

The pictorial notation of this control loop is shown in Figure 7.

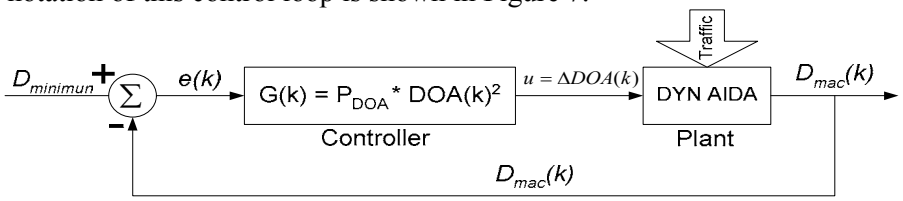

Figure 7: The control loop for the DYN AIDA scheme

As we will show in the evaluation, the current adaptive controller works best under a wide range of traffic scenarios under investigation.

4.3. AIDA Function Unit

The AIDA Aggregation Function Unit is responsible for the aggregation and de-aggregations of network units. This component builds four different types of aggregates, namely *Unicast, Manycast, Multicast* and *Broadcast*, in accordance with the set AIDA parameters and current state of the module.

- If there is only one network unit ready when the AIDA Control Unit is ready to aggregate (e.g. a time out occurs), the AIDA Function Unit will use Unicast to send the waiting unit out to the specified neighboring node. In this case, no aggregation is performed.

- If all network units to be aggregated are targeting the same next-hop node, AIDA sends out an aggregate using Manycast with the target specified.
- When network units to be aggregated have different next-hop addresses, the slightly more complex Multicast type is used to take advantage of the broadcast nature of wireless communication. In this case, AIDA merges network units, regardless of which neighbor each network unit targets, into a single aggregate and uses the MAC broadcast address as the destination. Every neighbor of the sending node will receive and de-aggregate this Multicast packet to determine whether or not a portion of the aggregated payload was destined to it.
- Finally, the Broadcast type of AIDA is used in the case where all aggregated network units are Broadcast messages.

Although a single packet format (Multicast) is logically enough to support all of the aforementioned scenarios, we argue that tailored packet formats for each scenario can reduce the AIDA header size and save bandwidth. These savings are beneficial in a resource constrained sensor network justifying the small amount of complexity added through AIDA typing.

4.4. Packet Format Details

Like most communication stack layers, AIDA adds meta-information to a packet in the form of a header. This header defines the aggregation format used for later de-aggregation, de-multiplexing, and seamless delivery to the appropriate network layer protocol. This header is placed in front of all aggregated network units and is included in the AIDA data units passed down to the MAC layer for transmission. Upon delivery at a node, the AIDA header can then be used to validate the specific aggregation mechanism used (in the case where multiple aggregation options are provided), assess the structure of the AIDA payload for de-aggregation, and potentially break apart, de-multiplex, and deliver each network unit to the appropriate network layer module.

The general form of the AIDA header is provided in Figure 8. It should be noted that some fields inside this general form are not used for different AIDA payload types.

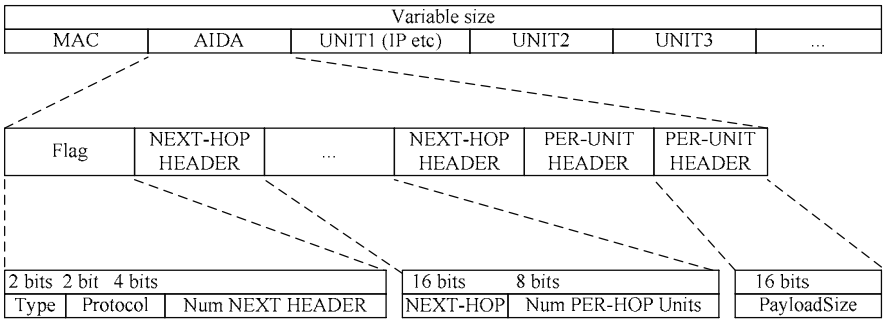

Figure 8: AIDA General Header format

4.4.1. FLAG for All Types

The first component of the AIDA header is an eight bit (1 byte) flag specifying information relevant to all aggregated network units. The Flag is composed of a Type field (2 bits), a protocol field (2 bits), and the number of Next Headers (4 bits).

- **Type Field**: The Type bits are used to specify whether the AIDA packet should be treated as a *Unicast*, *Manycast*, *Multicast*, or *Broadcast*.
- **Protocol Field:** The Protocol field (2 bits) of the AIDA Flag denotes to which network layer AIDA should de-multiplex network units.
- **Num-Next-Header Field**: The Num-Next-Header (4 bits) denotes how many headers follow. For Unicast, Manycast and Broadcast traffic, this field is set to the number of network units inside this

aggregate. For Multicast traffic this field will contain the number of neighbors receiving portions of this aggregate.

4.4.2. NEXT-HOP Header for Multicast Type

The NEXT-HOP header is only used by Multicast AIDA packets. Each header contains an ID specifying the intended recipient followed by the number of network units contained in this aggregate that are destined for the specified neighbor. In the case of Unicast, Manycast or Broadcast AIDA payloads, there is no need to differentiate between receiving nodes so this NEXT-HOP header is not used.

- **Next-Hop Field:** The Next-Hop field (2 bytes) of the NEXT-HOP Header contains a locally unique identifier of the node receiving a specified number of network units.
- **Num PER-HOP-UNITS Field:** The Num PER-HOP-UNITS field is an 8 bit (1 byte) field that identifies the number of aggregate network units that are destined for the neighbor specified in the Next-Hop Field.

4.4.3. PER-UNIT Header

PER-UNIT header is used during de-aggregation for delimiting the boundaries between network units. It consists of a 16 bit (2 byte) field that specifies the size of each network units. In the Unicast case there is no boundary to be identified, so the PER_UNIT HEADER is not used.

4.5. AIDA Header Overhead Analysis

From the structure of our AIDA header it is simple to assess the overhead incurred during the aggregation of network layer units. For comparison, the basic packet structure with and without AIDA is shown in Figure 9. With one AIDA Flag, one NEXT-HOP Header per applicable neighbor, and one PER-UNIT header per aggregated network unit, the overhead of AIDA is as follows:

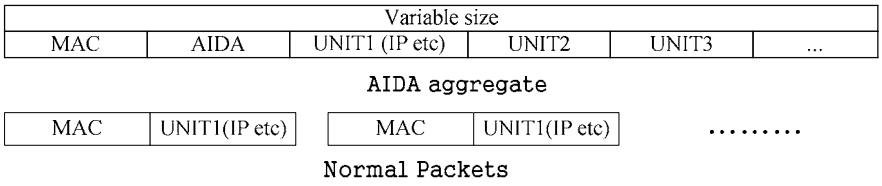

Figure 9: Format Comparison

- Unicast only uses the Flag field and therefore incurs a single byte of overhead per network unit.

- Besides the 1 byte flag, Manycast and Broadcast packets need use a per-unit header to delimitate the boundaries of multiple network units, thus incuring an average of (2+1/N) bytes overhead per network unit (where N is the number of network units aggregated into an AIDA payload).

- Because multiple next-hop node addresses need to be differentiated, Multicast payloads have a slightly larger overhead on the average (2+1/N+3/M) bytes per network unit (where N is the same as before and M is the average number of network units for each next-hop node),

It should be noted that AIDA aggregates several network units into one MAC payload; thereby it actually reduces the total header overhead per network unit. For example, in 802.11 the MAC header length is 28 bytes. To send out N network units without AIDA, the total header overhead would be 28*N bytes. Using AIDA we reduce the total header overhead to 28+N*(2+1/N+3/M) bytes. As long as the value of N (the DOA) is greater than 2, AIDA effectively reduces the total packet overhead incurred during transmission.

4.6. AIDA Savings Analysis

Adding header information to any transmission will intuitively increase transmission time for a single packet. We therefore only see savings in per transmission overhead costs when aggregating multiple upper layer payloads into a single transmission. By analyzing our AIDA header structure, we can see that savings differs for Unicast, Manycast, and Multicast transmissions. To better understand the potential benefits of aggregation, and to compare differing levels of aggregation under different traffic patterns, we provide a theoretical analysis to assess overhead with respect to transmission time. The analysis presented assumes optimal aggregation to the specified DOA without incurring any additional cost waiting for network layer payloads. We also assess savings without considering collisions and backoff, two factors that will ultimately increase the utility of AIDA.

The cost of packet transmission in the simple single sender, single receiver scenario with no channel contention and an arbitrary MAC layer is the time consumed by the MAC acquiring and setting up each transmission plus the time for sending the message, all multiplied by the number of individual transmissions. To maintain MAC layer independence, we simply assign the variable *M*, to the time (in msec) for performing MAC layer transmission preparation. For an 802.11 like MAC, this cost includes the channel sense, RTS, CTS, ACK, and intermittent wait times between control packets. For network units of size *S* transmitted at *R* bytes/second, the AIDA header overhead *O* (in bytes), and *DOA* number of packets aggregated. The cost C_{AIDA} (in msec) can be calculated with equation (8)

$$C_{AIDA} = M + (S * DOA + O) * R \tag{8}$$

compared with the cost of sending DOA number of packets without our aggregation scheme C_{None} (equation 9)

$$C_{None} = (M + S * R) * DOA \tag{9}$$

Hence, the saving in percentage is calculated as following:

$$SavingPercentage = (\frac{C_{None} - C_{AIDA}}{C_{None}}) = \left[1 - \frac{S * R}{M + S * R}\right] - \frac{M + O * R}{M + S * R} * \frac{1}{DOA} \tag{10}$$

From equation 10, we can see that the savings increases as the DOA increases when the cost at the MAC layer (M) is non-negligible. To demonstrate the utility of AIDA, we graph theoretical savings for our scheme under an 802.11 like MAC contention scheme for a 200 Kbps channel. The AIDA payload is passed down to a simplified 802.11 MAC that performs idle listening, RTS/CTS handshaking, and follows up each DATA packet with an acknowledgment. The control packet size for our theoretical MAC is 11 bytes. Contention also includes 5 msec's of idle listening and the DIFS and SIFS intervals are chosen at 10 and 5 msec's respectively in accordance with the current MICA specifications. We graph variable size network units to better understand the effect of packet size on potential savings.

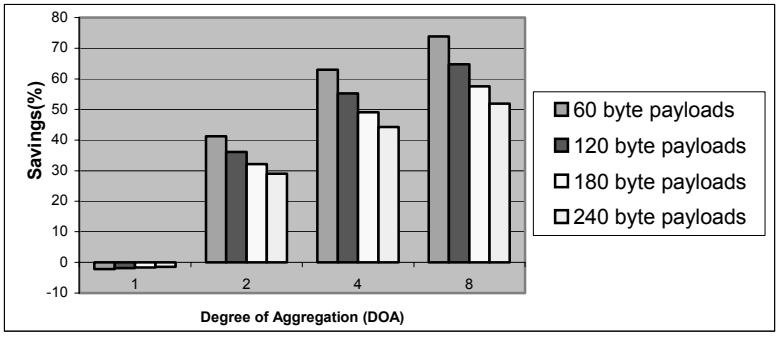

Figure 10: AIDA Theoretical Savings

Figure 10 demonstrates theoretical time savings as a percentage of the total time it would take to send the number of packets without AIDA. These savings are calculated by comparing the time to send a single AIDA aggregate, consisting of [DOA] network units with one MAC header, versus the time to send [DOA] separate packets without any AIDA header information or data aggregation performed. From this chart we can see that as the degree of aggregation increases, the percentage of savings in time increases drastically. We also note that as payload size increases, the relative time savings decreases. This occurs when data transmission time becomes a larger percentage of the total transmission time. Finally, we note that when AIDA fails to perform any aggregation as shown in Figure 10 when DOA = 1, the cost incurred is a single byte of data, which amounts to virtually no increase in transmission time.

5. Evaluation

We simulate AIDA in GloMoSim, a scalable discrete-event simulator developed at UCLA. This software provides a high fidelity simulation for wireless communication with detailed propagation, radio, MAC, and network layer components. Table 1 describes the detailed setup for our simulator. For our experiments the communication parameters are mostly chosen in accordance with Berkeley MICA mote specifications [32], the popular hardware platform on which sensor network research systems are currently deployed for testing. The current version of the MICA motes supports a 50kbps transmission rate and future versions are expected to provide higher than 1Mbps rates. Based on these considerations, we choose 200Kb/s as the effective bandwidth for our evaluation. Finally, we choose 802.11 as our MAC layer protocol, which has been implemented in a scaled down version on the MICA platform.

Routing	GF
MAC Layer	Simplified 802.11 DCF
Radio Layer	RADIO-ACCNOISE
Propagation model	TWO-RAY
Bandwidth	200Kb/s
Payload size	32 Byte
TERRAIN	(200m, 200m)
Number of Motes	100
Node placement	Uniform
Radio Range	40m

Table 1. Simulation settings

Since our work is the first we know of concerning data aggregation without utilizing application information, we evaluate our work based on different aggregation schemes we provide and a normal stack without aggregation support. In this evaluation we compare the performance of four schemes: No-aggregation, FIX, On-Demand, and DYN as previously defined. We show that DYN feedback is the best solution with better performance under all traffic scenarios tested.

In our evaluation, we analyze the following set of metrics: end-to-end delay, energy consumption, MAC control packets, degree of aggregation (DOA) and AIDA control overhead. These metrics are investigated under three sets of typical traffic patterns with a total of 72 different traffic loads, which allow us to access AIDA's adaptation capability under a wide range of traffic situations. Each plotted data point is the average of 10 runs generated from different random seed values. This ensured that 95% confidence intervals for our data are within 2~5% of obtained means. For legibility reasons we do not plot these confidence intervals in this paper. Full experimental data can be obtained from the authors upon request.

5.1. Work load Settings

We expect typical communication patterns inside a sensor network to be established based on request and retrieval semantics for data delivery between sensor nodes and a querying entity. One-to-one, many-to-one and many-to-many communication patterns are representative work loads in sensor networks. One-to-one commu-

nication happens when one sentry node detects some activity that needs to be reported to a remote entity. Alternatively, a query entity will require periodic reports from the whole sensor area, which take the form of many-to-one communication. It is more common that multiple applications run simultaneously and the traffic flows interleave with each other, which is a many-to-many cross-traffic pattern.

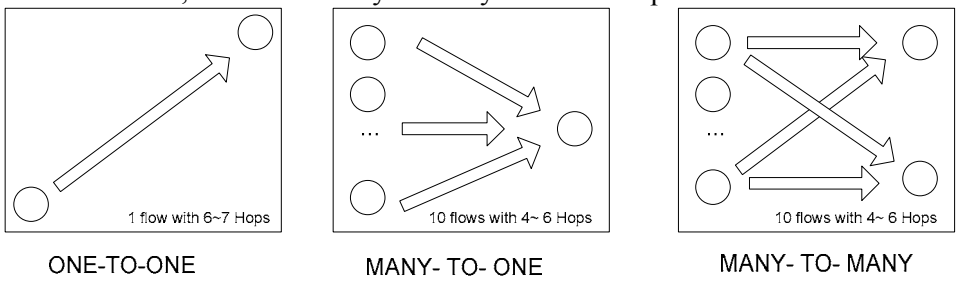

Figure 11: Traffic Load Settings

In our evaluation we focus on the aforementioned three representative communication patterns (Figure 11). To test the one-to-one scenario, we have a single node randomly placed on the left lower corner of our terrain send out a single CBR flow to the right upper corner of the terrain where the average route is approximately 6~7 hops. In the many-to-one scenario, 10 nodes on the left side of the terrain send out 10 CBR flows to the center-right side of the terrain where we place a single querying node. In many-to-many scenario, 5 nodes on the left side of the terrain send out 10 CBR flows (2 flows for each node) to the two querying nodes at the upper and lower right corner of the terrain, respectively. The sending rate of each CBR flow is incrementally increased to test the performance of AIDA under different traffic loads.

5.2. End-To-End Delay

5.2.1. End-to-end delay under different schemes

A major goal of the AIDA protocol is to achieve energy savings without jeopardizing end-to-end delay. AIDA not only doesn't add to the end-to-end delay, but in the presence of high degrees of aggregation, actually decreases end-to-end delay by reducing the number of control packets used at the MAC layer.

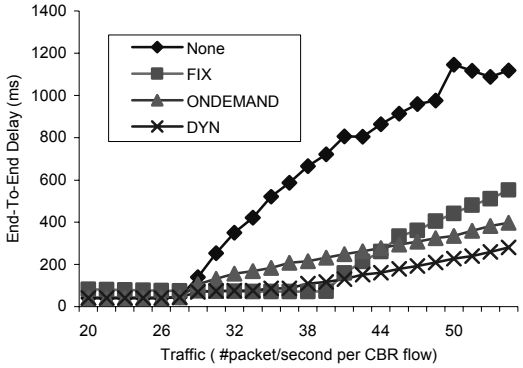

Figure 12: Avg E2E delay (one-to-one)

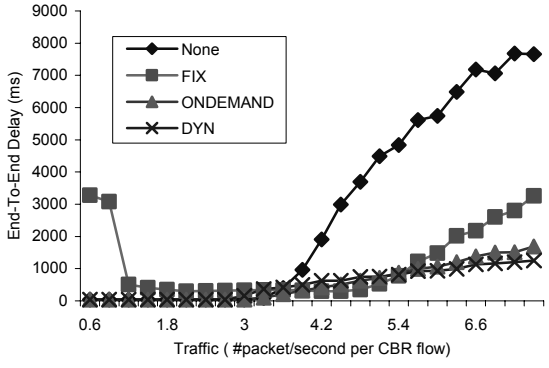

Figure 13: Avg E2E delay (many-to-one)

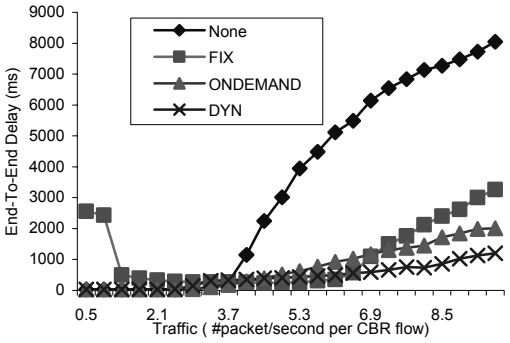

Figure 14: Avg E2E delay (many-to-many)

Figure 12, Figure 13 and Figure 14 graph end-to-end delay as a function of traffic loads under three traffic scenarios. These graphs show that end-to-end delay for CBR without performing aggregation increases dramatically as the overall traffic increases gradually. This is the typical case for multi-hop wireless networks where channel contention is much higher than in a single hop wireless LAN, As shown in figures, when traffic is low (e.g. below 3 packets/per flow in Figure 13), all schemes except FIX have very short end-to-end delay (abut 70~100ms). The reason for additional delay in FIX scheme is because the FIX scheme holds packets despite an available channel in order to obtain its specified degree of aggregation. The lower the sending rate is, the longer FIX scheme needs to wait. In contrast, the On-Demand and DYN schemes send out packets whenever possible, eliminating any additional end-to-end delay. On-Demand scheme performs well because of its reactive adaptive mechanism. The DYN scheme performs the best in all scenarios because it uses feedback control to dynamically adjust the required DOA and sending rate to the MAC when traffic is severely congested. In the presence of extremely heavy traffic, we show that DYN scheme is capable of reducing the end-to-end delay by as much as 80%, compared to non-aggregation case, when flow rate at 8.5 packets/second per flow (see Figure 14).

5.2.2. E2EDelay under different DOA setting for FIX scheme

In this experiment, we measure end-to-end delay for various traffic loads under different DOA settings in the FIX scheme. Figure 15 reveals the disadvantage of the FIX scheme and explains why dynamic adaptability is desired for such system. From Figure 15, we can see that there is no single DOA value that works well for every traffic pattern.

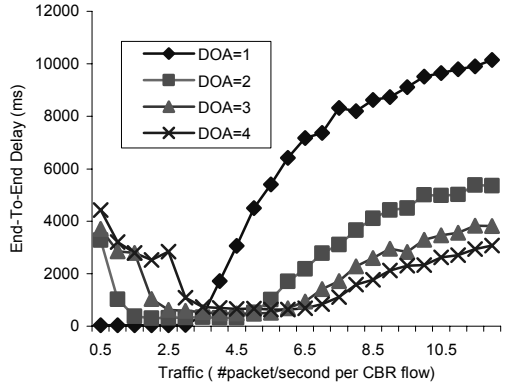

Figure 15: Avg E2E delay (many-to-one)

On one hand, a high DOA value setting doesn't perform well under low traffic loads. For example, only when the DOA is 1 (no aggregation is performed), no additional delay is incurred when the traffic load is as low as 0.5 packets/second per flow. The FIX scheme with DOA=4 introduces additional 4000ms delay under same traffic situation. The higher DOA settings tend to reduce congestion, but increase additional accumula-

tion delay in the AIDA component for packets waiting to be sent. On the other hand, low DOA value settings don't perform well under heavy traffic loads. For example, shown in Figure 15, the FIX scheme with DOA = 1 has nearly double the end-to-end delay as that with DOA=2 when the traffic is about 10 packet/second per flow.

The FIX scheme is insensitive to the traffic situations. To optimize for both light and heavy traffic, online adaptation is provided in On-Demand and DYN schemes, which can passively and proactively change the DOA value in accordance with these traffic patterns, respectively. Therefore, they exhibit a better overall performance as shown in Figure 12, Figure 13 and Figure 14.

5.3. Energy Consumption

5.3.1. Energy consumption under different schemes

With limited power resources, it is vital for sensor nodes to minimize energy consumption during radio communication to extend the lifetime of the sensor network. AIDA achieves such energy savings via several approaches. First, AIDA reduces MAC channel contention costs by distributing these costs across multiple network units. Second, by using less MAC control packets, AIDA dampens congestion and reduces the number of collisions resulting in fewer retransmissions. Finally, networking protocols designed for sensor networks usually adopt fixed packet sizes (e.g. TinyOS networking [14]), which leads to unnecessary padding costs. In our simulation with variable size support, AIDA take advantage of the first two approaches.

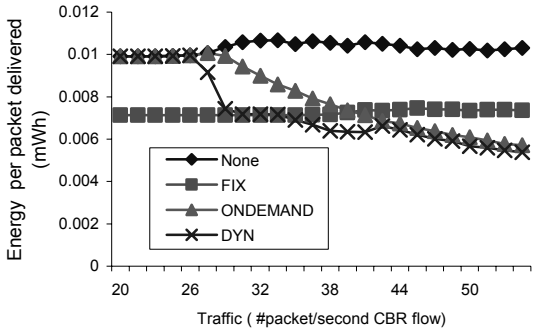

Figure 16: Energy per unit delivered (one-to-one).

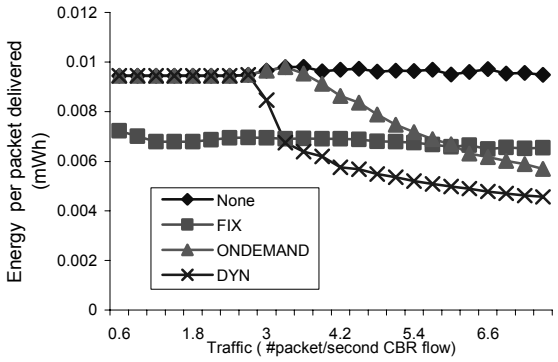

Figure 17: Energy per unit delivered (many-to-one)

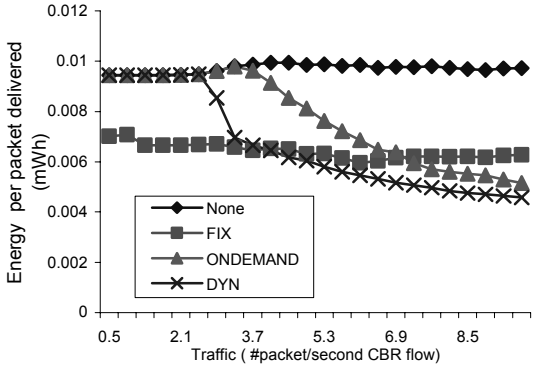

Figure 18: Energy per unit delivered (many-to-many)

In this experiment, we measure average transmission energy per delivered packet under 24 increasing traffic loads for three traffic patterns. In Figure 16, Figure 17 and Figure 18, our energy metrics show that the scheme without AIDA (None) demonstrates the worst performance. For example, None consumes double the energy as the DYN scheme when traffic load is about 6 packets/second per flow in Figure 18. the FIX scheme always aggregates 2 packets before sending which leads to nearly constant energy saving in both the low and

high traffic situation. However, in FIX, the DOA values are set and congestion levels are not taken into account resulting in worse performance than in DYN and On-Demand schemes under heavy traffic conditions. For example, shown in Figure 18, in DYN scheme, nodes consumes about 20% less energy per packet delivered as in FIX scheme, when traffic load is about 8 packets/second per flow.

5.3.2. Energy consumption under different DOA for FIX scheme

Figure 19 shows energy consumption per packet delivered for varying DOA's under the FIX scheme. This graph shows that for the FIX scheme, AIDA can achieve a higher percentage of energy savings by using higher DOA values. However, as we have shown in section 5.2, a higher DOA leads to additional delay when the network is lightly loaded, therefore taking end-to-end delay into account, it is not always beneficial to increase the DOA value.

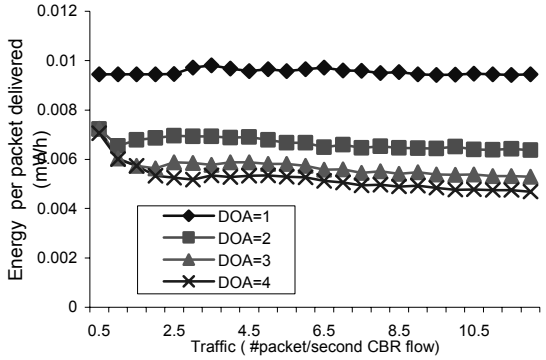

Figure 19: Energy per unit delivered (many-to-one)

5.4. MAC control packets

Even though our AIDA design is independent of any MAC layer protocol, it can reduce MAC overhead by sending longer, but less numerous payloads to the MAC layer for transmission. This reduces the number of channel access operations performed by the MAC. This section identifies the savings incurred through AIDA aggregation at the MAC layer. The data collected here is for the 802.11 MAC protocol although we would expect very similar results from other MAC protocols.

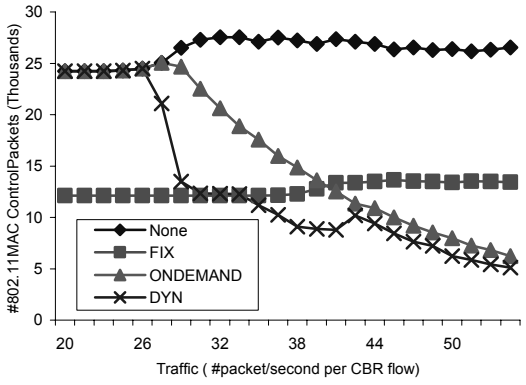

Figure 20: MAC control Packets (one-to-one)

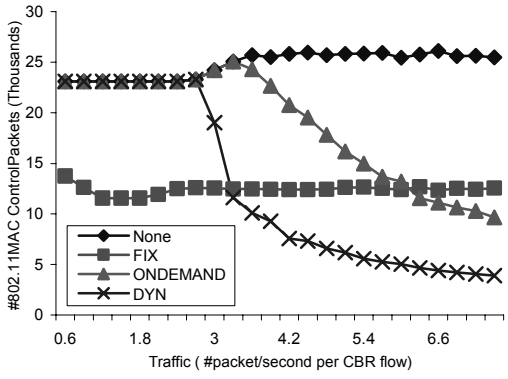

Figure 21: MAC control Packets (many-to-one)

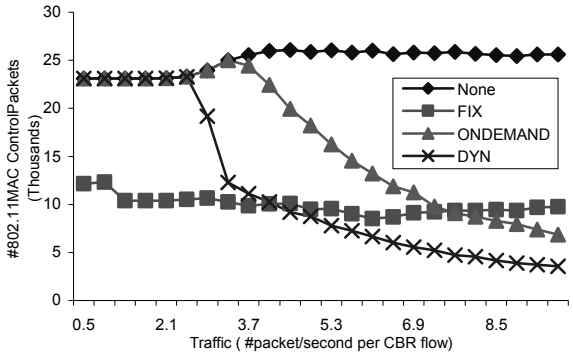

Figure 22: MAC control Packets (many-to-many)

Figure 20, Figure 21 and Figure 22 graph the number of control packets sent over various traffic loads. As shown in these graphs, the FIX scheme reduces the number of MAC control packets by approximately 50% when the DOA parameter is set to 2. On-Demand and DYN vary their DOA and therefore incrementally reduce MAC overhead as network congestion levels increase. For example shown in Figure 22, when per flow rate exceeds 9 packets/second, DYN only used about 20% of the control packets compared to the none-aggregation case. This leads to dramatically reduce congestion and energy consumption as shown in other evaluations.

5.5. Degree of Aggregation

As seen in the context of reducing MAC overhead, the degree of aggregation is a major indicator reflecting AIDA's ability to achieve energy savings and congestion dampening. Without aggregation , the DOA always equals one (e.g. None case in Figure 23). In the FIX scheme where DOA is set to 2, we can see a constant value for the degree of aggregation is achieved. In the On-Demand scheme, the DOA naturally follows traffic congestion levels. In DYN, the DOA is controlled by a feedback loop embedded inside AIDA.

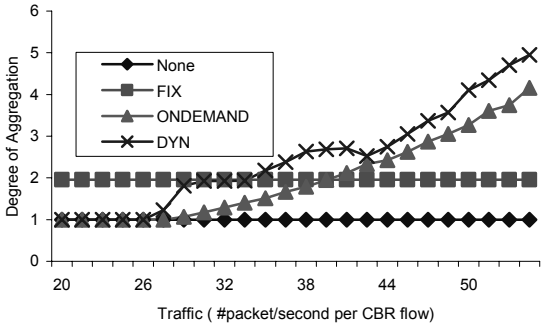

Figure 23: DOA (one-to-one)

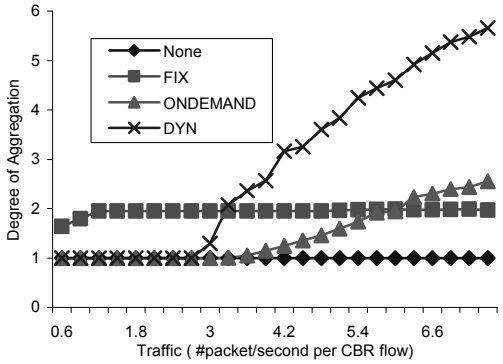

Figure 24: DOA (many-to-one)

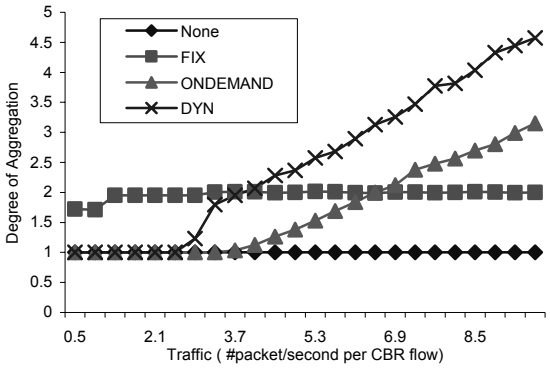

Figure 25: DOA (many-to-many)

Figure 23 , Figure 24 and Figure 25 graph the achieved DOA under various traffic conditions for the tested schemes. Figure 23 shows how DYN has roughly the same DOA value as the On-Demand scheme in the one-to-one pattern situation. However, in the more congested situations (Figure 24 and Figure 25), DYN achieves a higher DOA value than On-Demand resulting in more savings on channel bandwidth and energy consumption.

5.6. AIDA overhead

As shown in AIDA Header Overhead Analysis (section 4.5), AIDA's header overhead is about 3 bytes for Multicast packets, 2 bytes for Manycast, and 1 byte for Unicast and Broadcast. Figure 26, Figure 27 and Figure 28 graph per packet AIDA overhead under various traffic loads. As shown in Figure 27, under many-to-one conditions, FIX will send out only Manycast packets with its DOA value set to 2. This leads to an average of 2 bytes of AIDA header overhead. When the flow rate is very low (shown by the first two values for FIX scheme in Figure 27), the FIX scheme times out before it can reach its aggregation level of 2. When this happens the FIX scheme sends Unicast packets resulting in a smaller average AIDA overhead per network unit.

In one-to-one and many-to-one traffic patterns, AIDA uses Unicast when the network is not congested in order to avoid additional delay and Manycast when congestion is apparent. This is shown in Figure 26 and Figure 27 as congestion levels increase and the overhead approaches 2 bytes per header. In one-to-one and many-to-one traffic patterns, no multicast packets are sent out, explaining why AIDA overhead never exceeds 2 bytes per network unit.

On the contrary, in many-to-many situations, AIDA takes advantage of the broadcast nature of wireless networks, uses multicast packets to address multiple next-hop nodes in a single aggregation, which require 3 bytes overhead for each multicast packet. This is shown in Figure 28 where AIDA overhead is somewhere between 2 and 3 bytes for the FIX scheme.

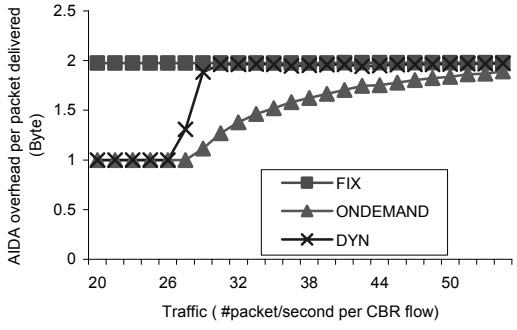

Figure 26: AIDA overhead (one-to-one)

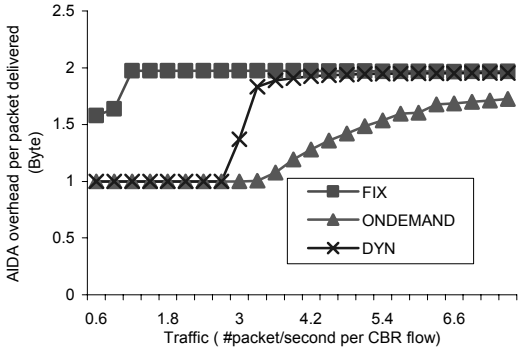

Figure 27: Aida overhead (many-to-one)

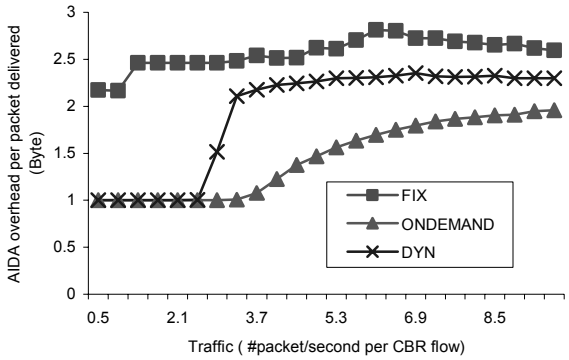

Figure 28: Aida overhead (many-to-many)

6. Implementation on the Berkeley Mote Test Bed

We have implemented the AIDA protocol on the Berkeley motes platform with a code size of 3,840 bytes (code is available at [9]). Three applications including data placement [26], target tracking [6], and CBR are built and tested on top of AIDA. Due to the physical limitation on the motes, it is extremely difficult to perform as extensive evaluation as we did in the wireless simulator. As a result, we only present partial results here as a study to better understand the effect of aggregation in developing a more complete adaptive solution. More detailed evaluation on upgraded versions of motes is left as future work.

In the experiment we use 25 motes to form a 5 by 5 grid. To evaluate the aggregation performance of AIDA we send three CBR flows from node 24 to node 0 (the requesting node). The experiment collects the number of packets relayed by intermediate motes (1~23) and compares this with the results obtained from a basic GF [20] protocol without AIDA. In most embedded designs, fixed packet sizes are supported for the sake of simplicity making padding costs large when sensor data payloads are small. AIDA takes advantage of this and aggregates multiple payloads into one packet to minimize padding costs. The savings achieved by AIDA are shown in Figure 29 graphing the number of packets sent at intermediate nodes under various DOA settings. We demonstrate that the transmission cost (packets sent) is reduced as the DOA value increases. For example, when the DOA value is 2, node 1 sends out nearly half as many packets as it did without aggregation. It is worth noting that with a fixed size packet, when the DOA reaches a certain value AIDA comes to a point where it cannot compact any more network units into the AIDA aggregate. For our experiment and payload size this occurred when the DOA was 5.

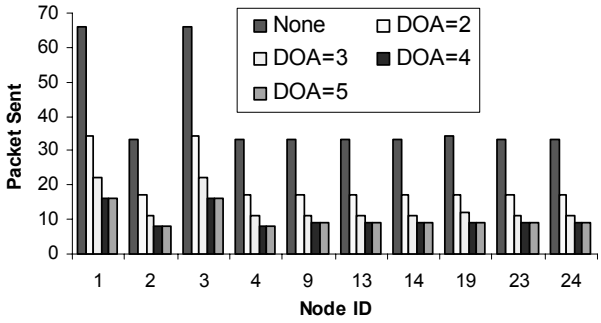

Figure 29: Packets Sent Under different DOA

7. Conclusion

In this paper we introduce AIDA, an adaptive application independent data aggregation mechanism for sensor networks. AIDA performs lossless aggregation by concatenating network units into larger payloads that are sent to the MAC layer for transmission. Due to the highly dynamic and unpredictable nature of wireless communication in sensor networks, a novel feedback-based scheduling scheme is proposed to dynamically adapt to changing traffic patterns and congestion levels. By isolating our work in a layer that sits between the networking and data-link components of the communication stack, AIDA is able to perform such aggregation without any necessary modifications or adjustments to upper or lower layer protocols. Moreover, very significantly, AIDA is a value-added compatible solution that can complement and augment the gain of application specific data aggregation (ADDA) schemes.

In our experiments we evaluate the performance gain achieved by AIDA. We show that by adaptively configuring our aggregation parameter (DOA), AIDA only introduces a small header overhead (around 2 bytes per network unit / negative overall header overhead) while reducing end-to-end delay by as much as 80% and transmission energy by 30~50% in heavy traffic conditions. As shown in our evaluation, AIDA running in the DYN (fully adaptive) scheme provides the best overall solution. The DYN feedback control loop dynamically tunes our DOA threshold and sending rate to optimize aggregation performance under varying traffic conditions by monitoring queuing delay to perform data aggregation without sacrificing end-to-end delay. The MAC control overhead is also reduced to allow for more efficient channel scheduling.

A physical implementation of AIDA on the Berkeley test bed provides initial evidence of the savings obtainable by an application independent aggregation scheme and pave the path for future implementations of our adaptive control based protocol.

8. References

[1] T. Abdelazaher, B. Blum, et. al. "EnviroTrack: An Environmental Programming Model for Tracking Applications in Distributed Sensor Networks". Technical Report CS-2003-02, University of Virginia 2003.

[2] M. Adamou, S. Khanna, I. Lee, I. Shin, S. Zhou, "Fair Real-time Traffic Scheduling over A Wireless LAN", In *Proceedings of the 22nd IEEE RTSS 2001*, London, UK, December 3-6, 2001

[3] ANSI/IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," ANSI/IEEE Std 802.11, 1999 Edition.

[4] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, Robert Morris "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks". In *Proc. of the 6th ACM MOBICOM Conf.,* Rome, Italy, July 2001.

[5] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. "MACAW: A Media Access Protocol For Wireless LAN's". In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 212-225, August 1994.

[6] Brian Blum, Prashant Nagaraddi, Anthony Wood, Tarek Abdelzaher, Sang Son, John Stankovic, "An Entity Maintenance and Connection Service for Sensor Networks," The *First International Conference on Mobile Systems, Applications, and Services* (*MobiSys*), San Francisco, CA, May 2003

[7] C. Fullmer and J.J. Garcia-Luna-Aceves, "Floor Acquisition Multiple Access (FAMA) for packet radio networks", *Computer Communication Review, vol. 25, (no. 4)*, ACM, Oct. 1995.

[8] C. Guo, L. C. Zhong and J. M. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks", In *Proceedings of IEEE GlobeCom 2001*, San Antonio, November 25-29, 2001

[9] T. He, L. Gu, B.Blum, Jun Xie "Nest Project Source Code Base" *at http://sourceforge.net/projects/vert/* Univeristy of Virginia

[10] Tian He, John A. Stankovic, Chenyang Lu, and Tarek F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks", In *International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, May 2003.

[11] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. "Building Efficient Wireless Sensor Networks with Low-Level Naming," In *Proceedings of the Symposium on Operating Systems Principles*, Lake Louise, Banff, Canada, October, 2001.

[12] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", In *HICSS '00*, January 2000.

[13] W.R. Heinzelman, J. Kulik and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks", In *MOBICOM, 1999*, Seattle, 174-185.

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, " System architecture directions for network sensors," in *ASPLOS 2000*.

[15] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," the

Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000), August 2000, Boston, Massachusetts.

[16] C. Intanagonwiwat, D. Estrin, R. Govindan, and J Heidemann. "Impact of Network Density on Data Aggregation in Wireless Sensor Networks", In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, IEEE. July, 2002.

[17] David B. Johnson and David A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks". In *Mobile Computing, Chapter 5, pages 153-181*, Kluwer Academic Publishers, 1996.

[18] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. W.Knightly, "Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constraints", In *IEEE MobiCOM 2001*, Rome, Italy, July 2001.

[19] Phil Karn. "MACA – A New Channel Access Method for Packet Radio". In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134-140, September 1990.

[20] Karp, B., "Geographic Routing for Wireless Networks", Ph.D. Dissertation, Harvard University, Cambridge, MA, October, 2000.

[21] B. Krishnamachari, Deborah Estrin, and Stephen Wicker. "Impact of data aggregation in wireless sensor networks". In *International Workshop on Distributed Event-Based Systems*, Vienna, Austria, July 2002.

[22] Alvin Lim, "Distributed Services for Information Dissemination in Self-Organizing Sensor Networks," the *Special Issue on Distributed Sensor Networks for Real-Time Systems with Adaptive Reconfiguration*, Journal of Franklin Institute, 2001.

[23] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks", In *IEEE RTAS 2002*, San Jose, CA, September 2002

[24] Rex Min, Manish Bhardwaj, Seong-Hwan Cho, Amit Sinha, Eugene Shih, Alice Wang, and Anantha Chandrakasan, "An Architecture for a Power-Aware Distributed Microsensor Node", *2000 IEEE Workshop on Signal Processing Systems (SiPS '00)*, October 2000

[25] Radhika Nagpal and Daniel Coore, "An Algorithm for Group Formation in an Amorphous Computer", In *Proceedings of the 10th International Conference on Parallel and Distributed Computing Systems* (PDCS'98), Nevada, Oct 1998.

[26] Sagnik Bhattacharya, Hyung Kim, Shashi Prabh, Tarek Abdelzaher , "Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks," *The First International Conference on Mobile Systems, Applications, and Services (MobiSys),* San Francisco, CA, May 2003.

[27] H.Takagi and L.Kleinrock. "Optimal Transmission Ranges For Randomly Distributed Packet Radio Terminals". *IEEE Trans. on Communication*, 32(3):246-257, March 1984

[28] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing," *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 2001.

[29] Wei Ye, John Heidemann and Deborah Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks". In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM 2002), New York, NY, USA, June, 2002.

[30] A. Woo and D. Culler. "A Transmission Control Scheme for Media Access in Sensor Networks," *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM 2001),* Rome, Italy, July 2001.

[31] Yuan Xue, Baochun Li. , "A Location-aided Power-aware Routing Protocol in Mobile Ad Hoc Networks", in *Proceedings of IEEE Globecom 2001*, Vol. 5, pp. 2837-2841, San Antonio, Texas, November 25-29, 2001

[32] http://www.xbow.com/Products/Product_pdf_files/MICA%20data%20sheet.pdf