AIDA: Adaptive Application-Independent Data Aggregation in Wireless Sensor Networks

TIAN HE, BRIAN M. BLUM, JOHN A. STANKOVIC and TAREK ABDELZAHER
Department of Computer Science, University of Virginia

Sensor networks, a novel paradigm in distributed wireless communication technology, have been proposed for various applications including military surveillance and environmental monitoring. These systems 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 an overall reduction in header overhead. Theoretical analysis, simulation, and a test-bed implementation on Berkeley's MICA motes are provided to validate our claims.

Categories and Subject Descriptors: C.2. [**Computer Communication Networks**]: Network Protocols

General Terms: Algorithms, Performance, Design

Additional Key Words and Phrases: Data aggregation, sensor networks, adaptive algorithms, feedback control, energy conservation, congestion control

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.

This work was supported, in part by, NSF grant CCR-0098269, the MURI award N00014-01-1-0576, and the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905).
Authors' address: T. He, B. M. Blum, J. A. Stankovic and T. Abdelzaher, Department of Computer Science, University of Virginia, Charlottesville, VA, 22904; email: tianhe@cs.virginia.edu.

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 multihop 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 [Hill et al. 2000; Woo et al. 2001; CrossBow 2002] and various other research institutions [Min et al. 2000] throughout the world. Using such hardware as a basis for development, the software architecture and communication stack residing on these devices are built taking into consideration the prolific research in the areas of *ad hoc* networking [He et al. 2003; Intanagonwiwat et al. 2000; Johnson and Maltz 1996], data aggregation [Intanagonwiwat et al. 2002; Krishnamachari et al. 2002; Madden et al. 2002], cluster formation [Nagpal and Corre 1998], distributed services [Lim 2001], group formation [Blum et al. 2003], channel contention [ANSI/IEEE 1999; Bharghavan et al. 1994; Fullmer and Aceves 1995; Karn 1990], and power conservation [Chen et al 2001; Yan et al. 2003]. Work targeted to these devices include research in query processing (e.g., TinyDB [Madden et al. 2003]), and aggregation (e.g., TAG [Madden et al. 2002]). Work on the utility of such innovative technologies has unearthed potential applications including event tracking [Abdelzaher et al. 2003], environmental monitoring, 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 solution. We develop 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, prior application dependent data aggregation (ADDA shown in Figure 1(b)) relies on application layer information and must have a bidirectional 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)). This component is generalized enough to be utilized over a wide range of applications (data types) without incurring the costs of rewriting components to support application-specific logic. Second,

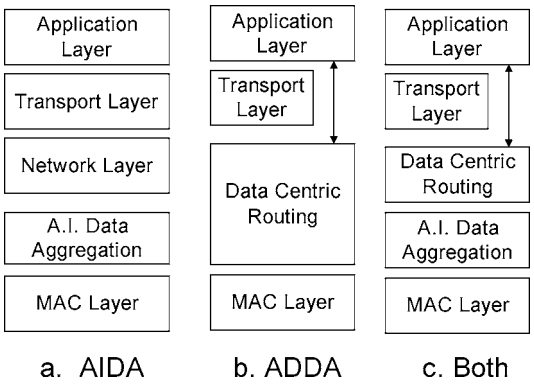

Fig. 1. Architectural designs.

no prior work in data aggregation adapts itself to the traffic situations 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 [Intanagonwiwat et al. 2000]) 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 cannot be obtained from the average). In contrast, AIDA performs lossless aggregation allowing the upper layer to decide whether information compression is appropriate at the time. Very importantly, 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.

This paper addresses 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 [ANSI/IEEE

1999], MACA [Karn 1990], MACAW [Bharghavan et al. 1994], FAMA [Fullmer and Aceves 1995], S-MAC [Ye et al. 2002], and multihop scheduling [Kanodia et al. 2001], 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 [He et al. 2003] and using a minimal hop path to reduce the total number of transmissions [Takagi and Kleinrock 1984]. Beyond the routing layer the communication stack in sensor networks becomes more amorphous. Clustering [Nagpal and Coore 1998], group formation [Blum et al. 2003], 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, hierarchical and grouping components often utilize various forms of data aggregation through consensus algorithms or other forms of local processing.

Basic schemes [Intanagonwiwat et al. 2002] 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.

Expressing queries [Madden et al. 2002] and utilizing those queries for data aggregation [Madden et al. 2002] present opportunities for in-network data aggregation. A popular data aggregation scheme for sensor networks, directed diffusion [Intanagonwiwat et al. 2000], 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 [Bhattacharya et al. 2003] 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 [Heinzelman et al. 2000] 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. 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 preconfiguration 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 [Lu et al. 2002] and SPEED [He et al. 2003] utilize neighborhood information to adjust priority levels or make more informed routing decisions in response to network congestion and changing traffic patterns. SPIN [Heinzelman 1999] makes adaptive decisions to participate in data dissemination based on current energy levels and the cost of communication. In Woo and Culler [2001], adaptive rate control is used at the data-link layer to fine tune contention parameters in response to local traffic conditions. GAF [Xu et al. 2001] 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 [Min et al. 2000], 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 [Guo et al. 2001; Yan et al. 2003]. At the network and routing layers, schemes work to minimize power along the transmission path [Bhattacharya et al. 2003], set routes according to the energy remaining at nodes along that path [Xue and Li 2001], and use mechanisms to save power through the distribution of messages among various paths from source to destination [He et al. 2003]. Finally, higher layer protocols that often incorporate routing semantics exist to form groups and rotate leadership responsibilities allowing nonleader nodes to sleep and conserve their energy [Chen et al. 2001]. 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 due to asymmetric channels, multihop 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 handshake, 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 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 evolvable provide us with insight into developing an 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 and logic to support these naming semantics. As a result, in an application-specific aggregation scheme, the logic of the components will need to be changed every time the operation or task changes. For example, different aggregation logic may be needed for mapping, counting, averaging, standard deviation, and so on. The more operations the applications have the more specific the aggregation logic needs to be, leading to time consuming modifications and a cumbersome design. AIDA seeks a solution without such cross-layer dependencies in order to be utilized over a wide range of data types and applications without incurring the costs of rewriting components. This reduction of interlayer dependencies leads to a lower cost to system evolvability.
- 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.
- The sensor networks we envision will be multipurpose systems. These systems should, therefore, support aggregation across different data types. An ADDA scheme will be limited and somewhat ineffectual as it is hard to aggregate temperature readings with light readings in an application-specific way. 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.
- 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. AIDA utilizes feedback control based on network traffic conditions when making aggregation decisions to adaptively optimize bandwidth while minimizing system energy consumption, which is underexploited by pervious aggregation schemes [Intanagonwiwat et al. 2002; Krishnamachari et al. 2002; Madden et al. 2002].

ADDA schemes have proven to be effective solutions for sensor networks. Given the research issues underexploited by 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 overhead, 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 maintain the benefit 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 are 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 deaggregates 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.

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 4.4) 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 online 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 refragmented

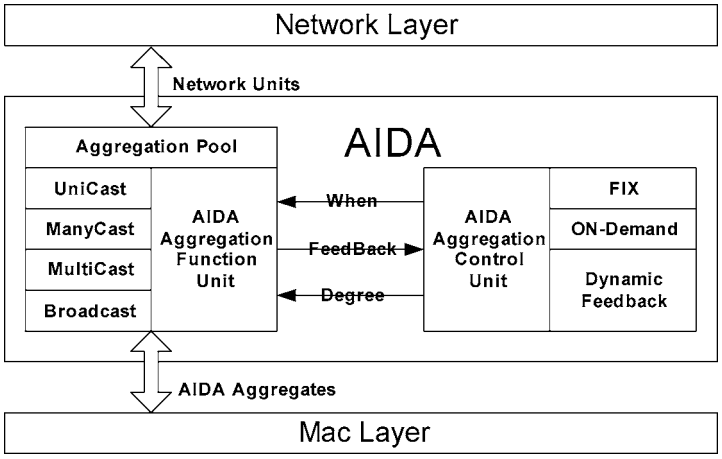

Fig. 2. AIDA components.

into their original network units of which each piece of the aggregate is passed up to the network layer for rerouting or application demultiplexing and delivery. Although we acknowledge that many aggregates may be bound for the same ultimate destination (it could be more efficient not to deaggregate and reaggregate at every intermediate node), we perform such deaggregation 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, RTS/CTS in MACAW, 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] $\times$ [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.

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. 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

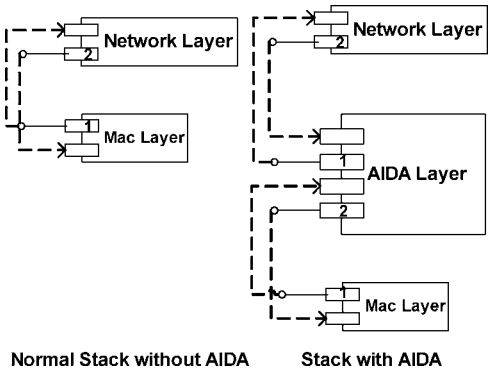

Fig. 3. AIDA implementation design.

AIDA. Versions of our architecture include the 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 $= N_{\text{fixed}}$). 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 do not 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 (unless the maximum MAC unit size is reached). 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

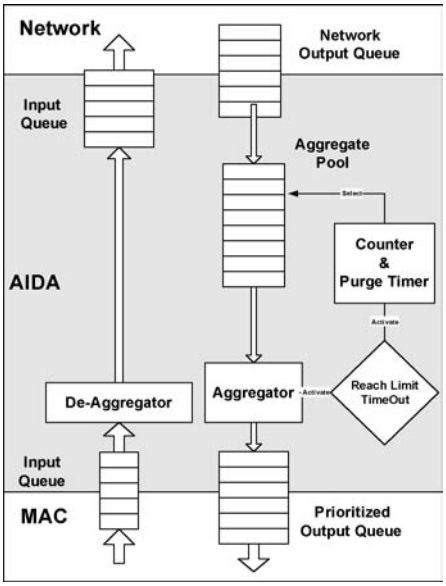

Fig. 4. AIDA FIX scheme.

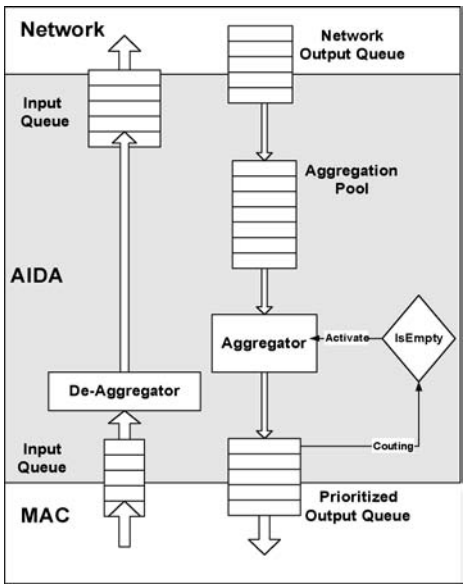

Fig. 5. AIDA on-demand scheme.

outbound message queue increases resulting in a proportional increase in the DOA.

As shown in Figure 5, the on-demand scheme only requires simple monitoring logic to test whether the outbound queue is empty or not. This simplicity of code is preferable for a constrained sensor node. It should be noted that by

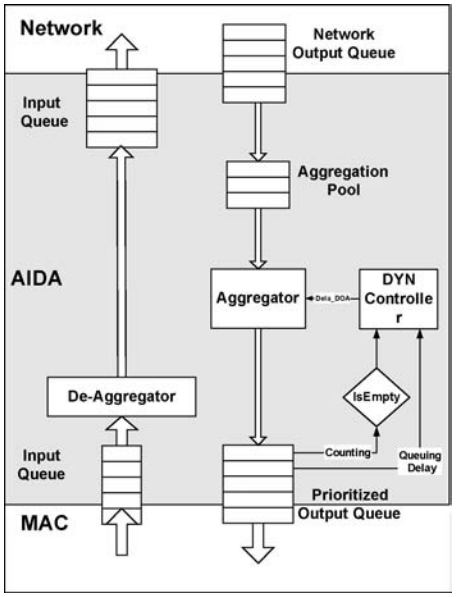

Fig. 6. AIDA dynamic feedback scheme.

aggregating a train of network units with one MAC header per aggregate, on-demand scheme can reduce the header overhead than the scheme that simply flushes all packets out of the queue.

4.2.4 *Dynamic Feedback Scheme.* 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 premise of control theory, our aggregation mechanism dynamically adjusts the degree of aggregation (DOA $= N_{\mathrm{DYN}}$) to converge the MAC delay to a certain set point. This scheme begins with N_{DYN} set to 1. 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 up and the contention delays transmission, our feedback loop adjusts our admission threshold (N_{DYN}) to allow a greater degree of aggregation prior to sending.

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.

It is common practice to use a time slotted approach (e.g., in ALHOA and CSMA) to analyze the performance of contention-based protocols and establish a system model. Here while our approach does not assume a slotted MAC, we adopt this analysis technique to simplify problem formulation. The modeling process is as following.

A general form for calculating the MAC delay can be defined as

$$D_{\mathrm{MAC}}(k) = D_{\mathrm{minimum}} + \#\ \mathrm{collisions}(k) \times D_{\mathrm{reslove}} \tag{1}$$

where $D_{\mathrm{MAC}}(k)$ is the MAC delay packets experience during time period $[k, k+1]$, D_{minimum} is the MAC delay when no collision is experienced, and it is the performance set point that control loop wants to achieve. # collisions(k) is the number of collisions a successful transmission encounters 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. It should be noted that (1) establishes the model for the MAC layer. The wait delay to build an AIDA packet is traffic dependent and should not be considered in the MAC modeling process.

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 that are compete for the channel. We denote the probability of a packet being transmitted at this time period by the symbol τ. This τ value is a function of the type of MAC protocol. An outgoing packet encounters a collision when it overlaps with the transmission of at least one other packet from the remaining $N(k)$/DOA$(k)-1$ packets. Accordingly, the average collision probability p can be calculated as

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

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

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

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

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

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

$$D_{\mathrm{MAC}}(k) = [D_{\mathrm{minimum}} - D_{\mathrm{reslove}}] + D_{\mathrm{reslove}}(1-\tau)^{1-(N(k)/\mathrm{DOA}(k))}. \tag{5}$$

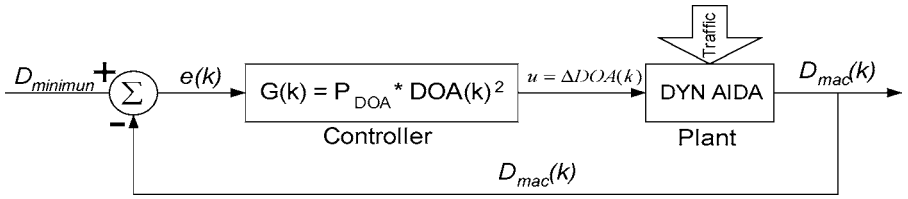

Fig. 7. The control loop for the DYN AIDA scheme.

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

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

where $\lambda_1 = D_{\text{reslove}}(1-\tau)N(k)(\ln(1-\tau),\ \lambda_2 = (1-\tau)^{-N(k)}$.

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 deriving 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:

$$\Lambda\text{DOA}(k) = G(k)\, e(k) \tag{7}$$

where $G(k) = P_{\text{DOA}} \times \text{DOA}(k)^2,\ e(k) = (D_{\text{MAC}}(k) - D_{\text{minimum}})$.

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. Thus AIDA is essentially modeled as a first-order system and therefore the gain $G(k)$ in equation (7) does not need to be constant for stability analysis, as long as $G(k)$ is bounded. The pictorial notation of this control loop is shown in Figure 7.

As we show in the evaluation, the current adaptive controller works best under a wide range of traffic scenarios under investigation. However, we acknowledge that the modeling portion of our work has room for improvement to precisely reflect the nonlinear behavior of the MAC contention.

4.3 AIDA Function Unit

The AIDA aggregation function unit (Figure 2) is responsible for the aggregation and deaggregations 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 uses 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.

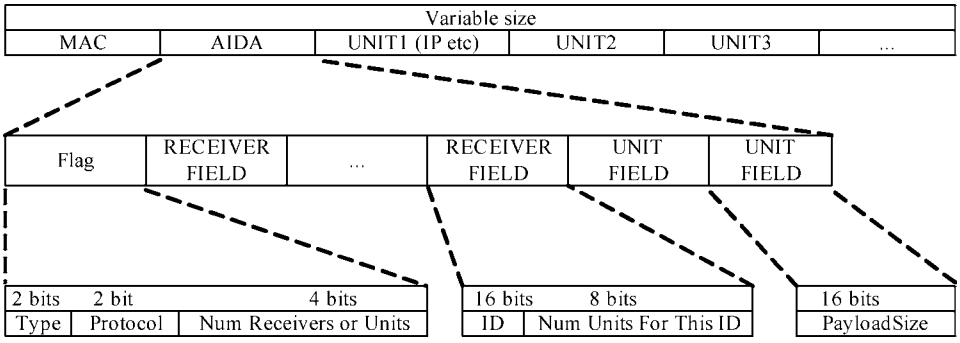

Fig. 8. AIDA general header format.

- 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 receives and deaggregates this Multicast packet to determine whether or not a portion of the aggregated payload was destined to it.
- Finally, the Broadcast type 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 deaggregation, demultiplexing, 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 deaggregation, and potentially break apart, demultiplex, and deliver each network unit to the appropriate network layer module.

It should be noted that by aggregating the network payloads, AIDA reduces the number of packets sent at the MAC layer, thus actually reducing overall header cost. The general form of the AIDA header is provided in Figure 8. Some fields inside this general form are *not* used for certain AIDA payload types.

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 demultiplex network units.
- Num Receiver/Units: This field (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 contains the number of neighbors receiving portions of this aggregate.

4.4.2 *Receiver Field for Multicast Type.* The Receiver field is *only* used by Multicast AIDA packets. Each field 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 field is not used.

- ID Field: The ID field (2 bytes) contains a locally unique identifier of the node receiving a specified number of network units.
- Num Units for this ID Field: This 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 ID Field.

4.4.3 *Unit Field.* The UNIT field is used during deaggregation for delimiting the boundaries between network units. It consists of a 16 bit (2 byte) field that specifies the size of each network unit. In the Unicast case there is no boundary to be identified, so the UNIT field is not used.

4.5 AIDA Header Overhead Analysis

First, it should be recalled that though AIDA introduces a new header, it actually reduces overall header overhead by aggregating several network units into one MAC payload. 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 $28 \times N$ bytes. Using AIDA, we reduce the total header overhead to $28 + \text{AIDAHeaderSize}$ bytes. As long as the value of N (the DOA) is greater than 1, AIDA effectively reduces the total packet overhead incurred during transmission.

It is simple to assess the overhead incurred during the aggregation of network units according to the description in Section 4.4. For comparison, the packet structure with and without AIDA is shown in Figure 9.

- Unicast only uses the Flag field and therefore incurs a single byte of overhead.
- Besides the 1 byte flag, Manycast and Broadcast packets need to delimitate the boundaries of multiple network units, thus incurring an average of

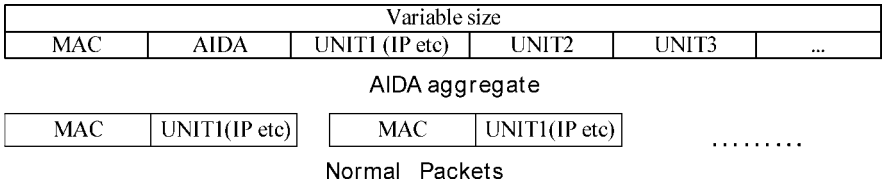

Fig. 9. Format comparison.

$(2+1/N)$ 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).

4.6 AIDA Savings Analysis

Adding header information to any transmission intuitively increases 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 differ for Unicast, Manycast, and Multicast transmissions. To better understand the potential benefits of aggregation, and to compare different 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 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 ms) 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/s, the AIDA header overhead is O(in bytes), and DOA is the number of packets aggregated. The cost C_{AIDA} (in ms) can be calculated from equation (8):

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

In contrast, the cost of sending DOA number of packets without the aggregation scheme C_{None} is

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

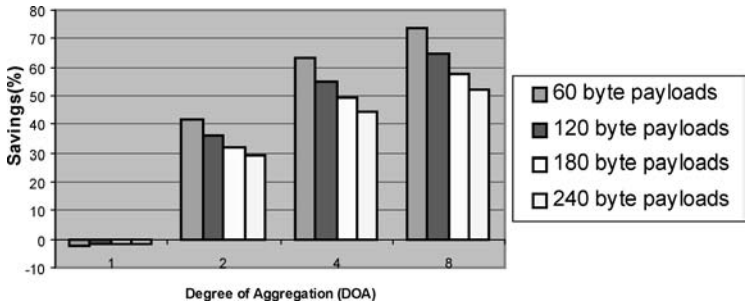

Fig. 10. AIDA theoretical savings.

Hence, the percentage saving in cost is calculated as follows:

$$\textit{Saving Percentage} = \left(\frac{C_{\text{None}} - C_{\text{AIDA}}}{C_{\text{None}}}\right) = \left[1 - \frac{S \times R}{M + S \times R}\right] - \frac{M + O \times R}{M + S \times R} \times \frac{1}{\text{DOA}}. \tag{10}$$

From equation (10), we can see that the saving 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 ms of idle listening and the DIFS and SIFS intervals are chosen at 10 and 5 ms, 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 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 DOA increases, the percentage of savings in time increases drastically. We also note that as payload size increases, the relative time saving 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 I describes the detailed setup for our simulator. For our experiments

Table I. Simulation Settings

Routing	GF
MAC layer	Simplified 802.11 DCF
Radio layer	RADIO-ACCNOISE
Propagation model	TWO-RAY
Bandwidth	40–200 Kbps
Payload size	32 Byte
Terrain	(200 m, 200 m)
Number of motes	100
Node placement	Uniform
Radio range	40 m

the communication parameters are mostly chosen in accordance with Berkeley MICA mote specifications [CrossBow 2002], the popular hardware platform on which sensor network research systems are currently deployed for testing. The current version of the MICA motes supports a 40 kbps transmission rate and the next generation is expected to provide higher than 1 Mbps rates. Based on these considerations, we choose 40–200 Kbps as the effective bandwidth for our evaluation (default 200 Kbps unless otherwise specified). Finally, we choose 802.11 as our MAC layer protocol, which has been implemented in a scaled down version on the MICA platform.

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, 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 the 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 Workload 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 workloads in sensor networks. One-to-one communication happens when one sentry node detects some activity that needs to be reported to a remote entity. Alternatively, a quering 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

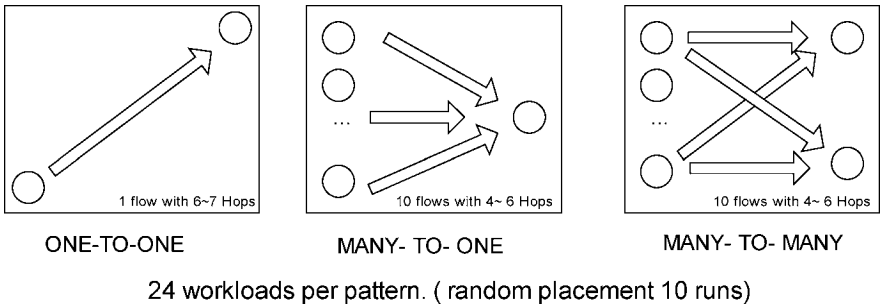

Fig. 11. Traffic load settings.

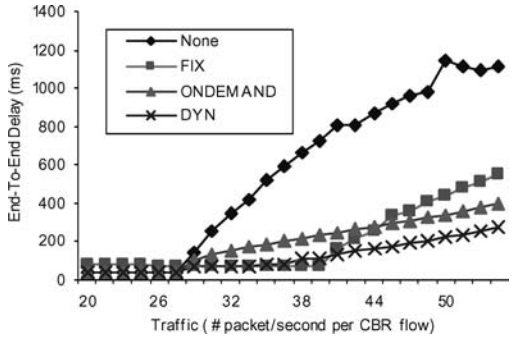

Fig. 12. Avg E2E delay (one-to-one 200 Kbps).

run simultaneously and the traffic flows interleave with each other, which is a many-to-many cross-traffic pattern.

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, five nodes on the left side of the terrain send out 10 CBR flows (two 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 does not 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.

Figures 12–14 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

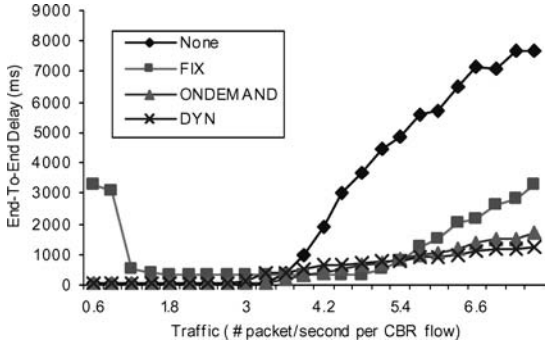

Fig. 13. Avg E2E delay (many-to-one 200 Kbps).

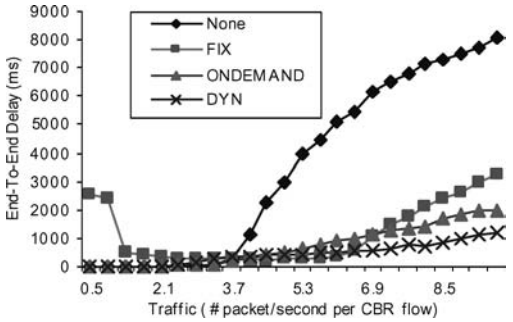

Fig. 14. Avg E2E delay (many-to-many 200 Kbps).

without performing aggregation increases dramatically as the overall traffic increases gradually. This is the typical case for multihop 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 the FIX have very short end-to-end delay (abut 70–100 ms). The reason for additional delay in the 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 the 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 dynamically adjusts the required DOA according to the MAC delay that the outgoing packets experience. In heavy traffic, it is beneficial to reduce number of node competing for the channel by reducing sending rate. 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 nonaggregation case, when flow rate at 8.5 packets/s per flow (see Figure 14).

5.2.2 *End-to-End Delay Under Different Available Bandwidth Settings.* In this experiment, we investigate the end-to-end delay under the different bandwidth settings. The workloads are chosen differently for each bandwidth setting

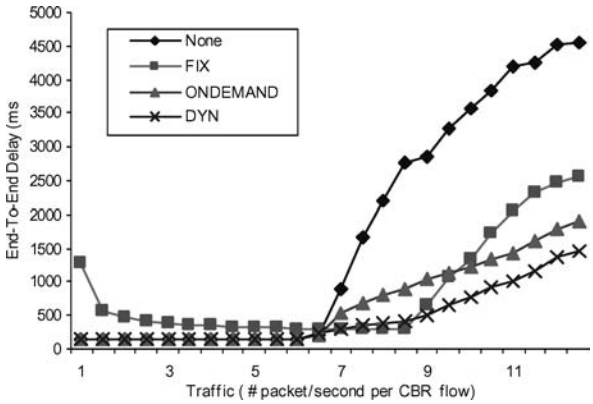

Fig. 15. E2E delay (one-to-one under 40 Kbps).

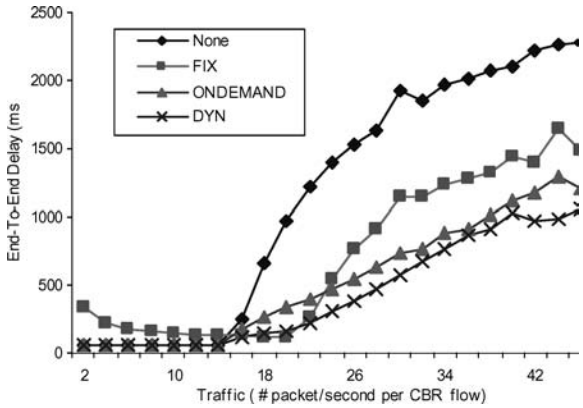

Fig. 16. E2E delay (one-to-one under 100 Kbps).

in order to compare the performance of each scheme from underutilized to saturated traffic situations.

Figures 12, 15, and 16 demonstrate that DYN scheme outperforms other schemes regardless the available bandwidth settings. This is mainly because that DYN can more effectively aggregate and schedule the packets according to the feedback of the current traffic situations than other schemes. Based on such an investigation, we conclude that the improvement made by the DYN scheme over other schemes is orthogonal to the available bandwidth settings, though the absolute performance gain may vary.

5.2.3 *End-to-end Delay Under Different DOA Setting for the FIX Scheme.* In this experiment, we measure packet end-to-end delay for various traffic loads under different DOA settings in the FIX scheme. Figure 17 reveals the disadvantage of the FIX scheme and explains why dynamic adaptability is desired for such a system. From Figure 17, we can see that there is no single DOA value that works well for every traffic pattern.

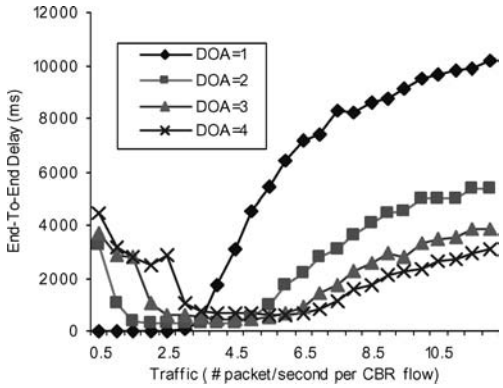

Fig. 17. Avg E2E delay (many-to-one).

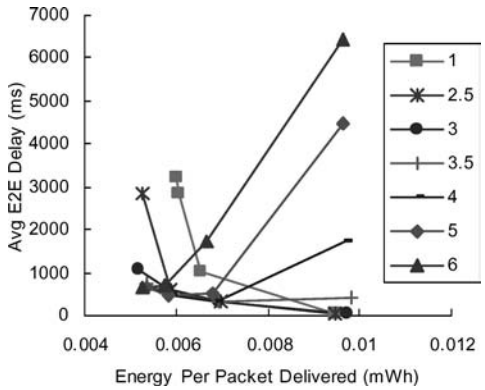

Fig. 18. E2E Delay versus energy (many-to-one).

On one hand, a high DOA value in the FIX scheme does not perform well under low traffic loads. For example, when the DOA is higher than 1, additional delay is incurred when the traffic load is 0.5 packets/second per flow or lower. The higher DOA settings tend to reduce congestion, but increase delay in the AIDA component for packets waiting to be sent. On the other hand, low DOA value settings do not perform well under heavy traffic. For example, shown in Figure 17, 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 or higher.

In addition, Figure 18 demonstrates the performance penalty due to the lack of adaptability in the FIX scheme. We plot the relationship between average end-to-end delay and average energy consumption per packet delivered under different CBR rates from 1 to 6 packets/s. Under the light traffic (e.g., 1 packet/s per CBR), the FIX scheme needs to hold back packets in order to reduce energy consumption. Under heavy traffic (e.g., 6 packet/s per CBR), the FIX scheme causes an increase in both delay and energy consumption by choosing a fixed DOA value that does not reflect the traffic load.

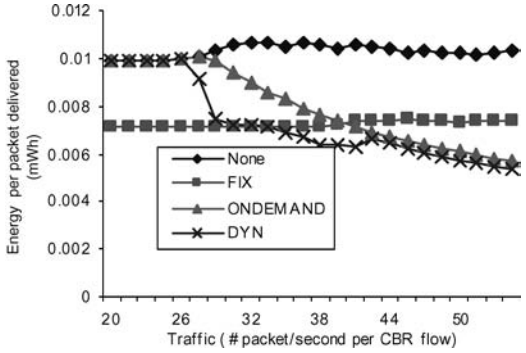

Fig. 19. Energy per unit delivered (one-to-one).

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 or proactively change the DOA value in accordance with these traffic patterns, respectively. Therefore, they exhibit a better overall performance as shown in Figures 12–14.

5.3 Energy Consumption

In this section, energy consumption, in transmission energy, is adopted as another revealing metric to evaluate the AIDA performance. Since transmission energy increases proportionally with the number of bits sent, it can adequately summarize and reflect the performance of other related metrics such as total header overhead, number of collisions, and total number of bit transmitted bytes.

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 [Hill et al. 2000]), which leads to unnecessary padding costs. In our simulation with variable size support, AIDA takes advantage of the first two approaches.

In this experiment, we measure average transmission energy per delivered packet under 24 increasing traffic loads for three traffic patterns. In Figures 19–21, 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/s per flow (in Figure 21). The FIX scheme always aggregates 2 packets before sending, which leads to nearly constant energy savings in both the low and high traffic situation. However, in the FIX scheme, the DOA values are set and congestion levels are not

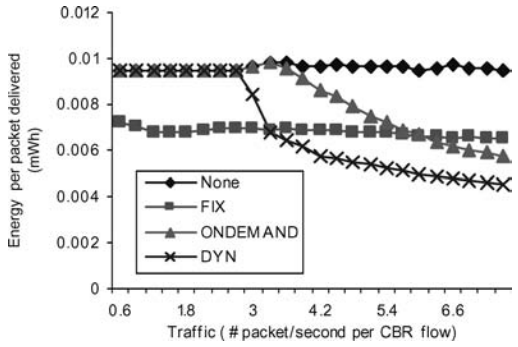

Fig. 20. Energy per unit delivered (many-to-one).

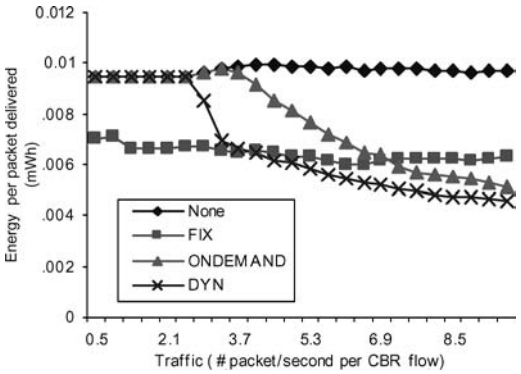

Fig. 21. Energy per unit delivered (many-to-many).

taken into account resulting in worse performance than in DYN and on-demand schemes under heavy traffic conditions. For example, shown in Figure 21, in the DYN scheme, nodes consumes about 20% less energy per packet delivered as in the FIX scheme, when traffic load is about 8 packets/s per flow.

5.3.2 *Energy Consumption Under Different DOA for the FIX Scheme.* Figure 22 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.

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 are for the

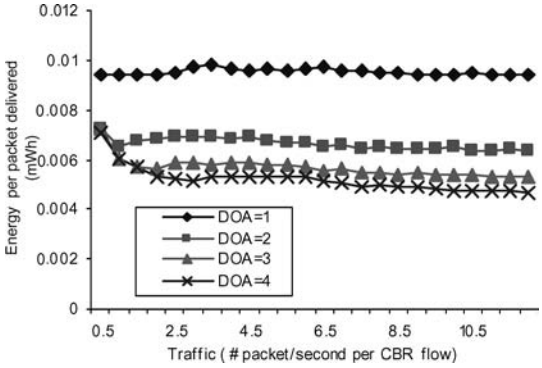

Fig. 22. Energy per unit delivered (FIX scheme).

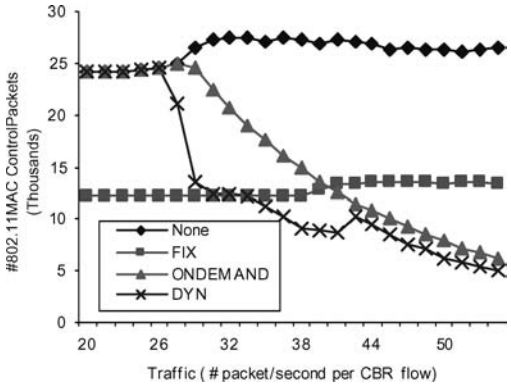

Fig. 23. MAC control packets (one-to-one).

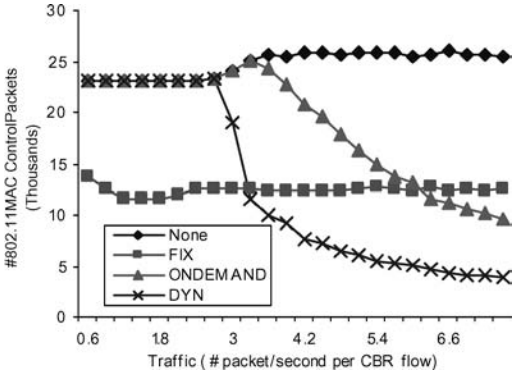

Fig. 24. MAC control packets (many-to-one).

802.11 MAC protocol although we would expect very similar results for other MAC protocols.

Figures 23–25 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.

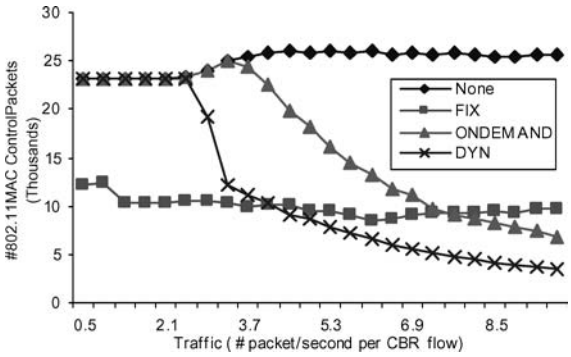

Fig. 25. MAC control packets (many-to-many).

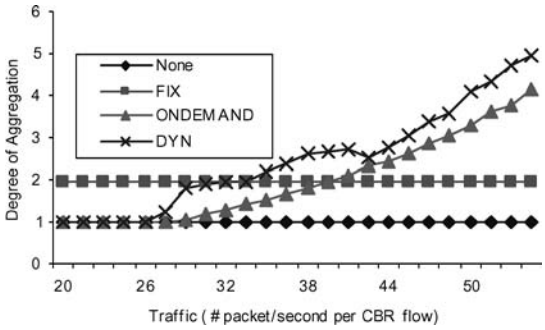

Fig. 26. DOA (one-to-one).

On-demand and DYN vary their DOA and, therefore, incrementally reduce the MAC overhead as network congestion levels increase. For example, shown in Figure 25, when per flow rate exceeds 9 packets/s, DYN only used about 20% of the control packets compared to the none-aggregation case. This dramatically reduces congestion and energy consumption as shown in other evaluations.

5.5 Degree of Aggregation

As seen in the context of reducing the 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 1 (e.g., None case in Figure 26). In the FIX scheme where DOA is set to 2, we can see that 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.

Figures 26–28 graph the achieved DOA under various traffic conditions for the tested schemes. Figure 26 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 (Figures 27–28), DYN achieves a higher DOA value than on-demand resulting in more savings on channel bandwidth and energy consumption.

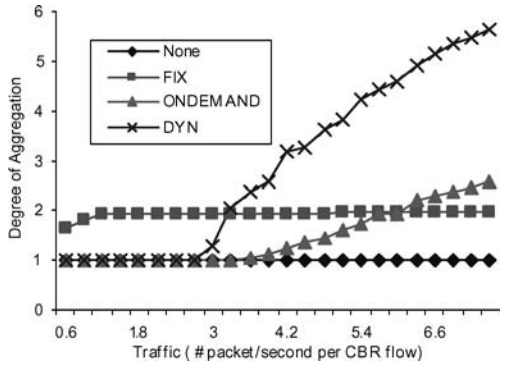

Fig. 27. DOA (many-to-one).

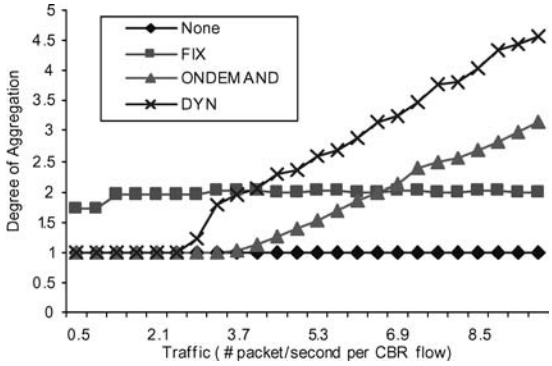

Fig. 28. DOA (many-to-many).

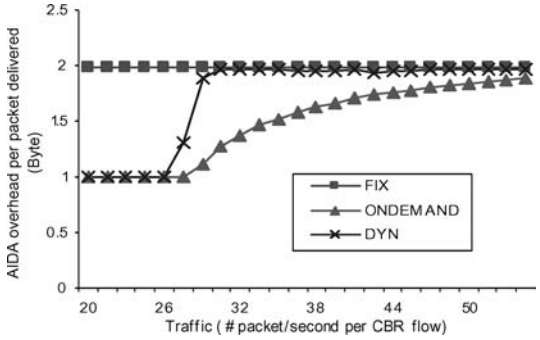

Fig. 29. AIDA overhead (one-to-one).

5.6 AIDA Overhead

As shown in the 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 per network unit. Figures 29–31 graph per packet AIDA overhead under various traffic loads. As shown in Figure 30, under many-to-one conditions, the FIX scheme sends out only Manycast packets with

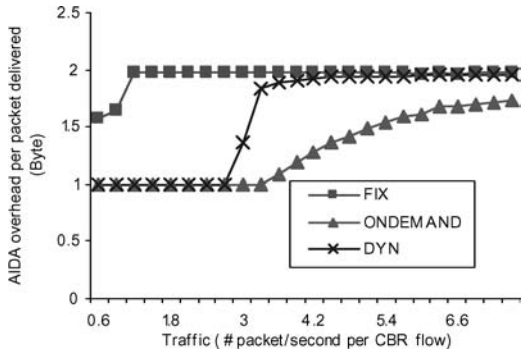

Fig. 30. AIDA overhead (many-to-one).

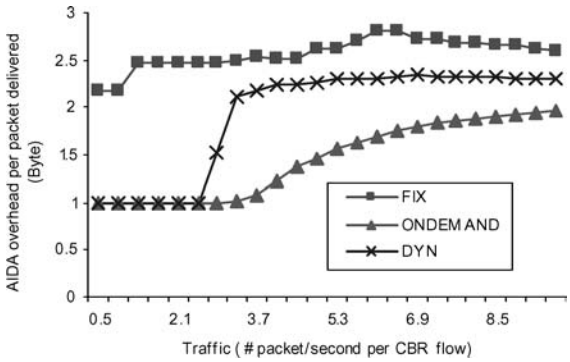

Fig. 31. AIDA overhead (many-to-many).

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 the FIX scheme in Figure 30), 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 Figures 29–30 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 of overhead for each multicast packet. This is shown in Figure 31 where AIDA overhead is somewhere between 2 and 3 bytes for the FIX scheme.

5.7 Comparisons and Summary

In summary, the FIX scheme does not take congestion into account and is not adaptable to changing traffic loads. There is no single DOA value that

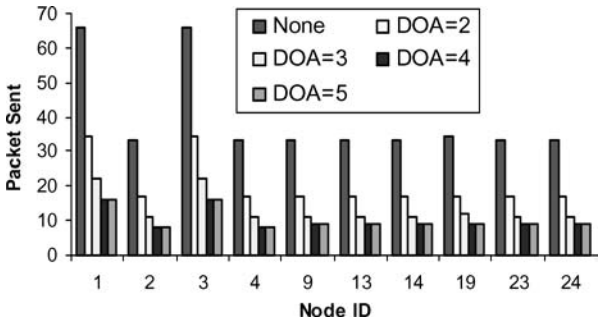

Fig. 32. Packets sent under different DOA.

works well for every traffic pattern. The feedback information utilized in the on-demand scheme is essential binary: the MAC component is either busy or free. This only provides limited information to the controller. In comparison, DYN obtains delay information that directly reflects the current traffic situation resulting in a better control model and, therefore, better performance.

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 He et al. [2002]). Three applications including data placement [Bhattacharya et al. 2003], target tracking [Blum et al. 2003], 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 (5 bytes payload) 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 [Karp 2000] protocol without AIDA. In some 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 32 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 concatenate any more network units into the AIDA aggregate. For our experiment and payload size this occurred when the DOA was 5. The latest version of TinyOS supports variable packet size during transmission. Under this, AIDA can achieve higher DOA values.

7. CONCLUSION

In this paper we introduce AIDA, an adaptive AIDA 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 incurring the costs of rewriting components 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 of around 2 bytes per network unit and reduces 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 testbed provides initial evidence of the savings obtainable by an application-independent aggregation scheme and paves the way for future implementations of our adaptive control-based protocol.

REFERENCES

ABDELZAHER, T. F., ET AL. 2003. *EnviroTrack: An Environmental Programming Model for Tracking Applications in Distributed Sensor Networks*. Tech. Rep. CS-2003-02, University of Virginia.

ADAMOU, M., KHANNA, S., LEE, I., SHIN, I., AND ZHOU, S. 2001. Fair real-time traffic scheduling over a wireless LAN. In *Proceedings of the 22nd IEEE RTSS* 2001, London, UK.

ANSI/IEEE. 1999. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *ANSI/IEEE Std 802.11*, 1999 Edition.

BHARGHAVAN, V., DEMERS, A., SHENKER, S., AND ZHANG, L. 1994. MACAW: A media access protocol for wireless LANs. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*. 212–225.

BHATTACHARYA, S., KIM, H., PRABH, S., ABDELZAHER, T. F. 2003. Energy-conserving data placement and asynchronous multicast in wireless sensor networks. In *The First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA.

BLUM, B., NAGARADDI, P., WOOD, A., ABDELZAHER, T., SON, S., AND STANKOVIC, J. A. 2003. An entity maintenance and connection service for sensor networks. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MobiSys),* San Francisco, CA.

CHEN, B., JAMIESON, K., BALAKRISHNAN, H., AND MORRIS, R. 2001. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of the 6th ACM MOBICOM Conference*, Rome, Italy.

CROSSBOW. 2003. Available from http://www.xbow.com/Products/Product_pdf_files/MICA%20data%20sheet.pdf.

FULLMER, C. AND GARCIA-LUNA-ACEVES, J. J. 1995. Floor acquisition multiple access (FAMA) for packet radio networks. *Comput. Commun. Rev. 25*, 4.

GUO, C., ZHONG, L. C., AND RABAEY, J. M. 2001. Low power distributed MAC for ad hoc sensor radio networks. In *Proceedings of IEEE GlobeCom 2001*, San Antonio.

HE, T., GU, L., AND BLUM, B. 2002. Nest Project Source Code Base. Available from http://sourceforge.net/projects/vert/.

HE, T., STANKOVIC, J. A., LU, C., AND ABDELZAHER, T. F. 2003. SPEED: A stateless protocol for real-time communication in sensor networks. In *International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, RI.

HEIDEMANN, J., SILVA, F., INTANAGONWIWAT, C., GOVINDAN, R., ESTRIN, D., AND GANESAN, D. 2001. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operating Systems Principles*, Lake Louise, Banff, Canada.

HEINZELMAN, W., CHANDRAKASAN, A., AND BALAKRISHNAN, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of HICSS '00*.

HEINZELMAN, W. R., KULIK, J., AND BALAKRISHNAN, H. 1999. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of MobiCOM 1999*, Seattle. 174–185.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. 2000. System architecture directions for network sensors. In *Proceedings of ASPLOS*.

INTANAGONWIWAT, C., ESTRIN, D., GOVINDAN, R., AND HEIDEMANN, J. 2002. 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.

INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. 2000. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of MobiCOM 2000*, Boston, MA.

JOHNSON, D. B. AND MALTZ, D. A. 1996. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*. Kluwer Academic Publishers, Boston, MA, 153–181, Chapter 5.

KANODIA, V., LI, C., SABHARWAL, A., SADEGHI, B., AND KNIGHTLY, E. W. 2001. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proceedings of MobiCOM 2001*, Rome, Italy.

KARN, P. 1990. MACA—A new channel access method for packet radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*. 134–140.

KARP, B. 2002. *Geographic Routing for Wireless Networks*. Ph.D. Dissertation, Harvard University, Cambridge, MA.

KRISHNAMACHARI, B., ESTRIN, D., AND WICKER, S. 2002. Impact of data aggregation in wireless sensor networks. In *International Workshop on Distributed Event-Based Systems*, Vienna, Austria.

LIM, A., 2001. Distributed services for information dissemination in self-organizing sensor networks. *Special Issue on Distributed Sensor Networks for Real-Time Systems with Adaptive Reconfiguration, Journal of Franklin Institute*.

LU, C., BLUM, B. M., ABDELZAHER, T. F., STANKOVIC, J. A., AND HE, T. 2002. RAP: A teal-time communication architecture for large-scale wireless sensor networks. In *IEEE RTAS 2002*, San Jose, CA.

MADDEN, S. R., HELLERSTEIN, M. J., AND HONG, W. 2002. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the ACM Symposium on Operating System Design and Implementation (OSDI)*.

MADDEN, S. R., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. 2003. The design of an acquisitional query processor for sensor networks. In *Proceedings of SIGMOD*.

MIN, R., BHARDWAJ, M., CHO, S.H., SINHA, A., SHIH, E., WANG, A., AND CHANDRAKASAN, A. 2000. An architecture for a power-aware distributed microsensor node. In *IEEE Workshop on Signal Processing Systems (SiPS '00)*.

NAGPAL, R. AND COORE, D. 1998. 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.

TAKAGI, H. AND KLEINROCK, L. 1984. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. Commun. 32*, 3, 246–257.

Woo, A. and Culler, D. 2001. A transmission control scheme for media access in sensor networks. In *Proceedings of MobiCOM 2001*, Rome, Italy.

Xu, Y., Heidemann, J., and Estrin, D. 2001. Geography-informed energy conservation for ad hoc routing. *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy.

Yan, T, He, T., and Stankovic, J. 2003. A differentiated surveillance service for sensor networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003),* Los Angeles, CA.

Ye, W., Heidemann, J. and Estrin, D. 2002. 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.

Received January 2003; revised June 2003; accepted August 2003