01 Jan 2006

Adaptive Distributed Fair Scheduling and Its Implementation in Wireless Sensor Networks

Maciej Jan Zawodniok
Missouri University of Science and Technology, mjzx9c@mst.edu

Jagannathan Sarangapani
Missouri University of Science and Technology, sarangap@mst.edu

Steve Eugene Watkins
Missouri University of Science and Technology, watkins@mst.edu

James W. Fonda
Missouri University of Science and Technology, fonda@mst.edu

Recommended Citation

M. J. Zawodniok et al., "Adaptive Distributed Fair Scheduling and Its Implementation in Wireless Sensor Networks," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2006, SMC'06*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2006.
The definitive version is available at https://doi.org/10.1109/ICSMC.2006.384641

2006 IEEE International Conference on
Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

Adaptive Distributed Fair Scheduling and Its Implementation in Wireless Sensor Networks

James W. Fonda*, Maciej Zawodniok, S. Jagannathan, and Steve E. Watkins

***Abstract*—A novel adaptive and distributed fair scheduling (ADFS) scheme for wireless sensor networks is shown through hardware implementation. In contrast to simulation, hardware evaluation provides valuable feedback to protocol and hardware development process. The proposed protocol focuses on quality-of-service (QoS) issues to address flow prioritization. Thus, when nodes access a shared channel, the proposed ADFS allocates the channel bandwidth proportionally to the weight, or priority, of the packet flows. Moreover, ADFS allows for dynamic allocation of network resources with little added overhead. Weights are initially assigned using user specified QoS criteria. These weights are subsequently updated as a function of delay, enqueued packets, flow arrival rate, and the previous packet weight. The back-off interval is also altered using the weight update equation. The weight update and the back-off interval selection ensure that global fairness is attained even with variable service rates. The algorithm is implemented using UMR/SLU motes for an industrial monitoring application. Results the hardware implementation demonstrates improved performance in terms of fairness index, flow rate, and delay.**

***Index Terms*— Fairness, Adaptive-fair-scheduling, Weight-adaptation, Quality-of-Service, Embedded System.**

I. Introduction

In this work hardware implementation of wireless sensor networks (WSNs) with the adaptive and distributed fair scheduling (ADFS) protocol are shown. Challenges for hardware implementation of ADFS on WSNs include memory limitations, low processing power, and selection of priority sensor flows. Selection of fair bandwidth allocations for the sensor flows, or based on quality of service (QoS), must also be established based on user requirements.

Introduction of the 802.15 standard has accelerated the application of WSNs in industrial environments. Use of small, low power, radio enabled networks provide observability in a cost effective and deployable platform. Research into WSNs has shown the ability to provide dynamic routing [1], intelligent processing of data, and observability in harsh environments. Since bandwidth is a major constraint in WSNs, one key in guaranteeing the QoS is to manage radio resources effectively and fairly. The focus of this work is to address challenges in, and present, a hardware implementation of a fair scheduling network protocol [2] on a WSN test-bed.

Fairness is a critical issue when accessing a shared wireless channel. Fair scheduling must then be employed in WSNs to provide proper flow of information. In the literature, many algorithms and protocols regarding QoS metrics are found; however they do not address hardware constraints [1-9].

A number of fair scheduling schemes exist in the literature; where some are centralized [3-6], and others are distributed [7, 8]. There has been work on achieving fairness using distributed MAC protocol for wireless networks [8]. A recent work [7] proposes a distributed fair scheduling protocol for wireless LANs. Distributed fair scheduling (DFS) [7] allocates bandwidth proportional to the weights of the flows. This protocol performs a fair allocation of bandwidth using a self-clocked fair queuing algorithm. However, this protocol may not be suitable for multi-hop networks with dynamic channel state and topology. With node mobility, the network state can change demanding weight updates. Moreover, DFS results in large delay variations, or jitter, in reception of packets at the destinations. Finally, selection of initial weights is not addressed in DFS. Unless weights are selected appropriately, fairness cannot be guaranteed even for wireless networks with stationary nodes.

In general these fair scheduling schemes determine appropriate weights in order to meet QoS criteria. In most schemes weights are assigned and not updated when dynamic network conditions apply, and thus do not provide the advantage seen in an ADFS enabled network [3-8].

The notion of fairness must be guaranteed among a set of contending flows. Moreover, the proposed scheme should be computationally distributed in its nature. Thus, any distributed solution to WSN fair scheduling must coordinate local interactions to achieve global performance. This should be achieved within the constraints imposed by the hardware. Therefore, any fair scheduling algorithm proposed for a multi-hop WSN must consider the following design criteria:

- ***Centralized vs. Distributed approaches.*** Distributed fair scheduling algorithm for WSN is preferred over a centralized scheme.
- ***Fairness Metric.*** Selection of an appropriate fairness metric is important from a design aspect. It should address the fair allocation of service proportional to weights selected by user-defined QoS metrics.
- ***Scalability.*** The scheduling scheme should deploy well in WSNs with dynamic topology and link failures.

Research supported in part by Dept. of Education GAANN Fellowship, Air Force Research Laboratory Grant (FA8650-04-C-704) and Intelligent Systems Center.

James W. Fonda, Macei Zawadniok, S. Jagannathan, are members of the Embedded Networking Systems Laboratory and the Electrical and Computer Engineering Department at the University of Missouri-Rolla, Rolla, MO 65409 (email: mjz9c@umr.edu, sarangap@umr.edu).

Steve E. Watkins is a member of the Electrical and Computer Engineering Department at the University of Missouri-Rolla, Rolla, MO 65409 (email: steve.e.watkins@ieee.org).

* Contact author: James W. Fonda can be reached via email: fonda@umr.edu, fax: (573) 341-4532, phone: (573) 308-6866

- ***Efficiency of the protocol.*** Since a trade-off exists between throughput and fairness, fair scheduling should render reasonable throughput to all flows.
- ***Persistency of Quality-of-Service.*** Fair scheduling should meet QoS of all flows during topology changes and dynamic channel states.

Applications for work presented here are based on industrial needs for distributed sensing. Consider the case of distributed sensing of air pressure and volumetric air flow in compressed air systems used for tooling. In this case sensed parameters must be communicated back to a base station without aggregation or sensor fusion to provide observability for each parameter. Thus, fair scheduling of sensor flows is needed to provide equal observability to all priority measurands. Moreover, non-aggregated data is required to allow analysis of the independent sensors.

Work presented in this paper focuses on implementation of the ADFS scheduling protocol [2]. The ADFS protocol was initially developed at UMR for ad-hoc networks. In this paper, the protocol is ported to WSN and implemented on a hardware platform developed at UMR. Hardware implementation is based on a platform developed at the University of Missouri-Rolla (UMR). Hardware testing results are shown to provide a comparison of the performance of ADFS scheme. Additional simulations results are available in [2]. Hardware implementation provides comparison of ADFS in a single hardware cluster. Inter-cluster scheduling provides useful bandwidth allocation to allow the cluster head (CH) to route the sensor information through the rest of the network using optimal energy delay sub-network routing (OEDSR) [1].

II. Adaptive and Distributed Fair Scheduling (ADFS) Protocol

The main goal of the proposed ADFS protocol [2] is to achieve fairness in WSNs in the presence of dynamic channel states since channel uncertainties such as shadowing affects available bandwidth. Dynamic weight adaptation is used to compensate for changing channel states. ADFS employs an adaptive scheduling algorithm to provide fairness among local queues, and the MAC protocol to provide fair channel access via dynamic backoff. ADFS performance was previously evaluated in the NS-2 simulator [2]. In this paper the performance is assessed on the UMR/SLU motes.

A. Protocol Implementation

To achieve fairness at the scheduling level the proposed ADFS protocol implements the start-time fair queuing (SFQ) [2, 3] scheme, defined as follows:

1) On arrival, the j^{th} packet of flow f, p_f^j, and has length l_{fj} and weight ϕ_{fj}, is stamped with start tag $S(p_f^j)$, defined as

$$S(p_f^j) = \max\left\{ v(A(p_f^j)), F(p_f^{j-1}) \right\} \qquad j \geq 1 \tag{1}$$

where $F(p_f^j)$, finish tag of packet p_f^j, is defined as

$$F(p_f^j) = S(p_f^j) + \frac{l_{fj}}{\phi_{fj}} \qquad j \geq 1 \tag{2}$$

where $F(p_f^0) = 0$.

2) Initially, the virtual time, *v(t)*, at a given wireless node is set to zero. During transmission, the node's virtual time at time *t*, *v(t)*, is defined to be equal to the start tag of the packet being transmitted at time t. At the end of a transmission, *v(t)* is set to the maximum of finish tag assigned to any packets that have been transmitted by time *t*.

3) Packets are transmitted in the increasing order of the start tags.

1) Dynamic Weight Adaptation

To account for the dynamic traffic demands and channel states that affect the fairness and end-to-end delay, the weights for the flows are updated dynamically. The actual weight for the i^{th} flow, j^{th} packet denoted by $\hat{\phi}_{ij}$, is updated as

$$\hat{\phi}_{ij}(k+1) = \alpha \hat{\phi}_{ij}(k) + \beta E_{ij} \tag{3}$$

where $\hat{\phi}_{ij}(k)$ is the previous weight of the packet, α and β are design constants, $\{\alpha, \beta\} \in [-1,1]$, and the network state, E_{ij}, is defined as:

$$E_{ij} = e_{ij,queue} + \frac{1}{e_{ij,delay}} \tag{4}$$

where $e_{ij,queue}$ is the error between the expected length of the queue and the actual size of the queue and $e_{ij,delay}$ is the error between the expected delay and the delay experienced by the packet so far. According to (3) and (4), as queues buildup, the packet weights will be increased to clear the backlog. Moreover, the weights of the packets with large end-to-end delays (delays closer to the expected delay) will be increased (due to smaller values of $e_{ij,delay}$) in order to speed up their transmission. However, packets experiencing delays greater than the expected delay will be dropped. Note that the term, E_{ij}, is a bounded value since the queue length and delay are finite values at each node. Equation (3) takes into account the channel state and buffer availability at each node via weight adaptation. The following two lemmas provide support to (3) [2].

Lemma 1: If the weights are updated using (3) for a sufficiently long interval $[t_1, t_2]$, then the weight error $\tilde{\phi}_{ij}(k+1)$ is bounded, provided $|\alpha| < 1$.

Lemma 2: The actual weights $\hat{\phi}_{ij}$ at each node using (3) converge close to their target values in a finite time.

To calculate the backoff interval and to implement the scheduling scheme, the updated weights at each node have to be transmitted in the data frame of the MAC protocol. To enable this, changes are made to the data packet header to accommodate the current weight of the packet. Whenever a packet is received, the current weight is used to update the

weights dynamically using (3). Then the weight field in the packet header is replaced with the updated weight.

2) MAC Protocol - Dynamic Backoff Intervals

The proposed ADFS protocol uses the CSMA/CA scheme similar to the IEEE 802.11 protocol. When multiple nodes of a wireless network compete to access the shared channel the selection of the backoff interval plays a critical role in deciding which node is granted access to the channel. In order to achieve global fairness, the nodes must access the channel in a fair manner.

The proposed ADFS scheme is implemented as a novel MAC protocol in order to provide access of the shared medium by adjusting the dynamic backoff intervals. ADFS calculates the backoff interval relative to the weight of the packet. Since the weights are updated using (3), the backoff interval is also updated at each node. Backoff interval, BI_{ij}, for i^{th} flow j^{th} packet with packet length l_{ij} and weight ϕ_{ij} is defined as

$$BI_{ij} = \left\lfloor \rho * SF * \frac{l_{ij}}{\phi_{ij}} \right\rfloor \tag{5}$$

where SF is the scaling factor and ρ is a random variable with mean 1. Collision handling mechanism is incorporated similar to the one in [7]. This results in fair allocation of the bandwidth. Next the fairness guarantee is presented with relevant theorems.

B. Fairness Guarantee

To prove that ADFS is fair, the bound on $\left| \frac{W_f(t_1,t_2)}{\phi_f} - \frac{W_m(t_1,t_2)}{\phi_m} \right|$ has to be obtained for a sufficiently long interval $[t_1,t_2]$ in which both flows, f and m, are backlogged, the proof for this can be found in [2].

Theorem 1: For any interval $[t_1, t_2]$ in which flows f and m are backlogged during the entire interval, the difference in the service received by two flows at a ADFS wireless node is given as

$$\left| \frac{W_f(t_1,t_2)}{\phi_{f,l}} - \frac{W_m(t_1,t_2)}{\phi_{m,l}} \right| \le \frac{l_f^{\max}}{\phi_{f,l}} + \frac{l_m^{\max}}{\phi_{m,l}} \tag{6}$$

Theorem 2: If Q is the set of flows served by an ADFS node following FC service model with parameters $(\lambda(t_1, t_2), \psi(\lambda))$, and $\sum_{n\in Q} \phi_{n,l} \le \lambda(t_1, t_2)$, then for all intervals $[t_1, t_2]$ in which flow f is backlogged throughout the interval, $W_f(t_1, t_2)$ is given as

$$W_f(t_1,t_2) \ge \phi_{f,l}(t_2 - t_1) - \phi_{f,l}\frac{\sum_{n\in Q} l_n^{\max}}{\lambda(t_1,t_2)} - \phi_{f,l}\frac{\psi(\lambda)}{\lambda(t_1,t_2)} - l_f^{\max} \tag{7}$$

Theorem 3: If Q is the set of flows served by an ADFS node following EBF service model with parameters $(\lambda(t_1, t_2), B, \omega, \psi(\lambda)), \gamma \ge 0$, and $\sum_{n\in Q} R_n(v) \le \lambda(t_1, t_2)$ for all v then the departure time of packet P_f^j at the node, denoted by $T_d(P_f^j)$, is given by

$$P\left(T_d(P_f^j) \le T_a(P_f^j, \phi_{f,j}) + \sum_{n\in Q \wedge n\neq f} \frac{l_n^{\max}}{\lambda(t_1,t_2)} + \frac{l_f^j}{\lambda(t_1,t_2)} + \frac{\psi(\lambda)}{\lambda(t_1,t_2)} + \frac{\gamma}{\lambda(t_1,t_2)} \right) \ge 1 - Be^{-\omega\gamma} \tag{8}$$

C. Overhead Analysis

Analysis is also performed to estimate the data transmission overhead in the case of the hardware implementation. The additional overhead in ADFS protocol is due to inclusion of the current weight value and time-stamp in the header of each packet. In the implementation the weight was encoded using 8-bits and the time-stamp using 32-bit variables. This information then added 5 bytes of overhead to each packet. This means a 5% overhead for the packet. Additional overhead is introduced due to the interface to the 802.15.4 module. Packetization of the data for the radio interface adds 7 more bytes of overhead. This results in another 7% overhead for a total of 12%. Addition of the ADFS protocol overhead is negligible compared to the benefits of the fairness guarantee and improved QoS.

D. Initial Weights Selection

The initial weights are selected according on the user specified QoS. Initial weights are assigned to the flows based on the class they belong to. The flows within the same class are assigned the same initial weight. For instance, in some simulations [2] (including both star and random topologies) the flows are classified into 4 different classes. On the other hand, for some simulation scenarios with n flows, it is assumed that all the flows belong to the same class and therefore the initial weight assigned to each flow is *1/n* and all the flows are treated equally. In both cases, the initial weights are selected in such a way that the overall sum of weights of all the flows is equal to unity.

Initial flow weights are used to assign a target QoS. Moreover, the proposed ADFS protocol uses dynamic weight adaptation to react to the varying channel state and network topology. Then the goal is to achieve an overall fair allocation of bandwidth proportional to the initial weights of the flows.

III. Hardware Implementation Description

In this section, an overview of the hardware implementation of the ADFS scheduling protocol is given. Use of customized hardware for development of sensing, processing, and networking is also presented. A description of capabilities, limitations, and support for networking applications is next presented. Also, in this section an overview of the software

architecture is given with respect to the ADFS protocol and its requirements on the hardware.

A. Hardware Description

Hardware for implementation of the ADFS was selected to be energy conservative, performance oriented, and of small form-factor. Use of Silicon Laboratories® 8051 variant hardware was selected for its ability to provide fast 8-bit processing, low-power consumption, and ease of interfacing to peripheral hardware components. A Maxstream XBee™ RF module was also employed for this work. The use of external RAM (XRAM), UART interfacing, and A/D sensing allow the microcontroller to perform the tasks needed for a sensor node platform. Next, a treatment of the hardware capabilities and limitations will be given.

1) Considerations and Limitations

Hardware implementation of any algorithm is constrained by the limitations of the hardware. Use of specific hardware must be weighed against the precision, speed, and criticality of an algorithm's implementation. For this protocol, low-power consumption was given the highest priority. In turn, the demand for low power limits the types of processor architectures that can be deployed. The selection of the Silicon Laboratories® 8051 variants was based on these criteria. Limitations for the implementation that are incurred through the use of the 8051 variant family are a small memory space and a maximum processing speed. In the next section, a description of the specifications for the hardware implemented nodes will be given.

B. Architecture of the Hardware System and Software

This section outlines the hardware and software components of the ADFS implementation. A presentation of hardware capabilities is given. The software implementation is also discussed in this section. Software architecture, control-flow, and hardware implications are shown. In this section the system architecture of the nodes is discussed.

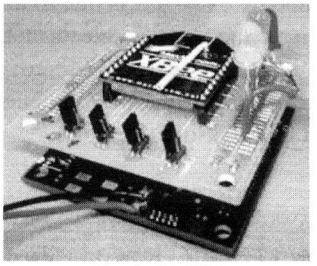

Fig. 1. G4-SSN

1) Sensor Node Hardware

The Generation-4 Smart Sensor Node (G4-SSN), seen in Fig. 1, was originally developed at UMR and subsequently updated at St. Louis University. The G4-SSN has various abilities for sensing and processing. The former include strain gauges, accelerometers, thermocouples, and general A/D sensing. The later include analog filtering, compact flash (CF) memory interfacing, and 8-bit data processing at a maximum of 100 MIPS.

2) G4-SSN Capabilities

The abilities of the UMR/SLU motes nodes are shown in Table I. As seen in the table the G4-SSN provides powerful 8-bit processing, a suitable amount of RAM, and a low-power small form-factor. Another strong point of the G4-SSN is the available code space found on the Silicon Laboratories® C8051F12x variant.

TABLE I
G4-SSN CAPABILITIES

	Ic @ 3.3V [mA]	Flash Memory [bytes]	RAM [bytes]	ADC Sampling Rate [kHz]	Form-Factor	MIPS
G4-SSN	35	128k	8448 bytes	100 @ 10/12-bit	100-pin LQFP	100

ADFS requires that the sensor nodes are synchronized. For this reason a test was performed on the G4-SSN real-time clock (RTC) capabilities. Experimentation consisted of a statistical analysis of the RTC accuracy. Extended use of the RTC without re-synchronization can cause drift to occur and this drift must be quantified to provide a confidence measure of the RTC. A 32.768 kHz quartz crystal is used to feed a timer on the 8051 and is used at the time-base of the RTC. The RTC was allowed to run for 10 minutes and the results are tabulated in Table II.

TABLE II
G4-SSN RTC DRIFT TESTING RESULTS

Test Time [min]	Δt [sec]	Variance [μ-sec]	Mean [sec]	STD [sec]	Error [sec]
10	0.05	50.45	0.0504	0.007	0.515

As seen in the table the RTC has a drift error of around a half-second over ten minutes, this translates to 3.5 seconds per hour. In the context of this application this drift is acceptable as the RTC will be synchronized with the BS every 30 seconds.

IV. HARDWARE IMPLEMENTATION RESULTS

In this section hardware implementation results are shown. Using the 802.15.4 standard with 250kbps RF data bandwidth the scheduling algorithm is tested. CBR traffic is generated on the source nodes and it routed to the base station via OEDSR [1]. The nodes internally provide 38.4 kbps throughput to the 802.15.4 module. There is no data aggregation, or data fusion, performed yet in the network since the considered application requires data from independent locations whereas the proposed scenario is a good test for queuing schemes for testing fairness. Due to hardware limitations of the 802.15.4 module interfacing the back-off interval time slots are constrained to a minimum of 15 mili-seconds. This limits the overall performance of the implementation; however the issue can be addressed in future work using the Chipcon CC2420.

Testing of the hardware implementation is now discussed. The results were obtained by use of a star topology with 5 source nodes and a single CH. CBR traffic is generated at each source node and the initial weight of each flow is equal to 1/5 and a value of α= 0.4, and β=0.6. Other parameters include SF=0.032 and the packet length of maximum 100 bytes with an 88 byte data payload. During testing the base station is used to record network activity for analysis. Performance of the ADFS implementation is evaluated using the exponential back-off scheme and drop-tail queuing. A comparison of these two methods shows the performance increase in the ADFS enabled network.

The ADFS scheduling scheme takes into account the weight of the packets and proportionally allocated bandwidth to the flows. In contrast, the networks without ADFS functionality lack the ability to differentiate QoS in this manner. Thus, poor fairness is observed in networks without ADFS.

In Table III the results for throughput and fairness index (FI) are shown. The fairness index (FI) [9] of the network is calculated using (9) and is used as a metric to further evaluate the performance of the ADFS protocol.

$$FI = \left(\sum_f \frac{T_f}{\phi_f}\right)^2 \Big/ \eta * \sum_f \left(\frac{T_f}{\phi_f}\right)^2 \qquad (9)$$

where T_f is the throughput of flow f, ϕ_f is the initial weight of flow f, and η is the number of flows.

In case of ADFS implementation, the throughput is higher for every flow since it is able to maintain a steady and proportional traffic thus reducing buffer overflows. The reference scheme is not able to distribute the available bandwidth proportionally to all flows, thus the buffer overflows occurs more often than in case of ADFS protocol. Overall the ADFS network achieves a *13.3%* increase in the throughput over the First-In-First-Out (FIFO) queued scheme.

Moreover, a larger FI value is observed for the ADFS implementation than for the reference scheme. The ADFS scheme allocates the resources proportionally to the packet weight thus adapting to a changing channel and network state.

TABLE III
THROUGHPUT AND FAIRNESS INDEX RESULTS FOR HARDWARE TESTING

	Flow 1 [kB/s]	Flow 2 [kB/s]	Flow 3 [kB/s]	Flow 4 [kB/s]	Flow 5 [kB/s]	Overall	FI
ADFS	88.8	85.1	84.7	81.7	81.0	**84.3**	**0. 9989**
FIFO	83.8	67.7	76.7	68.7	75.3	**74.4**	**0. 9938**

In Figs. 2 and 3 the throughput of each flow for a star-topology using drop-tail queuing and proposed ADFS respectively are shown. The non-ADFS protocol experiences high throughput variance, while the ADFS system provides more constant performance through establishing fairness of the scheduling. Moreover, ADFS achieves higher overall throughput since it allows all flows to share bandwidth in a more even manner over time when compared to FIFO queued network. This clearly demonstrates the fairness of ADFS protocol.

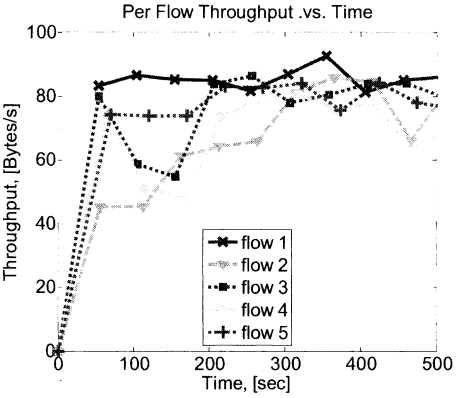

Fig.2 Throughput of FIFO Queued Network

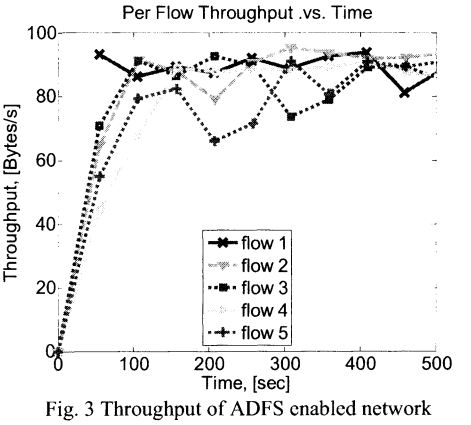

Fig. 3 Throughput of ADFS enabled network

In Table IV the results for average end-to-end delay and its standard variation (std) are shown. The mean delay values for both protocols are similar with ADFS achieving smaller delay for 4 flows out of 5 flows when compared to the FIFO protocol with Drop tail queue and exponential back-off. Overall the improvement of 2% in end to end delay is observed for the ADFS scheme over the FIFO queue even though there is an additional overhead with the proposed protocol. The advantage of ADFS is due to a fair allocation of radio resources since the ADFS selects a back-off interval proportionally to a packet weight. Moreover, the standard deviation is higher in case of the FIFO queued scheme due to higher variation in the back-off.

TABLE III
DELAY RESULTS FOR HARDWARE TESTING

[sec]		Flow 1	Flow 2	Flow 3	Flow 4	Flow 5	Overall
ADFS	mean	11.24	11.41	11.69	11.56	11.30	11.44
	std	1.011	1.167	1.068	1.015	0.834	0.17
FIFO	mean	11.11	11.60	11.71	11.61	12.30	11.66
	std	1.434	1.12	1.459	1.212	1.512	0.38

The results show that the proposed protocol can achieve fair allocation of bandwidth with a 13.3% increase in throughput, slightly lower end-to-end delays and delay variations reduced

by 55% (std), resulting in a better QoS. This shows the ability of the ADFS scheduler to be effectively applied to WSN systems to enhance network performance

V. UMR/SLU Mote Future Direction

Future work on the UMR/SLU motes includes increased capabilities for RF communications, memory density, and efficient networking implementations. Specifically, a discussion of the current state and direction of the RF communication module is now given. Currently a Maxstream XBee™ module is used to implement the RF layer of the G4-SSN. Future introduction of the Chipcon CC2420 transceiver chipset to the G4-SSN is now being developed at UMR. With this introduction several advantages are perceived. The primary perceived advantage is direct access to the physical layer in terms of parameter access and control. For example, direct and real-time sensing of channel access (CA) state will reduce back-off interval slots. Next, power savings in are expected with the CC2420 on the order of 62%. Additionally, more finely adjustable RF transmission power levels will allow for distributed power control in the WSN. The CC2420 also provides a smaller physical footprint to enable higher level of integration and miniaturization. Finally, the subsequent frame processing step on the XBee™ is avoided and frames are passed directly to the transceiver.

VI. Conclusions

In this paper, the implementation of a novel adaptive and distributed fair scheduling (ADFS) scheme for WSNs is discussed. The objective is to evaluate the hardware capabilities and implementation viability. The results indicate where hardware constraints must be alleviated, thus providing direction for future hardware redesign that meets the requirements of the ADFS protocol. More topologies will be utilized to evaluate the protocol.

In the proposed protocol, weights are updated and the updated weights are used in making the scheduling decisions and also in the calculation of the back-off intervals in the CSMA/CA paradigm. Initial weights are assigned to the flows based on the service they expect from the network.

The performance of the proposed ADFS scheme is presented and analytically evaluated. The effectiveness of the proposed ADFS protocol was evaluated through hardware experimentation. The results show that the proposed protocol can achieve fair allocation of bandwidth with a 13.3% increase in throughput, lower end-to-end delays and delay variations reduced by 55% (std), resulting in a better QoS. This shows the ability of the ADFS scheduler to be effectively applied to WSN systems to enhance network performance; however further study is needed to address the current hardware constraints and improve overall performance of the implementation.

Acknowledgment

Thanks go to Dr. Kyle Mitchell of St. Louis University, St. Louis, MO. His continued support and advice on the G4-SSN system and hardware implementation issues is much appreciated.

References

[1] S. Ratnaraj, S. Jagannathan and V. Rao, "OEDSR: Optimal Energy Delay Subnet Routing Protocol for Wireless Sensor Networks", Proc. of the IEEE Conference on Sensing, Networking, and Control, pp. 787-792, April 2006.

[2] N. Regatte and S. Jagannathan, "Adaptive and Distributed Fair Scheduling Scheme For Wireless Adhoc Networks", Proc. of the Word Wireless Congress, pp. 101-106, May 2004.

[3] P. Goyal, H.M. Vin, and H. Cheng, "Start-Time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks", IEEE/ACM Transactions on Networking, October 1997, vol. 5, pp. 690 – 704.

[4] S.J. Golestani, "A Self-clocked Fair Queuing Scheme for Broadband Applications", IEEE INFOCOM'94 Networking for Global Communications, June 1994, vol. 2, pp. 636 – 646.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", Proc. ACM SIGCOMM'89, pp. 3-12.

[6] J.C.R. Bennett and Hui Zhang, "WF2Q: Worst-Case Fair Weighted Fair Queuing", IEEE INFOCOM'96, March 1996, Vol. 1, pp. 120 – 128.

[7] N.H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN", 6th Annual International Conference on Mobile Computing & Networking, August 2000, pp. 167-178.

[8] H. Luo, P. Medvedev, J. Cheng, and S. Lu, "A Self-Coordinating Approach to Distributed Fair Queueing in Adhoc Wireless Networks", IEEE INFOCOM, 2001, pp. 1370 – 1379.

[9] R. Jain, G. Babic, B. Nagendra, and C. Lam, "Fairness, Call Establishment Latency and other Performance Metrics", Tech. Rep. ATM_Forum/96-1173, August 1996.