CONFLICT-FREE TRANSMISSION SCHEDULING FOR REAL-TIME QUERIES IN WIRELESS SENSOR NETWORKS

[1]NAIF ABDUL KHADER, [2]N.JAYARAJ

[1]Dept. of Electronics and Communication Engineering, The Oxford College of Engineering, Bengaluru, India
[2]Assistant Professor, Dept. of Electronics & Communication Engg., The Oxford College of Engineering, Bengaluru, India
E-mail: [1]nak_dasimplest@hotmail.com, [2]n_jaya_raj@yahoo.com

Abstract- In the past we have seen the emergence of wireless cyber-physical systems that must support real-time queries of physical environments through wireless sensor networks. This paper proposes Real-Time Query Scheduling (RTQS), a novel approach to conflict-free transmission scheduling for real-time queries in wireless sensor networks. First, it is shown that there is an inherent tradeoff between latency and real-time capacity in query scheduling. Then three new real-time schedulers are developed namely non-preemptive query scheduler, preemptive query scheduler and the slack stealing query scheduler. The non-preemptive query scheduler supports high real-time capacity but cannot provide low response times to high-priority queries due to priority inversions. The preemptive query scheduler eliminates priority inversions at the cost of reduced capacity. The slack stealing query scheduler combines the benefits of the preemptive and non-preemptive schedulers to improve the capacity while alongside meeting the end-to-end deadlines of queries. The schedulability analysis for each scheduler will be provided. The analysis and advantages of this approach are validated through NS2 simulations.

Keywords- Query scheduling, Schedulability analysis, Wireless sensor networks

I. INTRODUCTION

In the past we have seen the emergence of wireless cyber physical systems that must support real-time data collection at high data rates over wireless sensor networks (WSNs). Representative examples include emergency response, structural health monitoring, and process measurement and control. Such systems pose significant communication challenges. First, the system must handle multiple traffic classes with different deadlines. For example, during an earthquake, acceleration data from sensors deployed on a building must be delivered to the base station in a timely manner to detect any structural damage. Such traffic should have higher priority than temperature data collected for climate control. Thus, a WSN protocol should provide effective prioritization between traffic classes while meeting their respective deadlines. Second, the system must support high throughput because sensors may generate a high workload. For example, structural health monitoring requires numerous accelerometers to be sampled at high rates generating high network loads. Furthermore, because the system must deliver data to base stations within their deadlines, it is important for the system to achieve predictable and bounded end-to-end latencies.

Many cyber-physical systems use query services to periodically collect data from sensors to a base station over multi-hop wireless networks. In this paper, we propose Real-Time Query Scheduling (RTQS), a transmission scheduling approach for real-time queries in WSNs. RTQS includes three real-time query schedulers that exploit the unique characteristics of WSN queries including many-to-one communication and periodic sampling. We provide upper bounds on the response times of real-time queries for each scheduler. A unique aspect of our approach is that it bridges the gap between wireless sensor networks and schedulability analysis techniques, which have traditionally been applied to real-time processor scheduling.

This paper makes four contributions: First, it is shown that real-time query scheduling has an inherent tradeoff between latency and real-time capacity. Second, we developed three schedulers:

1) The non-preemptive query scheduler (NQS) achieves high capacity but cannot provide low response times to high-priority queries due to priority inversions.

2) The preemptive query scheduler (PQS) eliminates priority inversions at the cost of reduced capacity.

3) The slack stealing scheduler combines the advantages of preemptive and non-preemptive schedulers to improve real time capacity while meeting query deadlines. Thirdly, we provide a schedulability analysis for each scheduler that enables predictable real-time query services through online admission and rate control. Finally, the advantages of RTQS over existing contention-based and TDMA-based protocols in terms of real-time performance through simulations are demonstrated.

This paper is organized as follows: Section 2 reviews the literature survey. Section 3 describes the system models-query and network models. Sections 4 gives detail about the design and analysis of RTQS. Section 5 reviews the practical consideration. Section 6

provides simulation results. Section 7 concludes the paper.

II. LITERATURE SURVEY

The adoption of the WirelessHART standard has renewed the interest in real-time communication. A number of scheduling protocols have been proposed for effectively scheduling packet transmissions under the WirelessHART model. These solutions adopt a centralized approach to the construction of transmission schedules and do not support spatial reuse. As a result, the scalability of such approaches is limited. In contrast, RTQS overcomes these limitations by supporting spatial reuse and provides decentralized query schedulers.

In an earlier work, DCQS, a transmission scheduling technique for WSN queries. In contrast to traditional TDMA protocols designed to support general workloads, DCQS is specifically designed to exploit communication patterns and temporal properties of queries in WSNs. This allows DCQS to achieve high throughput and low latency. However, DCQS does not support query prioritization or real-time communication, which is the focus of this paper.

Real-time communication protocols adopt contention-based or TDMA-based approaches. Contention-based protocols support real-time communication through probabilistic differentiation. This is usually achieved by adapting the contention window and/or the initial back-off of the CSMA/ CA mechanism. Overviews of these approaches are presented in. Rate and admission control may be used with contention-based protocols to handle congestion. However, these approaches are unsuitable for high data rate and real-time application because 1) they provide highly variable communication latencies due to their random back-off mechanisms and 2) achieve low throughput under heavy load due to excessive channel contention.

In multi-hop networks, real-time communication may also be achieved through real-time routing. For example, the SPEED protocol builds on geographic routing to deliver packets at a uniform velocity in a multi-hop network. SPEED was extended to support multiple delivery speeds through multipath routing. These protocols usually build upon contention-based MAC protocols and, as a result, inherit their drawbacks.

TDMA protocols can achieve predictable communication latencies and higher throughput than contention-based protocols under heavy load. Several real-time TDMA protocols were designed for single-hop networks. IEEE 802.15.4 standard specifies Guaranteed Time Slots (GTS) for achieving predictable delays in single-hop networks. A flexible slot reservation mechanism was proposed, where slots are allocated based on delay or bandwidth requirements.

Symbol	Description
l, m, and h	Indices for low/medium/high priority queries
P_l	Period of query l
D_l	Deadline of query l
R_l	Response time of query l
$\square_l[i]$	Workload of query l on node i
$\square_l$	Start time of query l
u,v	Instance number of a query
$I_{l,u}$	The u^{th} instance of query l
$r_{l,u}$	Release time of instance $I_{l,u}$
$\rightarrow_{ab}$	Directed edge from a to b
L_l	Length of plan of query l
$T_l[i]$	Transmissions in step i in the plan of query l
T_l and V_l	The actual and reverse plan of query l
$I_{l,u}.i$	Step number i in the plan of instance (l,u)

Fig 1. Table of symbols

III. RTQS ALGORITHM WORKING PRINCIPLE

The RTQS approach relies on two components: a planner and a scheduler. The planner constructs a plan for executing all the instances of a query. A plan is an ordered sequence of steps, each comprised of a set of conflict-free transmissions. The scheduler runs on every node and determines the time slot in which each step of a plan is executed. To improve the capacity, the scheduler may execute steps from multiple instances in the same slot as long as they do not conflict. RTQS works as follows-

1) When a query is submitted, RTQS identifies a plan for its execution. RTQS usually reuses a previously constructed plan for the new query as multiple queries may be executed using the same plan, When no plan may be reused, the planner constructs a new one using the distributed planner.

2)RTQS determines if a query meets its deadline using the schedule analysis. The schedule analysis is performed on the base station using information collected during the distributed plan construction. If the query is schedulable, the parameters of the query are disseminated; otherwise, the query is rejected. Note that plans are cached even when the schedule analysis fails.

3)At runtime, the scheduler running on each node executes all admitted queries in a distributed fashion.

IV. PLAN CONSTRUCTION

4.1.Constructing Plans

Plan properties. A plan has two properties:

1) Each node is assigned sufficient steps to transmit its entire data report. $T_l[i]$ denote the set of

transmissions assigned to step i ($0 \leq i < L_l$) in the plan of query l, where L_l is the length of the plan.

2) The plan also must respect the precedence constraints introduced by aggregation: a node is assigned to transmit in a later step than any of its children. We remind the reader that RTQS will use packet merging as its default data aggregation function. This design decision is a tradeoff between query latency and energy consumption. The enforcement of the precedence constraints introduced by packet merging may increase query latency (i.e., length of the plan) as shorter plans may be constructed when these constraints are not enforced. However, the use of packet merging has two key advantages:

1) it reduces the communication workload and

2) the constructed transmission schedules have long contiguous periods of activity/inactivity: the node transitions from a sleep state to the active state just-in-time to receive the data from its children and transitions back to sleep after it completes collecting data from its children and relaying it to its parent. Such schedules are efficient because they reduce the wasted energy in transitions between sleep and active states.

Distributed planner. Since a node waits to receive the data reports from its children, the planner may reduce the query latency by assigning the transmissions of a node with a larger depth in the routing tree to an earlier step of the plan. This strategy reduces the query latency because it reduces the time a node waits for the data reports from all its children. More sophisticated heuristics may be developed. For example, we may account for the case when the routing tree is unbalanced to construct shorter plans. A difficulty associated with this approach is that the protocol would have to keep track of the size of the sub-trees as the topology changes. In contrast, our goal is to develop a heuristic that performs well in realistic scenarios (see Section 6) and allows for a simple distributed implementation on resource constrained WSN nodes rather than focusing on the construction of optimal schedules. A detailed description of the planner may be found in [16]; here, we overview the distributed planner and analyze its communication complexity. The planner works in two stages. In the first stage, the planner constructs a reversed plan (V) in which a node's transmission is assigned to an earlier step than its children. In the second stage, it constructs the actual plan (T) by reversing the order of the steps to enforce the precedence constraints. The planner associates with each node a priority given by the triple (depth, child count, ID) with the root having the highest priority. A node n waits until it is the highest priority and unscheduled node within its one-hop neighborhood. When this occurs, n collects the local plans of its two-hop neighbors that have higher priority. Using this information, n determines the steps in which its children will transmit: if n transmits (to its parent) in step t, each of its children will be assigned in the first step after t that does not conflict with the transmissions of the higher priority nodes. A child is assigned in sufficient steps to meet its workload demand. The first stage completes with node n disseminating its local plan to its two-hop neighbors. The second stage involves the reversal of the plan, which requires nodes to know the length of the plan. This is accomplished in two steps. By definition, a plan's length is the maximum step number in which a node transmits. This value may be computed at the base station using standard data aggregation with max as the aggregation function. Next, the plan's length is disseminated to all nodes by taking advantage of the already established routing tree. Once the value of the plan's length is received by a node, the reversal of the plan is a local operation. Upon the completion of the distributed planning process, each node in the network stores the time step in which it and its children transmit as well as the length of the plan. Note that the base station does not have a global view of the network but rather the plans are stored in a distributed manner.

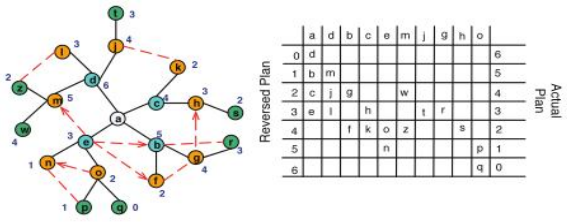


Fig 2. IC Graph and associated plan

Fig. 2 shows an IC graph and the actual and reversed plans constructed by the planner. The solid lines indicate the communication edges in the routing tree while the dashed lines indicate interference edges. Node a is the base station. The plan in Fig. 2 is constructed assuming that the data report generated by a node can be transmitted in a single step for each instance. The constructed plan meets the two constraints previously specified: 1) The planner assigns conflict-free transmissions in each step. For example, transmissions $\underset{ne}{\rightarrow}$ and $\underset{po}{\rightarrow}$ are assigned to step $T_l[1]$ because they do not conflict with each other. 2) The precedence constraints introduced by aggregation are respected. For example, nodes p and q are assigned in earlier steps than their parent o.

The message complexity of the distributed planner is $2(N_2+N)$, where N_2 is the size of the two-hop neighbors and N is the number of nodes (i.e., $N = |E|$). The first stage requires a node to collect the local plans of its two-hop neighbors and then disseminate its local plan to the same nodes. This contributes a total of $2*N_2$ messages. The second stage requires each node to transmit two packets, one during the computation of the plan length and the other during

its dissemination. This contributes an additional 2*N messages. The overhead of the distributed planner was evaluated through simulations in. Simulations results indicate that the cost of plan construction dominates the overall message overhead exceeding the cost of constructing the routing tree or the IC graph. More importantly, the simulations show the distributed planner scales well: message complexity increases linearly with the number of nodes when the node density remains constant. This is consistent with the above analysis.

Plan reuse. Due to the significant cost associated with constructing plans, we must minimize the number of constructed plans. We observed that the plan of a query l depends on the IC graph, the set of source nodes, and the aggregation function. Query instances executed at different times may need different plans if the IC graph changes. However, to handle dynamics in channel conditions, RTQS can construct plans that are robust against certain variations in the IC graph (as discussed in [19]). This allows instances executed at different times to be executed according to the same plan. Moreover, note that queries with the same aggregation function and sources but with different periods, start times, or can be executed according to the same plan. Furthermore, even queries with different aggregation functions may be executed according to the same plan. Let $\square_l[i]$ be the number of slots node i needs to transmit its data report to its parent for an instance of query l. If the planner constructs a plan for a query l, the same plan can be reused to execute a query h if $\square_l[i] = \square_h[i]$ for all nodes i. Examples of queries that share the same plan are the queries for the maximum temperature and the average humidity in a building. For both queries, a node transmits one data report in a single step (i.e., $\square_{max}[i] = \square_{avg}[i] = 1$ for all nodes i) if the slot size is sufficiently large to transmit a packet with two values. For the max query, the outgoing packet includes the maximum value of the data reports from itself and its children. For the average query, the packet includes the sum of the values and the number of data sources that contributed to the sum. We say that two queries belong to the same query class if they may be executed according to the same plan. Since queries with different temporal properties and aggregation functions may share a same plan, a WSN may only need to support a small number of query classes. This allows RTQS to amortize the cost of constructing a query plan over many queries and effectively reduces the overhead.

4.2 Overview of Scheduling Algorithm

RTQS supports both preemptive and non-preemptive query scheduling. NQS controls only the start of an instance: once an instance starts executing, the scheduler cannot preempt it.

In contrast, PQS will preempt an instance to allow a higher priority instance to execute when the two cannot be executed concurrently. The slack stealing scheduler determines dynamically whether to preempt instances. It avoids preemption to improve capacity when
it can still meet all deadlines, and performs preemption when needed to meet deadlines.

The scheduler executes a query instance according to the plan of its query. The scheduler improves the capacity by overlapping the transmissions of multiple instances (belonging to one or more queries) such that:

1) All steps executed in a slot are conflict free. Two steps of instances $I_{l,u}$ and $I_{h,v}$ are conflict free, that is $(I_{l,u}.i \parallel I_{h,v}.j)$, if all pairs of transmissions in $T_l[I_{l,u}.i] \cup T_h[I_{h,v}.j]$ are conflict free.

2) The steps of a plan are executed in order. If step $I_{l,u}.i$ is executed in slot s_i, step $I_{h,v}.j$ is executed in slot s_j and $s_j < s_i$ then $I_{l,u}.i > I_{h,v}.j$. This ensures that the precedence constraints required by aggregation are preserved. The local scheduler running on a node maintains a record of the start time, period, and priority of admitted queries. Plans are stored distributedly: each node knows when it transmits/receives packets and the plan's length.

The following is a brute-force approach for constructing a preemptive scheduler: in every slot s, a brute-force scheduler would consider the released instances in order of their priority and execute all steps that do not conflict in s. This solution has a high time complexity because each pair of steps must be checked for conflicts. Schedulers that have low time complexity are interested to be used as they will determine the steps to be in executed in a slot dynamically. To reduce the time complexity of the scheduler, the concept of minimum step distance is introduced (originally named the minimum inter-release time). Let $I_{l,u}.i$ and $I_{h,v}.j$ be two steps in the plans of instances $I_{l,u}$ and $I_{h,v}$. The step distance between $I_{l,u}.i$ and $I_{h,v}.j$ is defined as $|I_{l,u}.i - I_{h,v}.j|$. Then the minimum step distance $\Delta(l,h)$ is the smallest step distance between the instances $I_{l,u}$ and $I_{h,v}$ such that two steps $I_{l,u}.i$ and $I_{h,v}.j$ may be executed concurrently without conflict:

$$|I_{l,u}.i - I_{h,v}.j| \geq \Delta(l,h)$$
$$\Rightarrow (I_{l,u}.i \parallel I_{h,v}.j), \text{ for all } I_{l,u}.i < L_l ; I_{h,v}.j < L_h$$

where L_l and L_h are the plan lengths of queries l and h respectively.

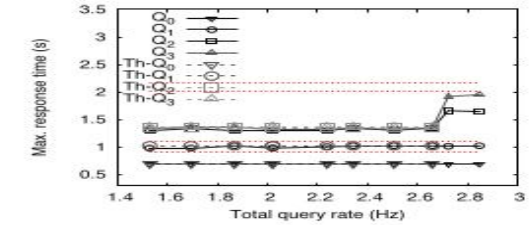
Thus, to ensure that no conflicting transmissions are executed in a slot, it suffices to enforce a minimum step distance between any two steps. The minimum step distance captures the degree of parallelism that may be achieved due to spatial reuse in a multihop WSN. Consider the case when $L = Lq = L_h$. In the worst case, when $\Delta(l,h) = L$, a single instance is executed in the network at a time. If $\Delta(l,h) = L/2$, then two instances can be executed in the network at the

same time. The minimum step distance Δ(l,h) depends on the IC graph and the plans of l and h. The number of minimum step distances that a scheduler stores is quadratic in the number of plans. Since a small number of plans is sufficient to meet the communication requirements of an application, then the number of step distances that must be stored is also small.

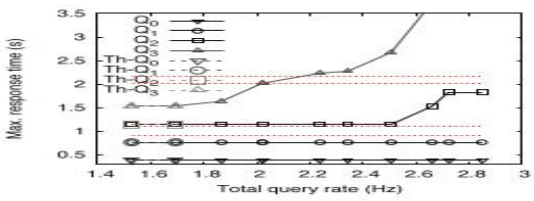
V. SIMULATIONS

In the following, we compare the RTQS algorithms under different workloads and validate our response time analysis. In all settings, we set the deadlines of the queries to be smaller or equal to the period as to match the conditions under for which we derived the schedulability analysis.

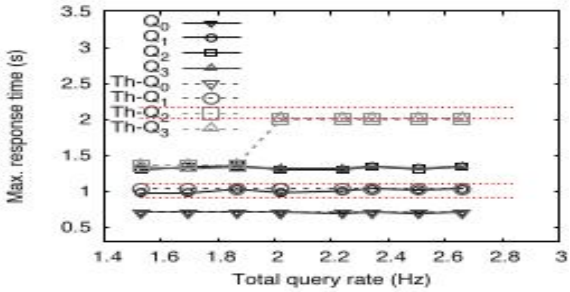
In this section, we compare the performance of all RTQS algorithms and validate their response time analysis. The ratios of their periods Q_0:Q_1:Q_2:Q_3 is 1.0:1.2:2.2:3.2. Following figures shows the response time of queries when workload is varied by changing query rates. Figs. 4.1a, 4.1b, and 4.1c show the measured and the theoretical maximum response times of NQS, PQS, and SQS under different total query rates. The dotted horizontal lines indicate the query deadlines. NQS meets all deadlines when the total query rate is within 2.85 Hz. In contrast, PQS supports a lower query rate since Q3 misses its deadline when the total query rate is 2.23 Hz. The long response time of Q3 is due to the high preemption cost suffered by the low-priority queries under PQS. This indicates that PQS is unsuitable for workloads in which the low-priority queries have tight deadlines. Similar to NQS, SQS can support a higher query rate than PQS without missing deadlines. In this experiment, the deadlines are lax, and hence, preemption is not necessary for meeting them. As such, SQS dynamically avoids preemption and the associated real-time capacity reduction. SQS achieves a slightly lower real-time capacity than NQS because it is limited by the conservative response time analysis.



(a) NQS response time



(b) PQS response time

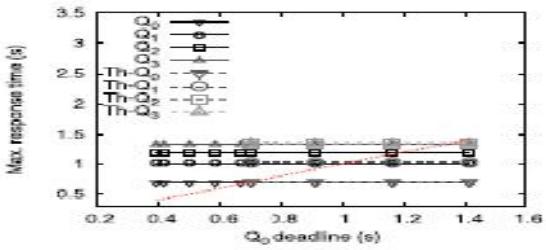


(c) SQS response time

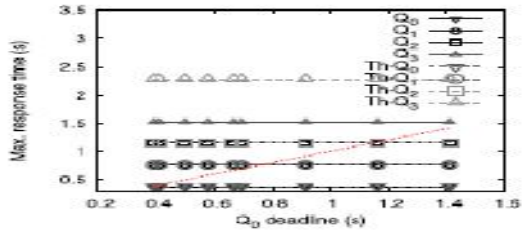
Fig 4 Response time of schedulers when workload is varied by changing query rates.

When the admission algorithm decides that the queries are unschedulable, it cannot find a slack assignment for the queries. Therefore, we cannot run SQS at a rate beyond its theoretical bound. In contrast, we may increase the rate further under NQS, which achieves a higher real time capacity than its theoretical bounds because its response time analysis is derived based on worst-case arrival patterns that do not always occur in our simulations.

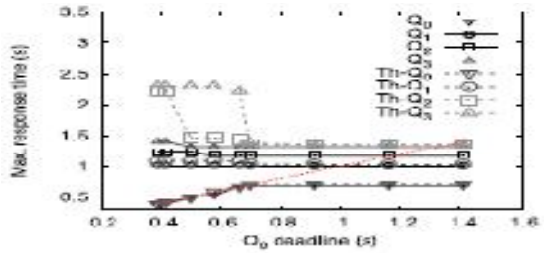
Following figures show the response time of queries when workload is varied by changing the deadlines of Q_0.



(a) NQS response time



(b) PQS response time



(c) SQS response time

Fig 5 Response time of schedulers when workload is varied by changing the deadlines of Q_0.

Figs. 5a, 5b, and 5c show the maximum response times of NQS, PQS, and SQS, respectively. For clarity, only Q_0's deadline is plotted because in this experiment the other queries always meet their deadlines. PQS meets Q_0's deadline when it is 0.39 s. In contrast, NQS meets its deadline only when Q0's deadline is bigger than 0.69 s. NQS misses Q_0's deadline when it is tight due to the priority inversion under non-preemptive scheduling. This indicates that NQS is unsuitable for high-priority queries with tight deadlines. Interestingly, under SQS, the response time of Q_0 changes depending on its deadline (Fig. 5.2c). As the deadline becomes tighter, the response time of Q_0 also decreases and remains below the deadline. It also can be seen there is an increase in the response times of the lower priority queries as Q_0's deadline is decreased. This is because as Q_0's deadline decreases the lower priority queries may steal less slack from Q_0. This shows that SQS adapts effectively based on query deadlines. Moreover, note that SQS provides smaller latencies for the lower priority instances than PQS. This is because SQS has a higher real-time capacity than PQS because it uses preemption only when it is necessary for meeting packet deadlines.

CONCLUSION

RTQS includes three real-time query schedulers that exploit the unique characteristics of WSN queries including many-to-one communication and periodic sampling. Upper bounds on the response times of real-time queries for each scheduler are provided. A unique aspect of this approach is that it bridges the gap between wireless sensor networks and schedulability analysis techniques, which have traditionally been applied to real-time processor scheduling. Firstly the inherent tradeoff between throughput and prioritization under conflict-free query scheduling is analyzed. Then the design and schedulability analysis of three new real-time scheduling algorithms for prioritized transmission scheduling is presented. It will observed that NQS achieves high throughput at the cost of priority inversion, PQS eliminates priority inversion at the cost of query throughput while SQS combines the advantages of NQS and PQS to achieve high query throughput while meeting query deadlines. NS2 simulations demonstrate that both NQS and PQS achieve significantly better real-time performance than representative contention-based and TDMA protocols. Moreover, SQS can maintain desirable real-time performance by adapting to deadlines. Real-time query scheduling provides a promising approach to provide predictable real-time queries in WSNs.

REFERENCES

[1] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-End Delay Analysis for Fixed Priority Scheduling in Wirelesshart Networks," Proc. IEEE 17th Real-Time and Embedded Technology and Applications Symp., 2010.

[2] H. Zhang, P. Soldati, and M. Johansson, "Optimal Link Scheduling and Channel Assignment for Convergecast in Linear WirelessHart Networks," Proc. Seventh Int'l Symp. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2009.

[3] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHart Networks," Proc. IEEE 31st Real-Time Systems Symp., 2010.

[4] O. Chipara, C. Lu, J. Stankovic, and G. Roman, "Dynamic Conflict- Free Transmission Scheduling for Sensor Network Queries," IEEE Trans. Mobile Computing, vol. 10, no. 5, pp. 734-748, May 2011. [3] Woodard D.L., Flynn P.J., ―Finger surface as a biometric identifier‖, CVIU, vol. 100, pp. 357–384, 2005.

[5] H. Zhu, M. Li, I. Chlamtac, and B. Prabhakaran, "A Survey of Quality of Service in IEEE 802.11 Networks," IEEE Wireless Comm., vol. 11, no. 4, pp. 6-14, Aug. 2004.

[6] W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee, "Distributed Mechanisms for Quality of Service in Wireless Lans," IEEE Wireless Comm., vol. 10, no. 3, pp. 26-34, June 2003.

[7] K. Karenos, V. Kalogeraki, and S. Krishnamurthy, "A Rate Control Framework for Supporting Multiple Classes of Traffic in Sensor Networks," Proc. IEEE 26th Real-Time Systems Symp., 2005

[8] G.-S. Ahn, A. Campbell, A. Veres, and L.-H. Sun, "Supporting Service Differentiation for Real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)," IEEE Trans. Mobile Computing, vol. 1, no. 3, pp. 192-207, July-Sept. 2002.

[9] M. Barry, A. Campbell, and A. Veres, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks," Proc. IEEE INFOCOM, 2001.

[10] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "Speed: a StatelessProtocol for Real-Time Communication in Sensor Networks," Proc. Int'l Conf. Distributed Computing Systems, 2003.

[11] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: Multipath Multi-Speed Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks," IEEE Trans. Mobile Computing, vol. 5, no. 6, pp. 738-754, June 2006.

[12] A. Koubaa, M. Alves, and E. Tovar, "I-GAME: An Implicit GTS Allocation Mechanism in IEEE 802.15.4 for Time-Sensitive Wireless Sensor Networks," Proc. Euromicro Conf. Real-Time Systems, 2006.

★★★