Dynamic Multilevel Priority Packet Scheduling Scheme for Wireless Sensor Network

Nidal Nasser, Lutful Karim, and Tarik Taleb

***Abstract*—Scheduling different types of packets, such as real-time and non-real-time data packets, at sensor nodes with resource constraints in Wireless Sensor Networks (WSN) is of vital importance to reduce sensors' energy consumptions and end-to-end data transmission delays. Most of the existing packet-scheduling mechanisms of WSN use First Come First Served (FCFS), non-preemptive priority and preemptive priority scheduling algorithms. These algorithms incur a high processing overhead and long end-to-end data transmission delay due to the FCFS concept, starvation of high priority real-time data packets due to the transmission of a large data packet in non-preemptive priority scheduling, starvation of non-real-time data packets due to the probable continuous arrival of real-time data in preemptive priority scheduling, and improper allocation of data packets to queues in multilevel queue scheduling algorithms. Moreover, these algorithms are not dynamic to the changing requirements of WSN applications since their scheduling policies are predetermined.**

In this paper, we propose a Dynamic Multilevel Priority (DMP) packet scheduling scheme. In the proposed scheme, each node, except those at the last level of the virtual hierarchy in the zone-based topology of WSN, has three levels of priority queues. Real-time packets are placed into the highest-priority queue and can preempt data packets in other queues. Non-real-time packets are placed into two other queues based on a certain threshold of their estimated processing time. Leaf nodes have two queues for real-time and non-real-time data packets since they do not receive data from other nodes and thus, reduce end-to-end delay. We evaluate the performance of the proposed DMP packet scheduling scheme through simulations for real-time and non-real-time data. Simulation results illustrate that the DMP packet scheduling scheme outperforms conventional schemes in terms of average data waiting time and end-to-end delay.

***Index Terms*—Wireless sensor network, packet scheduling, preemptive priority scheduling, non-preemptive priority scheduling, real-time, non-real-time, data waiting time, FCFS.**

I. Introduction

AMONG many network design issues, such as routing protocols and data aggregation, that reduce sensor energy consumption and data transmission delay, packet scheduling (interchangeably use as task scheduling) at sensor nodes is highly important since it ensures delivery of different types of data packets based on their priority and fairness with a minimum latency. For instance, data sensed for real-time applications have higher priority than data sensed for non-real-time applications. Though extensive research for scheduling the sleep-wake times of sensor nodes has been conducted [1]–[18], only a few studies exist in the literature on the packet scheduling of sensor nodes [19]–[22] that schedule the processing of data packets available at a sensor node and also reduces energy consumptions. Indeed, most existing Wireless Sensor Network (WSN) operating systems use First Come First Serve (FCFS) [23] schedulers that process data packets in the order of their arrival time and, thus, require a lot of time to be delivered to a relevant base station (BS). However, to be meaningful, sensed data have to reach the BS within a specific time period or before the expiration of a deadline. Additionally, real-time emergency data should be delivered to BS with the shortest possible end-to-end delay. Hence, intermediate nodes require changing the delivery order of data packets in their ready queue based on their importance (e.g., real or non-real time) and delivery deadline. Furthermore, most existing packet scheduling algorithms of WSN are neither dynamic nor suitable for large scale applications since these schedulers are predetermined and static, and cannot be changed in response to a change in the application requirements or environments [24]–[26]. For example, in many real-time applications, a real-time priority scheduler is statically used and cannot be changed during the operation of WSN applications.

In this paper, we propose a Dynamic Multilevel Priority (DMP) packet scheduling scheme for WSNs in which sensor nodes are virtually organized into a hierarchical structure. Nodes that have the same hop distance from the BS are considered to be located at the same hierarchical level. Data packets sensed by nodes at different levels are processed using a TDMA scheme. For instance, nodes that are located at the lowest level and one level upper to the lowest level can be allocated timeslots 1 and 2, respectively. Each node maintains three levels of priority queues. This is because we classify data packets as (i) real-time (priority 1), (ii) non-real-time remote data packet that are received from lower level nodes (priority 2), and (iii) non-real-time local data packets that are sensed at the node itself (priority 3). Non-real-time data traffic with the same priority are processed using the shortest job first (SJF) scheduler scheme since it is very efficient in terms of average task waiting time [23].

The remainder of the paper is organized as follows. In Section II, we discuss several existing WSN packet or task scheduling algorithms. Section III presents general assumptions and terminologies. Section IV presents the working principle and pseudo-code of the proposed DMP packet-

Manuscript received July 27, 2011; revised July 11, 2012; accepted December 2, 2012. The associate editor coordinating the review of this paper and approving it for publication was G. Zussman.

N. Nasser is with the Electrical & Computer Engineering Dept, College of Engineering, Alfaisal University, Saudi Arabia (e-mail: nnasser@alfaisal.edu).

L. Karim is with the School of Computer Science, University of Guelph, Canada (e-mail: lkarim@uoguelph.ca).

T. Taleb is with NEC Europe, Heidelberg, Germany (e-mail: talebtarik@ieee.org).

Digital Object Identifier 10.1109/TWC.2013.021213.111410

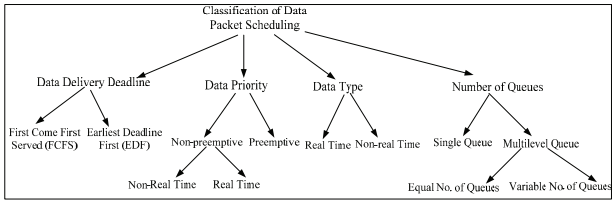


Fig. 1. Classification of packet scheduling schemes.

scheduling scheme. Section V provides a performance analysis of the proposed DMP packet-scheduling algorithm for different types of traffic in terms of average task waiting time and end-to-end data transmission delay. Section VI evaluates the performance of the DMP packet scheduling scheme through simulations and compares it against that of the existing FCFS and Multilevel Queue Scheduler algorithms [27]. Finally, Section VII concludes the paper defining some future research directions.

II. Related Works

In this section, we present existing packet or task scheduling schemes by classifying them based on several factors as is illustrated in Figure 1.

A. Factor: Deadline

Packet scheduling schemes can be classified based on the deadline of arrival of data packets to the base station (BS), which are as follows.

First Come First Served (FCFS): Most existing WSN applications use First Come First Served (FCFS) schedulers that process data in the order of their arrival times at the ready queue. In FCFS, data that arrive late at the intermediate nodes of the network from the distant leaf nodes require a lot of time to be delivered to base station (BS) but data from nearby neighboring nodes take less time to be processed at the intermediate nodes. In FCFS, many data packets arrive late and thus, experience long waiting times.

Earliest Deadline First (EDF): Whenever a number of data packets are available at the ready queue and each packet has a deadline within which it should be sent to BS, the data packet which has the earliest deadline is sent first. This algorithm is considered to be efficient in terms of average packet waiting time and end-to-end delay.

The research work done by Lu C. et al. [28] proposes a real-time communication architecture for large-scale sensor networks, whereby they use a priority-based scheduler. Data, that have travelled the longest distance from the source node to BS and have the shortest deadline, are prioritized. If the deadline of a particular task expires, the relevant data packets are dropped at an intermediate node. Though this approach reduces network traffic and data processing overhead, it is not efficient since it consumes resources such as memory and computation power and increases processing delay. The performance of the scheme can be improved by incorporating FCFS. Mizanian et al. [29] proposed RACE, a packet-scheduling policy and routing algorithm for real-time large-scale sensor networks that uses a loop-free Bellman-Ford algorithm to find paths with the minimum traffic load and delay between source and destination. RACE uses the Earliest Deadline First (EDF) scheduling concept to send packets with earliest deadline. It also uses a prioritized MAC protocol that modifies the initial wait time after the channel becomes idle and the back-off window increases the function of the IEEE 802.11 standard. Priority queues actively drop packets whose deadlines have expired to avoid wasting network resources. However, local prioritization at each individual node in RACE is not sufficient because packets from different senders can compete against each other for a shared radio communication channel.

B. Factor: Priority

Packet scheduling schemes can be classified based on the priority of data packets that are sensed at different sensor nodes.

Non-preemptive: In non-preemptive priority packet scheduling, when a packet t1 starts execution, task t_1 carries on even if a higher priority packet t_2 than the currently running packet t_1 arrives at the ready queue. Thus t_2 has to wait in the ready queue until the execution of t_1 is complete.

Preemptive: In preemptive priority packet scheduling, higher priority packets are processed first and can preempt lower priority packets by saving the context of lower priority packets if they are already running.

Min Y.U. et al. [30] present packet scheduling mechanisms that are used in TinyOS [25], [31] - the widely used operative system of WSN and classify them as either cooperative or preemptive. Cooperative scheduling schemes can be based on a dynamic priority scheduling mechanism, such as EDF and Adaptive Double Ring Scheduling (ADRS) [32], that uses two queues with different priorities. The scheduler dynamically switches between the two queues based on the deadline of newly arrived packets. If the deadlines of two packets are different, the shorter deadline packet would be placed into the higher-priority queue and the longer deadline packet would be placed into the lower-priority one. Cooperative schedulers in TinyOS are suitable for applications with limited system resources and with no hard real-time requirements. On the other hand, preemptive scheduling can be based on the Emergency Task First Rate Monotonic (EF-RM) scheme. EF-RM is an extension to Rate Monotonic (RM), a static priority scheduling, whereby the shortest-deadline job has the highest priority. EF-RM divides WSN tasks into Period Tasks, (PT) whose priorities are decided by a RM algorithm, and non-period tasks, which have higher priority than PTs and can interrupt, whenever required, a running PT.

C. Factor: Packet Type

Packet scheduling schemes can be classified based on the types of data packets, which are as follows.

Real-time packet scheduling: Packets at sensor nodes should be scheduled based on their types and priorities. Real-time data packets are considered as the highest priority packets among all data packets in the ready queue. Hence, they are processed with the highest priority and delivered to the BS with a minimum possible end-to-end delay.

Non-real-time packet scheduling: Non-real time packets have lower priority than real-time tasks. They are hence delivered to BS either using first come first serve or shortest job first basis when no real-time packet exists at the ready queue of a sensor node. These packets can be intuitively preempted by real-time packets.

Though packet scheduling mechanisms of TinyOS are simple and are used extensively in sensor nodes, they cannot be applied to all applications: due to the long execution time of certain data packets, real-time packets might be placed into starvation. Moreover, the data queue can be filled up very quickly if local data packets are more frequent that causes the discard of real-time packets from other nodes. To eliminate these drawbacks, Zhao Y. [24] proposed an improved priority-based soft real-time packet scheduling algorithm. Schedulers traverse the waiting queue for the data packets and choose the smallest packet ID as the highest priority to execute. Each packet is assigned an Execute Counter, EXECUTE_MAX_TIME, i.e., the largest initial task execution time. The management component compares the current packet ID with the previous packet ID. If it is the same, the system executes it and decrements the counting variable. Otherwise, if the counting variable is null, the management component terminates this packet and other packets get the opportunity to be executed. However, packet priorities are decided during the compilation phase, which cannot be changed during the execution time. If high priority packets are always in execution, the low priority packets cannot be implemented. If low-priority packets occupy the resources for a long time, the subsequent high-priority packets cannot get response in time.

D. Factor: Number of Queue

Packet scheduling schemes can also be classified based on the number of levels in the ready queue of a sensor node. These are as follows.

Single Queue: Each sensor node has a single ready queue. All types of data packets enter the ready queue and are scheduled based on different criteria: type, priority, size, etc. Single queue scheduling has a high starvation rate.

Multi-level Queue: Each node has two or more queues. Data packets are placed into the different queues according to their priorities and types. Thus, scheduling has two phases: (i) allocating tasks among different queues, (ii) scheduling packets in each queue. The number of queues at a node depends on the level of the node in the network. For instance, a node at the lowest level or a leaf node has a minimum number of queues whilst a node at the upper levels has more queues to reduce end-to-end data transmission delay and balance network energy consumptions. Figure 4 illustrates the main concept behind multi-level queue scheduling algorithms.

To eliminate problems in [24] Lee et al. [27] propose a multilevel queue scheduler scheme that uses a different number of queues according to the location of sensor nodes in the network. This approach uses two kinds of scheduling: simple priority-based and multi-FIFO queue-based. In the former, data enter the ready queue according to priority but this scheduling also has a high starvation rate. The multi-FIFO queue is divided into a maximum of three queues, depending on the location of the node in the network. If the lowest level is , nodes that are located at level have only one queue but there are two queues for nodes at level . Each queue has its priority set to high, mid, or low. When a node receives a packet, the node decides the packet's priority according to the hop count of the packet and accordingly sends it to the relevant queue. The work done by Karimi E. and Akbari B. [33] also proposes a priority queue scheduling algorithm for WMSN. In this scheduling scheme, buffer space of intermediate nodes is divided into four queues to hold three different types of video frames and one regular data frames. Data in the first three queues have the highest priority and are scheduled in round-robin fashion. Data in the fourth queue is transmitted when the first three queues are empty. However, these scheduling schemes do not consider variable number of queues based on the position of sensor nodes to reduce the overall end-to-end delay.

III. Preliminaries

In this section, we present general assumptions and define some terminologies that are used in designing the Dynamic Multilevel Priority (DMP) packet scheduling scheme.

A. Assumptions

We make the following assumptions to design and implement DMP packet scheduling scheme.

- Data traffic comprises only real-time and non-real-time data, e.g., real-time health data sensed by body sensors and non-real-time temperature data.
- All data packets (real-time and non-real-time) are of same size.
- Sensors are time synchronized.
- No data aggregation is performed at intermediate nodes for real-time data.
- Nodes are considered located at different levels based on the number of hop counts from BS.
- Timeslots are allocated to nodes at different levels using TDMA scheme, e.g., nodes at the lowest level, l_k are assigned timeslot 1. Details of timeslot allocation are explained in the "Terminologies" subsection.
- The ready queue at each node has maximum three levels or sections for real-time data (pr_1) non-real-time remote data (pr_2) and non-real-time local data (pr_3).
- The length of data queues is variable. For instance, the length of real-time data queue (pr_1) is assumed to be smaller than that of non-real-time data queues (pr_2 and pr_3). However, the length of the non-real-time pr_2 and pr_3 queues are same.
- DMP scheduling scheme uses a multichannel MAC protocol to send multiple packets simultaneously.

B. Terminologies

In this section, we define the following terminologies and factors that are used in designing the DMP packet scheduling scheme.

Routing Protocol: For the sake of energy efficiency and balance in energy consumption among sensor nodes, we

envision using a zone-based routing protocol [4, 8]. In a zone-based routing protocol, each zone is identified by a zone head (ZH) and nodes follow a hierarchical structure, based on the number of hops they are distant from the base station (BS). For instance, nodes in zones that are one hop and two hops away from the BS are considered to be at level 1 and level 2, respectively. Each zone is also divided into a number of small squares in such a way that if a sensor node exists in square S_1, it covers all neighboring squares. Thus, this protocol reduces the probability of having any sensing hole [34] in the network even if the neighboring squares of a node do not have any sensor node.

TDMA Scheme: Task or packet scheduling at each nodal level is performed using a TDMA scheme with variable-length timeslots. Data are transmitted from the lowest level nodes to BS through the nodes of intermediate levels. Thus, nodes at the intermediate and upper levels have more tasks and processing requirements compared to lower-level nodes. Considering this observation, the length of timeslots at the upper-level nodes is set to a higher value compared with the timeslot length of lower-level nodes. On the other hand, real-time and time-critical emergency applications should stop intermediate nodes from aggregating data since they should be delivered to end users with a minimum possible delay. Hence, for real-time data, the duration of timeslots at different levels is almost equal and short.

Fairness: This metric ensures that tasks of different priorities get carried out with a minimum waiting time at the ready queue based on the priority of tasks. For instance, if any lower-priority task waits for a long period of time for the continuous arrival of higher-priority tasks, fairness defines a constraint that allows the lower-priority tasks to get processed after a certain waiting time.

Priority: As discussed earlier, real-time and emergency data should have the highest priority. The priority of non-real-time data packets is assigned based on the sensed location (i.e., remote or local) and the size of the data. The data packets that are received by node x from the lower level nodes are given higher priority than the data packets sensed at the node x itself. However, if it is observed that the lower priority non-real-time local data cannot be transmitted due to the continuous arrival of higher priority non-real-time remote data, they are preempted to allow low-priority data packets to be processed after a certain waiting period. Nevertheless, these tasks can be preempted by real-time emergency tasks. In case of two same priority data packets the smaller sized data packets are given the higher priority.

IV. Proposed DMP Packet Scheduling Scheme

As discussed earlier, in non-preemptive packet scheduling schemes (interchangeably use as task scheduling in this paper), real-time data packets have to wait for completing the transmissions of other non-real-time data packets. On the other hand, in preemptive priority scheduling, lower-priority data packets can be placed into starvation for continuous arrival of higher-priority data. In the multilevel queue scheduling algorithm [5], each node at the lowest level has a single task queue considering that it has only local data to process.

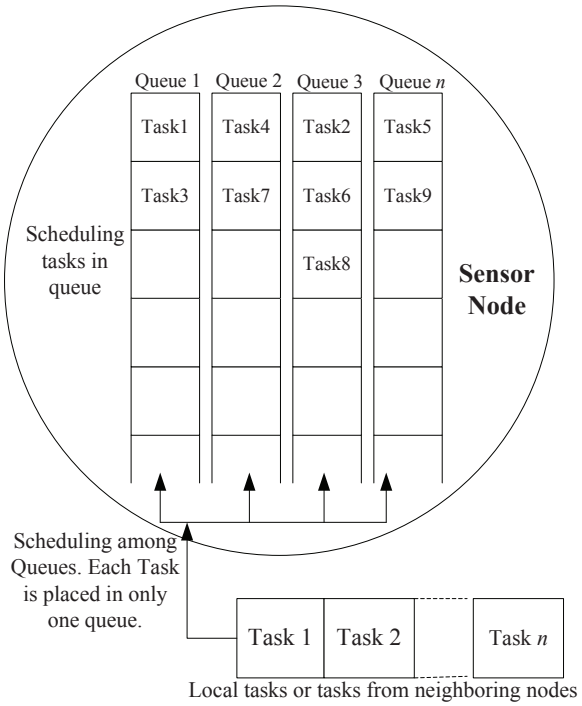


Fig. 2. Scheduling data among multiple queues.

However, local data can also be real-time or non-real time and should be thus processed according to their priorities. Otherwise, emergency real-time data traffic may experience long queuing delays till they could be processed. Thus, we propose a Dynamic Multilevel Priority (DMP) packet scheduling scheme that ensures a tradeoff between priority and fairness. In this section, we present the working principle of DMP packet scheduling scheme with its pseudo-code.

A. Working Principle

Scheduling data packets among several queues of a sensor node is presented in Figure 2. Data packets that are sensed at a node are scheduled among a number of levels in the ready queue. Then, a number of data packets in each level of the ready queue are scheduled. For instance, Figure 2 demonstrates that the data packet, $Data_1$ is scheduled to be placed in the first level, Queue1. Then, $Data_1$ and $Data_3$ of Queue1 are scheduled to be transmitted based of different criteria. The general working principle of the proposed DMP scheduling scheme is illustrated in Figure 3.

The proposed scheduling scheme assumes that nodes are virtually organized following a hierarchical structure. Nodes that are at the same hop distance from the base station (BS) are considered to be located at the same level. Data packets of nodes at different levels are processed using the Time-Division Multiplexing Access (TDMA) scheme. For instance, nodes that are located at the lowest level and the second lowest level can be allocated timeslots 1 and 2, respectively. We consider three-level of queues, that is, the maximum number of levels in the ready queue of a node is three: priority 1 (pr_1), priority 2 (pr_2), and priority 3 (pr_3) queues. Real-time data packets go to pr_1, the highest priority queue, and are

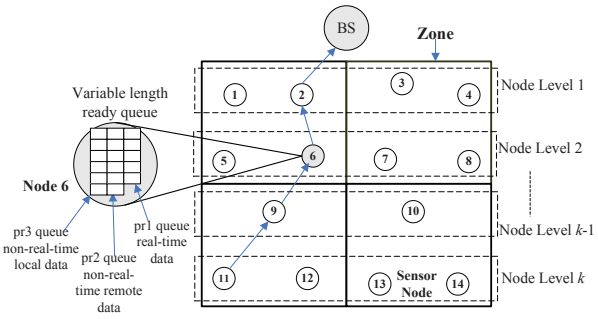


Fig. 3. Proposed dynamic multi-level priority (DMP) packet scheduling scheme.

processed using FCFS. Non-real-time data packets that arrive from sensor nodes at lower levels go to pr_2, the second highest priority queue. Finally, non-real time data packets that are sensed at a local node go to pr_3, the lowest priority queue. The possible reasons for choosing maximum three queues are to process (i) real-time pr_1 tasks with the highest priority to achieve the overall goal of WSNs, (ii) non real-time pr_2 tasks to achieve the minimum average task waiting time and also to balance the end-to-end delay by giving higher priority to remote data packets, (iii) non-real-time pr_3 tasks with lower priority to achieve fairness by preempting pr_2 tasks if pr_3 tasks wait a number of consecutive timeslots.

In the proposed scheme, queue sizes differ based on the application requirements. Since preemptive priority scheduling incurs overhead due to the context storage and switching in resource constraint sensor networks, the size of the ready queue for preemptive priority schedulers is expected to be smaller than that of the preemptable priority schedulers. The idea behind this is that the highest-priority real-time/emergency tasks rarely occur. They are thus placed in the preemptive priority task queue (pr_1 queue) and can preempt the currently running tasks. Since these processes are small in number, the number of preemptions will be a few. On the other hand, non-real-time packets that arrive from the sensor nodes at lower level are placed in the preemptable priority queue (pr_2 queue). The processing of these data packets can be preempted by the highest priority real-time tasks and also after a certain time period if tasks at the lower priority pr_3 queue do not get processed due to the continuous arrival of higher priority data packets. Real-time packets are usually processed in FCFS fashion. Each packet has an ID, which consists of two parts, namely level ID and node ID. When two equal priority packets arrive at the ready queue at the same time, the data packet which is generated at the lower level will have higher priority. This phenomenon reduces the end-to-end delay of the lower level tasks to reach the BS. For two tasks of the same level, the smaller task (i.e., in terms of data size) will have higher priority.

Moreover, it is expected that when a node x senses and receives data from lower-level nodes, it is able to process and forward most data within its allocated timeslot; hence, the probability that the ready queue at a node becomes full and drops packets is low. However, if any data remains in the ready queue of node x during its allocated timeslot, that data will be transmitted in the next allocated timeslot.

Timeslots at each level are not fixed. They are rather calculated based on the data sensing period, data transmission rate, and CPU speed. They are increased as the levels progress through BS. However, if there is any real-time or emergency response data at a particular level, the time required to transmit that data will be short and will not increase at the upper levels since there is no data aggregation. The remaining time of a timeslot of nodes at a particular level will be used to process data packets at other queues. Since the probability of having real-time emergency data is low, it is expected that this scenario would not degrade the system performance. Instead, it may improve the perceived Quality of Service (QoS) by delivering real-time data fast. Moreover, if any node x at a particular level completes its task before the expiration of its allocated timeslot, node x goes to sleep by turning its radio off for the sake of energy efficiency.

B. Pseudo-code

In our proposed DMP packet scheduling scheme, nodes at the lowest level,l_k, sense, process and transmit data during their allocated timeslots, whereas nodes at level l_{k-1} and upper levels receive data in addition to sensing, processing and transmitting data. Now, we present the pseudo-code of our proposed DMP packet scheduling scheme.

We consider only two levels in the ready queue of sensor nodes that are located at the lowest level since these nodes do not receive packets from any lower level nodes. Other nodes have three levels in the ready queue and place non-real time local tasks into pr_3 queue. We also consider that each node requires time to sense data packets and also process local and/or remote data packets. For instance, $t_1(k)$ in the pseudo-code represents the real-time data sensing time at a $node_i$. If the processing time of real-time data at $node_i$ is less than $t_1(k)$ then $node_i$ will have time remaining to process non-real-time pr_2 data packets. Similarly, if $node_i$ still has some remaining time, it can process non-real-time pr_3 data packets. The pseudo-code also shows that if the pr_1 queue is empty and pr_2 packets are processed α consecutive timeslots, the processing of pr_2 data packets will be preempted for j timeslots.

V. Performance Analysis

In this section, we analyze the performance of the proposed DMP task scheduling scheme in terms of end-to-end delay, and total waiting time of different types of traffic at the ready queues of active nodes.

A. End-to-End Delay

In the following, we formulate the average end-to-end delay of transmitting different priority data packets to the base station (BS). Again, we interchangeably use task and data to represent the data packets that are sensed at a sensor node.

Real-time Priority 1 Queue Data: Let us assume that a node x, residing at level l_k is sensing a real-time, emergency event, e.g., fire detection. This node transmits the emergency priority 1 data to BS through l_{k-1} intermediate levels. We consider the following scenario whereby every time a real-time data

while $task_{k,i}$ is received by $node_i$ at level k, i.e., l_k **do**
  if $Type(task_{k,i}) = real - time$ **then**
    put $task_{k,i}$ into pr_1 queue
  else if $node_i$ is not at lowest levels **then**
    if $task_{k,i}$ is not local **then**
      put $task_{k,i}$ into pr_2 queue
    else
      put $task_{k,i}$ into pr_3 queue
    end if
  else
    put $task_{k,i}$ into pr_2 queue
  end if
  Assume, the duration of a timeslot at $l_k \leftarrow t(k)$
  Data sensing time of $node_i$ at $l_k \leftarrow senseTime_k(t)$
  Remaining time after data sensing, $t_1(k) = t(k) - senseTime_k(t)$
  Let total real-time tasks for $node_i$ at $l_k \leftarrow n_k(pr_1)$
  Let $procTimepr_1(k) \leftarrow \sum_{j=1}^{n_k(pr1)} procTime(j)$
  if $procTimepr1(k) < t_1(k)$ **then**
    All pr_1 tasks of $node_i$ at l_k are processed as FCFS
    Remaining time $t_2(k) \leftarrow t_1(k) - procTimepr_1(k)$
    Let, total pr_2 tasks for $node_i$ at $l_k \leftarrow n_k(pr_2)$
    Let $procTimepr_2(k) \leftarrow \sum_{j=1}^{n_k(pr2)} procTime(j)$
    if $procTimepr_2(k) < t_2(k)$ **then**
      All pr_2 tasks are processed as FCFS
      pr_3 tasks are processed as FCFS for the remaining time, $t_3(k) \leftarrow t_2(k) - procTimepr_2(k)$
    else
      pr_2 tasks are processed for $t_2(k)$ time
      no pr_3 tasks are processed
    end if
  else
    only pr_1 tasks are processed for $t_1(k)$ time
    no pr_2 and pr_3 tasks are processed
  end if
  if pr_1 queue empty & pr_2 tasks are processed α consecutive timeslots since $t(k) \leq procTimepr_2(k)$ **then**
    pr_2 tasks are preempted at $\alpha+1, \ldots, \alpha+j$ timeslots by pr_3 tasks
    if pr_1 task arrives during any of $\alpha+1, \alpha+2, \ldots, \alpha+j$ timeslots **then**
      pr_3 tasks are preempted and pr_1 tasks are processed context are transferred again for processing pr_3 tasks
    end if
  end if
end while

packet reaches a neighboring active node, y at an upper level, a non-real time lower priority data is being processed at that node. Hence, data delivery at y is preempted to send real-time data.

Transmission time or delay that is required to place a real-time data from a node into the medium is equal to $\frac{data_{pr1}}{s_t}$. The propagation time, or delay to transmit data from the source to destination can be formulated as $\frac{d}{s_p}$. Considering the above mentioned scenario the end-to-end delay for sending a real-time data satisfies the following inequality.

$$delay_{pr1} \geq l_k \times (\frac{data_{pr1}}{s_t} + pr1_{proc}(t)) + \frac{d}{s_p} + (l_k \times t_{overhead}) \quad (1)$$

where $data_{pr1}$ denotes the real-time data size, s_t denotes the data transmission speed, d is the distance from the source node to BS, where $d = \sum_{i=1}^{l_k} d_i$, s_p denotes the propagation speed over the wireless medium, $pr1_{proc}(t)$ is the processing time of real-time tasks at each node, and $t_{overhead}$ is an overhead in terms of context switching and queuing time (including time for preemption). However, a real-time task t_1 has to wait if there is a number, n_{pr1}, of a real-time task ahead of t_1 at the pr_1 queue. We assume that all real-time data have the same size.

Therefore, the end-to-end delay for a real-time task t_1 considering that t_1 has n_{pr1} number of real-time tasks ahead of it,

$$delay_{t_1} \geq \sum_{i=1}^{n_{pr1}} (delaypr_1)_i \quad (2)$$

Non-real time Priority 2 Queue Data: Tasks at pr_2 queue can be preempted by real-time ones. Taking the scenario of Figure 3 as an example, we first consider the scenario when a real-time task is sensed at node 11 and is forwarded to BS through relay nodes 9, 6, and 2. It should be observed that tasks are available at the pr_2 queue at nodes 9, 6 and 2. Since one real-time task is available at the pr_1 queue of nodes 9, 6, and 2, real-time tasks will be processed and transmitted first during the timeslot of nodes 9, 6, and 2. The pr_2 tasks are processed in the remaining time of the timeslots. The transmission time or delay to place pr_2 data from a node into the medium can be therefore computed as $\frac{data_{pr2}}{s_t}$. Thus, the total end-to-end delay for a pr_2 task that can be processed in the same timeslot exceeds

$$l_k \times (\frac{data_{pr1}}{s_t} + \frac{data_{pr2}}{s_t} + pr1_{proc}(t) + pr2_{proc}(t)) + \frac{d}{s_p} + (l_k \times t_{overhead}) \quad (3)$$

Non-real time Priority 3 Queue Data: In the best case, when no task is available at the pr_1 and pr_2 queues, the end-to-end delay of the pr_3 tasks will be almost equal to that of the pr_1 queue tasks (Equation 1) although it can differ slightly based on the size of the pr_3 queue task. We assume that the pr_3 queue tasks are processed by preempting pr_2 queue tasks if for α consecutive timeslots there is no task at the pr_1 queue but there are tasks available at the pr_2 queue. Let t_k denote the length of a timeslot of nodes at level l_k. The transmission time or delay to place pr_3 data from a node into the wireless medium is equal to $\frac{data_{pr3}}{s_t}$. However, during the processing of the pr_3 queue tasks, these tasks can be preempted by real-time tasks. They are processed again after the completion of real-time tasks. Thus, the end-to-end delay for processing pr_3 tasks will be exceeding

$$\alpha \times t(k) + l_k \times (\frac{data_{pr3}}{s_t} + pr3_{proc}(t)) + \frac{d}{s_p}) + (l_k \times t_{overhead}) \quad (4)$$

B. Average Waiting Time

In the following, we formulate the average waiting time of tasks at different workloads. Let us assume that pr_{ij_i} represents the processing time of the j-th pr_i task at a node x, where, $1 \leq i \leq 3$ and $1 \leq j_i \leq n_i$.

Thus, total processing time, $pr_i(t) = \sum_{j_i=1}^{n_i} pr_{ij_i}(t)$. Let us denote the total number of levels as k, and the length of a timeslot at the level l_j as $t(j)$.

For real-time tasks, $i = 1$ (i.e., pr_1). Assuming that real-time and emergency tasks rarely occur and require a very short time to get processed, $pr_1(t) < t(k)$. Hence, all tasks,$1 \leq j_1 \leq n_1$, in the pr_1 queue complete processing and tasks in the pr_2 and pr_3 queues are processed for the remaining, $t_2(k) = t(k) - pr_1(t)$, period of time.

Since pr_1 tasks are processed as FCFS, the average waiting time for real-time, pr_1 tasks at node x is

$$AvgWaitingTimePr_1(t) = \frac{\sum_{j_1=1}^{n_1-1}\sum_{m=1}^{j_1} pr_{1,m}(t)}{n_1} \quad (5)$$

where the first pr_1 task has no waiting time and waiting time for the j-th pr_1 task is equal to $\sum_{m=1}^{j} pr_{1,m}(t)$. Now, let pr_2 tasks be sorted according to the ascending order of the processing time, $pr_{2j_2}(t)$, of pr_2 tasks at the ready queue so that we have $pr_{21}(t) \leq pr_{22}(t) \leq \ldots pr_{2n}(t)$. If pr_2 tasks are not preempted by pr_1 tasks and can be completed within the $t_2(k)$ time (i.e., within the same timeslot for the processing pr_1 tasks), the average waiting time for pr_2 tasks can be expressed as follows:

$$AvgWaitingTimePr_2(t) = \frac{\sum_{j_2=1}^{n_2-1}\sum_{m=1}^{j_2} pr_{2,m}(t)}{n_2} \quad (6)$$

If pr_2 tasks at i-th node at the level j require more than one timeslot to complete their processing,

$frameTime = \sum_{j=1}^{k} t(j)$, $node_{ij}$ waits $\tau = (\sum_{j=1}^{k} t(j)) - t(j)$.

If pr_2 queue tasks at $node_{ij}$ requires $\beta \geq 1$ timeslots to complete their processing and no pr_1 task preempts pr_2 tasks, then the average waiting time of pr_2 tasks is

$$\begin{aligned} AvgWaitingTimePr_2(t) \approx{}& \frac{\sum_{j_{21}=1}^{n_{21}}\sum_{m=1}^{j_{21}} pr_{2,m}(t)}{n_{21}} + \\ & \frac{\sum_{j_{22}=n_{21}+1}^{n_{22}}\sum_{m=1}^{j_{22}} pr_{2,m}(t)}{n_{22}} + \ldots \\ & + \frac{\sum_{j_{2,\beta-1}=n_{(2,\beta-2)}+1}^{n_{2,\beta-1}}\sum_{m=1}^{j_{2,\beta-1}} pr_{2,m}(t)}{n_{2,\beta}} + (\beta \times \tau) \end{aligned} \quad (7)$$

Where, $n_{21} + n_{22} + \ldots + n_{2,\beta} = n_2$ and $\beta \times \tau$ is the total waiting time of pr_2 tasks to work for β timeslots. Again, τ is the waiting time of pr_2 tasks of a node at a specific level to complete the remaining work in the next timeslot. Moreover, there is no other waiting time, even if a new pr_2 task arrives while processing other pr_2 tasks, since pr_2 tasks are non-preemptive (until a new pr_1 task preempts pr_2 tasks). In general, the average waiting time of a node for processing pr_2 tasks considering preemption by pr_1 tasks is

$$\begin{aligned} AvgWaitingTimePr_2(t) \geq{}& \frac{\sum_{j_{21}=1}^{n_{21}}\sum_{m=1}^{j_{21}} pr_{2,m}(t)}{n_{21}} \\ & + \frac{\sum_{j_{22}=n_{21}+1}^{n_{22}}\sum_{m=1}^{j_{22}} pr_{2,m}(t)}{n_{22}} + \ldots \\ & + \frac{\sum_{j_{2,\beta-1}=n_{(2,\beta-2)}+1}^{n_{2,\beta-1}}\sum_{m=1}^{j_{2,\beta-1}} pr_{2,m}(t)}{n_{2,\beta}} \\ & + (\beta \times \tau) + \sum_{m=1}^{\beta} \gamma_m \end{aligned} \quad (8)$$

where $1 \leq m \leq \beta$ and γ_m are the extra waiting time of pr_2 tasks for being preempted by pr_1 tasks in each of the m-th timeslots. If we consider that, at each of the timeslots, pr_2 tasks will be processed after processing pr_1 tasks for the $pr_1(t) = \sum_{j_i=1}^{n_1} pr_{1j_1}(t)$ period, then the processing time of pr_1 tasks will be considered as the waiting time of pr_2 tasks at each of the timeslot.

$$\begin{aligned} AvgWaitingTimePr_2(t) \geq{}& \frac{\sum_{j_{21}=1}^{n_{21}}\sum_{m=1}^{j_{21}} pr_{2,m}(t)}{n_{21}} \\ & + \frac{\sum_{j_{22}=n_{21}+1}^{n_{22}}\sum_{m=1}^{j_{22}} pr_{2,m}(t)}{n_{22}} + \ldots \\ & + \frac{\sum_{j_{2,\beta-1}=n_{(2,\beta-2)}+1}^{n_{2,\beta-1}}\sum_{m=1}^{j_{2,\beta-1}} pr_{2,m}(t)}{n_{2,\beta}} \\ & + (\beta \times \tau) + \sum_{m=1}^{\beta} \gamma_m + (\beta \times pr_1(t)) \end{aligned} \quad (9)$$

Now, consider that pr_2 tasks are preempted by pr_3 tasks if no pr_1 tasks exist at the ready queue and pr_2 tasks are processed for $\alpha \geq 2$ consecutive timeslots. If $\beta \leq \alpha$, then the average waiting time of pr_2 tasks will be the same as Equation 9. If $\beta > \alpha$ and the task processing controller are switched back to pr_2 tasks from pr_3 tasks after δ timeslots, the average waiting time of pr_2 tasks,

$$\begin{aligned} AvgWaitingTimePr_2(t) \geq{}& \frac{\sum_{j_{21}=1}^{n_{21}}\sum_{m=1}^{j_{21}} pr_{2,m}(t)}{n_{21}} + \\ & \frac{\sum_{j_{22}=n_{21}+1}^{n_{22}}\sum_{m=1}^{j_{22}} pr_{2,m}(t)}{n_{22}} + \ldots + \\ & \frac{\sum_{j_{2,\beta-1}=n_{(2,\beta-2)}+1}^{n_{2,\beta-1}}\sum_{m=1}^{j_{2,\beta-1}} pr_{2,m}(t)}{n_{2,\beta}} \\ & + (\beta \times \tau) + \sum_{m=1}^{\beta} \gamma_m + \\ & (\beta \times pr_1(t) + (\delta \times \sum_{j=1}^{k} t(j)) \end{aligned} \quad (10)$$

Equation 10 presents the waiting time of pr_2 tasks of the nodes at upper levels and have pr_1, pr_2 and pr_3 queues at each node. But the lowest-level nodes only have the pr_1 and pr_2

queues; therefore, pr_2 tasks are not preempted by pr_3 tasks at the lowest level. Similarly, we can formulate the average waiting time of pr_3 tasks of nodes at the upper level as follows.

If pr_3 tasks at a node get processed after α timeslots or frames due to the processing of pr_1 and pr_2 tasks, then pr_3 tasks are processed after waiting for $\alpha \times \sum_{j=1}^{k} t(j)$ time period. Then we assume that the pr_3 tasks require ψ timeslots to complete their tasks and, during these timeslots, the pr_3 tasks are preempted by pr_1 tasks for $\sum_{m=1}^{\psi} \gamma_m$ period.

Thus, the average waiting time of pr_3 tasks at a node, $AvgWaitingTimePr3(t)$, exceeds

$$AvgWaitingTimePr_3(t) \geq \frac{\sum_{j_{31}=1}^{n_{31}} \sum_{m=1}^{j_{31}} pr_{3,m}(t)}{n_{31}} + \frac{\sum_{j_{32}=n_{31}+1}^{n_{32}} \sum_{m=1}^{j_{32}} pr_{3,m}(t)}{n_{32}} + \ldots + \frac{\sum_{j_{3,\psi-1}=n_{(3,\psi-2)}+1}^{n_{3,\psi-1}} \sum_{m=1}^{j_{3,\psi-1}} pr_{3,m}(t)}{n_{3,\psi}} + (\psi \times \tau) + \sum_{m=1}^{\psi} \gamma_m + (\alpha \times \sum_{j=1}^{k} t(j)) \quad (11)$$

where $n_{31} + n_{32} + \ldots + n_{3,\psi} = n_3$ (i.e., the total number of pr_3 tasks at a node). Thus, using Equations 6 - 11 we formulate the average waiting time of pr_1, pr_2, and pr_3 tasks at the ready queue of a node x at a particular level.

VI. Performance Evaluation

The simulation model is implemented using the C programming language. It is used to evaluate the performance of the proposed DMP packet scheduling scheme, comparing it against the FCFS, and Multilevel Queue scheduling schemes. The comparison is made in terms of average packet waiting time, and end-to-end data transmission delay. We use randomly connected Unit Disk Graphs (UDGs) on a surface of 100 meter $\times$ 100 meter as a basis of our simulations. The number of simulated zones varies from 4 to 12 zones. Nodes are distributed uniformly over the zones. The ready queue of each node can hold a maximum of 50 tasks. Each task has a Type ID that identifies its type. For instance, type 0 is considered to be a real-time task. Data packets are placed into the ready queue based on the processing time of the task. Moreover, each packet has a hop count number that is assigned randomly, and the packet with the highest hop count number is placed into the highest-priority queue. We run the simulation both for a specific number of zones, and levels in the network until data from a node in each zone or level reach BS. Simulation results are presented for both real-time data and all types of data traffic. Table I presents simulation parameters, and their respective values.

Figures 4 and 5 illustrate the end-to-end data transmission delay of real-time tasks over a number of zones and levels, respectively. In both case, we observe that the proposed DMP scheduling scheme outperforms the existing FCFS, and Multilevel Queue scheduler. This is because the proposed scheduling scheme gives the highest priority to real-time tasks and also allows real-time data packets to preempt the processing of non-real time data packets. Thus, real-time data packets have lower data transmission delays.

TABLE I
SIMULATION PARAMETERS, AND THEIR RESPECTIVE VALUES

Parameter	**Value**
Network Size	100m X 100m
Number of Nodes	Maximum 200
Number of Zones	4 - 12
Base station position	55m X 101m
Transmission Energy Consumptions	50 $nJoule/bit$
Energy Consumption in free space or air	0.01 $nJoule/bit/m^2$
Initial Node Energy	2 Joule
Transmission Speed	$250Kbps$
Propagation Speed	$198 \times 10^6 meter/sec$

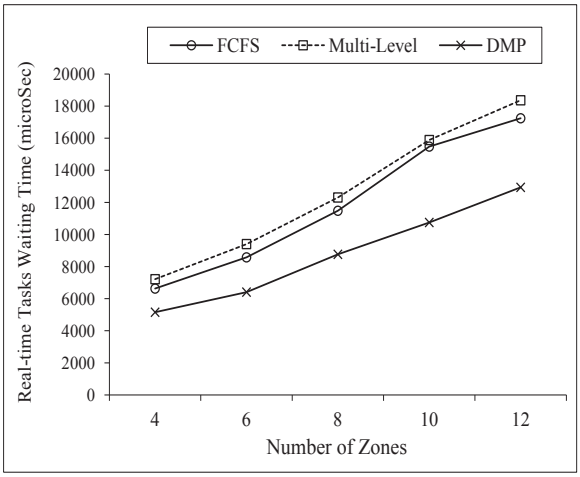


Fig. 4. End-to-end delay of real-time data over a number of zones.

We also validate these results using student's t-test at 95% confidence level. Figure 4 illustrates the p-values which are 0.0453 between FCFS and DMP schemes and 0.0137 between Multi-level queue and DMP schedulers.

Similarly, Figures 6 and 7 demonstrate the end-to-end delay of all types of data traffic over a number of zones and levels, respectively. From these results, we find that the DMP task scheduling scheme outperforms FCFS, and Multilevel Queue scheduler in terms of end-to-end data transmission delay. This is because in the proposed scheme, the tasks that arrive from the lower level nodes are given higher priority than the tasks at the current node. Thus, the average data transmission delay is shortened. Figure 6 shows the p-values of student's t-test, which are 0.01156 between Multi-level and DMP schedulers, 0.000000005 between FCFS and DMP schedulers. Thus, DMP outperforms both FCFS and Multi-level queue schedulers at 95% confidence interval.

Figures 8 - 11 demonstrate that the DMP task scheduler has better performance than the FCFS, and Multilevel Queue scheduler in terms of average task waiting time, both for real-time tasks, and all types of tasks. We have already explained the possible reasons for this performance differences. We also perform student's t-test at a 95% confidence level and find the p-value to be less than 0.05 in most cases. This test validates our claim about the performance of the proposed

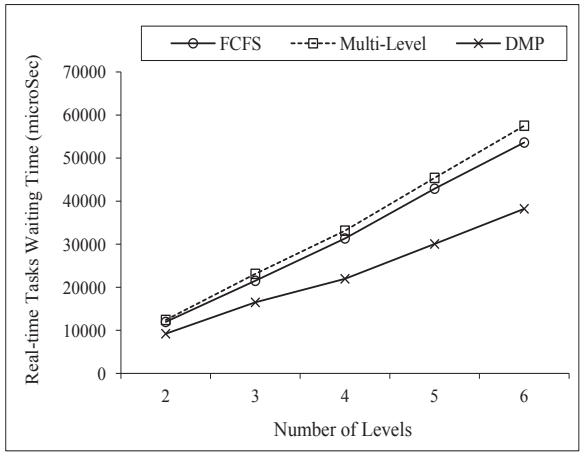


Fig. 5. End-to-end delay of real-time data over a number of levels.

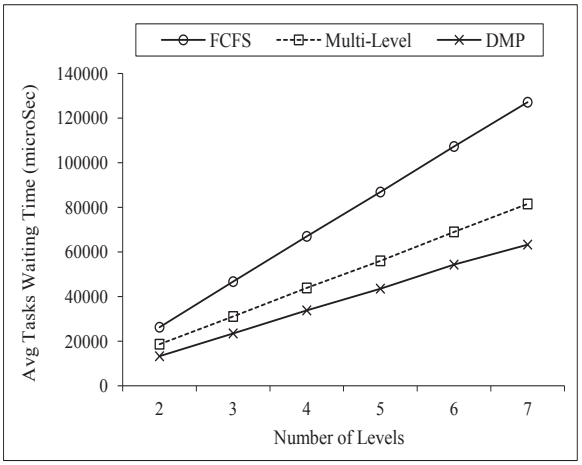


Fig. 7. End-to-end delay of all types of data over a number of levels.

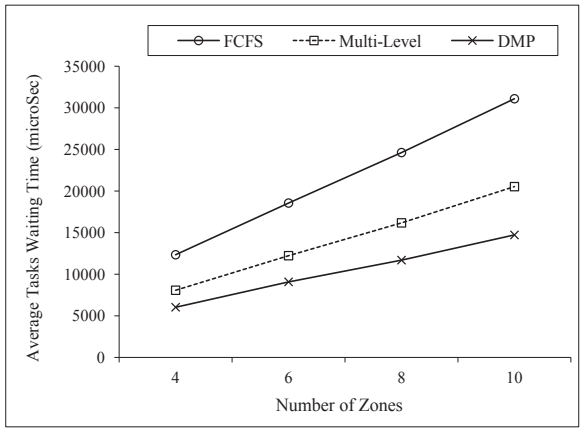


Fig. 6. End-to-end delay of all types of data over a number of zones.

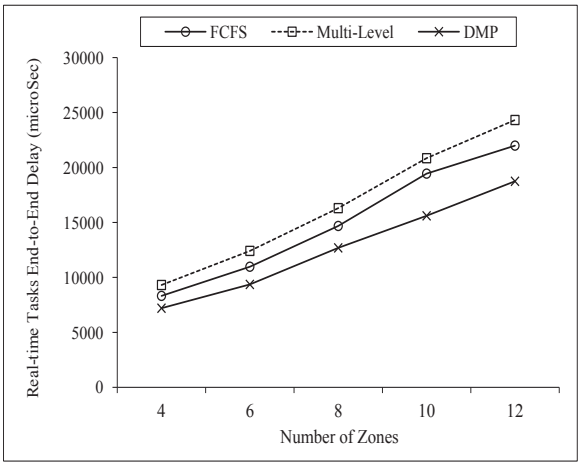


Fig. 8. Waiting time of real-time data over a number of zones.

DMP scheduling scheme.

We also measure, and compare the fairness of executing non-real-time task in terms of the total waiting time of non-real-time tasks over total waiting time of all tasks. Figure 12 illustrates that the fairness index of DMP scheduling scheme is higher or better than that of the other two approaches. The number of levels in the network topology increases as the number of zones multiplies, which increases the average waiting time for non-real-time tasks over real-time tasks. Thus, the fairness index slightly decreases or remains almost same as the number of zones increases.

Using the concept of three-level priority queues at each node, the proposed DMP task scheduling scheme allows different types of data packets to be processed based on their priorities. Since real-time, and emergency data should be processed with the minimum end-to-end delay, they are processed with the highest priority, and can preempt tasks with lower priorities located in the two other queues. On the other hand, in existing multilevel queue schedulers, a task with the highest hop count is given the highest priority. Hence, real-time tasks are prioritized over other task types only if their hop counts are higher than those of non-real-time tasks. Moreover, in FCFS and multilevel queue schedulers, the estimated processing time of a task is not considered when deciding the priority of a task. Thus, FCFS and Multilevel Queue schedulers exhibit longer task waiting times and end-to-end delays, in comparison to the DMP task scheduling scheme. Furthermore, the average waiting time of a task contributes largely to the experienced end-to-end data transmission delay, hence the strong correlation between the results of Figures 10 and 7.

In the DMP task scheduling approach, the source of a data packet is used to define the priority of data packets other than real-time. The priority of non-real time data packet will be more if it is sensed at remote node rather than the current sending node. Moreover, when no real-time tasks are available, pr_3 tasks can preempt pr_2 tasks if they are in starvation for a long time. This allows the processing of different types of tasks with fairness. The memory is also dynamically allocated to three queues and the size of the highest-priority queue is usually smaller than the two other queues (Figure 3) since pr_1 real-time tasks do not occur frequently compared to non-real-time tasks. As the memory capacity of a sensor node is limited, this also balances memory usages. Moreover, tasks are mostly non-real-time and are processed in the pr_2 and pr_3 queues. Non-real-time tasks that a node x receives from the lower level nodes are known as non-real-time remote tasks and processed with higher priority (pr_2) than the non-real-time local tasks that x senses. Thus, non-real-time remote

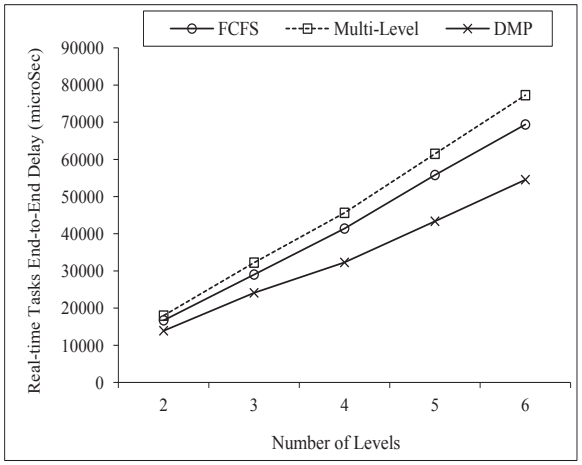


Fig. 9. Waiting time of real-time data over a number of levels.

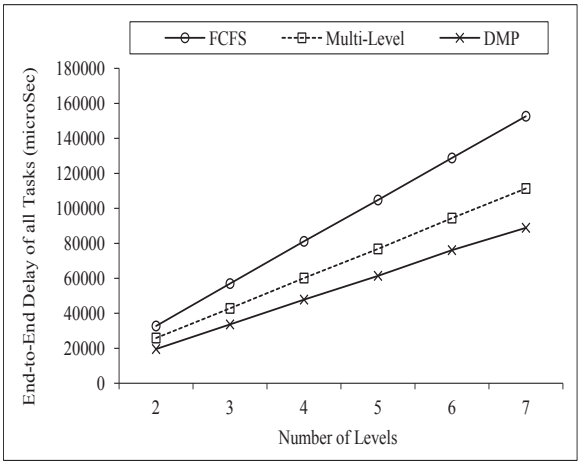


Fig. 11. Waiting time of all types of data over a number of levels.

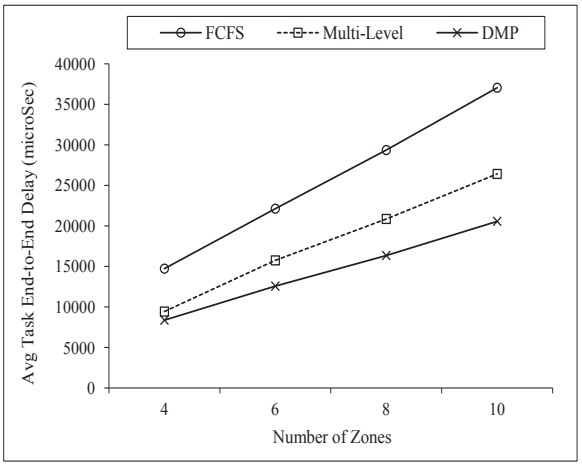


Fig. 10. Waiting time of all types of data over a number of simulated zones.

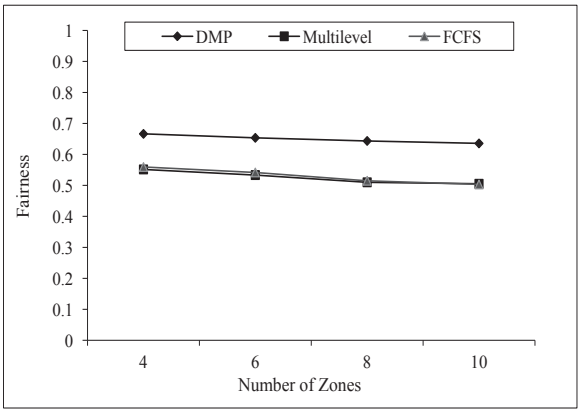


Fig. 12. Fairness in terms of the waiting time of non-real-time data.

tasks incur less average waiting time. In addition, the average waiting time will not be affected for real-time tasks that are processed using FCFS scheduling, since these real-time tasks occur infrequently with a short processing time.

Admittedly, one of the concerns regarding our proposed DMP task scheduling scheme pertains to its energy requirements. Indeed, the DMP task scheduling mechanism could be less energy efficient in comparison to the other two approaches since the DMP scheme requires a few more processing cycles to categorize and place the tasks into three different queues as well as for context saving and switching (for preemption). However, given the increased demand for WSN-based solutions that efficiently support real-time emergency applications and ensure them minimum average task waiting time and end-to-end delay, the proposed DMP task scheduling mechanism can be regarded as highly efficient.

VII. Conclusion and Future Work

In this paper, we propose a Dynamic Multilevel Priority (DMP) packet scheduling scheme for Wireless Sensor Networks (WSNs). The scheme uses three-level of priority queues to schedule data packets based on their types and priorities. It ensures minimum end-to-end data transmission for the highest priority data while exhibiting acceptable fairness towards lowest-priority data. Experimental results show that the proposed DMP packet scheduling scheme has better performance than the existing FCFS and Multilevel Queue Scheduler in terms of the average task waiting time and end-to-end delay.

As enhancements to the proposed DMP scheme, we envision assigning task priority based on task deadline instead of the shortest task processing time. To reduce processing overhead and save bandwidth, we could also consider removing tasks with expired deadlines from the medium. Furthermore, if a real-time task holds the resources for a longer period of time, other tasks need to wait for an undefined period time, causing the occurrence of a deadlock. This deadlock situation degrades the performance of task scheduling schemes in terms of end-to-end delay. Hence, we would deal with the circular wait and preemptive conditions to prevent deadlock from occurring. We would also validate the simulation result using a real test-bed.

References

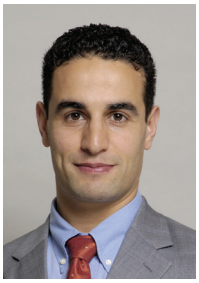
[1] G. Anastasi, M. Conti, and M. Di Francesco, "Extending the lifetime of wireless sensor networks through adaptive sleep," *IEEE Trans. Industrial Informatics*, vol. 5, no. 3, pp. 351–365, 2009.

[2] G. Bergmann, M. Molnar, L. Gonczy, and B. Cousin, "Optimal period length for the CQS sensor network scheduling algorithm," in *Proc. 2010 International Conf. Netw. Services*, pp. 192–199.

[3] E. Bulut and I. Korpeoglu, "DSSP: a dynamic sleep scheduling protocol for prolonging the lifetime of wireless sensor networks," in *Proc. 2007 International Conf. Advanced Inf. Networking Appl.*, vol. 2, pp. 725–730.
[4] S. Chachra and M. Marefat, "Distributed algorithms for sleep scheduling in wireless sensor networks," in *Proc. 2006 IEEE International Conf. Robot. Autom.*, pp. 3101–3107.
[5] P. Guo, T. Jiang, Q. Zhang, and K. Zhang, "Sleep scheduling for critical event monitoring in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 345–352, Feb. 2012.
[6] F. Liu, C. Tsui, and Y. J. Zhang, "Joint routing and sleep scheduling for lifetime maximization of wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 9, no. 7, pp. 2258–2267, July 2010.
[7] J. Liu, N. Gu, and S. He, "An energy-aware coverage based node scheduling scheme for wireless sensor networks," in *Proc. 2008 International Conf. Young Comput. Scientists*, pp. 462–468.
[8] O. Khader, A. Willig, and A. Wolisz, "Distributed wakeup scheduling scheme for supporting periodic traffic in wsns," in *Proc. 2009 European Wireless Conf.*, pp. 287–292.
[9] B. Nazir and H. Hasbullah, "Dynamic sleep scheduling for minimizing delay in wireless sensor network," in *Proc. 2011 Saudi International Electron., Communications Photon. Conf.*, pp. 1–5.
[10] D. Shuman and M. Liu, "Optimal sleep scheduling for a wireless sensor network node," in *Proc. 2006 Asilomar Conf. Signals, Syst. Comput.*, pp. 1337–1341.
[11] S. Paul, S. Nandi, and I. Singh, "A dynamic balanced-energy sleep scheduling scheme in heterogeneous wireless sensor network," in *Proc. 2008 IEEE International Conf. Netw.*, pp. 1–6, 2008.
[12] A. R. Swain, R. C. Hansdah, and V. K. Chouhan, "An energy aware routing protocol with sleep scheduling for wireless sensor networks," in *Proc. 2010 IEEE International Conf. Adv. Inf. Netw. Appl.*, pp. 933–940.
[13] Y. H. Wang, Y. L. Wu, and K. F. Huang, "A power saving sleep scheduling based on transmission power control for wireless sensor networks," in *Proc. 2011 International Conf. Ubi-Media Comput.*, pp. 19–24.
[14] Y. Wang, D. Wang, W. Fu, and D. P. Agrawal, "Hops-based sleep scheduling algorithm for enhancing lifetime of wireless sensor networks," in *Proc. 2006 IEEE International Conf. Mobile Adhoc Sensor Syst.*, pp. 709–714.
[15] Y. Xiao, H. Chen, K. Wu, B. Sun, Y. Zhang, X. Sun, and C. Liu, "Coverage and detection of a randomized scheduling algorithm in wireless sensor networks," *IEEE Trans. Comput.*, vol. 59, no. 4, pp. 507–521, Apr. 2010.
[16] X. Xu, Y. H. Hu, J. Bi, and W. Liu, "Adaptive nodes scheduling approach for clustered sensor networks," in *Proc. 2009 IEEE Symp. Comput. Commun.*, pp. 34–39.
[17] Y. Zhao, J. Wu, F. Li, and S. Lu, "VBS: maximum lifetime sleep scheduling for wireless sensor networks using virtual backbones," in *Proc. 2010 IEEE INFOCOM*, pp. 1–5.
[18] B. Zeng, Y. Dong, and D. Lu, "Cooperation-based scheduling algorithm in wireless multimedia sensor networks," in *Proc. 2011 International Conf. Wireless Commun., Netw. Mobile Comput.*, pp. 1–4.
[19] N. Edalat, W. Xiao, C. Tham, E. Keikha, and L. Ong, "A price-based adaptive task allocation for wireless sensor network," in *Proc. 2009 IEEE International Conf. Mobile Adhoc Sensor Syst.*, pp. 888–893.
[20] H. Momeni, M. Sharifi, and S. Sedighian, "A new approach to task allocation in wireless sensor actor networks," in *Proc. 2009 International Conf. Computational Intelligence, Commun. Syst. Netw.*, pp. 73–78.
[21] F. Tirkawi and S. Fischer, "Adaptive tasks balancing in wireless sensor networks," in *Proc. 2008 International Conf. Inf. Commun. Technol.: From Theory Appl.*, pp. 1–6.
[22] X. Yu, X. Xiaosong, and W. Wenyong, "Priority-based low-power task scheduling for wireless sensor network," in *Proc. 2009 International Symp. Autonomous Decentralized Syst.*, pp. 1–5.
[23] W. Stallings, *Operating Systems*, 2nd edition. Prentice Hall, 1995.
[24] Y. Zhao, Q. Wang, W. Wang, D. Jiang, and Y. Liu, "Research on the priority-based soft real-time task scheduling in TinyOS," in *Proc. 2009 International Conf. Inf. Technol. Comput. Sci.*, vol. 1, pp. 562–565.
[25] TinyOS. Available: http://webs.cs.berkeley.edu/tos, accessed June 2010.
[26] Available: http://webs.cs.berkeley.edu/tos, accessed June 2010.
[27] E. M. Lee, A. Kashif, D. H. Lee, I. T. Kim, and M. S. Park, "Location based multi-queue scheduler in wireless sensor network," in *Proc. 2010 International Conf. Advanced Commun. Technol.*, vol. 1, pp. 551–555.
[28] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: a real-time communication architecture for large-scale wireless sensor networks," in *Proc. 2002 IEEE Real-Time Embedded Technol. Appl. Symp.*, pp. 55–66.
[29] K. Mizanian, R. Hajisheykhi, M. Baharloo, and A. H. Jahangir, "RACE: a real-time scheduling policy and communication architecture for large-scale wireless sensor networks," in *Proc. 2009 Commun. Netw. Services Research Conf.*, pp. 458–460.
[30] M. Yu, S. J. Xiahou, and X. Y. Li, "A survey of studying on task scheduling mechanism for TinyOS," in *Proc. 2008 International Conf. Wireless Commun., Netw. Mobile Comput.*, pp. 1–4.
[31] P. A. Levis, "TinyOS: an open operating system for wireless sensor networks (invited seminar)," in *Proc. 2006 International Conf. Mobile Data Manag.*, p. 63.
[32] K. Lin, H. Zhao, Z. Y. Yin, and Y. G. Bi, "An adaptive double ring scheduling strategy based on tinyos," *J. Northeastern University Natural Sci.*, vol. 28, no. 7, pp. 985–988, 2007.
[33] E. Karimi and B. Akbari, "Improving video delivery over wireless multimedia sensor networks based on queue priority scheduling," in *Proc. 2011 International Conf. Wireless Commun., Netw. Mobile Comput.*, pp. 1–4.
[34] L. Karim, N. Nasser, and T. El Salti, "Efficient zone-based routing protocol of sensor network in agriculture monitoring systems," in *Proc. 2011 International Conf. Commun. Inf. Technol.*, pp. 167–170.

Nidal Nasser Dr. Nidal Nasser received his B.Sc. and M.Sc. degrees with Honors in Computer Engineering from Kuwait University, State of Kuwait, in 1996 and 1999, respectively. He completed his Ph.D. in the School of Computing at Queen's University, Kingston, Ontario, Canada, in 2004. He is currently an Associate Professor and Chairman of Electrical and Computer Engineering Department at Alfaisal University, Saudi Arabia. He worked in the School of Computer Science at University of Guelph, Guelph, Ontario, Canada (2004-2011). Dr. Nasser was the founder and Director of the Wireless Networking and Mobile Computing Research Lab @ Guelph (WiNG: http://wing.socs.uoguelph.ca). He has authored 129 journal publications, refereed conference publications and book chapters in the area of wireless communication networks and systems. He has also given tutorials in major international conferences. Dr. Nasser is currently serving as an associate editor of Wiley's *International Journal of Wireless Communications and Mobile Computing*, Wiley's *International Journal on Communication Systems*, Wiley's *Security and Communication Networks Journal* and *International Journal of Ad Hoc & Sensor Wireless Networks*. He has been a member of the technical program and organizing committees of several international IEEE conferences and workshops. Dr. Nasser is a member of several IEEE technical committees. He received Fund for Scholarly and Professional Development Award in 2004 from Queen's University. He received the Computing Faculty Appreciation Award from the University of Guelph-Humber. He received the Best Research Paper Award at the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'08), at the International Wireless Communications and Mobile Computing Conference (IWCMC'09) and at the International Wireless Communications and Mobile Computing Conference (IWCMC'11).

Lutful Karim Lutful Karim received Ph.D. and M.Sc degrees in Computer Science from the University of Guelph and University of Manitoba, Canada in 2012 and 2005, respectively. He is currently working as a Postdoctoral fellow in the School of Computer Science, University of Guelph, Canada. He worked as a faculty member of computer science for about 10 years. He has authored several refereed conference publications and journal publications, and been a member of organization committees and technical program committees in several international conferences. His research interest includes wireless and mobile sensor networks, mobile and wireless computing, ubiquitous and pervasive computing, communication protocols and algorithms, fault tolerant computing systems, and combinatorial optimizations.



Tarik Taleb Dr. Tarik Taleb is currently working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. Prior to his current position and till Mar. 2009, he worked as assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI. From Oct. 2005 till Mar. 2006, he was working as research fellow with the Intelligent Cosmos Research Institute, Sendai, Japan. He received his B. E degree in Information Engineering with distinction, M.Sc. and Ph.D. degrees in Information Sciences from GSIS, Tohoku Univ., in 2001, 2003, and 2005, respectively.

Dr. Talebs research interests lie in the field of architectural enhancements to 3GPP core networks, mobile cloud networking, mobile multimedia streaming, inter-vehicular communications, congestion control protocols, handoff and mobility management, and social media networking. Dr. Taleb has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPPs System Architecture working group. Dr. Taleb is an acting member of the IEEE Communications Society Standardization Program Development Board. As an attempt to bridge the gap between academia and industry, Dr. Taleb has founded and has been the general chair of the IEEE Workshop on Telecommunications Standards: from Research to Standards.

Dr. Taleb is/was on the editorial board of the *IEEE Wireless Communications Magazine*, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and a number of Wiley journals. He is serving as vice chair of the Wireless Communications Technical Committee, the largest in IEEE Communication Society (ComSoC). He also served as Secretary and then as Vice Chair of the Satellite and Space Communications Technical Committee of IEEE ComSoc (2006 - 2010). He has been on the technical program committee of different IEEE conferences, including Globecom, ICC, and WCNC, and chaired some of their symposia.

Dr. Taleb is the recipient of the 2009 IEEE ComSoc Asia-Pacific Young Researcher award (Jun. 2009), the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation (Mar. 2008), the 2007 Funai Foundation Science Promotion Award (Apr. 2007), the 2006 IEEE Computer Society Japan Chapter Young Author Award (Dec. 2006), the Niwa Yasujirou Memorial Award (Feb. 2005), and the Young Researcher's Encouragement Award from the Japan chapter of the IEEE Vehicular Technology Society (VTS) (Oct. 2003). Some of Dr. Talebs research work has been also awarded best paper awards at prestigious conferences. Dr. Taleb is a senior IEEE member.