Two-Level Priority Real-Time Scheduling Strategy for Node System in Wireless Sensor Network

Lan Yao, Fuxiang Gao, Xiuli Cui and Ge Yu
Collage of Information Science and Engineering
Northeastern University
Shenyang, China
{yaolan, gaofuxiang,cuixiuli, yuge}@ise.neu.edu.cn

Chao Shang
Collage of Software
Northeastern University
Shenyang, China
Shangchao@ise.neu.edu.cn

Abstract

Emerging applications such as forest fire monitoring have increasing demands on WSN to transmit data in real-time. In order to ensure real-time data transmission, it requires that the operating system of a node should schedule tasks in real-time. TinyOS is one of the most popular operation systems that support multifarious applications. However, its FIFO scheduling strategy does not guarantee requirements for hard real-time applications. A Two-Level Priority (TLP) Real-time Scheduling Strategy is proposed in this paper. Two tier priorities, static and dynamic, are designed and integrated in TinyOS task queue to guarantee the real-time task scheduling. We demonstrate this approach by a real-world case study: a WSN hardware node embedded with our task scheduling strategy is designed and implemented. The result demonstrates that our TLP real-time scheduling strategy performs efficiently in terms of packet throughput and task scheduling time.

1. Introduction

Different WSN practical applications require different level of real-time sensing data transmission. Some applications, such as environmental data monitoring in precision agriculture, tracking and monitoring of migratory birds, do not require too much about the real-time, While hard real-time scheduling is a demanding feature for many application domains. In real-world scenarios, real-time data transmission is critical to guarantee the quality of service. For example, forest fire monitoring and detection to invader for security.

The typical node system --TinyOS adopts non-preemptive FIFO task scheduling strategy. When the task queue is empty, the processor sleeps, and then the CPU is waken up to execute tasks by external events. Because all the tasks are equal, the priority of relative important or urgent task cannot be guaranteed. It is likely that over-load may even occur, which may cause the loss of task or the decrease of communication throughput. Thus the communication and operational efficiency based on this scheduling strategy of the whole system is limited.

This work is supported by the Major State Basic Research Development Program of China under grant (No. 2006CB303003).

Because TinyOS is a single task kernel, the resource utilization can be limited. Thus it is necessary to design a multi-task system. However, a mulit-task system raises the issue of real-time task scheduling. In order to make sure the system is real-time, the task scheduling strategy based on preemptive priority is widely adopted. Research efforts have been contributed to the study of multi-task scheduling strategy based on priority. [1] expends TinyOS to multi-task scheduling, and adds multi-task scheduling function to improve the responsiveness of the system. [2] proposes a priority scheduling based on the deadline to improve the real-time capability of WSN system. [3] puts forward a task priority scheduling algorithm to improve the throughput of the over-loaded nodes and thus solve the over-load of local node packets.

All the above scheduling strategies improve some aspects of the original FIFO scheduling strategy, however they all have some shortcomings. For instance, the deadline-based priority algorithm only takes into account the real-time task. As a result, it is inadequate for the over-loaded situation. The task-based priority algorithm improves the node over-loaded issue; however it ignores the real-time requirements.

In order to solve the limitations discussed above in TinyOS and other typical algorithms, a Two-Level Priority (TLP) Scheduling Strategy is designed and implemented. This strategy enables TinyOS to respond to important and hard real-time tasks. We also provide a hardware system that realizes and embeds TLP. Our solution effectively prevents over-load in a node, and hence improves the overall communication efficiency of the system.

The structure of the paper is as follows: in section 2 , the TLP scheduling strategy is introduced. The design and implementation of hardware system is given section 3. The performance of the TLP implemented on the hardware system is evaluated in section 4. Finally this paper concludes in section 5.

2. Design of TLP Scheduling Strategy

Communication routing task and local data processing task are divided according to different

functions of WSN tasks. Two relative static priorities *(H, L)* are given to these two types of tasks respectively, so that the over-loaded situation caused by the congestion of the local task emerging into the communication tasks can be processed.

Instead of the situation that a number of tasks share the same dynamic queue space caused by the FIFO scheduling strategy, different tasks occupy their own queues according to their levels. The task is running according to its queue space, and when the task is finished, the corresponding queue will be reclaimed and reallocated to new-arrival task. This makes the task processing more flexible.

At the mean while, based on the static priority, the dynamic scheduling strategy is adopted, called Earliest Deadline First (EDF). The task's dynamic priority is determined by the deadline and the running time of the task. The FIFO strategy is used when the tasks have the same priority. As a result, the capability of the running real-timely is improved.

The TLP scheduling strategy includes task submission and task scheduling.

Task submission: when a new task arrives, the scheduling strategy judges whether the queue is full. If the queue is not full, as it is shown in Fig.1, It submits the new task to the tail of the queue, and sort the queue according to the dynamic priority. If the queue is full, as it is shown in Fig.2, it sorts all the tasks in the queue according to static priority (steady sort), and judge static priority and dynamic priority of the current task and the tail task. It inserts the high priority task into a proper place of the queue, and dismisses the current task. In Fig.1 and Fig.2, *&task3_1* represents the third arrived task, which has high-level static priority; *&task2_0* represents the second arrived task, which has low-level static priority. The task submission of TLP scheduling strategy is shown in Fig.3.

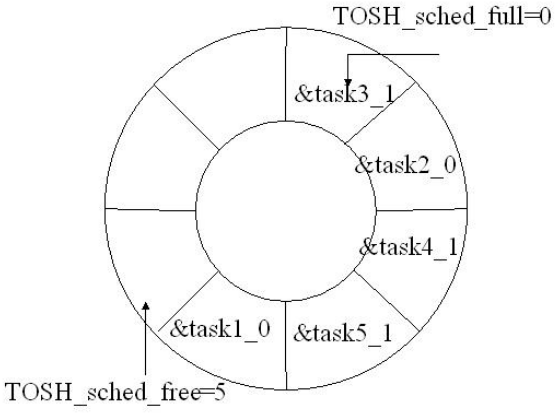


Fig.1 Dynamic priority in half-full queue

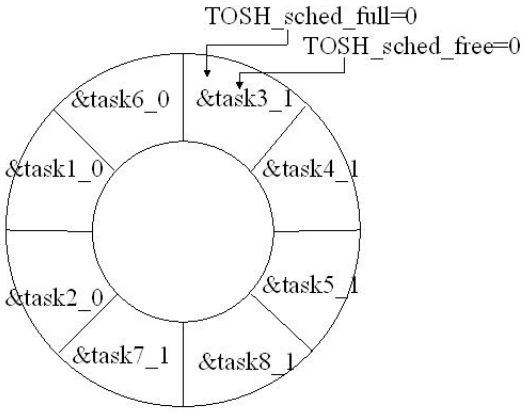


Fig. 2 Static priority in full queue

```
BEGIN
Initialization;
Close interruption;
If(Queue_is_full())
 {Sort _static();
 If (Curtask.sta_pri>Tailtask.sta_pri)
  { Exchange(&Curtask,& Tailtask);
   Sort _static();
        Sort_dynaminc();
        Open interruption;
        Return 1; }
  elseif(Curtask.dy_pri>Tailtask.dy_pri)
        {Exchange(&Curtask,& Tailtask);
         Open interruption;
             Return 1; }
       Else { Open interruption;
                  Return 0; }
   }
Else { Post_to_tail();Sort_dynamic();}
Open interruption;
Return 1;
END
```

Fig.3 TOS_task_post

Task scheduling: the hardware adopts Atmegal28L processor (details about this hardware will be introduced in section three), and takes the timer0 as the task-scheduling clock. At each time of the interruption, TLP updates the task's dynamic priority before schedule the task. The scheduling function chooses the task with the highest priority (the earliest deadline and the shortest running time) to execute. If the task is overdue, it will be dismissed, and a new head will be chosen. The timer0 is used in TinyOS as both the sleep/wake-up timer and the task-

scheduling timer. When the task queue is not empty, the timer0 is a task-scheduling timer, otherwise it is a sleep/wake-up timer. When the timer interruption arrives, TLP execute the EDF task scheduling. When the task queue is empty, CPU turns to the low-power sleep status. The task scheduling process is shown in Fig.4.

The TLP scheduling strategy effectively alleviates the concurrent overload situation, reduces the dropout rates, and thus improves the system throughput and communication efficiency.

```
BEGIN  Initialization;  Close interruption;
If (cur_task_is_end())
{ if (queue_isnot_empty())
  {update_queue_param(); exchange_stacks();
   get_head_task();  move_head_pointer();
     while (cur_task_overtime())
    { discard_cur_task();  get_head_task();
       move_head_pointer();}
   if (interrupted())
    { save_states(); run_cur_task();
         open initialization; return 1; }  }
        else {change_scheduling_to_sleep();
           open interruption;  return 0; }  }
else{update_cur_param();
     update_queue_param();
   if (curtask.dy_pri>headtask.dy_pri )
   { run_cur_task();open initialization;
      return 1;}
     else{savestates();
         exchange(&curtask,&hetask);
         while (cur_task_overtime())
    {discard_cur_task(); get_head_task();
        move_head_pointer();}
    if(interrupted()){save_states();
                              run_cur_task();
           open initialization; return 1} } }
END
```

Fig.4 Task Scedule

3. Design and Implementation of Hardware System

The node hardware adopts modular structure design. It consists of data processing modules, wireless communication module, power supply module and sensor module.

The node hardware system should satisfy the following conditions: small volume; low power with sleep mode enabled; high integrity and fast speed.

3.1 Data Processing Module

The data processing module is the core part of the node. In order to meet the needs of low-power and small volume, we choose *ATmega128L* micro processor[4] .It has 4K byte EEPROM, 4K byte SRAM, 53 general I/O lines, 32 general work registers, real-time clock RTC, JTAG interface compatible with the IEEE1149.1 stander, and 6 energy-saving mode which can be selected by software.

SCM *ATmegal128L* has limited data storage capability. So a manageable data storage chip is needed to store data.

512K serial FLASH AT45DB041B is used here to store data. Compared with common data memory, this chip has the features of low power consumption, small size, serial interface, and a simple external circuit. It is suitable for sensor nodes.

3.2 Wireless Communication Module

The wireless communication module should satisfy the requirements of low work voltage, low energy consumption and small volume.

We use RF *CC2420* chip. It is a wireless transceiver module compatible with 2.4Hz IEEE802.15.4 stander[5]. It is provided with programmable output intensity and transceiver frequency. Its external circuit mainly includes crystal oscillator circuit, antenna and impedance matching circuit, interface circuit, and decoupling filter circuit. Its maximum transceiver rate is 250kbps. It guarantees the efficiency and reliability of short-distance communication.

In the wireless sensor node communication module, the selection and the deployment of the antenna will affect the quality of the whole wireless communication networks directly. This node RF chip *CC2420* chooses metal inverted F PCD wired antenna and monopole antenna at the same time. The PCD wired antenna is a conductor printed on the circuit board, through which the antenna senses the airwaves and receives information.

3.3 Power Supply Module

The power supply module is a very important module in WSN. This module uses MAXIM company's *MAX604* chip[6]. *MAX604* chip has features including a low voltage difference, low energy consumption, and linear manostat to guarantee the stability of the system working voltage. As the time goes on, the voltage will decrease gradually. So it can not provide the system with a stable voltage. Thus, a MAX604 chip is added to the battery to keep the voltage at 3.3V.

The hardware structure of the node system is shown in Fig.5.

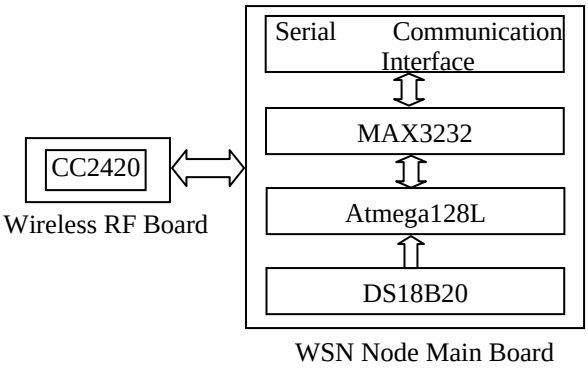


Fig. 5 Structure of node hardware

4. Performance Analysis

A TLP scheduling strategy is embedded in TinyOS and uploaded to nodes, which are implemented with *ATmega128L* and *CC2420* in this paper. 100 nodes are deployed randomly in the 100m×100m monitoring area. It is evaluated and the following conclusions are derived from the empirical results. As it is shown in Fig.6, when FIFO strategy is used, with the increase of the local task running time, the number of data packets sent per second decrease sharply. While after the TLP scheduling strategy is adopted, the number of data packets sent per second stays in a stable amount. The result demonstrates that our scheduling strategy and its implementation improves the node throughout significantly.

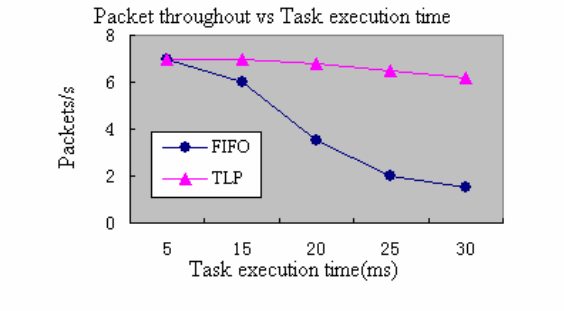


Fig.6 Sending throughout/local task running time

5. Conclusions

The TLP scheduling strategy is proposed to divide task priority into two tiers: static and dynamic to guarantee the real-time task scheduling on TinyOS, and solves the limitation that TinyOS can not satisfy the hard real-time requirement. Based on the independent development of ATmega128L hardware node, TLP is implemented. Our empirical evaluations demonstrate the efficiency of this solution. It remains our future work to compare our solution with other scheduling strategy and to further validate our solutions to other hard real-time scheduling application.

References

[1]J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX), Cambridge, MA, USA: 93-104, 2000.

[2]Seott A. Brnadt, Scott A.Banachowski, Caixue Lin, Timothy Bisson. Dynamic Integrated Scheduling of Hard Real-Time, Soft Real-Time and Non-Real-Time Processes. Proceedings of the 24th IEEE Real-Time Systems SymPosium, Cancun, Mexico: 396-407, Dec 2003.

[3]Venkita Subramonian, Huang-Ming Huang, Seema Datar, Chenyang Lu. Priority scheduling in TinyOS - A case study. Washington University, 2003.

[4]Atmel Corporation. ATmega128/ATmega128L datasheet. www.avrvi.com/down-9.html, 2001

[5]Chipcon AS.CC2420 Datasheet(1.3). http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf, 2005

[6]Maxim AS.MAX604 Datasheet(1.1). www.maxim-ic.com, 2004