CS497 Course Report on PMRT WSN Design

Qixin Wang
Instructor: Prof. Marco Caccamo
December, 2002

First, we study the system architecture of sensor devices and their networks. Based on the system architecture, we analyze the constraints on sensing frequency selections, so that real-time schedulability criterions are met.

Specifically, we study the sensor network of UC Berkeley mica motes [5][3], which uses ATmega103L micro-controllers [2].

Fig 1 is an illustration of the infrastructure of an ATmega103L mote with wireless communication module. The CPU, main memory and data bus etc. are all integrated in the ATMega103L micro-controller. The CPU clock frequency is 4MHz. One important peripheral to the ATMega103L micro-controller is the RFM™ TR1000 [6] wireless co-processor (named as "Radio CoPU" in Fig 1), which controls the wireless communication physical layer – the 916.5MHz radio channel. The CPU communicates with the Radio CoPU using *clock driven periodical querying* – the querying rate is 20kHz for receiving and 10kHz for sending. I.e. when the system is in radio receiving mode, there are exactly 4MHz/20kHz=200 CPU cycles between each two consecutive radio CoPU queries; and when in sending mode, there are 4MHz/10kHz=400 CPU cycles between each two consecutive radio CoPU queries. This design successfully enables ATMega103L mote with a 500Byte/sec application layer transmission rate [4].

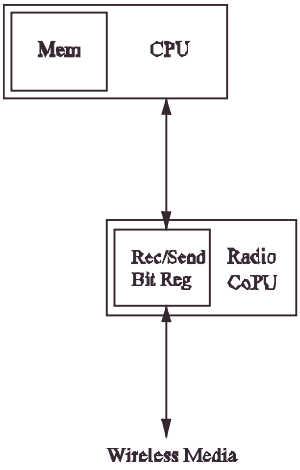

Fig 1 Mote Infrastructure

The success of mote's design supports the practicability of a slightly more complex system architecture – *Parallel Multiple hard Real-Time* (PMRT) wireless sensor device, illustrated by Fig 2.

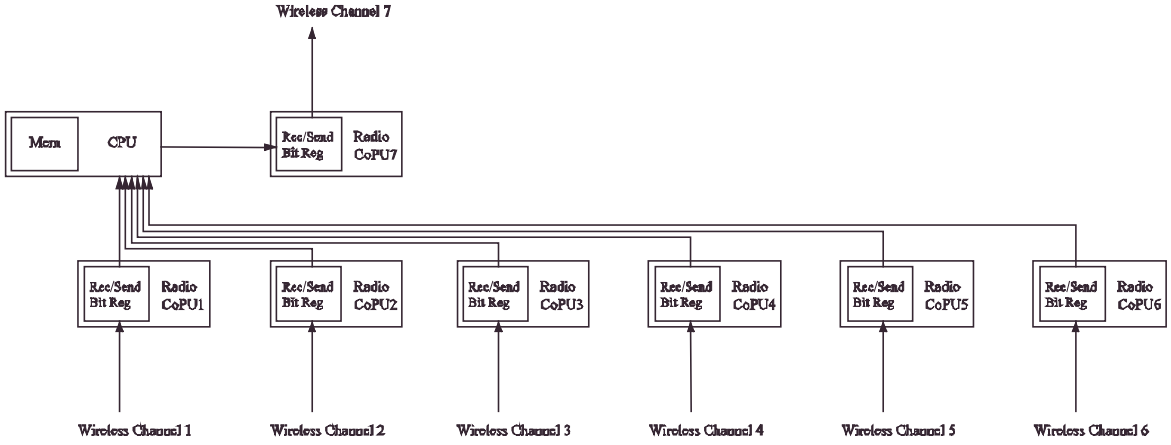

Fig 2 A Future Design of Multi-channel Mote (e.g., with 6 Rec Channels and 1 Sending Channel)

In this architecture, there can be several Radio CoPUs operating at different radio frequencies, each one functions either as a receiver or a sender (note: there can be several senders in one host, though Fig 2 only shows one). Each sender sends its packets according to *Non-preemptive EDF*. And every sender has a dedicated *EDF Scheduling Pool* [1] in the main memory to buffer/schedule its outgoing packets. The CPU accesses each receiver/sender by clock driven periodical querying. After each query, the CPU execute the following routings:

Routine 1: After queried a receiver:
1. Assemble the newly received bit in the memory buffer according to the query result.
2. If the bit is the last bit of a byte, assemble a newly received byte.
3. If the byte is the last byte of a packet, insert the packet to the right position of the right sender's EDF Scheduling Pool.

Routine 2: After queried a sender:
1. Pick the next bit-to-send from the current packet-to-send in the sender's EDF Scheduling Pool. This bit is the bit to give to the sender in the next query.
2. If the last bit of the current-packet-to send is already sent to sender, pick the next packet-to-send from the EDF Scheduling Pool as the new current packet-to-send and goto 1.
3. If there is no next packet-to-send, the corresponding sender becomes idle in the next query cycle.

First look at the inner-host schedulability. Suppose $C^{Routine1}, C^{Routine2}$ are the worst case execution time (in the units of "CPU cycles") for Routine 1 and Routine 2 respectively. Suppose R is the set of all receivers on the host and S is the set of all senders on the host. For a specific $receiver \in R$ the bandwidth is $B^{receiver}$, and for a specific $sender \in S$,

[1] Sometimes is also called *Scheduling Queue*. But *Pool* is a more proper word since the behavior may be much more complicated than *Queue*'s first-in-first-out behavior.

the bandwidth is B^{sender} (all are in the unit of "bits per second"). Let $k^{receiver}$ be the number of CPU queries to receive one bit on *receiver* and k^{sender} be the number of CPU queries to send one bit on *sender*. Let C^{CPU} to be the total CPU cycles available each second (i.e. in the unit of "CPU cycles per second"). Then the schedulability constraint set by the internal architecture of the host is (remember the internal CPU uses clock driven periodical scheduling):

$$\sum_{receiver \in R} k^{receiver} B^{receiver} C^{Routine1} + \sum_{sender \in S} k^{sender} B^{sender} C^{Routine2} \le C^{CPU} \qquad (1)$$

I.e., *if none of the receivers/senders are overloaded (on their bandwidths), inner-host schedulability is always guaranteed*. The manufacturers of PMRT wireless sensor devices should specify $B^{receiver}$ and B^{sender} according to (1).

The inter-host schedulability constraint analysis is as follows:

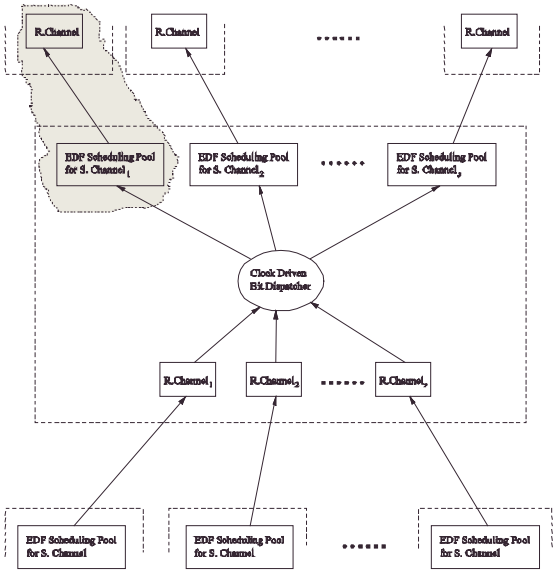

Fig 3(a) Channel with Single Receiver

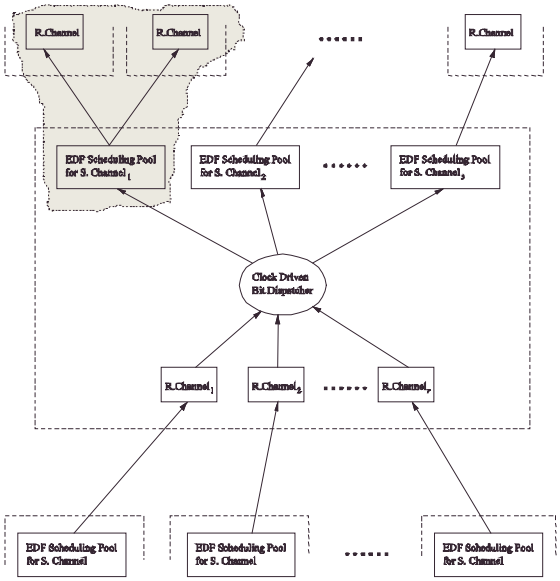

Fig 3(b) Channel with Multiple Receiver

In order to make things simple, we allow only one sender for each channel. On the other hand, a channel can have one (as shown by the shaded area in Fig 3(a)) or multiple (as shown by the shaded area in Fig 3(b)) receivers (where packet's receiver address is used to distinguish the effective receiver). No matter how many receivers are there in a channel, the bandwidth of a channel B (in bps) is defined as the maximal affordable bit rate that the sender, anyone of the receivers and the receiver's corresponding bit dispatcher (i.e., the internal CPU and Memory that runs Routine1) can handle.

Let's first look at the EDF Scheduling Pool of the sender of a channel. Suppose T is the set of connections (tasks) that need to go through this sender. For a connection (or say, in real-time's jargon – a task) $\tau \in T$, suppose it has a reporting frequency of f_τ (report

packets per second) and a report packet length of l_τ. Then each of its report, if there is no any other connections contending, should be sent to the receiver and dispatched to the receiver hop's corresponding EDF Scheduling Pool in time $\frac{l_\tau}{B}$. However, the physical channel media is a *non-preemptive* resource. I.e., if one packet started transmitting, it can not be preempted by any other packet until it is completely transmitted. Hereby, the *real-time scheduler of EDF Scheduling Pool should be a Nonpreemptive EDF scheduler*, to ensure both the EDF behavior and non-preemptive usage of the channel's physical media. Accordingly, the fine grain sufficient schedulability bound for the scheduler should be the following inequality system defined in [1].

Reference:

[1] G. C. Buttazzo, Hard real-time computing systems: predictable scheduling algorithm and applications, Kluwer Academic Publishers, 1997
[2] D. V. Gadre, Programming and Customizing the AVR Microcontroller, McGraw-Hill, 2001
[3] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, Kristofer S. J. Pister, System Architecture Directions for Networked Sensors, in Proc. of ASPLOS-IX, Cambridge, Mass. 2000
[4] Qixin Wang, Mote's throughput test report and source code, in http://www.andrew.cmu.edu/~weizhang/wsn/documents.html
[5] B. Warneke, B. Atwood, K. S. J. Pister, ``Smart Dust Mote Forerunners," in Proc. of the Fourteenth Annual International Conference on Microelectromechanical systems (MEMES 2001), Interlaken, Switzerland, Jan. 21-25, 2001, pp. 357-360.
[6] http://www.rfm.com/datasheet/tr1000.pdf