Reading Assignment #2

Marcella Wilson

"Timed Perturbation Analysis: An Approach for Non-Intrusive Monitoring of Real-Time Computations"

Madalene Spezialetti and Rajiv Gupta

What is Monitoring?

- Technique for developing and maintaining programs
- Results sent to remote site where it can be stored, analyzed or displayed.
- A user can request to monitor an application for 2 reasons
  1. Capture a portion of local state at specific point in the program
  2. Continuous tracing of a certain state (ie, a variable through execution)

Monitoring

- Monitoring a program during execution
  - Good for non-real time applications
  - Bad for real time applications
- What is a monitor?
  - A single process
  - Group of processes

What happens during monitoring?

- A process in task passes important information to monitor
- Some code from monitor incorporated into application modules
- Can change timing of program
- Can cause missed deadlines

Goal of Paper

- Non-intrusively monitor run-time activities
  - Find idle time during execution
  - Schedule task monitoring then
  - No deadlines missed

Some Definitions

- Task
  - Performs one function
  - Must complete by deadline
- Real Time application
  - Series of executions
- Idle Time
  - Waiting for data at input points
  - Execution suspended

One Non-Intrusive Approach

- If idle time < execution time of monitoring
  - Monitoring cant be done non-intrusively
- If idle time > execution time of monitoring
  - Monitoring can be done non-intrusively
- This approach assumes
  - Monitoring code an indivisible unit
  - Only one input point or one block of idle time

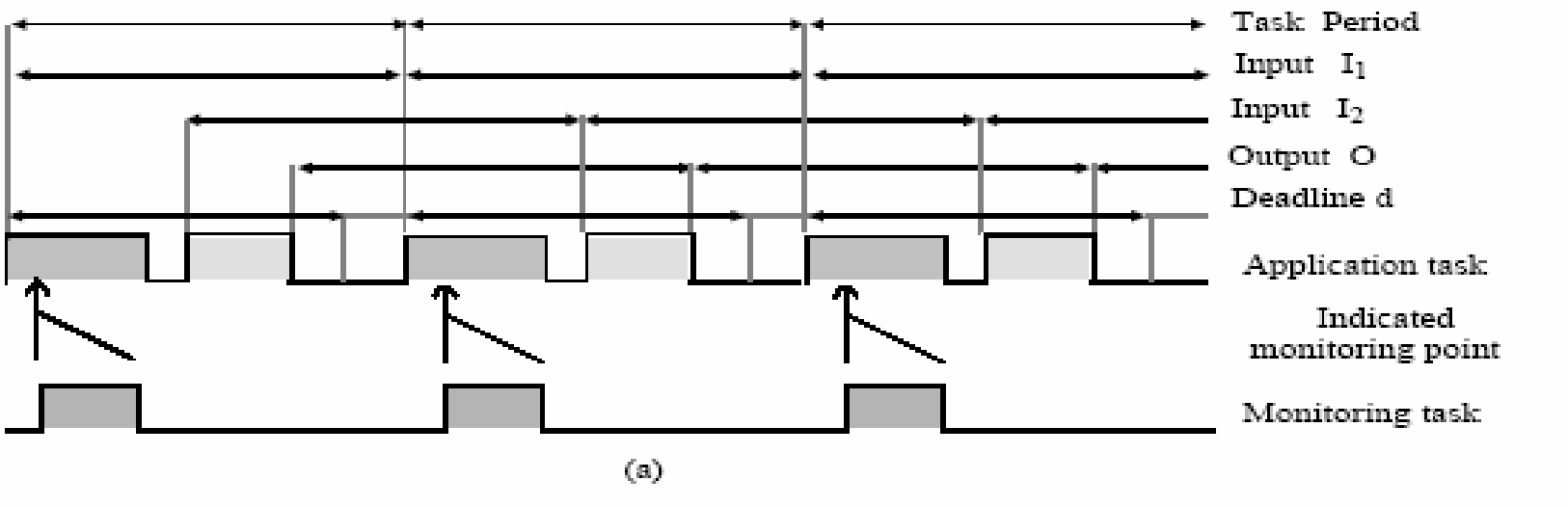
However…

- Monitoring code can be broken into many tasks
- Application can have more than one input point
- New Goal of Paper
  - Find 1 or more input points
  - Non-intrusively put monitoring code in application

More Definitions

- Real time application
  - repeated execution of code at regular intervals
- Pre-input idle time
  - execution suspended, waiting for data
- Post-completion idle time
  - Idle time between successive execution if first input needed to start task not available
- Pre-input and post-completion idle times absorb monitoring activity

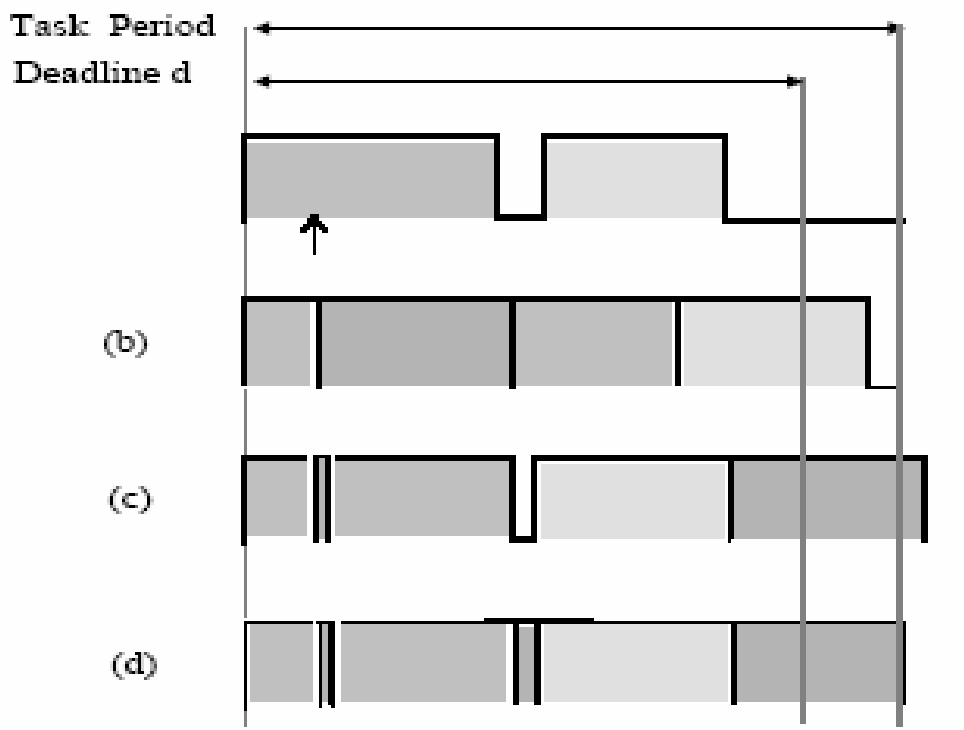
Example of Timing Behavior



- Code segment finishes execution before I_2 arrives
- I_2 has pre-input idle time (waiting for new input)
- After output, there is post-completion idle time

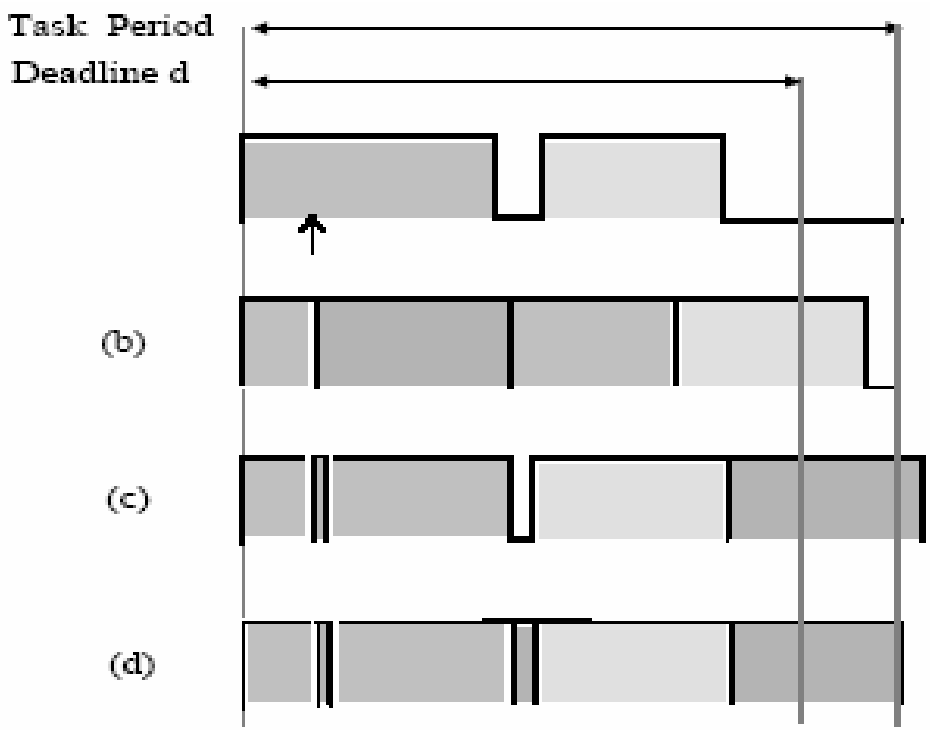
Three approaches to inserting monitoring code

1. Straightforward approach (b)
2. Split monitoring task approach (c)
3. Break up monitoring task with numerous input points approach (d)



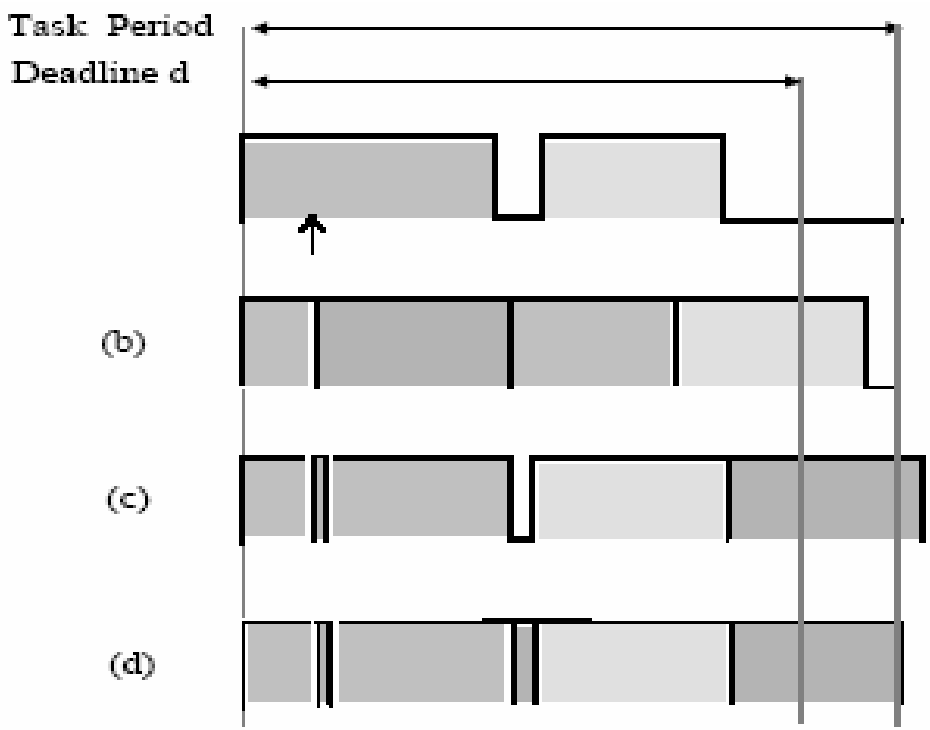
Straightforward Approach

- Insert monitoring task at specified point
- Monitoring task higher priority than application task
- May miss deadline



Split Monitoring Task Approach

- Divide monitoring task into 2 parts
  1. Local saving of data- takes small amount of time
  2. Packaging, transmission of monitoring data – time consuming
- Do local saving of data at specified point
- Do the rest at post-completion idle time

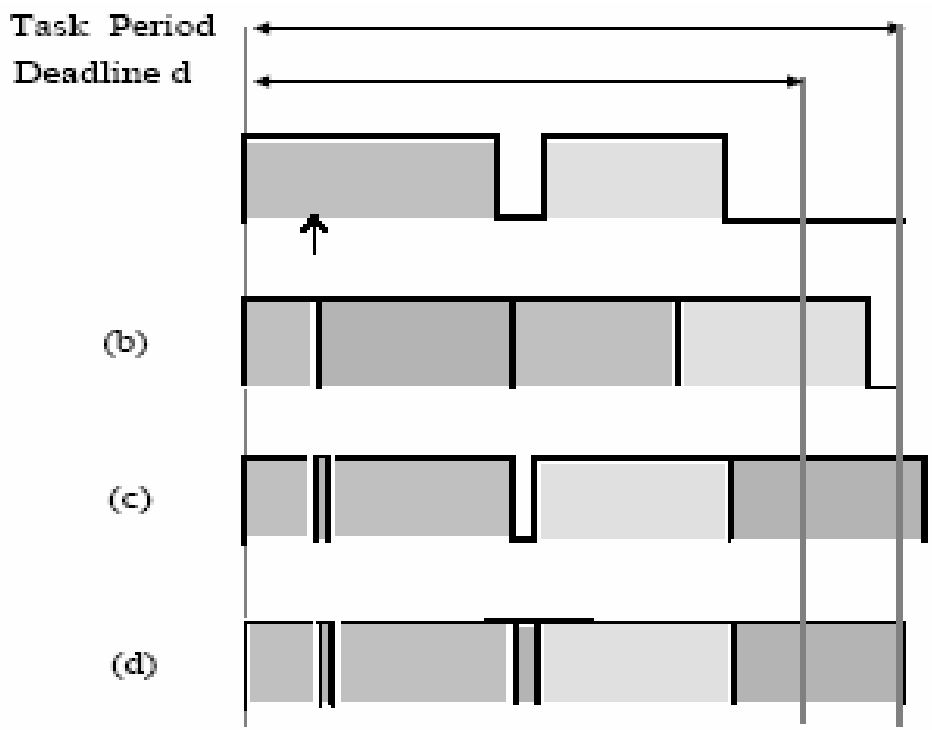


Cons of this approach

- Monitoring task may not finish execution before next iteration
  - Monitoring abandoned and left incomplete
  - If monitoring completes, delays next iteration
- Gives monitoring task lower priority but not effective

Break up monitoring task with numerous input points approach

- Do local saving of data at specified point
- Break up packaging, transmission of monitoring data into several tasks
- Execute at different times
  1. Post-completion idle time
  2. Pre-input idle time if more time needed



Break up monitoring task with numerous input points approach

- Best approach out of all three
- Assures that application task given highest priority
- Monitoring task done non-intrusively

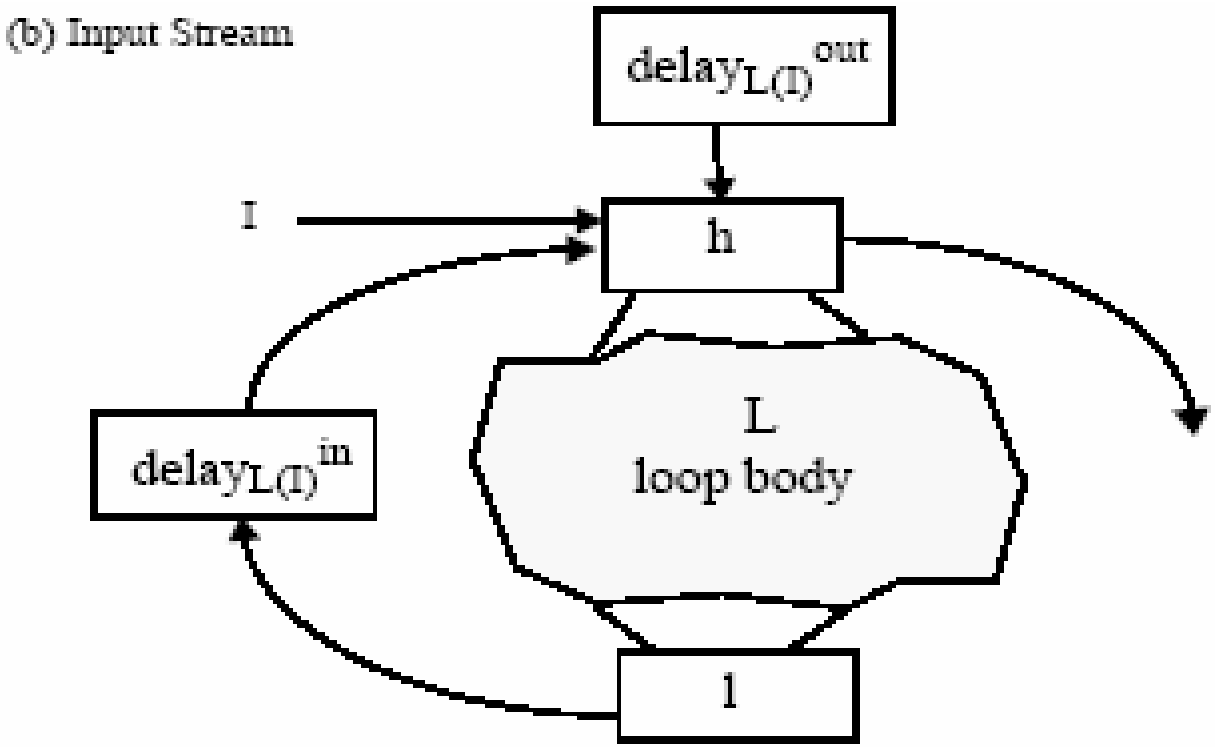
An approach to inserting monitoring code

- How real time application is modeled
  - 1 sequential task executed repeatedly (loop)
  - Output generated at specific intervals (must meet deadlines)
- Goal of Timing Analysis
  - Find start and end time of each statement
  - Find the maximum execution time

Timing Analysis

- Timing analysis performed using control flow graph
- Each interval has
  - Header node, h
  - Last node, L
  - Back edge from L to h (for delays)
- Timing analysis takes 2 passes – Why?
  - Timing information has 2 attributes

Control Flow Graph



The two attributes

- Synthesized attribute – loop execution time computed by examining statements within loop
- Inherited attribute – loop starting time depends on execution time of statements preceding loop

The Two Passes

- First Pass
  - Loop iteration execution time computed
  - Algorithm ComputeExecutionTimes
- Second Pass
  - Starting times of input and output statements computed
  - Uses information from first pass

Algorithm ComputeExecutionTimes

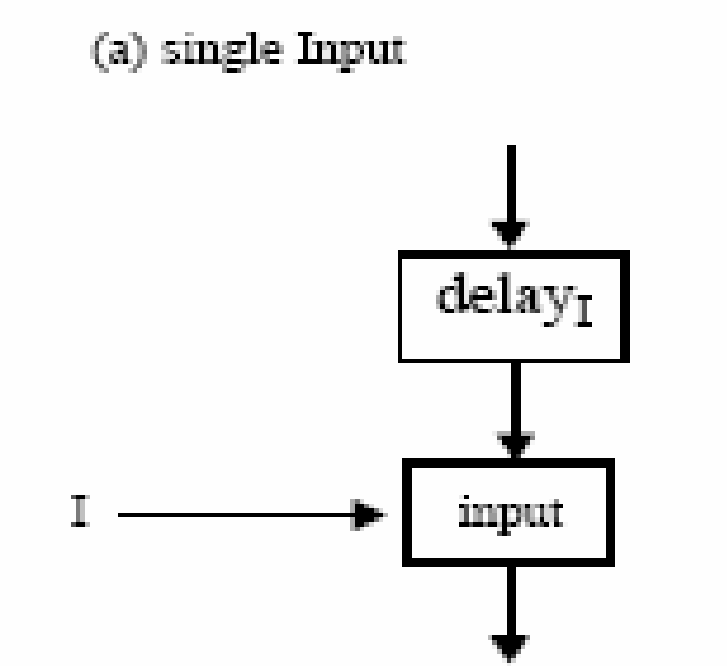
1. **Algorithm** ComputeExecutionTimes {
2. **for each** interval L, in innermost to outermost order **do**
3. – initialize the header node h
4. $start_h^{low_L} = 0$
5. – process the nodes in the interval
6. **for each** node in L in reverse-depth-first order **do**
7. – Let $Pred_i$ be the set of immediate predecessors of i
8. $start_i^{low_L} = MAX_{p \in Pred_i} finish_p$
9. $finish_i^{low_L} = start_i^{low_L} + exec_i$
10. **endfor**
11. – compute loop execution time
12. $iter_L = finish_l^{low_L}$
13. $finish_h^{high_L} = (high_L - low_L + 1) \times iter_L$
14. $exec_L = finish_h^{high_L}$
15. **endfor**
16. }

First Pass in Detail

- Start and completion times of all loop statements computed
- Execution time of single loop iteration
- Execution time of entire loop
- Input and output statement execution times unknown

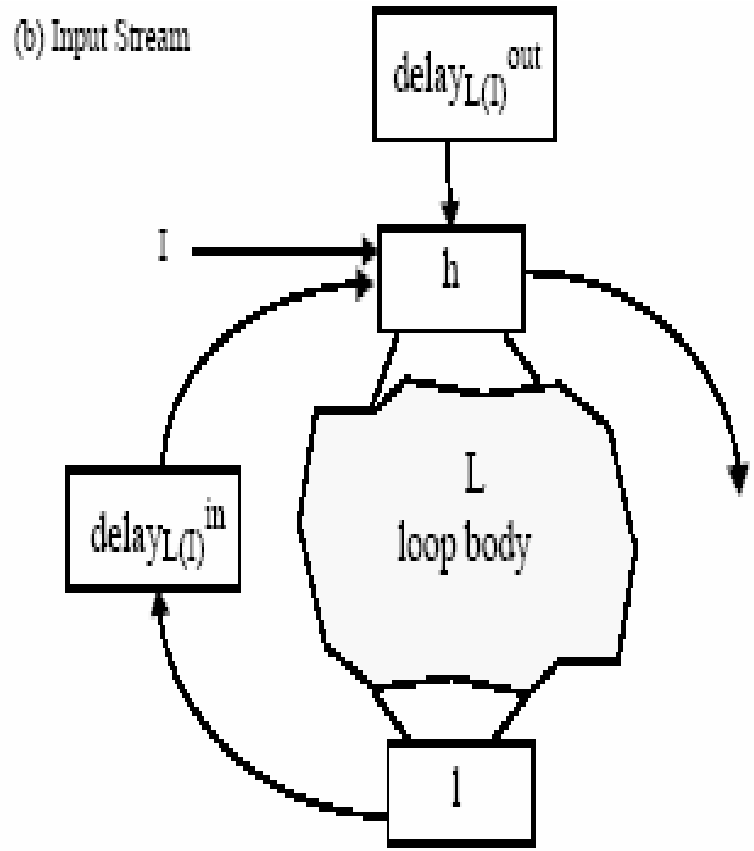
How to compute pre-input idle time

- When only 1 input executed during each iteration
  - If start time of input < when input is available
  - Then pre-input idle time = time when input is available minus start time of input
  - Delay put immediately before input



How to compute pre-input idle time

- If input statement is a stream of inputs
  - Compute initial start time
  - Check if start time of input < when input is available
  - Put delay preceding loop if needed
  - Compare loop iteration time to the interval when inputs become available
  - If delay needed, executed at loop back edge



How to Determine a Missed Deadline

- After timing analysis (see PlaceDelays Algorithm)
- If completion time of an output < output time of deadline
  - Deadline met
  - Can start computing total idle time available for monitoring tasks
- Else
  - Deadline missed

Computing total idle time available for monitoring tasks

- Time it takes to process S amount of data

$$T(S) = a \times S + b \times \lceil \frac{S}{M} \rceil.$$

  - First term = cost of packaging data
  - Second term = cost of transmitting messages of size M
- Therefore, max amount of data that can be processed during time t

$$T^{-1}(t) = S \ni T(S) \leq t < T(S+1).$$

Computing total idle time available for monitoring tasks

- Calculate Pre-input Delay

$$PreInput = \sum_{i=1}^{n} T^{-1}(delay_i) \times count_i$$

- Calculate Post-completion Delay

$$PostCompletion = T^{-1}(TaskInterval - finish_{end})$$

- Scheduling monitoring tasks is not critical
  - If scheduler finds idle time, will schedule monitoring tasks

Thank you!
Any Questions?