First Critique
• Critique the paper instead of just summarizing it. For example
  • Any assumptions not reasonable or unrealistic?
  • Not work in a certain situation?
  • Missed any important factors in design?
  • Do their results support their claims?
  • More…
• If you have critiqued the paper before
  • You must propose several ways to improve this paper
• Due midnight 09/08
  • Topic: CPU utilization control

Recap of Last Class
• Earliest Deadline First (EDF)
  • Utilization bound, implementation, evaluation
• Optimal scheduling algo when relative deadline < period
  • Deadline Monotonic Scheduling (DMS)
    • Response time analysis
  • Earliest Deadline First (EDF)
    • CPU time demand analysis
• Priority inversion
  • Sources
  • Unbounded priority inversion
  • Priority inheritance protocol

Previous Assumptions
• Single processor.
• All tasks are periodic.
• Zero context switch time.
• Relative deadline = period. (relaxed)
• No priority inversion. (relaxed)

Multi-Processor Systems
• Tight coupling among processors.
• Communicate through shared memory and on-board bus.
• Scheduled by a common scheduler/OS.
  • Global scheduling
  • Partitioned scheduling
• States of all processors available to each other.

Distributed Systems
• Loose coupling among processors
• Each processor has its own scheduler/OS
• Costly to acquire states of other processors
• Broad range of systems
  • Processor boards mounted on a VME bus
  • Automobile: hundreds of processors connected through Control Area Networks (CANs)
  • Air traffic control system on a wide area network
End-to-End Task Model

- An (end-to-end) task is composed of multiple subtasks running on multiple processors
  - Message/event
  - Remote method invocation
- Subtasks are subject to precedence constraints
  - Task = a chain/tree/graph of subtasks
  - E.g. ship navigation
- Subtasks: \( T_{i,1}, T_{i,2}, \ldots, T_{i,n(i)} \)
- \( n(i) \): the number of subtasks of \( T_i \)
- Precedence constraint: Job \( J_{i,j} \) cannot be released until \( J_{i,j-1} \) is completed.
- Task allocation
- Strategies
  - Offline, static allocation subject to resource availability
  - Allocate a task when it arrives dynamically
  - Re-allocate (migrate) a task after it starts
- Optimal solutions for maximum schedulability
  - How to meet all deadlines in an optimal way?
  - E.g., minimize the number of needed processors
  - NP-hard: heuristics needed
  - Non-deterministic polynomial-time hard

End-to-End Deadline

- A task is subject to an end-to-end deadline
  - Response time of a particular subtask is less important
  - How to meet end-to-end deadlines?

End-to-End Scheduling Framework

1. Task allocation
2. Synchronization protocol
3. Subdeadline assignment
4. Schedulability analysis

Bin Packing for Task Allocation

- Pack subtasks to bins (processors) with limited capacity
  - Size of a subtask
    - Utilization: \( C_j/P_i \)
  - Capacity of each bin is its utilization bound
  - Goal: minimize the number of bins subject to the capacity constraints
End-to-End Scheduling Framework

1. Task allocation
2. Synchronization protocol
3. Subdeadline assignment
4. Schedulability analysis

Synchronization Requirements

- Enforce precedence constraints
- Allow schedulability analysis
- Low worst-case response time
- Low overhead
- Low average response time

Greedy Protocol

- After a subtask is finished, the next subtask starts immediately
- Release job $J_{i,j,k}$ as soon as $J_{i,j-1,k}$ is completed
- Subsequent subtasks may not be periodic under a greedy protocol
  - Difficult for schedulability analysis
  - High-priority tasks arrive early → high worst-case response time for lower-priority tasks

Properties of Greedy Protocol

- Low overhead
- Low average response time
- Difficult schedulability analysis
  - Subsequent subtasks are no longer periodic
- High worst-case response time

Release Guard

- After a subtask is finished, the next subtask may wait for a while before release
- Every subtask (if not a first subtask) has a release guard, which
  - waits for the preceding subtask for a result/event
  - then releases the job
    - at the point of exact one period from the last release time (Rule 1)
    OR
    - whenever the processor becomes idles (Rule 2)
- Release guard strategy improves average response time without affecting schedulability
Release Guard Illustrated

T1 starts here

T1's deadline

Next release = 4+6 = 10

Release guard releases the job

T3 meets deadline

Properties of Release Guard
- Allow accurate schedulability analysis
- Subsequent subtasks are forced to be periodic
- Short worst-case response time
- Does not require global clock synchronization
- Improved average response time (if rule 2 is used)
- Works best for loosely coupled system!

Sync Protocols Comparison

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>Analysis</th>
<th>WC response time</th>
<th>Avg. response time</th>
<th>Global Info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>Y</td>
<td>N</td>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Release Guard</td>
<td>Y</td>
<td>Y</td>
<td>L</td>
<td>M/H</td>
<td>L</td>
</tr>
</tbody>
</table>

End-to-End Scheduling Framework
1. Task allocation
2. Synchronization protocol
3. Subdeadline assignment
4. Schedulability analysis

Subdeadline Assignment Algorithms
- Notations
  - (Relative) deadline \(D_i\) of task \(T_i\)
  - (Relative) subdeadline \(D_{ij}\) of subtask \(T_{ij}\) \((1 \leq j \leq n(i))\)

- Ultimate Deadline (UD): \(D_g = D_i\)

  Example
  - An end-to-end task T11, with deadline as 12, has 4 subtasks: T111, T12, T13 and T14 with execution times as: 3, 1, 1, 1.
  - \(D_{111} = D_{12} = D_{13} = D_{14} = D_1 = 12\)
  - But T11, T12, T13 must finish earlier than the end-to-end deadline such that T14 can have time to run.

Common Assignment Algorithms
- Proportional Deadline (PD):
  - Assign deadline proportionally to execution time
  \[D_0 = D \frac{C_{ij}}{\sum C_{ik}}\]
  - Example
    - An end-to-end Task T11, with deadline as 12, has 4 subtasks T111, T12, T13 and T14 with execution time as: 3, 1, 1, 1.
    - \(D_{111} = 12 \times \frac{3}{3+1+1+1} = 6\)
    - \(D_{12} = 12 \times \frac{1}{3+1+1+1} = 2\)
    - \(D_{13} = 12 \times \frac{1}{3+1+1+1} = 2\)
    - \(D_{14} = 12 \times \frac{1}{3+1+1+1} = 2\)
End-to-End Scheduling Framework

1. Task allocation
2. Synchronization protocol
3. Subdeadline assignment
4. Schedulability analysis
   • Decide an appropriate scheduling algorithm
     • RMS, DMS, EDF, etc
   • For each processor, conduct uniprocessor schedulability analysis

Summary

• Distributed real-time systems
  • End-to-end task model
  • End-to-end deadline
• End-to-end scheduling framework
  • Task allocation
    • Bin packing
  • Synchronization protocol
    • Greedy protocol, release guard
  • Subdeadline assignment
    • Ultimate deadline, proportional deadline
  • Schedulability analysis

Reading

• Textbook: Chapters 9.1, 9.2, 9.4