Dependable Real-Time Systems

Lecture #6

Professor Jan Jonsson

Department of Computer Science and Engineering
Chalmers University of Technology

Schedulability analysis

Recall from an earlier lecture that we have:

- The priority assignment problem
  - Given a set of tasks, *does there exist an assignment of priorities to these tasks* satisfying the property that the system can be scheduled by a priority-based run-time system such that all task instances will complete by their deadlines?
- The feasibility testing problem
  - Given a set of tasks, *and an assignment of priorities to these tasks*, can the system be scheduled by a priority-based run-time system such that all task instances will complete by their deadlines?

Schedulability analysis

Complexity of feasibility testing: (Leung, 1989; Baruah et al 1990)

The problem of deciding the feasibility of a schedule produced on m ≥ 1 processors by a particular static or dynamic priority assignment is NP-hard in the strong sense.

Observation:

- If an optimal priority assignment can be found in polynomial time, the complexity of schedulability analysis reduces to that of the feasibility testing problem.
- For single-processor systems, there exist optimal priority assignments that can be generated in polynomial time, and there exist pseudo-polynomial time exact feasibility tests.

Feasibility testing revisited

Feasibility testing − exactness of test:

- A feasibility test is sufficient if it with a positive answer shows that a set of tasks is definitely schedulable
  - A negative answer says nothing! A set of tasks can still be schedulable despite a negative answer
- A feasibility test is necessary if it with a negative answer shows that a set of tasks is definitely not schedulable
  - A positive answer says nothing! A set of tasks can still be impossible to schedule despite a positive answer
- An exact feasibility test is both sufficient and necessary
  - If the answer is positive the task set is definitely schedulable, and if the answer is negative the task set is definitely not schedulable

Feasibility testing revisited

Feasibility testing − existing techniques:

- Hyper period analysis
  - In an existing schedule no task execution may miss its deadline
- Guarantee bound analysis
  - The fraction of processor time that is used for executing the task set must not exceed a given bound
- Response time analysis
  - The worst-case response time for each task must not exceed the deadline of the task
- Processor demand analysis
  - The accumulated computation demand for the task set under a given time interval must not exceed the length of the interval

Hyper period analysis

Hyper period analysis:

- When it is not obvious which feasibility analysis should be used for a given task set and a given scheduler it is always possible to generate a schedule by simulating the execution of the tasks, and then check feasibility for individual tasks.
- The schedule interval that is sufficient to investigate is related to the hyper period of the task set, that is, the least common multiple (LCM) of the task periods.

  NOTE: Unless the periods of all tasks are harmonically related (multiples of each other) hyper-period analysis will in general have an exponential time complexity.

Hyper period analysis

Hyper period analysis – the trivial cases:

- For synchronous task sets:

  It is sufficient to investigate the interval $[0, P]$, where P is the hyper period of the task set.

- For asynchronous task sets: (special case)

  It is sufficient to investigate the interval $[0, P]$ if no task instance that arrives within the interval executes beyond time P.

  In all other cases it is necessary to investigate more than one hyper period.

Hyper period analysis

Hyper period analysis – the non-trivial cases:

- For asynchronous task sets: (general case)

  It is necessary to investigate an interval $\left[0, O_{\max} + N_{\text{stable}} \cdot P\right]$ where P is the hyper period of the task set, $O_{\max}$ is the largest offset in the task set, and N_{stable} is the minimum number of hyper periods required for the schedule to start repeating itself (always ≥ 2).

Guarantee bound analysis

Guarantee bound analysis:

- If the accumulated utilization U of all tasks does not exceed a guarantee bound, all timing constraints will be met
- The guarantee bound is expressed as a fraction of the available processing capacity of the system
  (= 100% multiplied by the number of processors)
- The utilization U_i of a task is expressed as the fraction of processing capacity used for executing the task
  - Thus, guarantee bound analysis has polynomial time complexity

$$\text{task utilization} = \frac{C_i}{T_i}$$

$$\text{accumulated utilization} = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Guarantee bound analysis

Guarantee bound analysis for RM: (Liu & Layland, 1973)

- A sufficient condition for RM priority assignment is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq U_{\text{RM}}$$

$$U_{\text{RM}} = n\left(2^{1/n} - 1\right)$$

- The test is only valid if all of the following conditions apply:
  1. Single-processor system, with fully preemptive scheduling
  2. Synchronous task sets
  3. Independent tasks
  4. Periodic or sporadic tasks, where $D_i = T_i$ for all tasks

Guarantee bound analysis

Guarantee bound analysis for EDF: (Liu & Layland, 1973)

- An exact condition for EDF priority assignment is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq U_{\text{EDF}}$$

$$U_{\text{EDF}} = 1$$

- The test is only valid if all of the following conditions apply:
  1. Single-processor system, with fully preemptive scheduling
  2. Synchronous task sets
  3. Independent tasks
  4. Periodic or sporadic tasks, where $D_i = T_i$ for all tasks

Response time analysis

Response time analysis:

- The response time R_i for a task τ_i represents the worst-case completion time of the task when execution interference from other tasks are accounted for.

$$R_i = C_i + I_i$$

$$I_i = \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Here, $hp(i)$ is the set of tasks with higher priority than τ_i

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Response time analysis

Response time analysis:

- An iterative procedure can be used to calculate the response time of a task:

$$R_i^{k+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil C_j$$

- The iteration starts with a value that is guaranteed to be less than or equal to the final value of R_i (e.g. $R_i^0 = C_i$)
- The iteration completes at convergence ($R_i^{n+1} = R_i^n$) or if the response time exceeds the deadline D_i

Response time analysis

Response time analysis − the test: (Joseph & Pandya, 1986)

- An exact condition for any static-priority assignment is

$$\forall i: R_i \leq D_i$$

- The test is only valid if all of the following conditions apply:
  1. Single-processor system, with fully preemptive scheduling
  2. Synchronous task sets
  3. Independent tasks
  4. Periodic or sporadic tasks, where $D_i \leq T_i$ for all tasks

Response time analysis

Recall from an earlier lecture that: (Leung & Whitehead, 1982)

> There exists a pseudo-polynomial time algorithm to decide if a synchronous task set can be scheduled using static priorities on one processor in such a way that all task instances will complete by their deadlines.

Proof:

- The deadline-monotonic priority assignment is optimal for synchronous task sets, and can be obtained in polynomial time
- An exact feasibility test for synchronous task sets on a single processor can be performed in pseudo-polynomial time (using response-time analysis).

Response time analysis

Response time analysis − time complexity:

Response time analysis has pseudo-polynomial time complexity

Proof:

- calculating the response-time for task τ_i requires no more than D_i iterations
- since $D_i \leq T_i$ the number of iterations needed to calculate the response-time for task τ_i is bounded above by T_i
- the procedure for calculating the response-time for each task is therefore of time complexity $O(\max\{T_i\})$
- the longest period of a task is also the largest number in the problem instance

Processor demand analysis

Processor demand analysis:

- The processor demand for a task τ_i in a given time interval $[0, L]$ is the amount of processor time that the task needs in the interval in order to meet the deadlines that fall within the interval.
- Let N_i^L represent the number of instances of τ_i that must complete execution before L .
- The total processor demand up to L is

$$C_P(0, L) = \sum_{i=1}^{n} N_i^L C_i$$

Processor demand analysis

Processor demand analysis:

- We can express N_i^L as

$$N_i^L = \left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1$$

- The total processor demand is thus

$$C_P(0, L) = \sum_{i=1}^{n} \left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i$$

Processor demand analysis

Processor demand analysis – the test: (Baruah et al., 1990)

- An exact condition for EDF priority assignment is

$$\forall L \in K : C_P(0, L) \leq L$$

- The test is only valid if all of the following conditions apply:
  1. Single-processor system, with fully preemptive scheduling
  2. Synchronous task sets
  3. Independent tasks
  4. Periodic or sporadic tasks, where $D_i \leq T_i$ for all tasks

Processor demand analysis

Processor demand analysis – the test: (Baruah et al., 1990)

- The set of control points K is in the general case:

$$K = \left\{ D_i^k \;\middle|\; D_i^k = kT_i + D_i,\ D_i^k \le L_{\max},\ 1 \le i \le n,\ k \ge 0 \right\}$$

$$L_{\max} = \mathrm{LCM}\left\{ T_1, \ldots, T_n \right\}$$

Observation:

The general case of processor demand analysis has the same time complexity as hyper period analysis, i.e., exponential time in the worst case.

Processor demand analysis

Recall from an earlier lecture that: (Baruah et al, 1990)

> There exists a pseudo-polynomial time algorithm to decide if a synchronous task set can be scheduled using dynamic priorities on one processor in such a way that all task instances will complete by their deadlines.

Proof:

- The earliest-deadline-first priority assignment is optimal for synchronous task sets, and can be obtained in polynomial time
- An exact feasibility test for synchronous task sets on a single processor can be performed in pseudo-polynomial time (using processor-demand analysis) if the total task utilization is < 1.

Processor demand analysis

Processor demand analysis – the test: (Baruah et al., 1990)

- The set of control points K is in most cases:

$$K = \left\{ D_i^k \;\middle|\; D_i^k = kT_i + D_i,\ D_i^k \le L_{\max},\ 1 \le i \le n,\ k \ge 0 \right\}$$

$$L_{\max} = \max\left\{ D_1, \ldots, D_n, \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{1 - U} \right\}$$

Observation:

$$L_{\max} \le \max\left\{ \max\{D_i\}, \frac{U}{1-U} \max\{T_i - D_i\} \right\} \le \max\left\{ \max\{T_i\}, \frac{U}{1-U} \max\{T_i\} \right\}$$

Processor demand analysis

Processor demand analysis − time complexity:

Processor demand analysis has pseudo-polynomial time complexity if total task utilization is less than 100%

Proof:

- the number of control points needed to check the processor demand is bounded above by

$$Q_L^{\max} = \max\left\{\max\{T_i\}, \frac{U}{1-U}\max\{T_i\}\right\} = \max\left\{1, \frac{U}{1-U}\right\} \bullet \max\{T_i\}$$

- since $U/(1-U)$ is a constant the procedure for calculating the processor demand is therefore of time complexity $O(\max\{T_i\})$
- the longest period of a task is also the largest number in the problem instance

Schedulability analysis

Recall from an earlier lecture that we have:

- The priority assignment problem
  - Given a set of tasks, *does there exist an assignment of priorities to these tasks* satisfying the property that the system can be scheduled by a priority-based run-time system such that all task instances will complete by their deadlines?
- The feasibility testing problem
  - Given a set of tasks, *and an assignment of priorities to these tasks*, can the system be scheduled by a priority-based run-time system such that all task instances will complete by their deadlines?

Schedulability analysis

Complexity of feasibility testing: (Leung, 1989; Baruah et al 1990)

The problem of deciding the feasibility of a schedule produced on m ≥ 1 processors by a particular static or dynamic priority assignment is NP-hard in the strong sense.

Observation:

- If an optimal priority assignment can be found in polynomial time, the complexity of schedulability analysis reduces to that of the feasibility testing problem.
- An optimal static-priority assignment can be generated in pseudo-polynomial time, by using the OPA algorithm and an OPA compatible pseudo-polynomial time feasibility test.

Schedulability analysis

OPA algorithm (Audsley, 1991)

```
for each priority level k, lowest first
{
   for each unassigned task τ
   {
      if τ is schedulable at priority k
      according to schedulability test S
      with all unassigned tasks assumed to
      have higher priorities
      {
         assign τ to priority k
         break (continue outer loop)
      }
   }
   return unschedulable
}
return schedulable
```

Schedulability analysis

Conditions for OPA compatibility: (Davis & Burns, 2009)

Condition 1: The schedulability of a task τ may, according to test S, depend on any independent properties of tasks with priorities higher than τ, but not on any properties of those tasks that depend on their relative priority ordering.

Condition 2: The schedulability of a task τ may, according to test S, depend on any independent properties of tasks with priorities lower than τ, but not on any properties of those tasks that depend on their relative priority ordering.

Condition 3: When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to test S, if it was previously schedulable at the lower priority.

Schedulability analysis

Conditions for OPA compatibility:

- Is the standard single-processor response-time test OPA compatible?

$$R_i^{k+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil C_j$$

$$\forall i : R_i \leq D_i$$

- We need to check whether or not all three conditions for OPA compatibility hold for the given test.