Exam for Real Time Systems

2024 Jan 11

Wang Yi

Important Instructions:

1. No course material or computer/calculator are allowed, only a pen and a dictionary.
2. Please mark which course you are registered for:

○ **5hp** (1DT063)	○ **10hp** (1DT004)
You need to solve problems 1–4 only (100p)	You need to solve *all* problems (150p)

3. For each problem, a number of choices/statements are given, that may be correct or wrong.

   You are asked to mark only the **correct** ones.

   If you mark all correct choices for each problem, it will give you all points of the problem; otherwise each wrong choice (marked) will result in "negative points" proportionally.

.....

Problem 1. *1. (5p) Why is it so difficult to estimate the Worst-Case Execution Time of a real-time task?*

( ) *it may have too many loops*
( ) *it may need too much memory in execution*
( ) *it may have too many instructions*
( ) *it may consume too much energy*
( ) *the platform may support mutlti-tasking*
( ) *the platform may have caches*
( ) *it is impossible to decide whether a program terminates or not*
( ) *the platform may have a multicore processor*

2. (5p) A real-time programming anguage like Ada Ada compiler should support:

( ) *object orientation*
( ) *multi-tasking*
( ) *implementation of real-time tasks*
( ) *task synchronization*
( ) *user-defined scheduling policies*

3. (5p) Resource servers can be useful for

( ) *upgrading the hardware platform of an existing system*
( ) *isolating applications of different types*
( ) *improving the average response times of soft real-time tasks*
( ) *guaranteeing the deadline of event-driven real-time tasks*

4. *(5p) When RMS is used, a task set is non-schedulable if its utilization is*

   ( ) *over* 100%

   ( ) *over* 69.3%, *but less than* 100%

   ( ) *less than* $n * (2^{1/n} - 1)$ *where* n *is the number of tasks*

5. *(5p) The advantages of fixed-priority scheduling include:*

   ( ) *it is stable*

   ( ) *it can be used to implement predictable systems*

   ( ) *it is the best policy for optimizing resource utilization*

   ( ) *it is the best policy for achieving the schedulability of real-time tasks*

   ( ) *it is the best policy for improving the average response time of real-time tasks*

   ( ) *it can be used to implement safety-critical systems*

6. *(5p) Assume a set of sporadic sets* (C_i, D_i, T_i) *where deadline* $D_i \leq T_i$. *If DMS is used, the task set is schedulable if*

   ( ) $(\sum C_i/D_i) \leq 100\%$

   ( ) $(\sum C_i/T_i) \leq 100\%$

   ( ) $(\sum C_i/D_i) \leq n * (2^{1/n} - 1)$ *where* n *is the number of tasks*

   ( ) $(\sum C_i/T_i) \leq n * (2^{1/n} - 1)$ *where* n *is the number of tasks*

7. *(5p) A system with periodic tasks may contain dependent tasks with input/output relations, which causes release jitters. Assume that the dependent tasks have the same period and task B depending on task A is defined as follows: the computed result of a job of task A is used as the input of a job of task B (and thus task B suffers a release jitter due to the varying response times of task A).*

   Is it possible to avoid this and thus avoid release jitters?

   ( ) *this is not possible at all*

   ( ) *yes, deliver the result of a job when it is completed*

   ( ) *yes, deliver the result of a job at the worst-cast response time estimated*

   ( ) *yes, deliver the result of a job by the end of each period*

   ( ) *yes, deliver the result of a job by the given deadline if the tasks are schedulable*

8. *(5p) For a set of periodic task scdueduled on a single processor, what are the differences between DMS and EDF?*

   ( ) *EDF is stable; DMS is non stable*

   ( ) *EDF is non stable; DMS is stable*

   ( ) *DMS uses fixed priorities; EDF uses dynamic priorities*

   ( ) *EDF is good for resource utilization; DMS is not as good as EDF*

   ( ) *EDF is optimal; DMS is not*

   ( ) *EDF is optimal; DMS is also optimal*

9. *(5p) Assume that a set of periodic tasks with deadlines equal to periods is schedulable on a single processor system with fixed-priority scheduling accoding to a user-defined fixed-priority assignment. Then the task set is also schedulable using*

   ( ) *non-preemptive RMS*

   ( ) *RMS*

   ( ) *non-Preemptive EDF*

   ( ) *EDF*

   ( ) *non-preemptive DMS*

   ( ) *DMS*

( ) *FIFO*

( ) *Shortest Job First Scheduling*

10. *(5p) When you use CAN, you should*

    ( ) *make sure that messages sent from different nodes have different identities*

    ( ) *know that sending messages with the same identities by different nodes may lead to errors*

    ( ) *send messages with periodic tasks*

    ( ) *assign the priorities of messages according to the periods of the sending tasks*

    ( ) *make sure that the priority ordering of messages is a total order according to their id's with the lower id number, the higher priority for messages*

    ( ) *send messages with receiver's address*

Problem 2. *(20p) Assume a CAN bus running at 1 Mbits per second, connecting 3 stations (nodes) A, B and C. There are 12 messages of fixed size (200 bits each), with 12 identities (1 to 12), sent by the nodes as follows:*

1. *On node A, there are four tasks each sending messages with identities 1,2,3,4 at most every 1ms, 2ms, 2ms and 7ms respectively.*
2. *On node B, there are four tasks each sending messages with identities 5,6,7,8 at most every 8ms, 10ms, 18ms and 20ms respectively.*
3. *On node C, there are four tasks each sending messages with identities 9, 10, 11, 12 at most every 30ms, 31ms, 43ms and 50ms respectively.*

The worst case transmission delay (i.e. the time period from queuing to completed message transmission) for message with identity 1 is bounded by:

( ) $0.10ms$

( ) $0.30ms$

( ) $0.50ms$

( ) $0.70ms$

( ) $0.90ms$

( ) $1.10ms$

The worst case transmission delay for message with identity 6 is bounded by:

( ) $0.2ms$

( ) $0.40ms$

( ) $0.60ms$

( ) $0.80ms$

( ) $1.00ms$

( ) $1.20ms$

( ) $2.20ms$

( ) $3.20ms$

( ) $4.20ms$

( ) $5.20ms$

Problem 3. *(15p) The two standard operations P and V on semaphores are implemented according to the following pseudo-code:*

P(S)	V(S)
(1) Disable-interrupt; (2) if S.counter > 0 (3) then S.counter -- 1; (4) else { (5) insert(current-task, S.queue); (6) schedule()}; (7) Enable-interrupt	(8) Disable-interrupt; (9) If non-empty(S.queue) (10) then { (11) next-to-run := get-first(S.queue); (12) insert(next-to-run, Ready-queue); (13) schedule()} (14) else S.counter ++ 1; (15) Enable-interrupt

To implement HLP (Immediate Priority Inheritance Protocol), the following changes may be made:

( ) *between line (3) and (4), it should be extended by* "save the priority of the current-task and raise it to the ceiling of S"

( ) *between line (4) and (5), it should be extended by* "save the priority of the current-task and raise it to the ceiling of S"

( ) *between line (14) and (15), it should be extended by* "restore the priority of the current-task to the previously saved priority"

Problem 4. *(15p) Assume an event-driven task with computing times C_i and deadlines D_i (assume $C_i \leq D_i$). The jobs of the task may arrive sporadically but the minimal distance between two jobs is $T_i > D_i$. Design a polling server for the task with period T and capacity $C = C_i$ such that no deadline is violated when RMS is used for scheduling. Assume that $C \leq T$ and R is the response time estimated using RMS analysis when the server is considered as a periodic task. Further, assume $R \leq T$. Identify correct statements:*

( ) $T \leq D_i$ *is necessary*

( ) $T \leq D_i$ *is sufficient*

( ) $T + T \leq D_i$ *is necessary*

( ) $T + T \leq D_i$ *is sufficient*

( ) $T + R \leq D_i$ *is necessary*

( ) $T + R \leq D_i$ *is sufficient*

( ) $T + C \leq D_i$ *is necessary*

( ) $T + C \leq D_i$ *is sufficient*

Problem 5. *(20p) Assume a set of sporandic tasks running on a platform of N processors each with the same speed S (or processing capacity). Assume that the task set is schedulable with global scheduling. Identify the correct statements:*

( ) *The task set with some tasks removed is also schedulable.*

( ) *The task set with the computing time of some tasks reduced is also schedulable.*

( ) *The task set is also schedulable on the same platform with more memory.*

( ) *The task set is also schedulable on the same platform with more caches.*

( ) *The task set is also schedulable with a platform of more processors of speed S.*

( ) *The task set is also schedulable with a platform of $10 * N$ processors of speed $S/10$.*

( ) *The task set is also schedulable with a single powerful processor of speed $N * S$.*

( ) *The task set is surely feasible with partioned scheduling.*

( ) *The task set is surely feasible with partitioned scheduling (allowing job-splitting) if its utilzation is below $N * 69.3\%$.*

Problem 6. *(10p) Assume a set of periodic tasks with periods as deadlines and utilizations as follows:* $0.80, 0.90, 0.9, 0.4, 0.7, 0.7, 0.3, 0.8, 0.9, 0.2, 0.1, 0.2, 0.1$

If you partition the tasks onto a multiprocessor platform and then run the respective scheduling policy, what is the least number of processors you need?

( ) *9-12 with EDF*

( ) *5-8 with EDF*

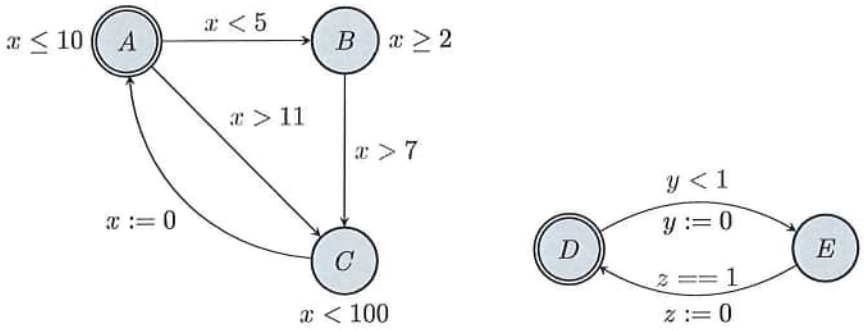
( ) *1-4 with EDF*

( ) *9-12 with RMS*

( ) *5-8 with RMS*

( ) *1-4 with RMS*

Problem 7. *(20p)*



(a) Timed automaton T_1

(b) Timed automaton T_2

Figure 1: Two timed automata in which x, y and z are clocks. Note that both automata are separate systems, they do *not* run together.

Take a look at the timed automaton T_1 in Figure 1(a). Identify correct statements:

( ) *Location B is reachable*

( ) *x can be below 5 at location C*

( ) *There are no edges that can be removed without changing the set of reachable states*

( ) *It satisfies $E <> T_1.B$*

( ) *It satisfies $E <> (T_1.C$ and $x > 5)$*

( ) *It satisfies $E <> (T_1.C$ and $x < 5)$*

( ) *It satisfies $A <> x \geq 1$*

Take a look at the timed automaton T_2 in Figure 1(b). Identify correct statements:

( ) *It is possible that T_2 deadlocks*

( ) *No matter what, T_2 will eventually deadlock*