Real-Time Systems

Solutions to final exam March 18, 2019 (version 20190318)

PROBLEM 1

a) False: For a sporadic task the time interval between two, subsequent, arrivals is guaranteed to never be less than a minimum value.

b) True: TinyTimber's AFTER() construct allows the programmer to call a method after a delay relative to the calling method's baseline, thereby eliminating any systematic time skew.

c) False: For an NP-complete problem to have pseudo-polynomial time complexity the largest number in the problem cannot be bounded by the input length (size) of the problem.

d) True: ICPP cannot cause deadlock if all tasks use the protocol to access shared resources.

e) False: The term false path in execution-time analysis refers to a part of the program code that will never be executed at run-time.

f) True: If we know that the task set is not schedulable then a *sufficient* test must have resulted in the outcome 'False'. This is because, for sufficient tests, the outcome 'True' always means that the task set is schedulable.

PROBLEM 2

a) In single-processor systems, the mutual exclusion is guaranteed by disabling the processor's interrupt service mechanism ("interrupt masking") while the critical region is executed. This way, unwanted task switches in the critical region (caused by e.g. timer interrupts) are avoided. This method is not used in multi-processor systems since interrupt management is typically not synchronized between the processors.

b) In multi-processor systems with shared memory, a test-and-set instruction is used for handling critical regions. A test-and-set instruction is a processor instruction that reads from and writes to a variable in one atomic operation. In a multi-processor system, the atomic operation is guaranteed by locking (disabling access to) the memory bus during the entire operation.

PROBLEM 3

a) The WCET of `Control` is dependent on the WCET of function `Calc`.

WCET of "Calc":

$WCET(Calc(x)) =$
$\{Declare, i\} + \{Declare, r\} + \{Assign, i\}) + \{Assign, r\} +$
$(3+1) \cdot \{Compare, i < 3\} + 3 \cdot (\{Multiply, r * x\} + \{Assign, r\} + \{Add, i + 1\} + \{Assign, i\}) +$
$\{Subtract, r - 1\} + \{Assign, r\} + \{Return, r\} \ =$
$1+1+1+1+4 \cdot 2+3 \cdot (5+1+3+1)+3+1+2 = 4+8+30+6 = 48$

WCET of "Control":

$WCET(Control) =$
$\{Declare, c\} + \{Declare, r\} + \{Assign, c\}) +$
$\{Call, Calc(c)\} + WCET(Calc(c)) + \{Divide, Calc(c)/3\} + \{Assign, r\} + \{Compare, r <= 800\} +$
$\max(\{Shift, r\} + \{Assign, r\}, \{Multiply, 3 * r\} + \{Divide, 3 * r/289\}) + \{Add, 3 * r/289 + 2\}) +$
$\{Assign, r\}) \ =$
$1+1+1+2+ WCET(Calc(c)) + 8+1+2+\max(2+1, 5+8+3+1)+1 =$
$17 + \max(3, 17) + WCET(Calc(c))$

The function `Calc(x)` calculates the polynomial $x^4 - 1$ which, with the given input port data range $[-9, +9]$, has the largest value $9^4 - 1 = 6560$. The comparison in the `if`-statement in `Control` then becomes $6560/3 = 2187 \leq 800$, which is false. Thus, the longer path in the `if`-statement will be executed.

$WCET(Control) = 17 + \max(3, 17) + WCET(Calc(c)) = 17 + 17 + 48 = 82 > 73$

The deadline is not met!

b) We notice that, if the shorter path in the `if`-statement would be executed, we get:

$WCET(Control) = 17 + 3 + WCET(Calc(c)) = 17 + 3 + 48 = 68 < 73$

Thus, in order to find the largest input port data range for which `Control` will meet its deadline we must make sure that the shorter path is always taken. This happens when $Calc(c)/3 \leq 800$, that is, when $Calc(c) \leq 2400$. Since `Calc(x)` calculates the polynom $x^4 - 1$ the largest permitted data range is $[-7, +7]$, since $7^4 - 1 = 2400$.

c) The new function `Calc(x)` is functionally compatible with the old function since $(x^2-1)((x^2-1)+2) = (x^2-1)(x^2+1) = x^4 - 1$. However, the WCET of the new function is significantly smaller:

$WCET(Calc(x)) =$
$\{Declare, r\} + \{Multiply, x * x\} + \{Subtract, x * x - 1\} + \{Assign, r\} +$
$\{Add, r + 2\} + \{Multiply, r * (r + 2)\} + \{Assign, r\} \ + \{Return, r\} \ =$
$1+5+3+1+3+5+1+2 = 21$

With the original input port data range $[-9, +9]$ we get:

$WCET(Control) = 17 + \max(3, 17) + WCET(Calc(c)) = 17 + 17 + 21 = 55 < 73$

The deadline is met!

PROBLEM 4

a), b) The tasks T1, T2 and T3 should normally reside in three separate objects, but since their execution is precedence-constrained a solution where they share one object (A) is also correct.

```
Time max_wcet = 0;
Time start;
Time diff;

void T1(TaskObj *self, int u) {

    start = CURRENT_OFFSET();

    Action400(); // Do work for 400 microseconds

    diff = CURRENT_OFFSET() - start;

    BEFORE(USEC(1500), self, T2, 0); // Keep current baseline
}

void T2(TaskObj *self, int u) {

    start = CURRENT_OFFSET();

    Action600(); // Do work for 600 microseconds

    diff += CURRENT_OFFSET() - start;

    BEFORE(USEC(2000), self, T3, 0); // Keep current baseline
}

void T3(TaskObj *self, int u) {

    start = CURRENT_OFFSET();

    Action700(); // Do work for 700 microseconds

    diff += CURRENT_OFFSET() - start;

    if (diff > max_wcet)
        max_wcet = diff;

    SEND(USEC(2300), USEC(700), self, T1, 0);
}

void kickoff(TaskObj *self, int u) {

    BEFORE(USEC(700), &A, T1, 0);
}

main() {
    return TINYTIMBER(&A, kickoff, 0);
}
```

PROBLEM 5

The two versions of response-time analysis are:

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

$$R_i^{k+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^k}{(T_j - \alpha)} \right\rceil C_j \qquad \forall i : R_i \leq D_i \tag{2}$$

a) No, we cannot provide a guarantee. This is because the schedulability of task τ_i based on the analysis in Eq. (1) only considers the amount of interference by the higher-priority task τ_j regardless of whether τ_j meets its deadline or not. The following example task set demonstrates such a situation, for the DM priority assignment.

	C_i	D_i	T_i
τ_1	2	3	5
τ_2	4	7	10
τ_3	2	12	20

The response time of the lowest-priority task τ_3 is $R_3 = 10$. However, task τ_2 (which has priority higher than τ_3) misses its deadline ($R_2 = 8$). Therefore, the lower-priority task τ_3 meets its deadline while the higher-priority task τ_2 misses its deadline.

b) The interference by task τ_j on a lower-priority task τ_i using Eq. (2) is never smaller than the interference by task τ_j on τ_i using Eq. (1), because $\alpha \geq 0$ and hence

$$\left\lceil \frac{R_i^k}{(T_j - \alpha)} \right\rceil \geq \left\lceil \frac{R_i^k}{T_j} \right\rceil$$

Consequently, if the response-time analysis in Eq. (2) is satisfied for τ_i, then task τ_i must meet its deadline.

If, on the other hand, the response-time analysis in Eq. (2) would indicate failure with $\alpha > 0$ we cannot draw any conclusions regarding schedulability because the interference may then be overestimated. Thus, the response-time analysis in Eq. (2) is only a sufficient test for $\alpha > 0$.

c) We will calculate the response time of each task based on the analysis in Eq. (2), assuming $\alpha = 2$, and compare it against the corresponding task deadline.

$R_1 = C_1 = 4 = 4 \leq T_1 = 10$ (task τ_1 meets its deadline)

Assume that $R_2^0 = C_2 = 5$.

$$R_2^1 = C_2 + \lceil \frac{R_2^0}{T_1 - 2} \rceil \cdot C_1 = 5 + \lceil \frac{5}{10-2} \rceil \cdot 4 = 5 + 1 \cdot 4 = 9$$

$$R_2^2 = C_2 + \lceil \frac{R_2^1}{T_1 - 2} \rceil \cdot C_1 = 5 + \lceil \frac{9}{10-2} \rceil \cdot 4 = 5 + 2 \cdot 4 = 13$$

$$R_2^3 = C_2 + \lceil \frac{R_2^2}{T_1 - 2} \rceil \cdot C_1 = 5 + \lceil \frac{13}{10-2} \rceil \cdot 4 = 5 + 2 \cdot 4 = 13$$

Since $R_2^3 = R_2^2 = 13 \leq T_2 = 18,$ we have $R_2 = 13$ (task τ_2 meets its deadline).

Assume that $R_3^0 = C_3 = 4$

$$R_3^1 = C_3 + \lceil \frac{R_3^0}{T_1 - 2} \rceil \cdot C_1 + \lceil \frac{R_3^0}{T_2 - 2} \rceil \cdot C_2 = 4 + \lceil \frac{4}{10-2} \rceil \cdot 4 + \lceil \frac{4}{18-2} \rceil \cdot 5 = 4 + 1 \cdot 4 + 1 \cdot 5 = 13$$
$$R_3^2 = C_3 + \lceil \frac{R_3^1}{T_1 - 2} \rceil \cdot C_1 + \lceil \frac{R_3^1}{T_2 - 2} \rceil \cdot C_2 = 4 + \lceil \frac{13}{10-2} \rceil \cdot 4 + \lceil \frac{13}{18-2} \rceil \cdot 5 = 4 + 2 \cdot 4 + 1 \cdot 5 = 17$$
$$R_3^3 = C_3 + \lceil \frac{R_3^2}{T_1 - 2} \rceil \cdot C_1 + \lceil \frac{R_3^2}{T_2 - 2} \rceil \cdot C_2 = 4 + \lceil \frac{17}{10-2} \rceil \cdot 4 + \lceil \frac{17}{18-2} \rceil \cdot 5 = 4 + 3 \cdot 4 + 2 \cdot 5 = 26$$

Since $R_3^3 = 26 > T_2 = 20,$ task τ_2 cannot be guaranteed to meet its deadline.

However, since the response-time analysis in Eq. (2) is only a sufficient test for $\alpha = 2$, we cannot conclude anything regarding the schedulability of the task set. The task set is in fact schedulable, as can be shown by using the exact response-time analysis in Eq. (1), or by using the new response-time analysis in Eq. (2) assuming $\alpha = 1$.

PROBLEM 6

We apply processor-demand analysis to determine the maximum value of C_1. The hyper-period of the task set is LCM{10, 20, 40} = 40.

The set of control points are $K = \{6, 8, 10, 18, 26, 28, 38\}$.

Consider $L = 6$.

$N_1^L \cdot C_1 = (\lfloor \frac{6-8}{10} \rfloor + 1) \cdot C_1 = 0 \qquad N_2^L \cdot C_2 = (\lfloor \frac{6-6}{20} \rfloor + 1) \cdot C_2 = 2$

$N_3^L \cdot C_3 = (\lfloor \frac{6-10}{40} \rfloor + 1) \cdot C_3 = 0$

$C_P(0, L) = C_P(0, 6) = 0 + 2 + 0 = 2 \leq L = 6.$

Consider $L = 8$.

$N_1^L \cdot C_1 = (\lfloor \frac{8-8}{10} \rfloor + 1) \cdot C_1 = C_1 \qquad N_2^L \cdot C_2 = (\lfloor \frac{8-6}{20} \rfloor + 1) \cdot C_2 = 2$

$N_3^L \cdot C_3 = (\lfloor \frac{8-10}{40} \rfloor + 1) \cdot C_3 = 0$

$C_P(0, L) = C_P(0, 8) = C1 + 2 + 0 = C_1 + 2.$

To satisfy the deadline, we must have $C_P(0, L) = C_1 + 2 \leq L = 8$. Therefore, $C_1 \leq 6$.

Consider $L = 10$.

$N_1^L \cdot C_1 = (\lfloor \frac{10-8}{10} \rfloor + 1) \cdot C_1 = C_1 \qquad N_2^L \cdot C_2 = (\lfloor \frac{10-6}{20} \rfloor + 1) \cdot C_2 = 2$

$N_3^L \cdot C_3 = (\lfloor \frac{10-10}{40} \rfloor + 1) \cdot C_3 = C_3 = 2C_1$

$C_P(0, L) = C_P(0, 10) = C1 + 2 + 2C_1 = 3C_1 + 2.$

To satisfy the deadline, we must have $C_P(0, L) = 3C_1 + 2 \leq L = 10$. Therefore, $C_1 \leq 8/3$. Since C_1 is an integer, the value of C_1 is upper bounded by $\lfloor 8/3 \rfloor$. Consequently, $C_1 \leq 2$.

Consider $L = 18$.

$N_1^L \cdot C_1 = (\lfloor \frac{18-8}{10} \rfloor + 1) \cdot C_1 = 2C_1 \qquad N_2^L \cdot C_2 = (\lfloor \frac{18-6}{20} \rfloor + 1) \cdot C_2 = 2$

$N_3^L \cdot C_3 = (\lfloor \frac{18-10}{40} \rfloor + 1) \cdot C_3 = C_3 = 2C_1$

$C_P(0, L) = C_P(0, 18) = 2C1 + 2 + 2C_1 = 4C_1 + 2.$

To satisfy the deadline, we must have $C_P(0, L) = 4C_1 + 2 \leq L = 18$. Therefore, $C_1 \leq 4$. Since for the control point $L = 10$ we must have $C_1 \leq 2$, the schedulability for all $L = 6, 8, 10, 18$ is satisfied if $C_1 \leq 2$.

Consider $L = 26$.

$N_1^L \cdot C_1 = (\lfloor \frac{26-8}{10} \rfloor + 1) \cdot C_1 = 2C_1 \qquad N_2^L \cdot C_2 = (\lfloor \frac{26-6}{20} \rfloor + 1) \cdot C_2 = 4$

$N_3^L \cdot C_3 = (\lfloor \frac{26-10}{40} \rfloor + 1) \cdot C_3 = C_3 = 2C_1$

$C_P(0, L) = C_P(0, 26) = 2C1 + 4 + 2C_1 = 4C_1 + 4.$

To satisfy the deadline, we must have $C_P(0, L) = 4C_1 + 4 \leq L = 26$. Therefore, $C_1 \leq 22/4 = 5.5$. Since for the control point $L = 10$ we must have $C_1 \leq 2$, the schedulability for all $L = 6, 8, 10, 18, 26$ is satisfied if $C_1 \leq 2$.

Consider $L = 28$.

$N_1^L \cdot C_1 = (\lfloor \frac{28-8}{10} \rfloor + 1) \cdot C_1 = 3C_1 \qquad N_2^L \cdot C_2 = (\lfloor \frac{28-6}{20} \rfloor + 1) \cdot C_2 = 4$

$N_3^L \cdot C_3 = (\lfloor \frac{28-10}{40} \rfloor + 1) \cdot C_3 = C_3 = 2C_1$

$C_P(0, L) = C_P(0, 28) = 3C1 + 4 + 2C_1 = 5C_1 + 4.$

To satisfy the deadline, we must have $C_P(0, L) = 5C_1 + 4 \leq L = 28$. Therefore, $C_1 \leq 24/5 = 4.8$. Since for the control point $L = 10$ we must have $C_1 \leq 2$, the schedulability for all $L = 6, 8, 10, 18, 26, 28$is satisfied if $C_1 \leq 2$.

Consider $L = 38$.

$N_1^L \cdot C_1 = (\lfloor \frac{38-8}{10} \rfloor + 1) \cdot C_1 = 4C_1 \qquad N_2^L \cdot C_2 = (\lfloor \frac{38-6}{20} \rfloor + 1) \cdot C_2 = 4$

$N_3^L \cdot C_3 = (\lfloor \frac{38-10}{40} \rfloor + 1) \cdot C_3 = C_3 = 2C_1$

$C_P(0, L) = C_P(0, 38) = 4C1 + 4 + 2C_1 = 6C_1 + 4.$

To satisfy the deadline, we must have $C_P(0, L) = 6C_1 + 4 \leq L = 38$. Therefore, $C_1 \leq 34/6 = 5.66$. Since for the control point $L = 10$ we must have $C_1 \leq 2$, the schedulability for all $L = 6, 8, 10, 18, 26, 28, 38$ is satisfied if $C_1 \leq 2$.

The maximum value of C_1 is 2.

PROBLEM 7

a) The underlying causes for the weak theoretical framework of global scheduling are:

- Dhall's effect
  - With RM, DM and EDF, some low-utilization task sets can be non-schedulable regardless of how many processors are used. Thus, any utilization guarantee bound would become so low that it would be useless in practice.
  - This is in contrast to the single-processor case where we have utilization guarantee bounds of 69.3% (RM) and 100% (EDF).
- Hard-to-find critical instant
  - A critical instant does not always occur when a task arrives at the same time as all its higher-priority tasks.
  - Note that this is in contrast to the uniprocessor case.

b) One way to find the smallest number of processors required to schedule the task set is to use schedulability analysis based on processor utilisation. For static-priority scheduling on multiprocessor systems there are two approaches that offer such analysis, namely RM-US (for global scheduling) and RMFF (for partitioned scheduling).

The task set has a total utilization $U_{\text{total}} = 4/20 + 5/5 + 12/40 + 3/10 + 20/100 = 2.0$.

We will find the smallest number of processors required for RM-US global scheduling based on its utilization bound $m^2/(3m-2)$. Since $U_{\text{total}} = 2.0$, we have

$$\begin{aligned} m^2/(3m-2) &\geq 2.0 \\ \text{or,}\ \ m^2 - 6m + 4 &\geq 0 \end{aligned}$$

By solving the quadratic equation $m^2 - 6m + 4 = 0$, we have $m = \frac{6 \pm \sqrt{36-16}}{2} = 3 \pm \sqrt{5}$. Since the number of processors $m \geq 1$, we have $3 + \sqrt{5} = 5.23$. Since the number of processors m is an integer, we need at least 6 processors for the RM-US algorithm.

Next, we will find the smallest number of processors required for RMFF partitioned scheduling based on its utilization bound $m(\sqrt{(}2) - 1)$. Since $U_{\text{total}} = 2.0$, we have

$$\begin{aligned} m(\sqrt{(}2) - 1) &\geq 2.0 \\ \text{or,}\ \ m &\geq 2.0/(\sqrt{(}2) - 1) = 4.82 \end{aligned}$$

Since the number of processors m is an integer, we need at least 5 processors for the RM-FF algorithm.

Therefore, RM-FF partitioned scheduling requires a smaller number of processors.