Estimating Latency

Goal:
given information about events and their handlers . . .
. . . estimate response times . . .
. . . or devise scheduling to insure that deadlines are met.

Definitions:

Latency [of an interrupt handler or dæmon task to an event]:
Time from event occurrence to start of its handler or dæmon task.

Response time [of interrupt handler or dæmon task to an event]:
Time from event occurrence to response. In class response assumed to be generated at completion of event's handler.

Let an event occur at t_1, its handler start at t_2 and finish at t_3 . . .
. . . then the latency is $t_2 - t_1$ and response time $t_3 - t_1$.

Run time [of interrupt handler or dæmon task]:
The time needed to run[1] *on an unloaded system.*

Actual run time [of interrupt handler or dæmon task]:
The time needed to run in a particular situation (considering other CPU activity, etc.).

[1] For handlers (which run something like operating system subroutines), run time is total run time. For dæmons (which are tasks managed like any other tasks the OS is running) it's the amount of run time needed to generate a response. After the response the dæmon will wait for another event, in contrast to a handler which actually finishes execution. At a subsequent event a waiting dæmon moves from the wait state (if the event is handled by a dæmon), while a "fresh" call to a handler is made (if the event is handled by a handler).

Worst-case latency [of an int. handler or dæmon task to an event]:
The longest possible latency given constraints on when events can occur, etc.

Worst-case response time [of an int. handler or dæmon task to an event]:
The longest possible response time given constraints on when events can occur, etc.

Worst-case run time [of an int. handler or dæmon task to an event]:
The longest possible actual run time given constraints on when events can occur, etc.

For a particular event, response time is always latency plus actual run time.

However, the worst-case response time is not necessarily the worst-case latency plus the worst-case actual run-time.

Unless otherwise stated, all latencies, actual run times, and response times are worst case.

Classes of problems:

- *One-shot.*
  Assume that each event can occur at most once.
- *Periodic exhaustive.*
  Events periodic, solution found exhaustively.
- *Periodic statistical.*
  Events periodic, solution found using statistical methods.

Events, Event Types, and Interrupts

An *event* is some occurrence in the process.

For example, temperature exceeding a threshold.

(This is the definition used throughout the semester.)

An *event type* is a kind of event.

As a result of an *event*, which is of a certain *event type*, an *interrupt* is requested; as a result a *handler* is run.

For example, consider a system where CDT marks trigger interrupts:

Event type: a clock mark passing under the mark reader of the coded displacement transducer.

Event: at $t = 30$ the clock mark passes under the mark reader of the coded displacement transducer.

Interrupt: the interruption caused by this event.

Handler: the CDT code that is run for this interrupt.

Each of these four terms has a different meaning, for brevity they sometimes will be used interchangeably.

Archetypical Problems

Timing Estimation Problem:

Given:

- Events. When the events can occur.
- Handlers and tasks for events. How long these will run.

  Handlers and tasks generate a *response* by the end of their run.
- Scheduling algorithm, interrupt system, and related details.

Find:

- Worst-case response time for each event.

Scheduling Problem:

Given:

- Events. When the events can occur.
- Deadlines for responses to these events.
- Handlers and tasks for events. How long these will run.
- Some details of interrupt hardware and scheduling.

Find:

- Scheduling priorities, algorithm, or other details so that deadlines are met.

Trivial Example Problem

A RTS must react to a single event. Exactly 5 μs *after the event occurs the NMI-request line will be asserted. (The delay is due to a slow sensor response.) The handler for this event requires* 100 μs *to generate a response to the event. The following are the timings of other system functions: context save,* 100 ns*; task switch,* 50 μs*; scheduler run time,* 100 μs*; worst-case instruction completion,* 50 ns*; masking and jump to address specified in IVT,* 20 ns*. Find the worst-case response time.*

Solution:

The following occurs from event to response:

Total Time/ μs	Time/ μs	Activity
0.00	0.00	The event.
5.00	5.00	Time for sensor to generate request.
5.05	.05	Instruction completion.
5.07	.02	Mask and jump.
5.17	.10	Save context.
105.17	100.00	Run handler.

Note: In most problems the following times are ignored:

Context save, instruction completion, mask and jump.

One-Shot Events

An event type is said to be *one-shot* if it can occur no more than once and if it can occur at any time.

A system is said to have *one-shot events* if all event types are one-shot.

In most systems an event can occur more than once, however...

... the solution technique to be presented can be applied to problems in which the time for an event to re-occur is sufficiently large.

General solutions will be provided for these cases:

- Pure strong-priority interrupt selection.

  This is similar to priority scheduling in a task-preemptive OS with an infinite quantum.

- Pure weak-priority interrupt selection.

  This is similar to priority scheduling in a non-task-preemptive OS with an infinite quantum.

- Both strong- and weak-priority interrupt selection.

Problems will include these cases, and variations on these cases. (So that the general solutions will have to be modified.)

Pure Strong Priority, One-Shot Events

This covers two possible sets of conditions:

Interrupts:

- Responses are generated by interrupt handlers.
- There is no more than one event and handler for each interrupt level.
- Interrupts can only be masked by other interrupts.

There are no tasks or other unrelated activities which can mask interrupts.

- Only the run time of the interrupt handlers is significant.

Tasks:

- Responses are generated by tasks, with exactly one task per event.
- Priority scheduling is used with at most one task per priority level.
- The OS is task-preemptive.
- Only tasks specified in the problem need be considered.

(No other tasks will preempt or be selected before the tasks needed for responses.)

- The quantum is infinite.

Interrupt Latency of Pure Strong Priority, One-Shot Events

Let

$\mathcal{E}$ be the set of event types

$(e.g., \mathcal{E} = \{\text{button3}, \text{tempAlarm}, \text{eventX}, 3, A\}),$

all events in $\mathcal{E}$ be one shot,

$t_h(e)$ be the run time of handler for $e \in \mathcal{E}$,

$p(e)$ be the strong priority level of the handler for $e \in \mathcal{E}$,

no two events have the same strong priority,

then worst case latency for event $e \in \mathcal{E}$ is

$$\sum_{\substack{i \in \mathcal{E} \\ p(i) > p(e)}} t_h(i).$$

In words: sum of run times of all higher priority interrupts.

Worst-case latency is encountered by e if. . .

. . .at the same time as e. . .

. . .one or more higher priority events occur. . .

. . .and all other higher priority events occur before e gets to run.

Pure Strong Priority, One-Shot Example Problem

Find the response time for all events in a system using pure-strong-priority interrupts and one-shot events. The events and interrupt handlers are described below.

Event	Priority	Run Time
A	3	$10\,\mu$s
B	7	$15\,\mu$s
C	1	$8\,\mu$s

Solution:

Worst-case latency for A: $t_h(B) = 15\,\mu$s.

Response time of A is $t_h(B) + t_h(A) = 25\,\mu$s.

Worst-case latency for B is 0.

Response time of B is $t_h(B) = 15\,\mu$s.

Worst-case latency for C: $t_h(A) + t_h(B) = 25\,\mu$s.

Response time of C is $t_h(A) + t_h(B) + t_h(C) = 33\,\mu$s.

Pure Weak Priority, One-Shot Events

This covers two possible sets of conditions:

Interrupts:

- Responses are generated by interrupt handlers.

◇ *All interrupts share a single IRQ input.*

◇ *Interrupt to run determined by polling sequence.*

- Interrupts only masked by other interrupts.
  (*I.e.*, no tasks or other unrelated activities mask interrupts.)

- Only handler run time is significant.
  E.g., context-switch and service-routine time can be ignored.

Tasks:

- Responses generated by tasks.

- Exactly one task per event.

- Priority scheduling used, at most one task per priority level.

◇ *The OS is <u>not</u> task-preemptive.*

- Only tasks specified in problem considered.
  (No other tasks will preempt or be selected before the tasks needed for responses.)

- The quantum is infinite.

Items that differ from the pure-strong-priority case are printed using slanted type and diamonds for bullets.

Interrupt Latency of Pure Weak Priority, One-Shot Events

Let

$\mathcal{E}$ be a set of one-shot event types

(*e.g.*, $\mathcal{E} = \{\text{button3}, \text{tempAlarm}, \text{eventX}, 3, A\}$),

$t_h(e)$ be the run time of the handler for event $e \in \mathcal{E}$,

$p(e)$ be weak priority level of handler for $e \in \mathcal{E}$.

and no two events have the same weak priority,

then worst case latency for event $e \in \mathcal{E}$ is

$$\left(\sum_{\substack{i\in\mathcal{E}\\ p(i)>p(e)}} t_h(i) \right) + \max_{\substack{i\in\mathcal{E}\\ p(i)<p(e)}} t_h(i).$$

In words:

Sum of run times of all higher-priority interrupts…

…plus longest run time of lower-priority interrupts.

Worst-case latency is encountered by e if…

…just before e…

…the lower-priority event with the longest handler occurs…

…and before this event's handler finishes…

…all events of higher priority than e occur.

Pure Weak Priority, One-Shot Example Problem

Find the worst-case response time for all events in a system using pure-weak-priority interrupts and one-shot events. The events and interrupt handlers are described below.

Event	Priority	Run Time
A	3	$10\,\mu$s
B	7	$15\,\mu$s
C	1	$8\,\mu$s

Solution:

Latency for A: $t_h(B) + t_h(C) = 23\,\mu$s.

Response time of A is $t_h(B) + t_h(C) + t_h(A) = 33\,\mu$s.

Latency for B is $t_h(A) = 10\,\mu$s.

Response time of B is $t_h(A) + t_h(B) = 25\,\mu$s.

Latency for C: $t_h(A) + t_h(B) = 25\,\mu$s.

Response time of C is $t_h(A) + t_h(B) + t_h(C) = 33\,\mu$s.

Notes:

A and C cannot simultaneously have worst-case response times.

Highest weak priority level <u>does not</u> guarantee lowest response time. (This will be seen in an example below.)

Strong and Weak Priority, One-Shot Events

This covers two possible sets of conditions:

Interrupts:

• Responses are generated by interrupt handlers.

♢ *An IRQ input can be used by any number of events.*

♢ The interrupt to service determined by polling sequence…
…*using a different sequence for each strong priority level.*

• Interrupts can only be masked by other interrupts.
There are no tasks or other unrelated activities which can mask interrupts.

• Only handler run time is significant.

Tasks:

• Responses are generated by tasks, exactly one task per event.

♢ *Scheduling done in two rounds, both use priority scheduling.*

♢ *First round is task-preemptive, second round is non-task-preemptive.*

• Only tasks specified in the problem need be considered.
(No other tasks will preempt or be selected before the tasks needed for responses.)

• The quantum is infinite.

Diamonds (♢) indicate differences with pure-weak-priority.

Interrupt Latency of Strong and Weak Priority, One-Shot Events

Let

$\mathcal{E}$ be a set of event types

($e.g.$, $\mathcal{E} = \{\text{button3}, \text{tempAlarm}, \text{eventX}, 3, A\}$),

all events in $\mathcal{E}$ be one shot,

$t_h(e)$ be the run time of the handler for event $e \in \mathcal{E}$,

$p_1(e)$ be the strong priority level of the handler for event $e \in \mathcal{E}$,

$p_2(e)$ be weak priority level of handler for $e \in \mathcal{E}$.

Then worst-case latency for event $e \in \mathcal{E}$ is

$$\sum_{\substack{i\in\mathcal{E}\\ p_1(i)>p_1(e)}} t_h(i) + \sum_{\substack{i\in\mathcal{E}\\ p_1(i)=p_1(e)\\ p_2(i)>p_2(e)}} t_h(i) + \max_{\substack{i\in\mathcal{E}\\ p_1(i)=p_1(e)\\ p_2(i)<p_2(e)}} t_h(i).$$

In words:

Sum of the run times of…

…all higher strong-priority and weak-priority handlers…

…plus longest run time of…

…the lower weak-priority handlers…

…that are at the same strong priority level as e.

This worst-case latency is encountered by e if:

The longest-run-time handler at the same strong priority level. . .

. . .starts to run just before e. . .

. . .and all higher priority interrupts occur before e's handler starts.

Strong and Weak Priority, One-Shot Example Problem

Find the worst-case response time for all events in a system using strong and weak priority interrupts and one-shot events. The events and interrupt handlers are described below.

Event	Strong Priority	Weak Priority	Run Time
A	3	1	$10\,\mu$s
B	2	3	$15\,\mu$s
C	2	2	$8\,\mu$s
D	2	1	$50\,\mu$s
E	1	2	$1\,\mu$s
F	1	1	$2\,\mu$s

Solution:

Event A

Event order: A occurs any time.

Latency for A: $0\,\mu$s.

Response time of A is $t_h(A) = 10\,\mu$s.

Event B

Possible event order: D, **B**, A.

Latency for B is $t_l(B) = t_h(A) + t_h(D) = 60\,\mu$s.

Response time of B is $t_l(B) + t_h(B) = 75\,\mu$s.

Event C

Possible event order: D, **C**, B, A.

Latency for C: $t_l(C) = t_h(A) + t_h(B) + t_h(D) = 75\,\mu$s.

Response time of C is $t_l(C) + t_h(C) = 83\,\mu$s.

Example, continued.

Event	Strong Priority	Weak Priority	Run Time Time
A	3	1	$10\,\mu$s
B	2	3	$15\,\mu$s
C	2	2	$8\,\mu$s
D	2	1	$50\,\mu$s.
E	1	2	$1\,\mu$s.
F	1	1	$2\,\mu$s.

Event D

Possible event order: B, **D**, C, A.

Latency for D: $t_l(D) = t_h(A) + t_h(B) + t_h(C) = 33\,\mu$s.

Response time of D is $t_l(D) + t_h(D) = 83\,\mu$s.

Event E

Possible event order: F, **E**, A, B, C, D.

Latency for E: $t_l(E) = \displaystyle\sum_{e\in\{A,B,C,D,F\}} t_h(e) = 85\,\mu$s.

Response time of E is $t_l(E) + t_h(E) = 86\,\mu$s.

Event F

Possible event order: E, **F**, A, B, C, D.

Latency for F: $t_l(F) = \displaystyle\sum_{e\in\{A,B,C,D,E\}} t_h(e) = 84\,\mu$s.

Response time of F is $t_l(F) + t_h(F) = 86\,\mu$s.

Example Problem Solution Details

Simulator Output Format

Output divided into sections divided by lines like:

```
================= B Worst-Case Latency and Run Time ====================
```

Each section demonstrates worst-case behavior for a different event.

Handler timing given in lines like:

```
Handler for F (1) finished: lat. 99., dur. 2., resp. 101.
```

which indicates that occurrence 1 of event F had a latency of 99, actual run time (duration) of 2, and a response time of 101.

This timing information is shown for *all* events, only a few of which suffer worst-case delays. (The event suffering worst-case delays is indicated in the section heading, but watch for multiple occurrence of the same event.)

Each section heading is printed at $t = 4 \pmod{1000}$, the first event in each section occurs at time $t = 5 - \epsilon \pmod{1000}$ and the event suffering worst case latency and response time occurs at time $t = 5 \pmod{1000}$, where ϵ is a very small time interval.

Simulator Output

```
** Time:        4
================= B Worst-Case Latency and Run Time ====================
** Time:        5.
Interrupt D (0) requested.
Handler for D (0) starting.
** Time:        5
Interrupt B (0) requested.
** Time:        6
Interrupt A (0) requested.
Handler for D (0) preempted.
Handler for A (0) starting.
** Time:       16
Handler for A (0) finished: lat. 0, dur. 10, resp. 10
Handler for D (0) resumed.
** Time:       65.
Handler for D (0) finished: lat. 0., dur. 60., resp. 60.
Handler for B (0) starting.
** Time:       80.
Handler for B (0) finished: lat. 60., dur. 15., resp. 75.
** Time:    1,004
================= C Worst-Case Latency and Run Time ====================
** Time:    1,005.
Interrupt D (1) requested.
Handler for D (1) starting.
** Time:    1,005
Interrupt C (0) requested.
```

```
** Time:     1,006
Interrupt A (1) requested.
Handler for D (1) preempted.
Handler for A (1) starting.
** Time:     1,007
Interrupt B (1) requested.
** Time:     1,016
Handler for A (1) finished: lat. 0, dur. 10, resp. 10
Handler for D (1) resumed.
** Time:     1,065.
Handler for D (1) finished: lat. 0., dur. 60., resp. 60.
Handler for B (1) starting.
** Time:     1,080.
Handler for B (1) finished: lat. 58., dur. 15., resp. 73.
Handler for C (0) starting.
** Time:     1,088.
Handler for C (0) finished: lat. 75., dur. 8., resp. 83.
** Time:     2,004
================= D Worst-Case Latency and Run Time ====================
** Time:     2,005.
Interrupt B (2) requested.
Handler for B (2) starting.
** Time:     2,005
Interrupt D (2) requested.
** Time:     2,006
Interrupt A (2) requested.
Handler for B (2) preempted.
Handler for A (2) starting.
** Time:     2,007
Interrupt C (1) requested.
** Time:     2,016
Handler for A (2) finished: lat. 0, dur. 10, resp. 10
Handler for B (2) resumed.
** Time:     2,030.
Handler for B (2) finished: lat. 0., dur. 25., resp. 25.
Handler for C (1) starting.
** Time:     2,038.
Handler for C (1) finished: lat. 23., dur. 8., resp. 31.
Handler for D (2) starting.
** Time:     2,088.
Handler for D (2) finished: lat. 33., dur. 50., resp. 83.
** Time:     3,004
================= E Worst-Case Latency and Run Time ====================
** Time:     3,005.
Interrupt F (0) requested.
Handler for F (0) starting.
** Time:     3,005
Interrupt E (0) requested.
** Time:     3,006
Interrupt A (3) requested.
Handler for F (0) preempted.
Handler for A (3) starting.
** Time:     3,007
```

```
Interrupt B (3) requested.
** Time:     3,008
Interrupt C (2) requested.
** Time:     3,009
Interrupt D (3) requested.
** Time:     3,016
Handler for A (3) finished: lat. 0, dur. 10, resp. 10
Handler for F (0) resumed.
Handler for F (0) preempted.
Handler for B (3) starting.
** Time:     3,031
Handler for B (3) finished: lat. 9, dur. 15, resp. 24
Handler for F (0) resumed.
Handler for F (0) preempted.
Handler for C (2) starting.
** Time:     3,039
Handler for C (2) finished: lat. 23, dur. 8, resp. 31
Handler for F (0) resumed.
Handler for F (0) preempted.
Handler for D (3) starting.
** Time:     3,089
Handler for D (3) finished: lat. 30, dur. 50, resp. 80
Handler for F (0) resumed.
** Time:     3,090.
Handler for F (0) finished: lat. 0., dur. 85., resp. 85.
Handler for E (0) starting.
** Time:     3,091.
Handler for E (0) finished: lat. 85., dur. 1., resp. 86.
** Time:     4,004
================== F Worst-Case Latency and Run Time =====================
** Time:     4,005.
Interrupt E (1) requested.
Handler for E (1) starting.
** Time:     4,005
Interrupt F (1) requested.
** Time:     4,005
Interrupt A (4) requested.
Handler for E (1) preempted.
Handler for A (4) starting.
** Time:     4,007
Interrupt B (4) requested.
** Time:     4,008
Interrupt C (3) requested.
** Time:     4,009
Interrupt D (4) requested.
** Time:     4,015
Handler for A (4) finished: lat. 0, dur. 10, resp. 10
Handler for E (1) resumed.
Handler for E (1) preempted.
Handler for B (4) starting.
** Time:     4,030
Handler for B (4) finished: lat. 8, dur. 15, resp. 23
Handler for E (1) resumed.
Handler for E (1) preempted.
Handler for C (3) starting.
** Time:     4,038
Handler for C (3) finished: lat. 22, dur. 8, resp. 30
Handler for E (1) resumed.
Handler for E (1) preempted.
Handler for D (4) starting.
** Time:     4,088
Handler for D (4) finished: lat. 29, dur. 50, resp. 79
Handler for E (1) resumed.
** Time:     4,089.
Handler for E (1) finished: lat. 0., dur. 84., resp. 84.
Handler for F (1) starting.
** Time:     4,091.
Handler for F (1) finished: lat. 84., dur. 2., resp. 86.
Simulation completed.
```

Perturbations

One-shot assumption too limiting.

Useful problems based on perturbation of one-shot assumption.

These problems solved by adapting one-shot solution procedure.

Adapting Solution

Instead of using latency formulas (from previous slides)…

…find an ordering of events…

…meeting problem restrictions…

…while resulting in worst-case behavior.

Two Perturbations from One-Shot

- Timing restrictions on events.
- Multiple occurrences of an event type.

Timing Restrictions

Usually minimum or maximum time between events.

E.g., event A occurs $\geq 12\,\mu$s after B.

Multiple Event Occurrences

Event may happen several times.

Bound on number and timing of occurrences will be given.

E.g., event A occurs ≤ 3 times $> 5\,\mu$s apart.

Multiple Occurrences Notation

Subscript used to indicate occurrence.

E.g., A_1, first occurrence of event type A; A_2 second.

Handling Multiple Occurrences

In class, the handler must be run for each occurrence.

E.g., if A occurs twice (A_1 and A_2),…

…handler for A must be run twice…

…even if A_2 occurs before handler for A_1 runs.

Perturbations Example Problem

Find the worst-case response time for all events in a system using strong and weak priority interrupts and one-shot events. The events and interrupt handlers are described below. The handler must be run once for each event occurrence, even if an event occurs a second time before the first the handler was run for the first occurrence.

Event	Strong Priority	Weak Priority	Run Time	Event Occurrences
A	3	1	$10\,\mu$s	Occurs once, any time.
B	2	3	$15\,\mu$s	Occurs twice, any times.
C	2	2	$8\,\mu$s	Occurs once, 45 to $50\,\mu$s after event A.
D	2	1	$50\,\mu$s	Occurs once, any time.
E	1	2	$1\,\mu$s	Occurs three times, with $> 100\,\mu$s separation.
F	1	1	$2\,\mu$s	Occurs once.

Solution:

Event A

Event order: A can occur any time.

Latency for A: $0\,\mu$s.

Response time of A is $t_h(A) = 10\,\mu$s.

Event B

Since event-type B can occur twice...
...an event B might have to wait...
...for a previous occurrence of B to be handled.

Possible event order: D, B_1, $\mathbf{B_2}$, A.

Latency for B_2 is $t_l(B) = t_h(A) + t_h(B) + t_h(D) = 75\,\mu$s.

Response time of B_2 is $t_l(B) + t_h(B) = 90\,\mu$s.

Example, continued.

Event	Strong Priority	Weak Priority	Run Time	Event Occurrences
A	3	1	$10\,\mu$s	Occurs once, any time.
B	2	3	$15\,\mu$s	Occurs twice, any times.
C	2	2	$8\,\mu$s	Occurs once, 45 to $50\,\mu$s after event A.
D	2	1	$50\,\mu$s	Occurs once, any time.
E	1	2	$1\,\mu$s	Occurs three times, with $> 100\,\mu$s separation.
F	1	1	$2\,\mu$s	Occurs once.

Event C

C only occurs after event A so:

Possible event order: D, **C**, B_1, B_2.

Latency for C: $t_l(C) = 2t_h(B) + t_h(D) = 80\,\mu$s.

Response time of C is $t_l(C) + t_h(C) = 88\,\mu$s.

Event D.

C and A cannot both occur within D's latency period.

Possible event order: B_1, **D**, A, B_2.

Latency for D: $t_l(D) = t_h(A) + 2t_h(B) = 40\,\mu$s.

Response time of D is $t_l(D) + t_h(D) = 90\,\mu$s.

Example, continued.

Event	Strong Priority	Weak Priority	Run Time	Event Occurrences
A	3	1	$10\,\mu$s	Occurs once, any time.
B	2	3	$15\,\mu$s	Occurs twice, any times.
C	2	2	$8\,\mu$s	Occurs once, 45 to $50\,\mu$s after event A.
D	2	1	$50\,\mu$s	Occurs once, any time.
E	1	2	$1\,\mu$s	Occurs three times, with $> 100\,\mu$s separation.
F	1	1	$2\,\mu$s	Occurs once.

Event E

Both A and C can occur during E's latency.

Possible event order: F, **E**, A, B_1, B_2, D, C

Latency for E: $t_l(E) = \displaystyle\sum_{e\in\{A,B_1,B_2,C,D,F\}} t_h(e) = 100\,\mu\text{s}.$

Response time of E is $t_l(E) + t_h(E) = 101\,\mu$s.

Event F

E's second occurrence is after F starts.

Possible event order: E_1, **F**, A, B_1, B_2, D, C

Latency for F: $t_l(F) = \displaystyle\sum_{e\in\{A,B_1,B_2,C,D,E\}} t_h(e) = 99\,\mu\text{s}.$

Response time of F is $t_l(F) + t_h(F) = 101\,\mu$s.

Example Problem Solution Details

Simulator Output

```
** Time:          4
================== B Worst-Case Latency and Run Time ======================
** Time:          5.
Interrupt D (0) requested.
Handler for D (0) starting.
** Time:          5
Interrupt B (0) requested.
** Time:          5.
Interrupt B (1) requested.
** Time:          6
Interrupt A (0) requested.
Handler for D (0) preempted.
Handler for A (0) starting.
** Time:         16
Handler for A (0) finished: lat. 0, dur. 10, resp. 10
Handler for D (0) resumed.
** Time:         65.
Handler for D (0) finished: lat. 0., dur. 60., resp. 60.
Handler for B (0) starting.
** Time:         80.
Handler for B (0) finished: lat. 60., dur. 15., resp. 75.
Handler for B (1) starting.
** Time:         95.
Handler for B (1) finished: lat. 75., dur. 15., resp. 90.
** Time:      1,004
================== C Worst-Case Latency and Run Time ======================
** Time:      1,005.
Interrupt D (1) requested.
Handler for D (1) starting.
** Time:      1,005
Interrupt C (0) requested.
** Time:      1,005.
Interrupt B (2) requested.
** Time:      1,005.
Interrupt B (3) requested.
** Time:      1,055.
Handler for D (1) finished: lat. 0., dur. 50., resp. 50.
Handler for B (2) starting.
** Time:      1,070.
Handler for B (2) finished: lat. 50., dur. 15., resp. 65.
Handler for B (3) starting.
** Time:      1,085.
Handler for B (3) finished: lat. 65., dur. 15., resp. 80.
Handler for C (0) starting.
** Time:      1,093.
Handler for C (0) finished: lat. 80., dur. 8., resp. 88.
** Time:      2,004
================== D Worst-Case Latency and Run Time ======================
** Time:      2,005.
Interrupt B (4) requested.
Handler for B (4) starting.
** Time:      2,005
```

```
Interrupt D (2) requested.
** Time:     2,006
Interrupt A (1) requested.
Handler for B (4) preempted.
Handler for A (1) starting.
** Time:     2,007
Interrupt B (5) requested.
** Time:     2,016
Handler for A (1) finished: lat. 0, dur. 10, resp. 10
Handler for B (4) resumed.
** Time:     2,030.
Handler for B (4) finished: lat. 0., dur. 25., resp. 25.
Handler for B (5) starting.
** Time:     2,045.
Handler for B (5) finished: lat. 23., dur. 15., resp. 38.
Handler for D (2) starting.
** Time:     2,095.
Handler for D (2) finished: lat. 40., dur. 50., resp. 90.
** Time:     3,004
================= E Worst-Case Latency and Run Time =====================
** Time:     3,005.
Interrupt F (0) requested.
Handler for F (0) starting.
** Time:     3,005
Interrupt E (0) requested.
** Time:     3,006
Interrupt A (2) requested.
Handler for F (0) preempted.
Handler for A (2) starting.
** Time:     3,007
Interrupt B (6) requested.
** Time:     3,008
Interrupt B (7) requested.
** Time:     3,009
Interrupt D (3) requested.
** Time:     3,016
Handler for A (2) finished: lat. 0, dur. 10, resp. 10
Handler for F (0) resumed.
Handler for F (0) preempted.
Handler for B (6) starting.
** Time:     3,031
Handler for B (6) finished: lat. 9, dur. 15, resp. 24
Handler for F (0) resumed.
Handler for F (0) preempted.
Handler for B (7) starting.
** Time:     3,046
Handler for B (7) finished: lat. 23, dur. 15, resp. 38
Handler for F (0) resumed.
Handler for F (0) preempted.
Handler for D (3) starting.
** Time:     3,051
Interrupt C (1) requested.
** Time:     3,096
```

```
Handler for D (3) finished: lat. 37, dur. 50, resp. 87
Handler for F (0) resumed.
Handler for F (0) preempted.
Handler for C (1) starting.
** Time:     3,104
Handler for C (1) finished: lat. 45, dur. 8, resp. 53
Handler for F (0) resumed.
** Time:     3,105.
Handler for F (0) finished: lat. 0., dur. 100., resp. 100.
Handler for E (0) starting.
** Time:     3,106.
Handler for E (0) finished: lat. 100., dur. 1., resp. 101.
** Time:     4,004
================= F Worst-Case Latency and Run Time =====================
** Time:     4,005.
Interrupt E (1) requested.
Handler for E (1) starting.
** Time:     4,005
Interrupt F (1) requested.
** Time:   4,005.5
Interrupt A (3) requested.
Handler for E (1) preempted.
Handler for A (3) starting.
** Time:     4,007
Interrupt B (8) requested.
** Time:     4,008
Interrupt B (9) requested.
** Time:     4,009
Interrupt D (4) requested.
** Time:   4,015.5
Handler for A (3) finished: lat. 0., dur. 10., resp. 10.
Handler for E (1) resumed.
Handler for E (1) preempted.
Handler for B (8) starting.
** Time:   4,030.5
Handler for B (8) finished: lat. 8.5, dur. 15., resp. 23.5
Handler for E (1) resumed.
Handler for E (1) preempted.
Handler for B (9) starting.
** Time:   4,045.5
Handler for B (9) finished: lat. 22.5, dur. 15., resp. 37.5
Handler for E (1) resumed.
Handler for E (1) preempted.
Handler for D (4) starting.
** Time:     4,051
Interrupt C (2) requested.
** Time:   4,095.5
Handler for D (4) finished: lat. 36.5, dur. 50., resp. 86.5
Handler for E (1) resumed.
Handler for E (1) preempted.
Handler for C (2) starting.
** Time:   4,103.5
Handler for C (2) finished: lat. 44.5, dur. 8., resp. 52.5
Handler for E (1) resumed.
** Time:     4,104.
Handler for E (1) finished: lat. 0., dur. 99., resp. 99.
Handler for F (1) starting.
** Time:     4,105
Interrupt E (2) requested.
** Time:     4,106.
Handler for F (1) finished: lat. 99., dur. 2., resp. 101.
Handler for E (2) starting.
** Time:     4,107.
Handler for E (2) finished: lat. 1., dur. 1., resp. 2.
Simulation completed.
```

Strong and Weak Priority One-Shot Scheduling Problem

Responses in a RTS are generated by interrupt handlers. Strong and weak priority can be used. The events, the run time of the event handlers, and the timing constraints on the responses are given in the table below. Assign priorities so that the deadlines are met and the minimum number of strong priority levels are used.

Event Name	Run Time	Response Time Constraint	Occurrence
A	$50\,\mu$s	$< 90\,\mu$s	Once, any time.
B	$20\,\mu$s	$< 30\,\mu$s	Once, any time.
C	$10\,\mu$s	$\leq 80\,\mu$s	Once, any time.

Solution:

Event Name	Strong Priority	Weak Priority	Latency	Response Time
A	1	1	$30\,\mu$s	$80\,\mu$s
B	2	1	$0\,\mu$s	$20\,\mu$s
C	1	2	$70\,\mu$s	$80\,\mu$s

Periodic Events

An event type is said to be *periodic* if...

...the time between occurrences is fixed...

...the time of occurrences (phase) is arbitrary.

The *period* is the time between occurrences.

$t_b(X)$ will denote the period of event type X.
(The b is for *between*.)

Example

Let $t_b(A) = 10\,\mathrm{s}$, where A is an event type.

Then A might occur at $t/\mathrm{s} = 10$, 20, 30, ...

Or A might occur at $t/\mathrm{s} = 12$, 22, 32, ...

Relative Timing

Since time of occurrences arbitrary...

...in a system with several periodic event types,...

...the relative timing of the events is arbitrary.

Relative Timing Example

Suppose event-type A is periodic with $t_b(A) = 10\,\mathrm{ms}$...

...and B is periodic also with $t_b(B) = 10\,\mathrm{ms}$.

Then A and B could occur at:...

...the same time,...

...or A could follow B by any time $< 10\,\mathrm{ms}$.

But, because their periods are the same,...

...the time from A to B is fixed,...

...even though this time is not known in advance.

Latency Estimation with Periodic Events

Exhaustive Method

By trial-and-error, find the worst-case scenario.

Statistical Method

Find average effect of relatively short handlers.

Periodic-Interrupts Example Problem

A RTS has three event types, A, B, and C. All event types are periodic; their periods, the run time of their handlers, and their priority levels appear in the table below. Find the latency and response time for each event type.

Event Name	Strong Priority	Period $t_b/\,\mu$s	Handler Run Time $t_h/\,\mu$s
A	3	23	5
B	2	100	20
C	1	36	2

Solution:

Latency for A: $0\,\mu$s.

Response time of A is $t_r(A) = t_h(A) = 5\,\mu$s.

Latency for B: $t_l(B) = t_h(A) = 5\,\mu$s.

Response time of B is $t_r(B) = 2t_h(A) + t_h(B) = 30\,\mu$s.

(B occurs just after A; before B finishes A occurs a second time.)

Latency for C: $t_l(C) = 2t_h(A) + t_h(B) = 30\,\mu$s.

Response time of C is $t_r(C) = 2t_h(A) + t_h(B) + t_h(C) = 32\,\mu$s.

Statistical Method of Latency Estimation

Motivation: exhaustive method may be too time consuming.

Statistical Method Idea

Consider average—not exact—number of times. . .

. . .one handler interrupts another.

Illustration

Consider two periodic events, A and B.

Let interrupt A have higher strong priority than B.

Let $t_h(A) = 100\,\mu\text{s}$, $t_b(A) = 300\,\mu\text{s}$, $t_h(B) = 450\,\mu\text{s}$, and $t_b(B) = 10\,\text{s}$.

Exhaustive Method on Illustration

B can be interrupted by A 2 or 3 times.

WC run-time for B then $t_a(B) = t_h(B) + n\,t_h(A)$,

where n is number of times B interrupted:

Substituting yields $t_a(B) = 750\,\mu\text{s}$.

WC Run Using Average Number of Interruptions

Ignoring fact that n is an integer…

…let $n = t_a(B)/t_b(A)$.

This average contains unknown value, $t_a(B)$.

Solving for $t_a(B)$ yields $t_a(B) = \dfrac{t_h(B)}{1 - \frac{t_h(A)}{t_b(A)}}$.

Average Method on Illustration

Substituting yields $t_a(B) = 675\,\mu\text{s}$.

Difference, $750\,\mu\text{s} - 675\,\mu\text{s} = 75\,\mu\text{s}$, too big to ignore…

…but as n increases the difference drops.

Important Quantity in Average Method: *Loading Factor*

Consider actual run time equation:

$$t_a(B) = \frac{t_h(B)}{1 - \frac{t_h(A)}{t_b(A)}}.$$

Average $n = t_h(A)/t_b(B)$ not explicit.

But $1 - \frac{t_h(A)}{t_b(A)}$ is.

So $1 - \frac{t_h(A)}{t_b(A)}$ will be used instead of average n.

$1 - \frac{t_h(A)}{t_b(A)}$ is example of *loading factor.*

Intuitive Definition of Loading Factor

Fraction of CPU time that A leaves for B.

B's run time computed for slower CPU using loading factor.

First definitions, then details of the statistical method.

Definitions

Load (noun) [that a handler places on system]

The *load* of a periodic event's handler is

$l = t_h / t_b$,…

…where t_h is run time of handler…

…and t_b is event's period.

In words: fraction of CPU time needed by handler.

A system will be overloaded if the sum of all loads is greater than one.

Load (verb)

Event X is said to *load* event Y…

…if strong priority of X is higher than Y…

…and $\theta_l t_h(X) < t_h(Y)$,…

…where θ_l is a constant called the *loading threshold.*

Loading Threshold Values

Higher values give more precise answers.

Unless stated otherwise $\theta_l = 50$.

Load Set

Let Y denote an event.

The *load set* for Y is ...
... the set of events that load Y.

Loading Factor

Let Y denote an event ...
... and $\mathcal{X}$ be the load set for Y.

Then the *loading factor* for Y's handler is

$$l_f(Y) = 1 - \sum_{e \in \mathcal{X}} \frac{t_h(e)}{t_b(e)}.$$

In words: Y's loading factor is fraction of CPU time ...
... available to Y's handler ...
... after accounting for $\mathcal{X}$, the load set.

Loaded Duration

Let Y denote an event with loading factor $l_f(Y)$.

Then the *loaded duration* of event Y's handler is

$$t'_h(Y) = \frac{t_h(Y)}{l_f(Y)}.$$

Note: the actual duration, $t_a(Y)$, can be longer.

General Solution Technique

Start at the highest strong-priority level.

For all events in a strong-priority level:

- Find the loading factor, compute the loaded duration.
- Exhaustively compute the actual duration (consider only higher-priority events which do not load the event being considered).
- Exhaustively compute the latency.
- Exhaustively compute the response time.

Repeat for the next lower strong-priority level or finish if at the lowest strong priority.

Statistical-Latency-Estimation Example Problem

A RTS has three event types, all periodic. Their period, priority, and the run time of their handlers is listed in the table below. Find the latency and response time for each event.

Event Name	Strong Priority	Period t_b	Handler Run Time, t_h
A	3	$10\,\mu$s	$0.5\,\mu$s
B	2	50 ms	$37\,\mu$s
C	1	1 s	$40\,\mu$s

Solution:

- Event A

  Latency, $t_l(A) = 0\,\mu$s.

  Response time, $t_r(A) = t_l(A) + t_h(A) = 0.5\,\mu$s.

- Event B

  $\dfrac{t_h(B)}{t_h(A)} = 74 > \theta_l = 50$, therefore A loads B.

  Loading factor for B is $l_f(B) = 1 - \dfrac{t_h(A)}{t_b(A)} = 0.95$.

  Loaded duration: $t'_h(B) = \dfrac{t_h(B)}{l_f(B)} = 38.95\,\mu$s.

  Latency, $t_l(B) = t_h(A) = 0.5\,\mu$s.

  Response time, $t_r(B) = t_l(B) + t'_h(B) = 39.45\,\mu$s.

Example, continued.

Event Name	Strong Priority	Period t_b	Handler Run Time, t_h
A	3	$10\,\mu$s	$0.5\,\mu$s
B	2	50 ms	$37\,\mu$s
C	1	1 s	$40\,\mu$s

- Event C

  $$\frac{t_h(C)}{t_h(A)} = 80 > \theta_l; \ \frac{t_h(C)}{t_h(B)} = 1.08 < \theta_l,$$

  therefore A loads C but B does not.

  Loading factor, $l_f(C) = 1 - \dfrac{t_h(A)}{t_b(A)} = 0.95.$

  Event C's worst-case latency due to A *followed by* one loaded B:

  Latency, $t_l(C) = t_h(A) + t'_h(B) = 39.45\,\mu$s.

  Event B can occur either during C's latency or run time, but not both:

  Response time, $t_r(C) = t_h(A) + t'_h(B) + \dfrac{t_h(C)}{l_f(C)} = 81.55\,\mu$s.

Perturbations

Quasi-Periodic Events

Occur regularly, but not with fixed period.

Examples:

- Period ranging from 100 ms to 80 ms.
- No more than 5 times in 10 ms interval.
- At $t = 0$ and exactly 27 μs after previous occurrence handled.

Load of Handlers for Quasi-Periodic Events

Can sometimes find a worst-case t_b.

This would be used in load factors.

The method for determining the WC t_b depends upon details of quasi-periodic event.

Another Statistical Latency Estimation Example Problem

A RTS must react to six event types. The names of the event types, their occurrence times, the priorities of their respective interrupts, and the run time of their handlers is listed in the table below. For each event type find the latency, duration, and response time. Also find the total system load.

Event Name	Strong Pri.	Weak Pri.	Handler Run Time	Occurrence
A	4	2	$3\,\mu$s	Periodic, $t_b(A) = 20\,\mu$s.
B	4	1	$2\,\mu$s	From $7\,\mu$s to $13\,\mu$s after event A, if at all.
C	3	1	$700\,\mu$s	Periodic, $t_b(C) = 27\,$ms.
D	3	2	$11\,\mu$s	No more than 3 times in any 1 ms interval.
D				At most 2 unresponded events held.
E	2	1	1 ms	Periodic, $t_b(E) = 100\,$ms.
F	1	1	500 ms	Anytime after resp. to prev. occur.

Solution:

Events A and B

From table above, cannot occur at same time.

A possible event sequence: A.

B possible event sequence: B.

And so latency: $t_l(A) = t_l(B) = 0\,\mu$s.

Response time: $t_r(A) = t_a(A) = 3\,\mu$s.

Response time: $t_r(B) = t_a(B) = 2\,\mu$s.

Example, continued.

Event Name	Strong Pri.	Weak Pri.	Handler Run Time	Occurrence
A	4	2	$3\,\mu$s	Periodic, $t_b(A) = 20\,\mu$s.
B	4	1	$2\,\mu$s	From $7\,\mu$s to $13\,\mu$s after event A, if at all.
C	3	1	$700\,\mu$s	Periodic, $t_b(C) = 27\,$ms.
D	3	2	$11\,\mu$s	No more than 3 times in any 1 ms interval.
D				At most 2 unresponded events held.
E	2	1	1 ms	Periodic, $t_b(E) = 100\,$ms.
F	1	1	500 ms	Anytime after resp. to prev. occur.

Event C Run Time

Loading: A, B.

effective $t_b(B)$ same as $t_b(A)$.

$$l_f(C) = 1 - \frac{t_h(A)}{t_b(A)} - \frac{t_h(B)}{t_b(A)} = 0.75.$$

Actual run time $t_a(C) = \dfrac{t_h(C)}{l_f(C)} = 933\dfrac{1}{3}\,\mu$s.

Event D Run Time

Use exhaustive method, note variation in B's timing.

Possible event sequence: **D**, B_1, A, B_2.

$t_a(D) = t_h(D) + 2t_h(B) + t_h(A) = 18\,\mu$s.

Example, continued.

Event Name	Strong Pri.	Weak Pri.	Handler Run Time	Occurrence
A	4	2	$3\,\mu$s	Periodic, $t_b(A) = 20\,\mu$s.
B	4	1	$2\,\mu$s	From $7\,\mu$s to $13\,\mu$s after event A, if at all.
C	3	1	$700\,\mu$s	Periodic, $t_b(C) = 27\,$ms.
D	3	2	$11\,\mu$s	No more than 3 times in any 1 ms interval.
D				At most 2 unresponded events held.
E	2	1	1 ms	Periodic, $t_b(E) = 100\,$ms.
F	1	1	500 ms	Anytime after resp. to prev. occur.

Event C Latency

Event D can occur 3 times in an interval…

…in which A can occur 3 times and B twice.

Possible event sequence: D_0, A_0, **C**, B_0, D_1, A_1, D_2, B_1, A_2.

$t_l(C) = 3t_h(D) + 3t_h(A) + 2t_h(B) = 46\,\mu\text{s}$.

Event D Latency

Event D_2 must wait for C…

…and an earlier instance of D to finish.

Possible event sequence: $C*$, D_1, $\mathbf{D_2}$, B_1, A, B_2.

$C*$ indicates C and events that load C's handler.

$t_l(D) = t_a(C) + t_a(D) = 951.33\,\mu\text{s}$.

Example, continued.

Event Name	Strong Pri.	Weak Pri.	Handler Run Time	Occurrence
A	4	2	$3\,\mu$s	Periodic, $t_b(A) = 20\,\mu$s.
B	4	1	$2\,\mu$s	From $7\,\mu$s to $13\,\mu$s after event A, if at all.
C	3	1	$700\,\mu$s	Periodic, $t_b(C) = 27\,$ms.
D	3	2	$11\,\mu$s	No more than 3 times in any 1 ms interval.
D				At most 2 unresponded events held.
E	2	1	1 ms	Periodic, $t_b(E) = 100\,$ms.
F	1	1	500 ms	Anytime after resp. to prev. occur.

Event C Response Time

The event sequences used...

...for computing C's latency and duration...

...could happen <u>both</u> during the latency and run intervals...

...therefore response time is sum of two:

Response time $t_r(C) = t_l(C) + t_a(C) = 979.33\,\mu$s.

Event D Response Time

The response time of D includes two runs of D.

The event sequences used in computing $t_a(C)$...

...had D interrupted by two B's and an A.

This cannot happen for two consecutive runs of D.

Possible event sequence: $C*$, D_1, $\mathbf{D_2}$, B_1, A_1, B_2 A_2.

Response time $t_r(D) = t_a(C) + 2(t_h(D) + t_h(A) + t_h(B)) = 965.33\,\mu$s.

Example, continued.

Event Name	Strong Pri.	Weak Pri.	Handler Run Time	Occurrence
A	4	2	$3\,\mu$s	Periodic, $t_b(A) = 20\,\mu$s.
B	4	1	$2\,\mu$s	From $7\,\mu$s to $13\,\mu$s after event A, if at all.
C	3	1	$700\,\mu$s	Periodic, $t_b(C) = 27\,$ms.
D	3	2	$11\,\mu$s	No more than 3 times in any 1 ms interval.
D				At most 2 unresponded events held.
E	2	1	1 ms	Periodic, $t_b(E) = 100\,$ms.
F	1	1	500 ms	Anytime after resp. to prev. occur.

Event E

Loaded by A, B, and D:

$$l_f(E) = 1 - \frac{t_h(A)}{t_b(A)} - \frac{t_h(B)}{t_b(B)} - \frac{t_h(D)}{t_b(D)} = 0.717.$$

Worst-case latency includes

…time for C…

…plus time for D's waiting after C finishes…

…plus 3 A's and 2 B's:

$$t_l(E) = 3t_h(D) + 3t_h(A) + 2t_h(B) + t_a(C) = 979.33\,\mu\text{s}.$$

Actual duration is similar, except E occurs before C.

$$t_a(E) = \frac{t_h(E)}{l_f(E)} + t_a(C) + 3t_a(D) + 3t_a(A) + 2t_a(B) = 2.374\,\text{ms}.$$

Response time must not count C twice:

$$t_r(E) = t_l(E) + \frac{t_h(E)}{l_f(E)} = 2.374\,\text{ms}.$$

Example, continued.

Event Name	Strong Pri.	Weak Pri.	Handler Run Time	Occurrence
A	4	2	$3\,\mu$s	Periodic, $t_b(A) = 20\,\mu$s.
B	4	1	$2\,\mu$s	From $7\,\mu$s to $13\,\mu$s after event A, if at all.
C	3	1	$700\,\mu$s	Periodic, $t_b(C) = 27\,$ms.
D	3	2	$11\,\mu$s	No more than 3 times in any 1 ms interval.
D				At most 2 unresponded events held.
E	2	1	1 ms	Periodic, $t_b(E) = 100\,$ms.
F	1	1	500 ms	Anytime after resp. to prev. occur.

Event F

All other events load F:

$$l_f(F) = 1 - \sum_{e \in \{A,B,C,D,E\}} \frac{t_h(e)}{t_b(e)} = 0.681.$$

Duration

$$t_a(F) = \frac{t_h(F)}{l_f(F)} = 734.1\,\text{ms}.$$

Latency of F is the same as response time of E since there is only one interrupt at strong levels 1 and 2 and since neither interrupt will recur until after handler finishes.

$t_l(F) = t_r(E) = 2.374\,$ms.

$t_r(F) = t_l(F) + t_a(F) = 736.5\,$ms.

Because of event F, worst-case load is 1.

Example Problem Solution Details

Simulator Output Format

Output shows worst-case scenarios for some events.

As with simulator output appearing above, latency and response times shown are only worst case for a few events, see title at the head of each run.

Events in a handler's load set are not shown while a handler is running. For example, *C* is loaded by *A* and *B*, so while *C* is running occurrences of *A* and *B* are not shown.

Just before a handler starts a message is printed for each active event in its load set, indicating that those events will be ignored (and the run time of the handler is computed using the loading factor), for example, "Event A simulated using handler load." If the event is already being ignored no message is printed. Similarly, before a handler starts a message is printed for each event which will no longer be ignored. For example, "Normal simulation of A resuming." Sometimes both messages are printed (the software is not yet polished), use the second message.

```
– Loaded Event Handlers –
Event C: ld. factor, 0.75; ld. dur, 933.333 load set A, B
Event E: ld. factor, 0.75; ld. dur, 1333.33 load set A, B
Event F: ld. factor, 0.714074; ld. dur, 700207. load set A, B, C, E
** Starting simulation...
=============== C(0) Worst-Case Latency and Response. ===================
** Time:   1,000.
Interrupt D (0) requested.
Handler for D (0) starting, time remaining 11
** Time:   1,000.
Interrupt A (0) requested.
Handler for D (0) preempted.
Handler for A (0) starting, time remaining 3
** Time:   1,000
Interrupt C (0) requested.
** Time:   1,000.
Interrupt D (1) requested.
** Time:   1,003.
Handler for A (0) finished:
        latency 0. + duration 3. = response time 3.
Handler for D (0) resumed, time remaining 11..
** Time:   1,007
Interrupt B (0) requested.
Handler for D (0) preempted.
Handler for B (0) starting, time remaining 2
** Time:   1,009
Handler for B (0) finished:
        latency 0 + duration 2 = response time 2
Handler for D (0) resumed, time remaining 7..
** Time:   1,016.
Handler for D (0) finished:
        latency 0. + duration 16. = response time 16.
Handler for D (1) starting, time remaining 11
** Time:   1,020.
Interrupt A (1) requested.
```

```
Handler for D (1) preempted.
Handler for A (1) starting, time remaining 3
** Time:     1,023.
Handler for A (1) finished:
               latency 0. + duration 3. = response time 3.
Handler for D (1) resumed, time remaining 7..
** Time:     1,025
Interrupt D (2) requested.
** Time:     1,027
Interrupt B (1) requested.
Handler for D (1) preempted.
Handler for B (1) starting, time remaining 2
** Time:     1,029
Handler for B (1) finished:
               latency 0 + duration 2 = response time 2
Handler for D (1) resumed, time remaining 3..
** Time:     1,032.
Handler for D (1) finished:
               latency 16. + duration 16. = response time 32.
Handler for D (2) starting, time remaining 11
** Time:     1,040.
Interrupt A (2) requested.
Handler for D (2) preempted.
Handler for A (2) starting, time remaining 3
** Time:     1,043.
Handler for A (2) finished:
               latency 0. + duration 3. = response time 3.
Handler for D (2) resumed, time remaining 3..
** Time:     1,046.
Handler for D (2) finished:
               latency 7. + duration 14. = response time 21.
Event A simulated using handler load.
Event B simulated using handler load.
Handler for C (0) starting, time remaining 700
** Time:  1,979.33
Handler for C (0) finished:
               latency 46. + duration 933.333 = response time 979.333
```

```
– Loaded Event Handlers –
Event C: ld. factor, 0.75; ld. dur, 933.333 load set A, B
Event E: ld. factor, 0.75; ld. dur, 1333.33 load set A, B
Event F: ld. factor, 0.714074; ld. dur, 700207. load set A, B, C, E
Event A simulated using handler load.
Event B simulated using handler load.
** Starting simulation...
================= D(1) Worst-Case Response. =====================
** Time:    1,000.
Interrupt C (0) requested.
Handler for C (0) starting, time remaining 700
** Time:    1,000
Interrupt D (0) requested.
** Time:    1,000.
Interrupt D (1) requested.
** Time:  1,933.33
Handler for C (0) finished:
           latency 0. + duration 933.333 = response time 933.333
Normal simulation of A resuming.
Normal simulation of B resuming.
Handler for D (0) starting, time remaining 11
** Time:    1,934.
Interrupt B (0) requested.
Handler for D (0) preempted.
Handler for B (0) starting, time remaining 2
** Time:    1,936.
Handler for B (0) finished:
           latency 0. + duration 2. = response time 2.
Handler for D (0) resumed, time remaining 10.3333.
** Time:    1,941.
Interrupt A (0) requested.
Handler for D (0) preempted.
Handler for A (0) starting, time remaining 3
** Time:    1,944.
Handler for A (0) finished:
           latency 0. + duration 3. = response time 3.
Handler for D (0) resumed, time remaining 5.33333.
** Time:  1,949.33
Handler for D (0) finished:
           latency 933.333 + duration 16. = response time 949.333
Event A simulated using handler load.
Event B simulated using handler load.
Normal simulation of A resuming.
Normal simulation of B resuming.
Handler for D (1) starting, time remaining 11
** Time:    1,954.
Interrupt B (1) requested.
Handler for D (1) preempted.
Handler for B (1) starting, time remaining 2
** Time:    1,956.
Handler for B (1) finished:
           latency 0. + duration 2. = response time 2.
Handler for D (1) resumed, time remaining 6.33333.
```

```
** Time:      1,961.
Interrupt A (1) requested.
Handler for D (1) preempted.
Handler for A (1) starting, time remaining 3
** Time:      1,964.
Handler for A (1) finished:
             latency 0. + duration 3. = response time 3.
Handler for D (1) resumed, time remaining 1.33333.
** Time:   1,965.33
Handler for D (1) finished:
             latency 949.333 + duration 16. = response time 965.333
```

```
– Loaded Event Handlers –
Event C: ld. factor, 0.75; ld. dur, 933.333 load set A, B
Event E: ld. factor, 0.717; ld. dur, 1394.7 load set A, B, D
Event F: ld. factor, 0.681074; ld. dur, 734135. load set A, B, C, D, E
Event A simulated using handler load.
Event B simulated using handler load.
Event D simulated using handler load.
================= E(0) and F(0) Worst-Case Latency and Response.==========
** Time:    1,000.
Interrupt C (0) requested.
Normal simulation of D resuming.
Handler for C (0) starting, time remaining 700
** Time:    1,000
Interrupt F (0) requested.
** Time:    1,000
Interrupt E (0) requested.
** Time:  1,288.33
Interrupt D (0) requested.
** Time:  1,621.67
Interrupt D (1) requested.
** Time:  1,933.33
Handler for C (0) finished:
            latency 0. + duration 933.333 = response time 933.333
Event D simulated using handler load.
Normal simulation of A resuming.
Normal simulation of B resuming.
Normal simulation of D resuming.
Handler for D (0) starting, time remaining 11
** Time:    1,934.
Interrupt A (0) requested.
Handler for D (0) preempted.
Handler for A (0) starting, time remaining 3
** Time:    1,937.
Handler for A (0) finished:
            latency 0. + duration 3. = response time 3.
Handler for D (0) resumed, time remaining 10.3333.
** Time:    1,941.
Interrupt B (0) requested.
Handler for D (0) preempted.
Handler for B (0) starting, time remaining 2
** Time:    1,943.
Handler for B (0) finished:
            latency 0. + duration 2. = response time 2.
Handler for D (0) resumed, time remaining 6.33333.
** Time:  1,949.33
Handler for D (0) finished:
            latency 645. + duration 16. = response time 661.
Event A simulated using handler load.
Event B simulated using handler load.
Event D simulated using handler load.
Normal simulation of A resuming.
Normal simulation of B resuming.
Normal simulation of D resuming.
```

```
Handler for D (1) starting, time remaining 11
** Time:     1,954.
Interrupt A (1) requested.
Handler for D (1) preempted.
Handler for A (1) starting, time remaining 3
** Time:     1,955.
Interrupt D (2) requested.
** Time:     1,957.
Handler for A (1) finished:
            latency 0. + duration 3. = response time 3.
Handler for D (1) resumed, time remaining 6.33333.
** Time:     1,961.
Interrupt B (1) requested.
Handler for D (1) preempted.
Handler for B (1) starting, time remaining 2
** Time:     1,963.
Handler for B (1) finished:
            latency 0. + duration 2. = response time 2.
Handler for D (1) resumed, time remaining 2.33333.
** Time:  1,965.33
Handler for D (1) finished:
            latency 327.667 + duration 16. = response time 343.667
Event A simulated using handler load.
Event B simulated using handler load.
Event D simulated using handler load.
Normal simulation of A resuming.
Normal simulation of B resuming.
Normal simulation of D resuming.
Handler for D (2) starting, time remaining 11
** Time:     1,974.
Interrupt A (2) requested.
Handler for D (2) preempted.
Handler for A (2) starting, time remaining 3
** Time:     1,977.
Handler for A (2) finished:
            latency 0. + duration 3. = response time 3.
Handler for D (2) resumed, time remaining 2.33333.
** Time:  1,979.33
Handler for D (2) finished:
            latency 10.3333 + duration 14. = response time 24.3333
Event A simulated using handler load.
Event B simulated using handler load.
Event D simulated using handler load.
Handler for E (0) starting, time remaining 1000
** Time:  3,374.03
Handler for E (0) finished:
            latency 979.333 + duration 1394.7 = response time 2374.03
Event C simulated using handler load.
Event E simulated using handler load.
Handler for F (0) starting, time remaining 500000
** Time:  737,509.
Handler for F (0) finished:
            latency 2374.03 + duration 734135. = response time 736509.
```