Real-Time Scheduling of Mixed-Criticality Systems: What are the “X” Factors?

Risat Pathan
Chalmers University of Technology, Sweden

Workshop on Mixed Criticality Systems, Rome, Italy
December 2, 2014

Mixed-Criticality (MC) Systems

- Tasks have different criticalities

- Criticality specifies "importance" of a task
  - Higher criticality => higher importance
  - Correctly executing higher criticality tasks is more important

- What is correct execution?
  - The functional output is **right**, i.e., 1+2=3
  - The functional output is **timely**, i.e., deadline is met

The “X” Factors

- Correct execution of high-critical tasks can be **threatened** at runtime by events (“X” factors)

- Examples of “X” factors for MC systems:
  - WCET overrun (Steve Vestal, RTSS 2007)
    - ✓ Difficulty in estimating WCET
  - Occurrences of errors and the need for recovery
    - ✓ Hardware problems, environmental effects, software bugs

This research

- Scheduling constrained-deadline sporadic tasks with three constraints:
  - Meeting ***hard deadlines***
  - Error recovery using ***time-redundant execution*** (called, backups)
  - Respecting criticality to facilitate ***certification***

- An instance of a task is called a **job** that must
  - generate the correct output, and
  - meet its deadline

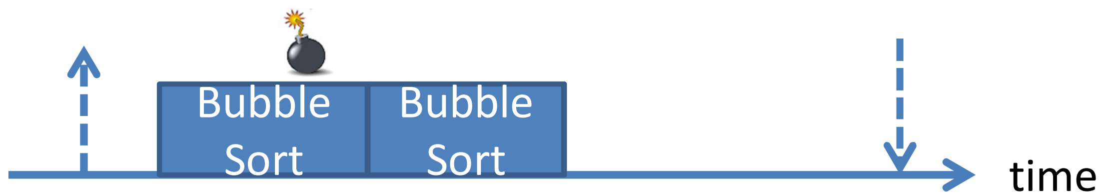
Error Model

- ***Task (Job) Errors***
  - wrong path, wrong output, etc.
  - Why do we have errors?
    - Errors are caused by **faults**
      - **Hardware Transient Faults**
        - » temporary malfunctioning of the computing unit
        - » ***happen for a short time and then disappear (not permanent)***
      - **Software Bugs**
        - » Bugs may remain undetected after testing

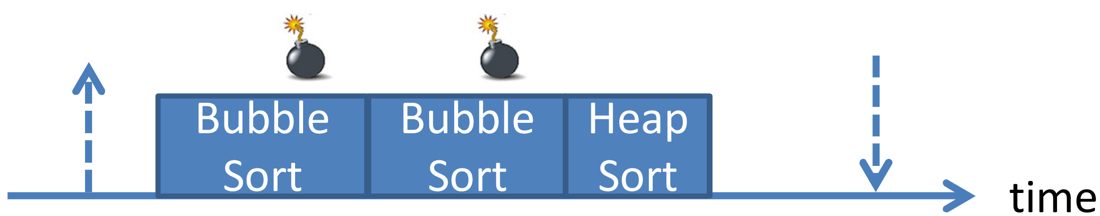
Error Recovery

- Hardware Transient Faults
  - By **re-execution**

Primary and backup have same WCET



- Software Bugs
  - Re-execution may **not** be effective (permanent error)
  - By executing a **diverse implementation** of the same task



Primary and backup may have different WCET

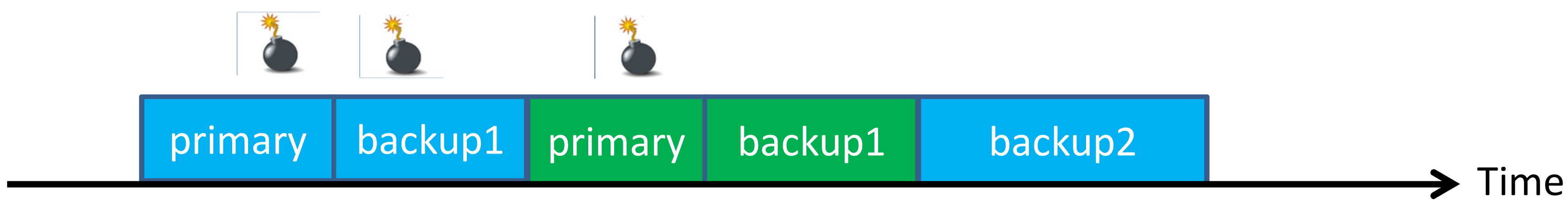
Error Recovery

Time-redundant Backups:

re-execution, or
diverse-implementation execution

Each task has one **primary** and several **backups**

- Backups are executed until it is detected to be non-erroneous



Tolerating **multiple** errors are considered

- the ***same*** job may be erroneous multiple times
- ***different*** jobs of the same or different tasks may be erroneous

How errors are detected?

- Based on existing error detection mechanisms
  - Examples
    - HW based: watchdog processor, illegal opcode detection, etc.
    - SW based: assertions, duplication and comparison
- Undetected errors have to be tolerated using **space redundancy** (**Not** covered in this work)

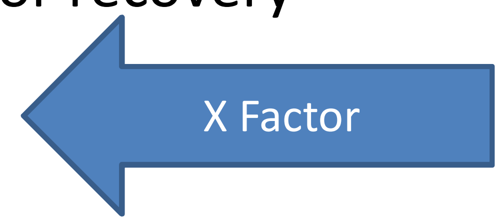
Certification and Mixed-Criticality

Certification of MC Systems

- Certification is about assurance
  - ✓ higher criticality=>higher assurance in meeting deadlines

- Different WCETs of the **same task** [Vestal,RTSS07]
  - ✓ Higher assurance => larger WCET
  - ✓ C^{LO} and C^{HI} where $C^{LO} \leq C^{HI}$

- Different numbers of errors in each interval $\leq D_{max}$
  - ✓ Higher assurance => higher number of error recovery
  - ✓ **f** and **F** where $f \leq F$

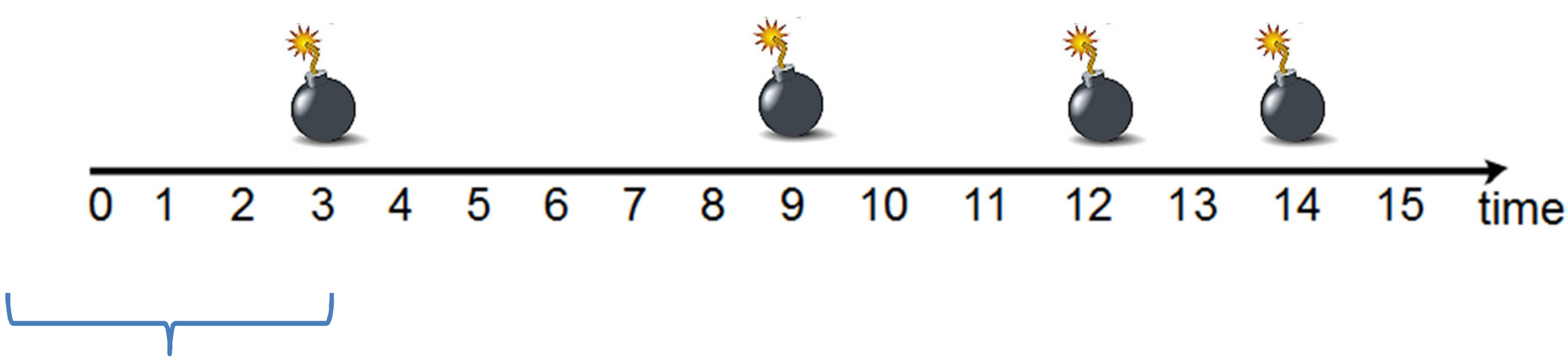


Different numbers of errors in each interval $\leq D_{max}$

- ✓ f and F where $f \leq F$

What does it mean by particular number errors in each interval $\leq D_{max}$?

$f=1 \quad F=3 \quad D_{max}=4$

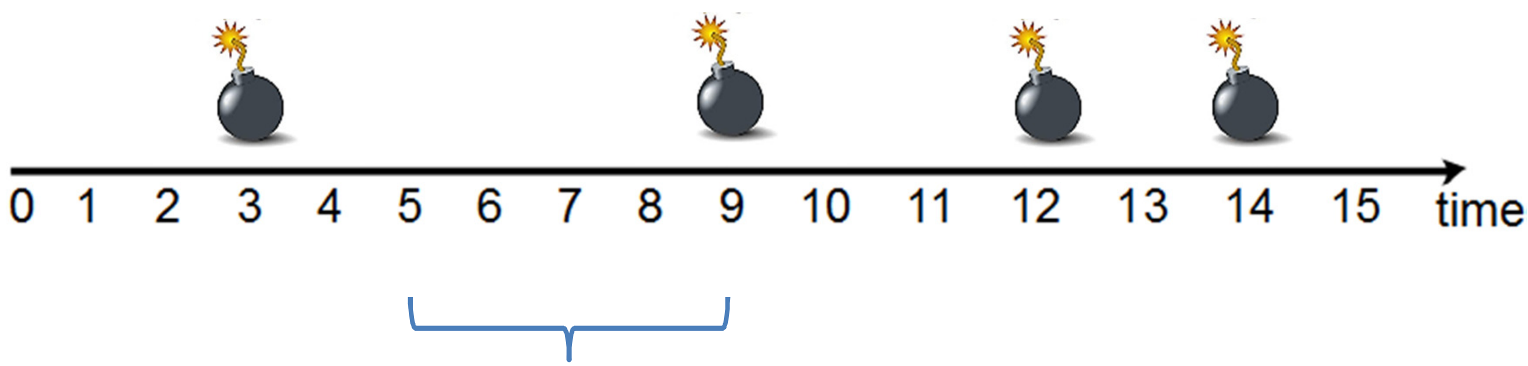


Different numbers of errors in each interval ≤ D_{max}

- ✓ f and F where f ≤ F

What does it mean by particular number errors in each interval ≤ D_{max}?

$f=1 \quad F=3 \quad D_{max}=4$

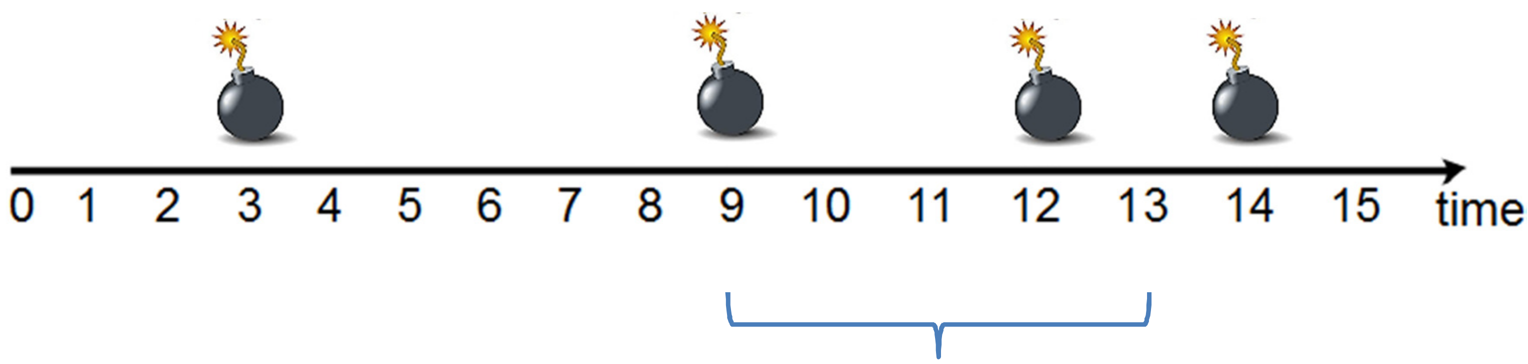


Different numbers of errors in each interval $\leq D_{max}$

- ✓ f and F where $f \leq F$

What does it mean by particular number errors in each interval $\leq D_{max}$?

$f=1 \quad F=3 \quad D_{max}=4$

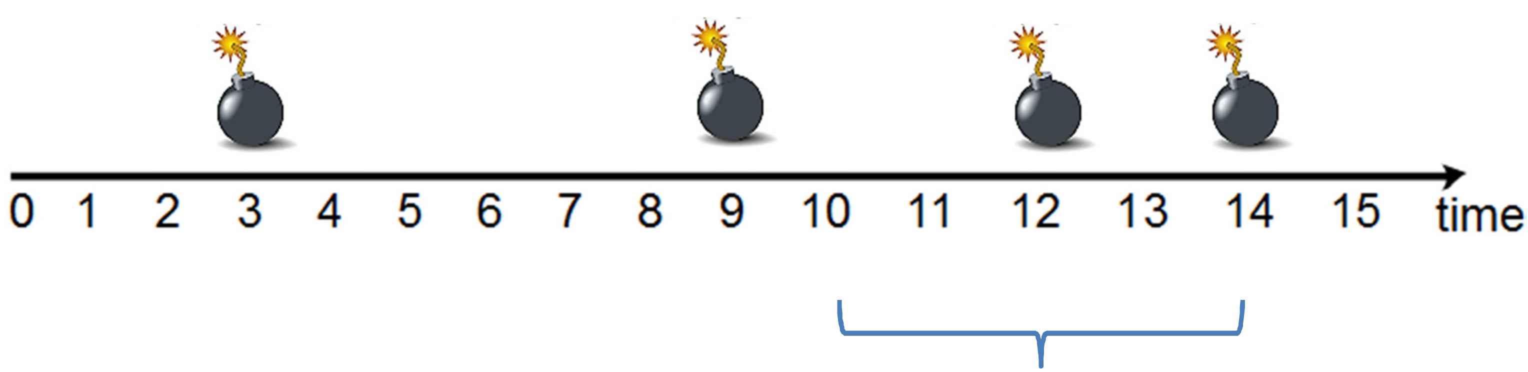


Different numbers of errors in each interval $\leq D_{max}$

✓ f and F where $f \leq F$

What does it mean by particular number errors in each interval $\leq D_{max}$?

$f=1 \quad F=3 \quad D_{max}=4$



Task Model

- Total *n* sporadic tasks
  - Task $\tau_i \equiv (L_i, C_i, B_i, D_i, T_i)$

✓ dual-criticality **$L_i \in$ {LO, HI}**

✓ Different WCETs of primary and backups of a task

- Primary: $C_i = \langle C_i^{LO}, C_i^{HI} \rangle$ where $\mathbf{C_i^{LO} \leq C_i^{HI}}$
- Backups: $B_i = \langle B_1, B_2, \ldots B_f, \ldots B_F \rangle$
  - where $B_1 = \langle B_1^{LO}, B_1^{HI} \rangle$ where $\mathbf{B_1^{LO} \leq B_1^{HI}}$
  - where $B_2 = \langle B_2^{LO}, B_2^{HI} \rangle$ where $\mathbf{B_2^{LO} \leq B_2^{HI}}$

    .

    .
  - where $B_F = \langle B_F^{LO}, B_F^{HI} \rangle$ where $\mathbf{B_F^{LO} \leq B_F^{HI}}$
- relative deadline ≤ period, i.e., $\mathbf{D_i \leq T_i}$

Task Model

- Total *n* sporadic tasks
  - Task $\tau_i \equiv (L_i, C_i, B_i, D_i, T_i)$
- Tasks are given **fixed priorities**
  - Primary and backups of a task have the same priority
  - **hp(i)**: the set of higher priority tasks of task τ_i
    - Higher-priority and LO-Critical tasks
    - Higher-priority and HI-Critical tasks
- Tasks are executed on **uniprocessor**

Scheduling Problem Statement

How to ensure that all the deadlines are met on uniprocessors ?

- different freq. of errors for different assurance levels
- different WCETs of the primary and backups for different assurance levels

Outline

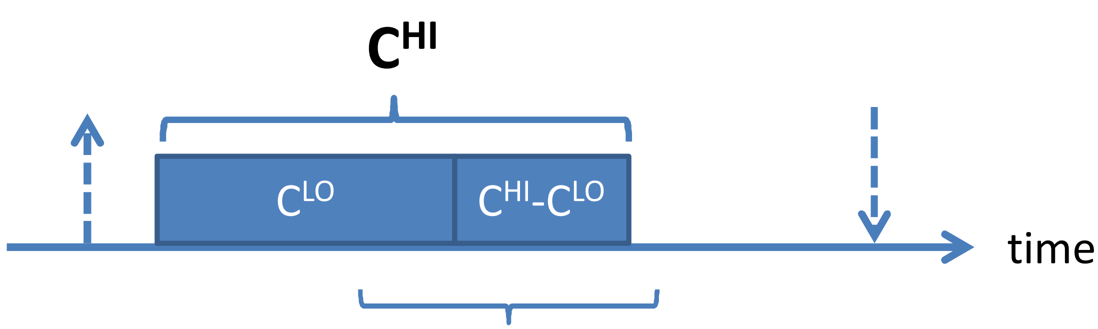
- Task model
  - Criticality Behaviors
- Scheduling Algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

Outline

- Task model
  - Criticality Behaviors
- Scheduling Algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

Criticality behavior

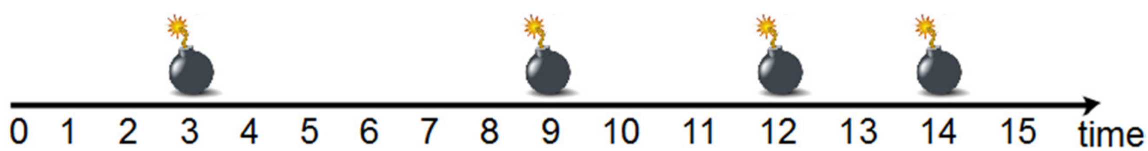
- Different assumptions regarding the two X factors
  - Assumptions that hold at runtime determines **criticality behavior**
- Exhibits **LO-Crit behavior** as long as
  - LO-Crit assumptions regarding ***all*** X factors hold
- Switches to **HI-Crit behavior** when
  - LO-Crit assumptions regarding **at least one X factor** does not hold
    - Actual exec. time of some primary/backup exceeds C^{LO}/ B^{LO} , or



Criticality behavior

- Different assumptions regarding the two X factors
  - Assumptions that hold at runtime determines **criticality behavior**
- Exhibits **LO-Crit behavior** as long as
  - LO-Crit assumptions regarding ***all*** X factors hold
- Switches to **HI-Crit behavior** when
  - LO-Crit assumptions regarding **at least one X factor** does not hold
    - Actual exec. time of some primary/backup exceeds C^{LO}/ B^{LO} , or
    - The $(f+1)^{th}$ error is detected in an interval $\leq D_{max}$

$f=1 \quad F=3 \quad D_{max}=4$



Criticality behavior

- Different assumptions regarding the two X factors
  - Assumptions that hold at runtime determines **criticality behavior**
- Exhibits **LO-Crit behavior** as long as
  - LO-Crit assumptions regarding ***all*** X factors hold
- Switches to **HI-Crit behavior** when
  - LO-Crit assumptions regarding **at least one X factor** does not hold
    - Actual exec. time of some primary/backup exceeds C^{LO}/ B^{LO} , or
    - The $(f+1)^{th}$ error is detected in an interval $\leq D_{max}$
- After criticality switches, the system exhibits HI-Crit behavior

Criticality Behavior (X Factors = WCET, freq. of errors)

$f=1 \quad F=3 \quad D_{max}=4$

LO-crit Behavior [0, 12)

Both LO and HI-crit tasks executes

No task executes more than C^{LO}/B^{LO}

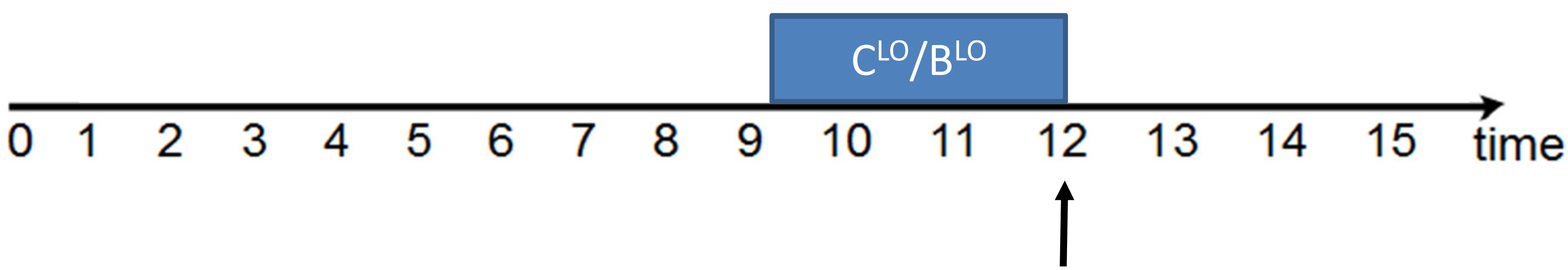
At most f=1 errors in an interval $\leq D_{max} = 4$

HI-crit Behavior [12, α)

Only HI-crit tasks executes

Task executes at most C^{HI} / B^{HI}

At most F=3 errors in an inteval $\leq D_{max} = 4$



Some task does not signal completion after executing for C^{LO}/B^{LO} time units

Criticality behavior switches from LO to HI at t=12

Criticality Behavior (X Factors = WCET, freq. of errors)

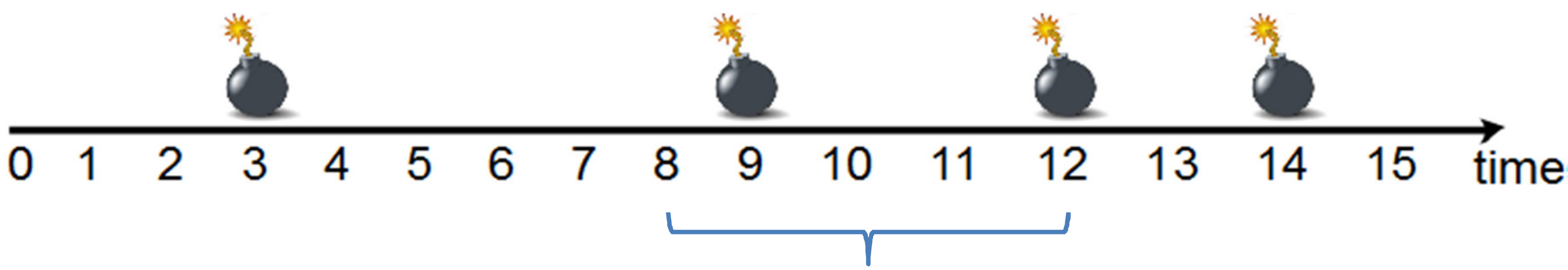
$f=1 \quad F=3 \quad D_{max}=4$

LO-crit Behavior [0, 12)

Both LO and HI-crit tasks executes
No task executes more than C^{LO}/B^{LO}
At most f=1 errors in an interval $\leq D_{max} = 4$

HI-crit Behavior [12, α)

Only HI-crit tasks executes
Task executes at most C^{HI} / B^{HI}
At most F=3 errors in an
inteval $\leq D_{max} = 4$



At time t=12, the $(f+1)^{th}=2^{nd}$ error
is detected in an interval $\leq D_{max}=4$

Criticality behavior switches from LO to HI at t=12

Outline

- Task model
  - Criticality Behaviors
- Scheduling Algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

FTMC: Fault-Tolerant Mixed-Criticality Scheduling

FTMC scheduling is same as FP scheduling on uniprocessor

+ execute a backup if an error is detected

+ criticality-switch detection

- if the $(f+1)^{th}$ error is detected in an interval $\leq D_{max}$
- if some primary/backup executes $\geq C^{LO}/B^{LO}$

+ drop all LO-crit tasks **after** switching

Outline

- Task model
  - Criticality Behaviors
- Scheduling Algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

FTMC: Schedulability Analysis

- **Correctness:** The system is ***schedulable*** in all LO- and HI-criticality behaviors.
  - **LO criticality: All (HI- and LO-critical) tasks** meet their deadlines in all LO-crit behaviors
  - **HI criticality: All HI-critical tasks** meet their deadlines in all HI-crit behaviors

FTMC: Schedulability Analysis

Response-time analysis for LO- and HI-crit behaviors to find

$\mathbf{R_i^{LO}}$: Response-time at LO-crit behavior

$\mathbf{R_i^{HI}}$: Response-time at HI-crit behavior

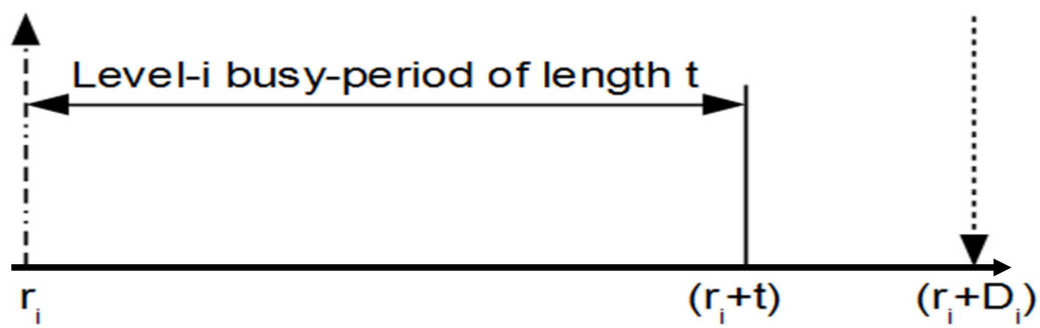
Non-MC and Non-FT

Response-time analysis of task τ_i:

$$t \leftarrow C_i + \sum_{\tau_k \in hp(i)} \lceil \frac{t}{T_k} \rceil C_j$$

Set of jobs of $\tau_i \cup hp(i)$ during the busy period are

$$\text{JobSet(t)} = \bigcup_{\tau_k \in \tau_i \cup hp(i)} \left\{ J_{k,1}, J_{k,2}, \ldots J_{k,\lceil \frac{t}{T_k} \rceil} \right\}$$



Non-MC and Non-FT

Response-time analysis of task τ_i:

$$t \leftarrow C_i + \sum_{\tau_k \in hp(i)} \lceil \frac{t}{T_k} \rceil C_j$$

Set of jobs of $\tau_i \cup hp(i)$ during the busy period are

$$\text{JobSet(t)} = \bigcup_{\tau_k \in \tau_i \cup hp(i)} \left\{ J_{k,1}, J_{k,2}, \ldots J_{k,\lceil \frac{t}{T_k} \rceil} \right\}$$

If these jobs recover E errors, then what is the total workload in the busy period?

Work(JobSet(t), E) =?

Example Task Set (F=2)

	L_i	C_i^{LO}	B_1^{LO}	B_2^{LO}	C_i^{HI}	B_1^{HI}	B_2^{HI}	D_i	T_i
τ_1	HI	1	1	2	2	1	2	7	8
τ_2	LO	2	3	2	–	–	–	12	14
τ_3	HI	3	3	3	4	3	3	14	28

Primary

Example Task Set (F=2)

	L_i	C_i^{LO}	B_1^{LO}	B_2^{LO}	C_i^{HI}	B_1^{HI}	B_2^{HI}	D_i	T_i
τ_1	HI	1	1	2	2	1	2	7	8
τ_2	LO	2	3	2	–	–	–	12	14
τ_3	HI	3	3	3	4	3	3	14	28

First Backup

Example Task Set (F=2)

	L_i	C_i^{LO}	B_1^{LO}	B_2^{LO}	C_i^{HI}	B_1^{HI}	B_2^{HI}	D_i	T_i
τ_1	HI	1	1	2	2	1	2	7	8
τ_2	LO	2	3	2	–	–	–	12	14
τ_3	HI	3	3	3	4	3	3	14	28

Second
Backup

Example Task Set (F=2)

	L_i	C_i^{LO}	B_1^{LO}	B_2^{LO}	C_i^{HI}	B_1^{HI}	B_2^{HI}	D_i	T_i
τ_1	HI	1	1	2	2	1	2	7	8
τ_2	LO	2	3	2	–	–	–	12	14
τ_3	HI	3	3	3	4	3	3	14	28

Work({J}, E) = total exec. by job J to recover E errors

If J is a job of task τ_3 and E=2, then

Work({J_{LO}}, 2) = 3+3+3 = 9

Work({J_{HI}}, 2) = 4+3+3 = 10

Work(JobSet(t), E)=?

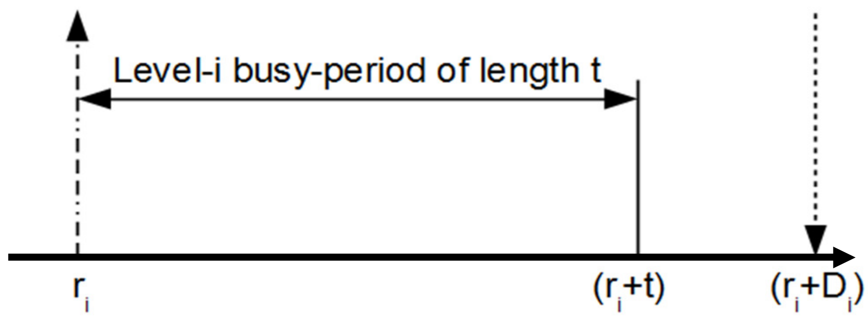
Steps to Compute R_i^{LO} and R_i^{HI}

- Find **Jobset(t)**: the jobs that are eligible to execute in the busy period are determined.
- **Characterize** each job J as J_{LO} or J_{HI}.
- **Workload** is computed considering maximum number of errors in the busy period.
- A **recurrence** is formulated to find the response time.

Finding R_i^{LO}

R_i^{LO}: Schedulability Analysis

We compute the workload in the level-*i* busy period



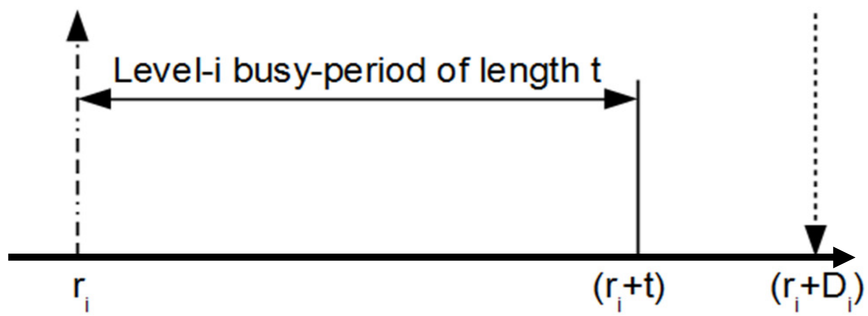
Both LO- and HI-Crit jobs are executed in LO-criticality behavior and at most f errors can occur

Critical instant (Audsley et al. 1991) for sporadic tasks applies:

- when all tasks arrive simultaneously, and
- when jobs of the tasks arrive strictly periodically.

R_i^{LO}: Schedulability Analysis

We compute the workload in the level-*i* busy period

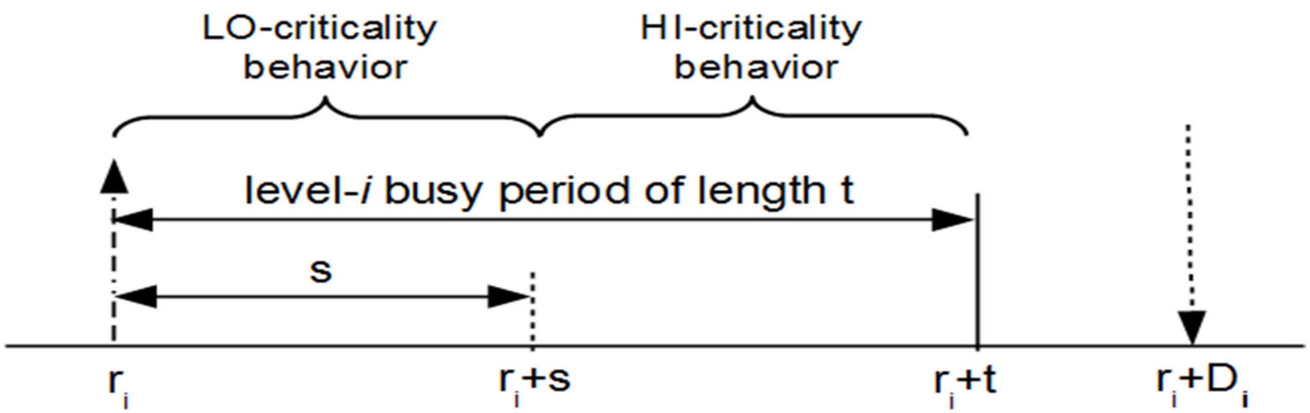


$$\textbf{JobSet(t)} = \bigcup_{\tau_k \in \tau_i \cup hp(i)} \left\{ J_{k,1}, J_{k,2}, \ldots J_{k,\lceil \frac{t}{T_k} \rceil} \right\}$$

$$t \leftarrow \text{Work}(\textbf{JobSet(t)}, \mathbf{f})$$

Finding R_i^{HI}

R_i^{HI}: Schedulability Analysis



LO-crit tasks executes ONLY during LO-criticality behavior

HI-crit tasks executes during LO- and HI-criticality behavior

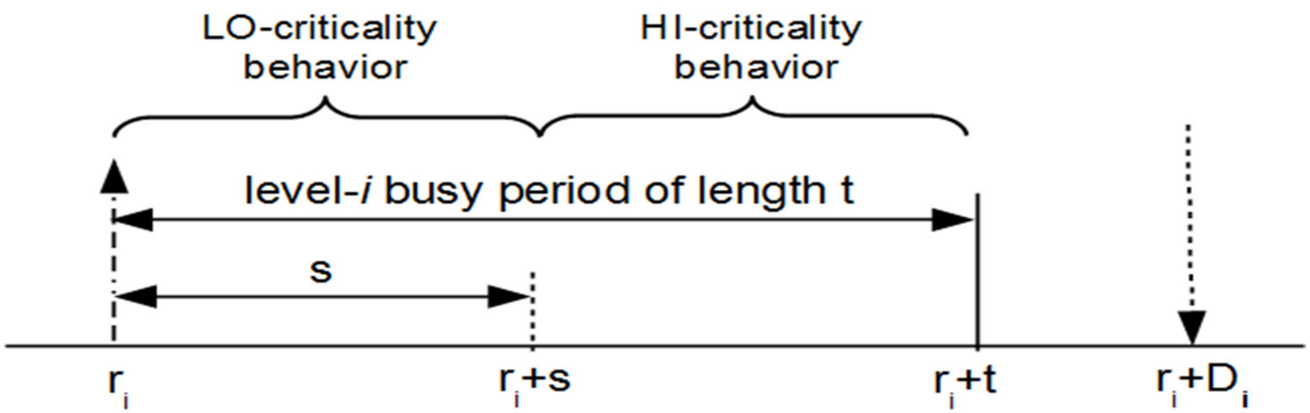
Sanjoy Baruah, Alan Burns, and Robert Davis. [Response-time analysis for mixed criticality systems](). **Proceedings of the IEEE Real-Time Systems Symposium (RTSS)**, 2011.

X = set of LO-Crit jobs that execute in LO-Crit behavior

Y = set of HI-Crit jobs that execute in HI-Crit behavior

Z = set of HI-Crit jobs that execute in LO-Crit behavior

R_i^{HI}: Schedulability Analysis



LO-crit tasks executes ONLY during LO-criticality behavior

HI-crit tasks executes during LO- and HI-criticality behavior

Sanjoy Baruah, Alan Burns, and Robert Davis. Response-time analysis for mixed criticality systems. **Proceedings of the IEEE Real-Time Systems Symposium (RTSS)**, 2011.

JobSet(t,s) = X U Y U Z

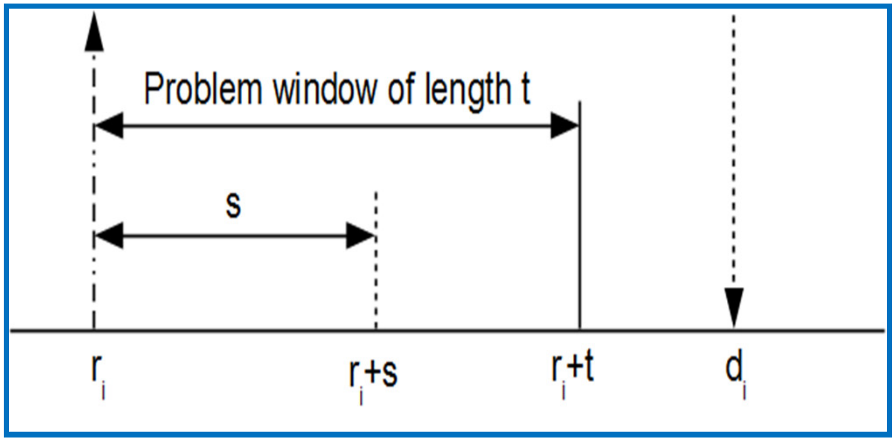
R_i^{HI}: Schedulability Analysis

Jobs in "JobSet(t,s)"" are executed in the busy period where at most F errors can occur

$R_{i,s}^{HI}$ is the solution of

Work(**JobSet(t,s)**, **F**)

$$R_i^{HI} = \max \{ R_{i,s}^{HI} \}$$



Priority Assignment

- Deadline-monotonic is not optimal for uniprocessor MC system [Vestal, RTSS07]

How to assign the fixed-priorities for MC scheduling on uniprocessor?

Audsley's Optimal Priority Assignment (OPA)

Audsley' OPA algorithm

for each priority level k, lowest first

for each priority unassigned task τ_i

If $R_i^{HI} \leq D_i$ and $R_i^{LO} \leq D_i$ assuming higher priorities
for the other priority unassigned task, then

assign τ_i to priority k

break (continue outer loop)

return "unschedulable"

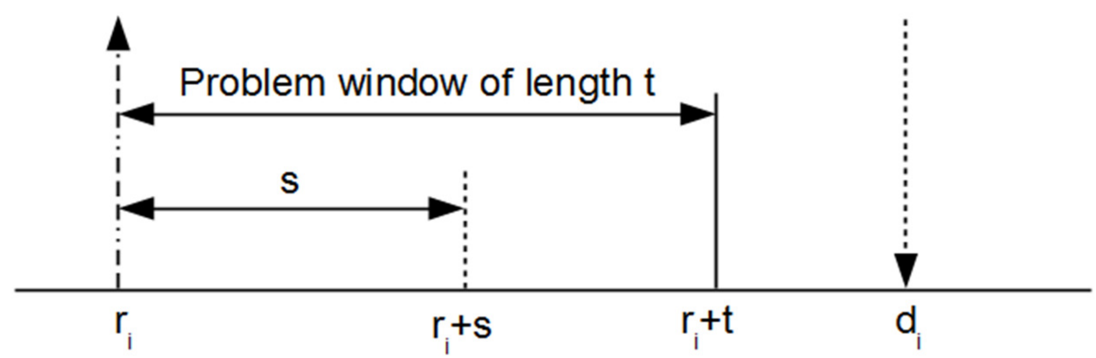
return "schedulable"

Evaluation

Schedulability Tests

Three tests are evaluated

- DM-FTMC: Response time tests with deadline-monotonic priority assignment
- OPA-FTMC: Response time tests with OPA
- UBound test: Necessary Test
  - This is an upper bound on the schedulable task sets by FTMC algorithm.



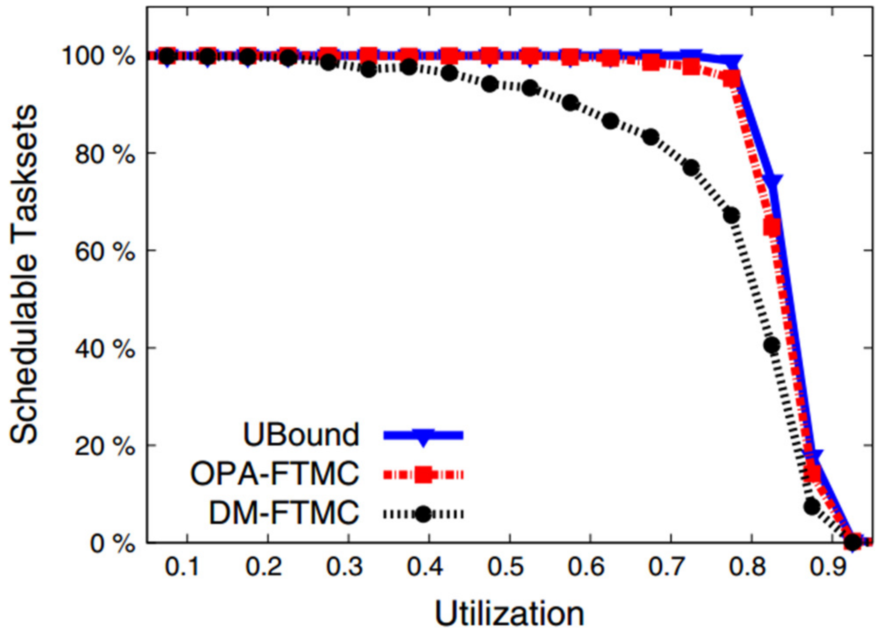
Simulation Parameters

- Random mixed-criticality task sets are generated
  - **U** : total utilization of a task set $(\sum C_{LO}/T)$
  - **n** : number of tasks in a task set
  - **f**, **F** : Frequency of errors
  - **CF** : $C^{HI} = C^{LO} \times CF$

- Backups are same as the primary (i.e., re-excution)

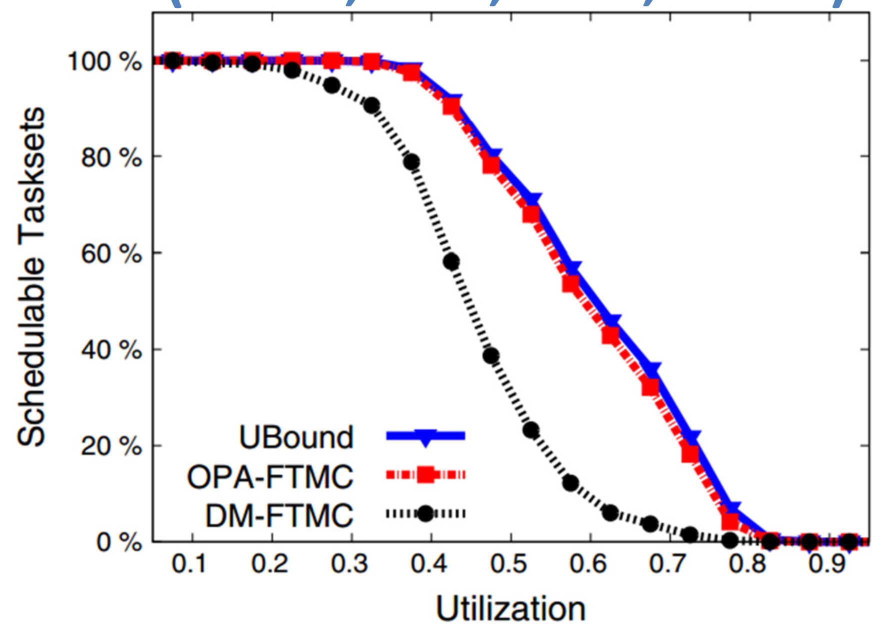
Results

(n = 20, f = 0, F = 1, CF = 1)



No pessimism regarding WCET

(n = 20, f = 1, F = 2, CF = 2)



Pessimism regarding WCET and freq. of errors

Conclusion

- FP scheduling of sporadic tasks on uniprocessor
  - Real time, fault tolerance, and mixed criticality

- Priority assignment with Audsley's OPA

- Applicable to more than two criticality levels
  - **Reference: Risat Mahmud Pathan, "Fault-Tolerant and Real-Time Scheduling for Mixed-Criticality Systems", Real-Time Systems Journal, Vol. 50, Issue. 4, July 2014.**

Future work: Apply it for multiprocessors, probabilistic analysis

What are the other X factors?

Thank You

Email: risat@chalmers.se

Risat Mahmud Pathan, "Fault-Tolerant and Real-Time Scheduling for Mixed-Criticality Systems", Real-Time Systems Journal, Vol. 50, Issue. 4, July 2014.