THE UNIVERSITY of York

Inria
INVENTORS FOR THE DIGITAL WORLD

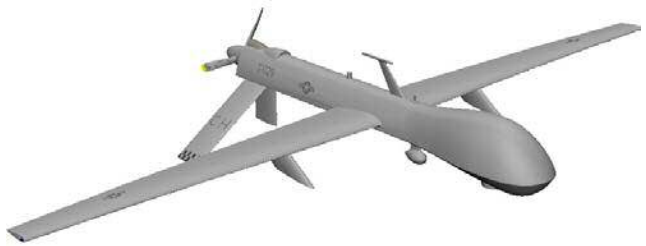
Response Time Analysis for Mixed Criticality Systems with Arbitrary Deadlines

Alan Burns[1] and Robert Davis[1,2]

[1] *Real-Time Systems Research Group, University of York*
[2] *Inria, Paris, France*

Mixed Criticality Systems

- MCS
  - Applications of different criticality levels on the same HW platform
    - E.g. Safety Critical, Mission Critical, Non-critical
  - Driven by SWaP and cost requirements

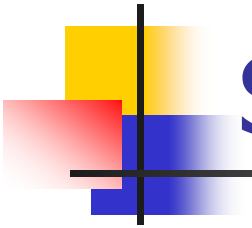


- Examples
  - Aerospace: e.g. UAVs
    - Flight Control Systems v. Surveillance
  - Automotive:
    - Electronic Power Assisted Steering (EPAS) vs. Cruise Control

- This research considers: Dual-Criticality Systems
  - Applications with HI- and LO-criticality

Mixed Criticality Systems

- Key requirements
  - *Separation* – must ensure LO-criticality applications cannot impinge on those of HI-criticality
  - *Sharing* – want to allow LO- and HI-criticality applications to use the same resources for efficiency
- Real-Time behaviour
  - Concept of a criticality mode (LO or HI)
  - System starts in LO-criticality mode
  - LO- and HI-criticality tasks must meet their deadlines in LO-criticality mode
  - Only HI-criticality tasks need meet their deadlines in HI-criticality mode
- Initial Research (Vestal 2007)
  - Idea of different LO- and HI-criticality WCET estimates for the same code
  - Certification authority requires pessimistic approach to $C(HI)$
  - System designers take a more realistic approach to $C(LO)$



System Model

- Uniprocessor
- Scheduling based on fixed priorities
- Sporadic task sets
  - T_i – Period or minimum inter-arrival time (sporadic behaviour)
  - D_i – Relative deadline
  - L_i – Criticality level (LO or HI)
  - HI-criticality tasks have both $C_i(HI)$ and $C_i(LO)$ worst-case execution time estimates with $C_i(HI) > C_i(LO)$
  - LO-criticality tasks need only have $C_i(LO)$

Adaptive Mixed Criticality (AMC)

- AMC scheduling scheme
  - If a HI-criticality task executes for its $C(LO)$ without signalling completion then no further jobs of LO-criticality tasks are started[1] and the system enters HI-criticality mode
  - This frees up processor bandwidth to ensure that HI-criticality tasks can meet their deadlines in HI-criticality mode

- Analysis of AMC
  1. Check all tasks are schedulable in LO-criticality mode
  2. Check HI-criticality tasks are schedulable in HI-criticality mode
  3. Check HI-criticality tasks are schedulable over the mode change

[Note analysis for the mode transition usually also covers HI-criticality mode]

[1]Any partially executed job of a LO-criticality task may complete

AMC Analysis methods

- AMC-rtb
  - Uses a *response time bound* (rtb) to limit the time interval during a busy period in which LO-criticality jobs are considered to execute
- AMC-max
  - More precise analysis which considers a number of different times at which a mode change transition could occur and takes the *max* response time over all of them
- Scope
  - Original analysis is limited to constrained deadline tasks
- Contribution of this work
  - Extend AMC-rtb and AMC-max analysis to tasks with arbitrary deadlines
- Motivation
  - If arbitrary deadlines are possible without breaking time constraints and can be supported, for example by buffering inputs, then schedulability can be improved

Recap: Analysis of FPPS with arbitrary deadlines

- Approach
  - With arbitrary deadlines, multiple jobs of a task τ_i can be active at the same time
  - Need to compute the response time for each job q of task τ_i in the priority level-i busy period starting from synchronous release
  - Completion time of job q from the start of the busy period is given by:

$$r_i(q) = (q+1)C_i + \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{r_i(q)}{T_j} \right\rceil C_j$$

  - Examine jobs $q = 0 \ldots p$ where p is the first job with completion before the next release: $r_i(p) \leq (p+1)T_i$.
  - Response time of each job is given by:

$$\forall q_{(0 \leq q \leq p)} : \quad R_i(q) = r_i(q) - qT_i$$

  - Worst case response time is therefore:

$$R_i = \max_{\forall q_{(0 \leq q \leq p)}} \{R_i(q)\}$$

[K. W. TindellA. BurnsA. J. Wellings "An extendible approach for analyzing fixed priority hard real-time tasks" . Real-Time Systems March 1994, Volume 6, Issue 2, pp 133–151]

Recap: AMC-rtb analysis for constrained deadlines

- LO-criticality mode
  - All tasks must be schedulable

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

- HI-criticality mode
  - Only HI-criticality tasks need be schedulable
  - Consider behaviour in HI-criticality mode and across the mode change

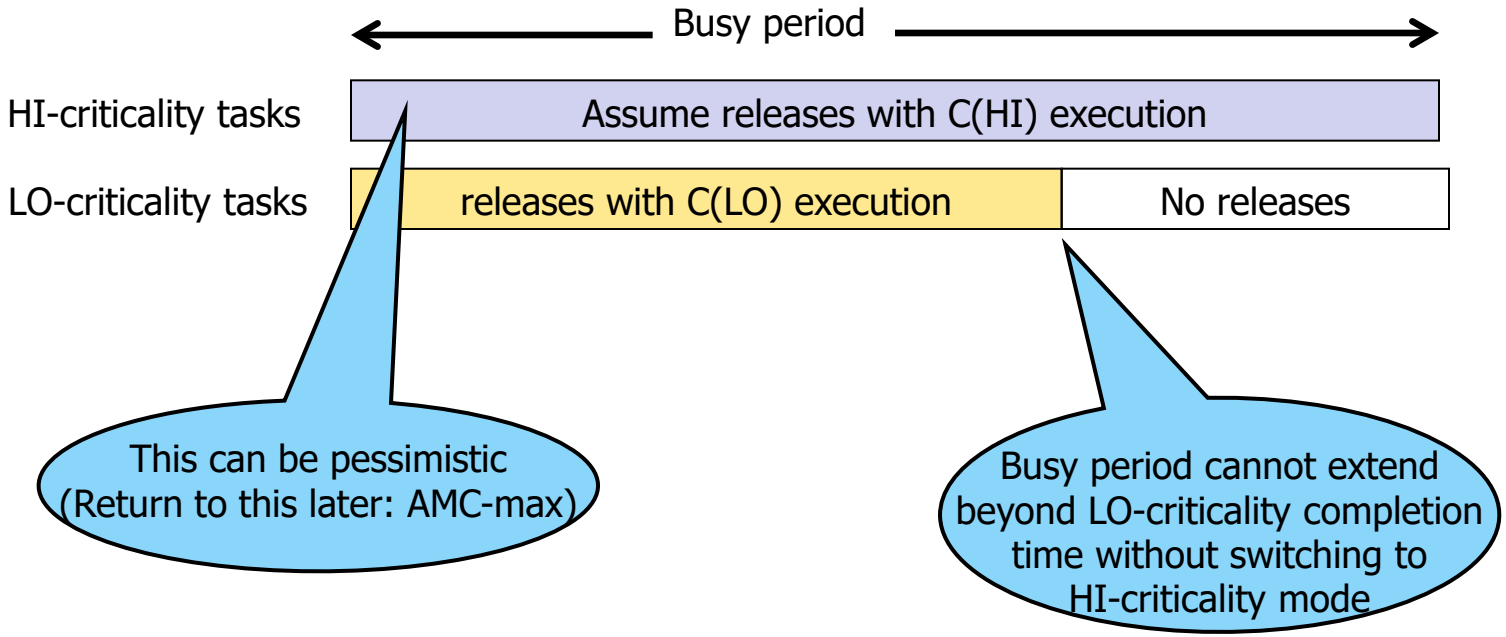
$$R_i(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI) + \sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k(LO)$$

Interference from higher priority LO-criticality tasks cannot be released beyond the completion time in LO-criticality mode

[S.K. Baruah, A. Burns, R.I. Davis "Response Time Analysis for Mixed Criticality Systems" . In proceedings 32nd IEEE Real-Time Systems Symposium (RTSS'11) , pages 34-43, Nov 29th - Dec 2nd, 2011]

AMC-rtb
key simplifying idea

- **Analysis for mode change transition**
  - Considers HI-criticality tasks executing for C(HI) over all of the busy period
  - Considers LO-criticality task releases for as long as possible – which is until the job of task τ_i would have completed if the system had stayed in LO-criticality mode



New: AMC-rtb-Arb analysis for arbitrary deadlines

- LO-criticality mode
  - All tasks must be schedulable
  - Compute completion time of each job q of task τ_i in the busy period

$$r_i^L(q) = (q+1)C_i(LO) + \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{r_i^L(q)}{T_j} \right\rceil C_j(LO)$$

  - Worst-case response time of each job

$$\forall q_{(0 \le q \le p)} : \quad R_i(LO)(q) = r_i^L(q) - qT_i$$

  - Examine jobs $q = 0 \ldots p$ where p is the first job with completion before the next release, and so is the last job in the busy period

$$r_i^L(p) \le (p+1)T_i$$

  - Worst-case response time of the task

$$R_i(LO) = \max_{\forall q_{(0 \le q \le p)}} \{R_i(LO)(q)\}$$

This is just arbitrary deadline analysis for FPPS

New: AMC-rtb-Arb analysis for arbitrary deadliness

- HI-criticality mode
  - Only HI-criticality tasks need be schedulable
  - Consider behaviour in HI-criticality mode and across the mode change

$$r_i^H(q) = (q+1)C_i(HI) + \sum_{j \in \mathbf{hpH}(i)} \left\lceil \frac{r_i^H(q)}{T_j} \right\rceil C_j(HI) + \sum_{k \in \mathbf{hpL}(i)} \left\lceil \frac{r_i^L(\min(q,p))}{T_k} \right\rceil C_k(LO)$$

Interference from higher priority LO-criticality tasks cannot be released after the LO-criticality completion time of job q

Interference from LO-criticality tasks cannot be released after completion of job p

  - Job v is the first job where completion occurs before the next release and is the last job in the busy period. (Note $v > p$ is possible, where p is the last job in the LO-criticality busy period)
  - Worst-case response time of each job

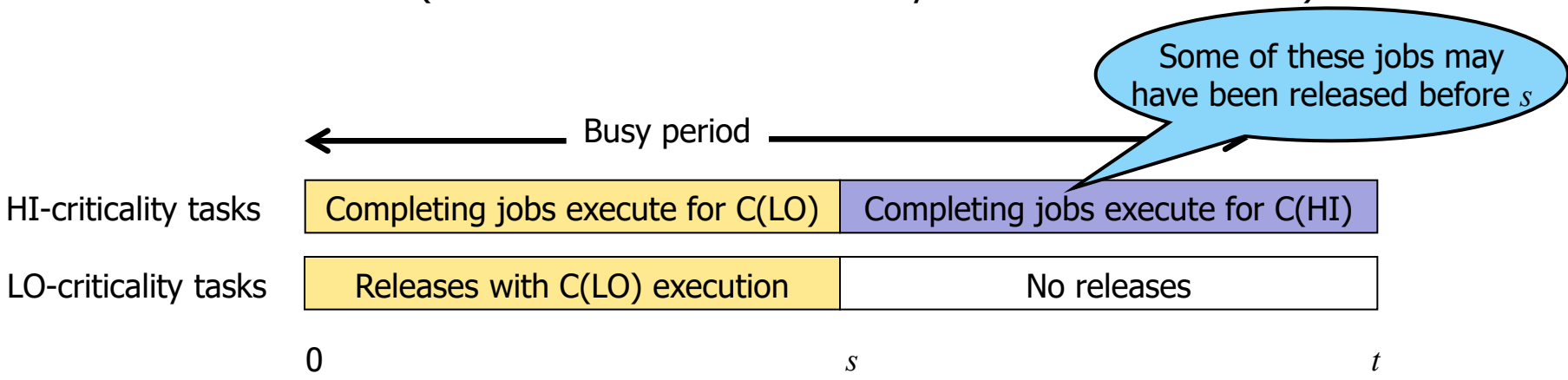
$$\forall q (0 \le q \le v): \quad R_i(HI)(q) = r_i^H(q) - qT_i$$

  - Worst-case response time of the task

$$R_i(HI) = \max_{\forall q (0 \le q \le v)} \{R_i(HI)(q)\}$$

AMC-max
key simplifying idea

- **Analysis for mode change transition**
  - Consider some interval of length s from start of busy period of length t
  - At time s, the system switches to HI-criticality mode
  - Compute the maximum response time considering all relevant values of s
  - Since interference from LO-criticality tasks only increases at values of s that are multiples of their periods, those are the relevant values to consider (Interference from HI-criticality tasks decreases with s).



  - AMC-max dominates AMC-rtb due to its less pessimistic treatment of interference from HI-criticality tasks at the beginning of the busy period

Recap: AMC-max analysis for constrained deadlines

- LO-criticality mode
  - Same as AMC-rtb
- HI-criticality mode
  - Only HI-criticality tasks need be schedulable
  - Consider behaviour in HI-criticality mode and across the mode change
  - Compute response time with a mode change at time s

$$R_i^s(HI) = C_i(HI) + I_L(s) + I_H(s)$$

  - Take maximum over relevant values of s

$$R_i^*(HI) = \max_{\forall s}(R_i^s(HI))$$

Recap: AMC-max analysis for constrained deadlines

- Interference from LO-criticality tasks
  - Prevented from being released after the mode change at time s

$$I_L(s) = \sum_{j \in \mathbf{hpL}(i)} \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1 \right) C_j(LO)$$

- Interference from HI-criticality tasks
  - Upper bound on the number of jobs in the busy period that can execute after s

$$\left\lceil \frac{t - s - (T_k - D_k)}{T_k} \right\rceil + 1$$

Effectively jittered by D_k

  - The above cannot be more than the number of jobs released in *t*, so

$$M(k,s,t) = \min \left\{ \left\lceil \frac{t - s - (T_k - D_k)}{T_k} \right\rceil + 1, \left\lceil \frac{t}{T_k} \right\rceil \right\}$$

[Note: $M(k,s,t)$ cannot be less than zero]

  - Hence

$$I_H(s) = \sum_{k \in \mathbf{hpH}(i)} \left\{ M(k,s,t)\ C_k(HI) + \left(\left\lceil \frac{t}{T_k} \right\rceil - M(k,s,t) \right) C_k(LO) \right\}$$

Recap: AMC-max analysis for constrained deadlines

- Putting it all together

$$R_i^s(HI) = C_i(HI) + \sum_{j \in \mathbf{hpL}(i)} \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1 \right) C_j(LO) +$$

$$\sum_{k \in \mathbf{hpH}(i)} \left\{ M(k, s, R_i^s(HI)) C_k(HI) + \left(\left\lceil \frac{R_i^s(HI)}{T_k} \right\rceil - M(k, s, R_i^s(HI)) \right) C_k(LO) \right\}$$

  - Take the maximum overall all values of s equating to integer multiples of LO-criticality task periods

$$R_i^*(HI) = \max_{\forall s}(R_i^s(HI))$$

[S.K. Baruah, A. Burns, R.I. Davis "Response Time Analysis for Mixed Criticality Systems" . In proceedings 32nd IEEE Real-Time Systems Symposium (RTSS'11) , pages 34-43, Nov 29th - Dec 2nd, 2011]

New: AMC-max analysis for arbitrary deadlines

- HI-criticality mode
  - Compute completion time of each job q of task τ_i

$$r_i^s(q) = XC_i(HI) + YC_i(LO) + I_L(s) + I_H(s)$$

  - where $X + Y = q + 1$.
  - Upper bound on the number of jobs of an interfering task that can execute after s is the same equation as with constrained deadlines

$$\left\lceil \frac{t - s - (T_k - D_k)}{T_k} \right\rceil + 1$$

Effectively jittered by D_k which can now be > T_k

  - Upper bound of the number of jobs of task τ_i that can execute after s is given by (note $q + 1$ jobs in all)

$$X = \min\left(\left\lceil \frac{t - s + (D_i - T_i)}{T_i} \right\rceil + 1, q + 1\right)$$

New: AMC-max analysis for arbitrary deadlines

- Putting it all together

$$r_i^s(q) = XC_i(HI) + YC_i(LO) + I_L(s) + I_H(s)$$

  - where $X = \min\left(\left\lceil \frac{t - s + (D_i - T_i)}{T_i} \right\rceil + 1, q + 1\right)$ and $X + Y = q + 1$.

$$I_L(s) = \sum_{j \in \mathbf{hpL}(i)} \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1\right) C_j(LO)$$

$$I_H(s) = \sum_{k \in \mathbf{hpH}(i)} \left\{ M(k,s,t)\ C_k(HI) + \left(\left\lceil \frac{t}{T_k} \right\rceil - M(k,s,t)\right) C_k(LO) \right\}$$

  - Iterate with $t = r_i^s(q)$ until convergence or deadline for job q exceeded
  - Gives the worst-case completion time of job q given a transition from LO-criticality to HI-criticality mode at time s

New: AMC-max analysis for arbitrary deadlines

- Putting it all together (continued)
  - Compute response times for all values of s corresponding to multiples of LO-criticality task periods up to completion of job q
  - Determine worst-case completion time of job q

$$r_i^*(q) = \max_{\forall s} (r_i^s(q))$$

  - Determine worst-case response time for each job q up to job p which is the first job where completion occurs before the next release

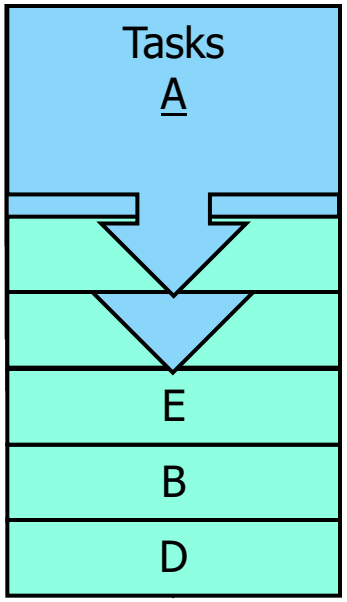
$$\forall q_{(0 \le q \le p)} : \quad R_i(HI)(q) = r_i^*(q) - qT_i$$

  - Finally we have the worst-case response time for the task

$$R_i(HI) = \max_{\forall q (0 \le q \le p)} \{R_i(HI)(q)\}$$

Recap: Optimal Priority Assignment (OPA) Audsley's algorithm

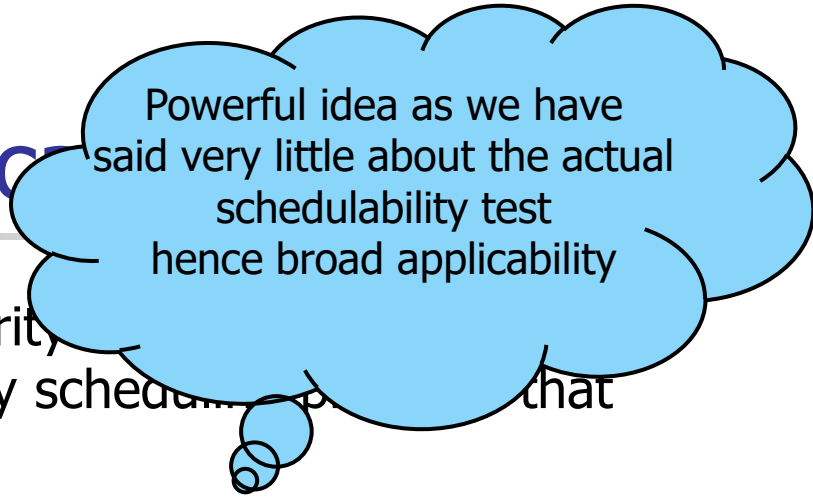
```
for each priority level i, lowest first {
    for each unassigned task τ {
        if τ is schedulable at priority i
        assuming that all unassigned tasks are
        at higher priorities {
            assign task τ to priority level i
            break (exit for loop)
        }
    }
    if no tasks are schedulable at priority i {
        return unschedulable
    }
}
return schedulable
```



$n(n+1)/2$ schedulability tests rather than $n!$
by exhaustively exploring all possible orderings

[N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", Technical Report YCS 164, Dept. Computer Science, University of York, UK, 1991.]
[N.C. Audsley, "On priority assignment in fixed priority scheduling", Information Processing Letters, 79(1): 39-44, May 2001.]
[K. Bletsas, and N.C. Audsley, "Optimal priority assignment in the presence of blocking". *Information Processing Letters* Vol. 99, No. 3, pp83-86, August. 2006]

OPA algorithm applic



- OPA algorithm provides optimal priority
  schedulability test S for fixed priority schedu… that three conditions are met…

  Condition 1: Schedulability of a task may, according to the test, be dependent on the set of higher priority tasks, but not on their relative priority ordering

  Condition 2: Schedulability of a task may, according to the test, be dependent on the set of lower priority tasks, but not on their relative priority ordering

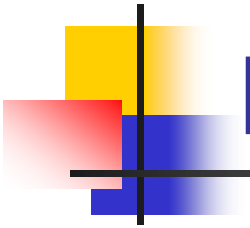
  Condition 3: When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to the test, if it was previously deemed schedulable at the lower priority

Tests meeting these conditions are referred to as **OPA-compatible**

[R.I. Davis and A. Burns "Improved Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-Time Systems". Real-Time Systems, (2011) Volume 47, Number 1, pages 1-40, (First published online 22nd September 2010). DOI 10.1007/s11241-010-9106-5.]

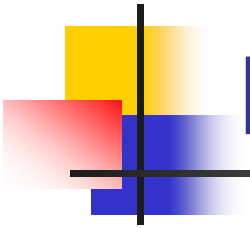
Priority assignment for AMC-rtb-Arb and AMC-max-Arb

- AMC-rtb-Arb and AMC-max-Arb schedulability tests
  - By inspection of the schedulability tests we can see that all three necessary and sufficient **Conditions** hold so both tests are **OPA-compatible**
  - Hence we can use Audsley's algorithm for priority assignment … and do so in all our experiments



Evaluation

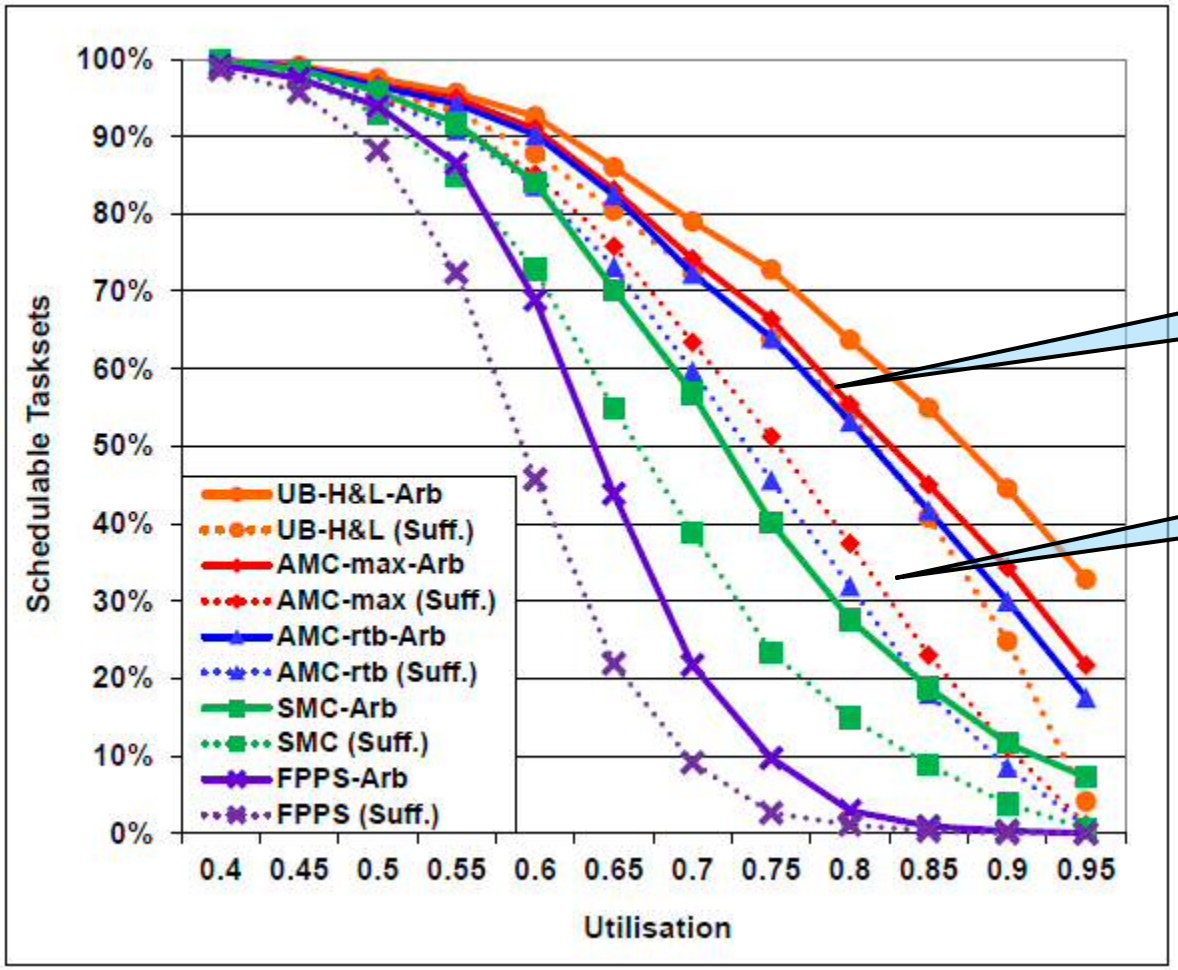
- Compared the following schemes:
  - **UB-H&L-Arb**: A *necessary test* which checks if all of the tasks are schedulable in LO-criticality mode and if the HI-criticality tasks are schedulable in HI-criticality mode (with no LO-criticality tasks executing). It ignores the mode switch
  - **AMC-max-Arb**: New AMC-max analysis for arbitrary deadlines
  - **AMC-rtb-Arb**: New AMC-rtb analysis for arbitrary deadlines
  - **SMC-Arb**: Arbitrary deadline analysis for the SMC scheme (LO-criticality tasks continue to be released in HI-criticality mode)
  - **FPPS-Arb**: Arbitrary deadline analysis for FPPS (LO-criticality tasks also have to be schedulable in HI-criticality mode)
- Also
  - Sufficient tests artificially restricting deadlines to no more than period **AMC-max (suff.), AMC-rtb (suff.), SMC (suff.), FPPS (suff.)**



Evaluation

- Task set generation:
  - Number of tasks (Default n = 20)
  - Periods: Log-uniform distribution (Default 10ms – 100ms)
  - Deadlines: Log-uniform distribution ([0.25,4.0]Period)
  - Utilisation values U_i generated using Uunifast
  - LO-criticality execution times set via $C_i(LO) = U_i T_i$
  - HI-criticality execution times $C(HI) = CF.\ C(LO)$ where CF is the criticality factor (Default CF = 2.0)
  - Probability CP of a task being HI-criticality (Default CP = 0.5)

Success ratio



AMC-rtb-Arb nearly as good as AMC-max-Arb.

Big gap to sufficient tests with deadline restriction

Weighted schedulability

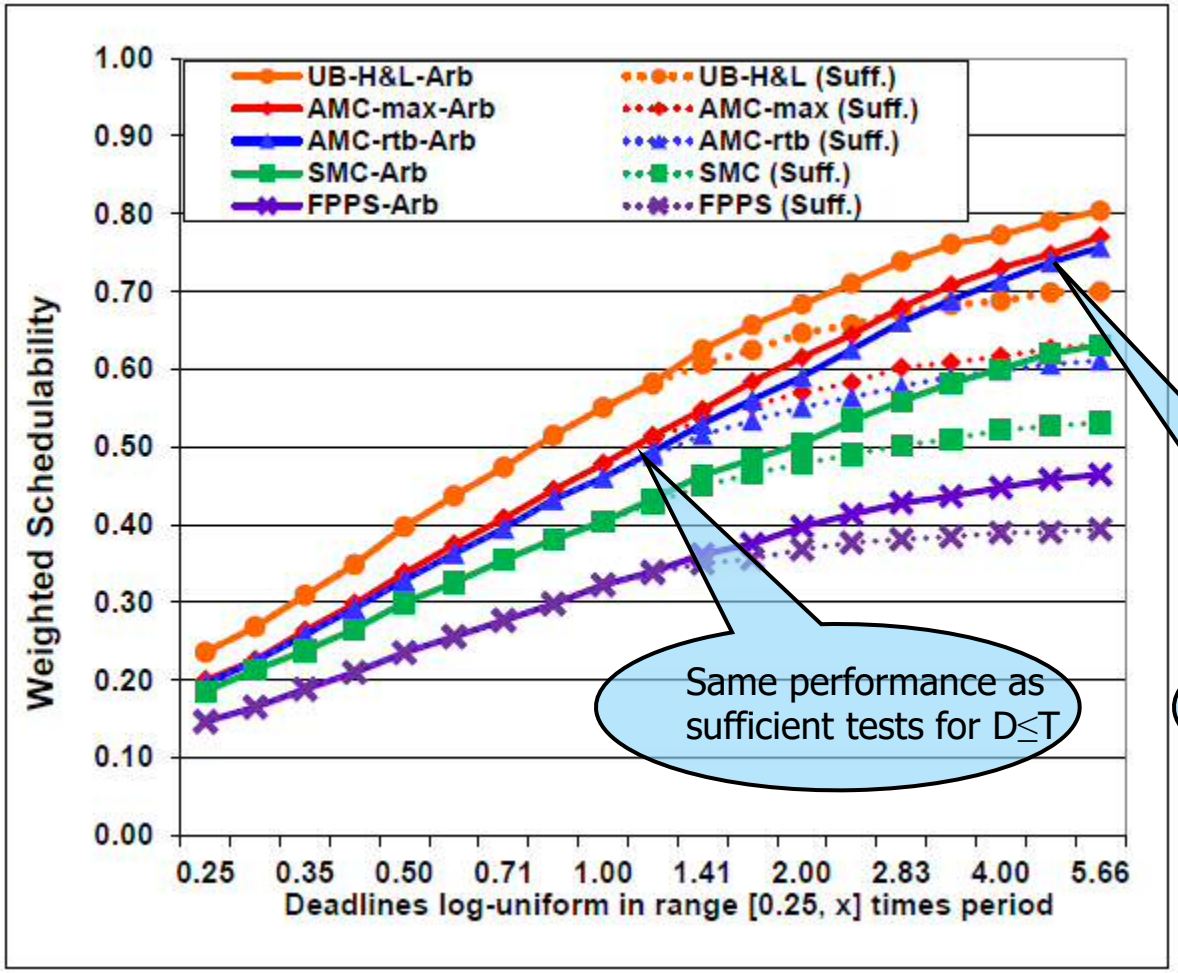
- Weighted schedulability
  - Enables overall comparisons when varying a specific parameter (not just utilisation)
  - Combines results from all of a set of equally spaced utilisation levels
  - Weighted schedulability:

$$Z_y(p) = \frac{\sum_{\forall \tau} S_y(\tau).U(\tau)}{\sum_{\forall \tau} U(\tau)}$$

  - Collapses all data on a success ratio plot for a given method, into a single point on a weighted schedulability graph

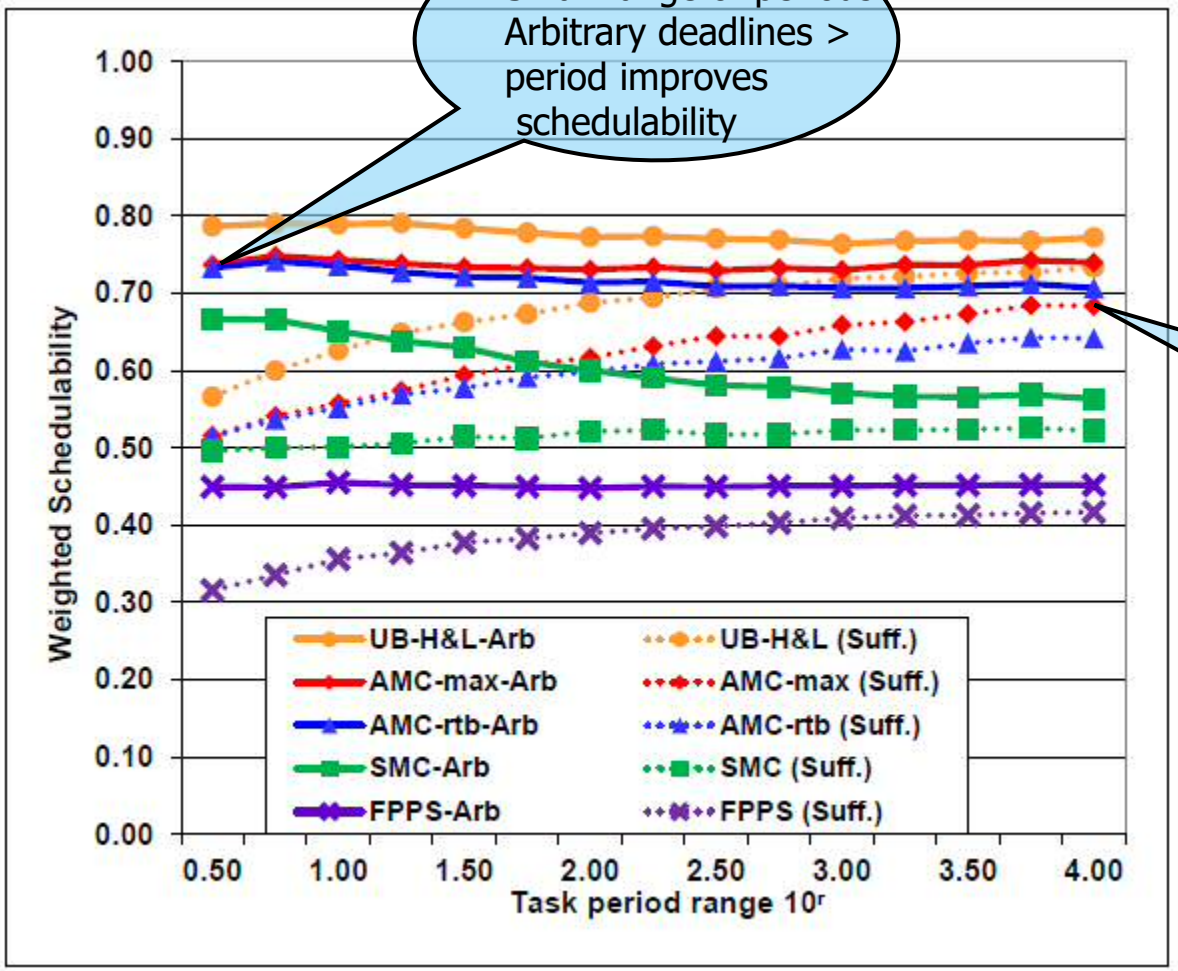
Weighted schedulability is effectively a weighted version of the area under a success ratio curve biased towards scheduling higher utilisation task sets

Weighted schedulability: varying deadlines



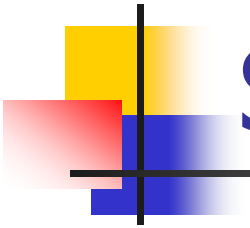
Varying deadlines log-uniform in the range [0.25, x]Period where x goes from 0.25 to >4.0

Weighted schedulability: range of task periods



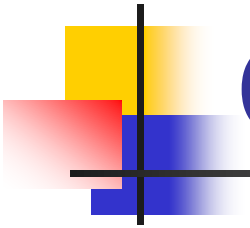
Varying task period range from $10^{0.5} \approx 3$ to $10^4 = 10{,}000$

Large range of periods
Schedulability already good without arbitrary deadlines



Summary and Conclusions

- Summary
  - AMC scheme is the most effective fixed priority pre-emptive scheduling approach for Mixed Criticality Systems
- Main contributions
  - Extended analysis for AMC to arbitrary deadlines: AMC-rtb-Arb and AMC-max-Arb analysis
  - Surprisingly, as in the constrained deadline case, the simple AMC-rtb analysis is close to AMC-max in performance
  - Evaluation shows a useful improvement in schedulability when deadlines are permitted to be arbitrary, with gains most evident when the range of task periods is relatively small e.g. factor 100 or less (as in many real systems)



Questions?

Why does AMC-rtb-Arb analysis perform so well?

- AMC-rtb-Arb analysis for the mode change

$$r_i^H(q) = (q+1)C_i(HI) + \sum_{j \in \mathbf{hpH}(i)} \left\lceil \frac{r_i^H(q)}{T_j} \right\rceil C_j(HI) +$$

$$\sum_{k \in \mathbf{hpL}(i)} \left\lceil \frac{r_i^L(\min(q,p))}{T_k} \right\rceil C_k(LO)$$

Interference from LO-criticality tasks cannot be released after completion of job p

  - Job p is the last job in the LO-criticality busy period
  - Rare that we get more than one job of task τ_i in the LO-criticality busy period
  - Hence difference between AMC-max-Arb and AMC-rtb-Arb is similar to that between AMC-max and AMC-rtb