Sistemas Distribuidos de Tiempo Real

Apuntes: TEMA 9

Por: J. Javier Gutiérrez gutierjj@unican.es
http://www.ctr.unican.es/

Grupo de Computadores y Tiempo Real, Universidad de Cantabria

Sistemas distribuidos de tiempo real

PARTE IV: Análisis de Sistemas de Tiempo Real Distribuidos

- **TEMA 8. Modelo transaccional de sistema distribuido**
- **TEMA 9. Análisis de sistemas distribuidos**

9. Analisys of Distributed Real-Time Systems

9.1 Handling Task Suspension
9.2 Analysis of End-to-End Deadlines
9.3 Holistic analysis technique
9.4 Offset-based techniques
9.5 Optimized offset-based analysis
9.6 Analysis of EDF Scheduling
9.7 Analysis of heterogeneous FP and EDF scheduling
9.8 Scheduling parameters assignment

9.1 Handling Task Suspension

In many systems a task suspends itself waiting for a response from a remote resource.

Analysis of the suspending task: we can treat the suspension interval in two different (pessimistic) ways:

- treat the suspension time as execution time,
- or, treat each portion of the response as a different event.

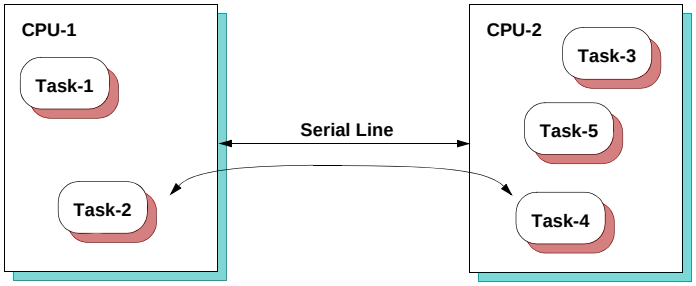
Analysis of lower priority tasks: treat each portion of the response as a different event, but:

- include the jitter effect of the second portion, or eliminate jitter.

Analysis with offsets can also be used for suspending tasks

- less pessimism introduced

Example of a Suspending Task:



Task parameters

Tasks 1, 2, 3, and 5 are periodic, and task 4 is a server task

Task	C_i	T_i	D_i	P_i
1	4	20	20	High
2	20+30	150	150	Low
3	5	30	30	High
4	15	-	-	Medium
5	100	200	200	Low

The transmission times in the serial line: 25 and 34 ms., respectively for the request and the reply message.

The end-to-end deadline for task-2 is 150 ms

Solution with Suspending Task

Analysis in CPU-1:

- **treat suspension as execution time:**

$$E_2 = \left\lceil \frac{a_k}{T_1} \right\rceil C_1 + C_{21} + M_1 + E_4 + M_2 + C_{22} = 165ms$$

- **treat each portion as a different task:**

$$E_{21} = \left\lceil \frac{a_k}{T_1} \right\rceil C_1 + C_{21}$$

$$E_{22} = \left\lceil \frac{a_k}{T_1} \right\rceil C_1 + C_{22}$$

$$E_2 = E_{21} + M_1 + E_4 + M_2 + E_{22} = 145ms$$

Notas:

The response time for task-1 is equal to C_1=4 ms.

For the response time of task-2 we must take into account the suspension time, which is the transmission time of the request message (M_1), plus the response time to task-4 (E_4), and plus the transmission time for the reply message (M_2). M_1 and M_2 are 25 and 34 ms. respectively, because the serial line is not used by any other task; otherwise a complete analysis should be performed for the communications line. E_4 is 20 ms.; this value was obtained from the analysis of CPU-2, which is shown in the following slide.

The response time for task-2 can be determined using two different approaches:

- Treating the suspension time as execution time: in this case the result is 165 ms.
- Treating each portion of task-2 as a different task, and adding the two response times plus the suspension time: in this case the result is 145 ms.

Both results may be pessimistic, i.e., they represent upper bounds to the worst-case execution time. Therefore, we can take the smallest (145 ms.) as the worst-case response time.

Solution (continued)

Analysis in CPU-2:

$$E_3 = C_3 = 5ms$$

$$E_4 = \left\lceil \frac{a_k}{T_3} \right\rceil C_3 + C_4 = 20ms$$

$$E_5 = \left\lceil \frac{a_k}{T_3} \right\rceil C_3 + \left\lceil \frac{a_k + E_{21} + M_1}{T_4} \right\rceil C_4 + C_5 = 160ms$$

The term E_{21}+M_1 incorporates the effect of jitter in the activation of task-4.

Notas:

The analysis of CPU-2 is conventional, except that task-4 is activated with jitter. This jitter has a schedulability penalty on lower priority tasks (i.e., on task-5). The amount of jitter that the activation of task-4 suffers is the variability of the first portion of task-2, and the request message. Supposing that both could sometimes be very short (near to zero) and in other occasions the worst case could happen (E_{21}=28 ms. and M_1=25 ms.), the total jitter could be 53 ms.

Notice that if we had not included the jitter effect in the equations, the result for task 5 would have been 140 ms., less that the actual worst-case that is 160 ms.

9.2. End-to-End Deadlines

It is possible to add response times in different resources to find out the worst-case end-to-end response time

The portions of the distributed response after the first are not initiated periodically. We can use two approaches:

- **Analyze the effect of jitter**
  - **the analysis is complex, because jitter in one resource affects response times in other resources, and viceversa**
- **Or eliminate jitter by using purely periodic activation or a sporadic server**

The analysis of the network is done like the analysis in a CPU, but:

- **the scheduling policy is usually different than in the CPUs**

Notas:

The main problem that appears in a distributed system when a response is comprised of portions that run in different resources, is that all portions except the first are not activated periodically. Since each portion is activated when its predecessor has finished, it has jitter in its activation, which influences the response time of lower priority tasks. We can use two approaches to handle jitter:

- Analyze the effect that jitter has on the schedulability of lower priority responses. We must take into account that the jitter in one resource influences the response times in that resource, which in turn influences the jitter in other resources. However, since the effect of increasing the jitter can only increase the response times, we can apply the schedulability equations iteratively, first assuming that there is no jitter, and then using the jitter from the previous iteration, until we reach a stable result.
- The second approach consists of eliminating jitter by activating each portion of the response at periodic intervals. These intervals are calculated so that when one portion is started, there is guarantee that the previous portion had already completed. The same effect can be used by scheduling each portion of the response with a sporadic server, that will guarantee that the effects on lower priority tasks are not worse than those of an equivalent periodic task.

The schedulability of LANs is usually determined in the same way as in a CPU: messages are treated like tasks, and message transmission times are like task execution times. However, we must be careful, because most LANs do not operate like CPUs, in a priority preemptive way. The analysis must take into account the scheduling policy of the LAN.

Analysis of End-to-End Deadlines

The analysis can be done in two different ways:

- **Holistic analysis: use the schedulability analysis for single processor systems, as if all resources were independent:**
  - **Jitter depends on response times**
  - **Response times depend on jitter**
  - **Because the dependency of the response time on jitter is monotonic, we can start with zero jitter and then iterate over the analysis until a stable solution is achieved**
  - **This analysis is pessimistic**
- **Using task offsets:**
  - **More complex analysis**
  - **Much less pessimistic**

Notas:

The analysis can be done in two different ways:

- Using the schedulability analysis for single processor systems, as if all resources were independent. The problem with this analysis is that jitter depends on response times, and the response times in turn depend on jitter. Because the dependency of the response time on jitter is monotonic, we can start with zero jitter and then iterate over the analysis until a stable solution is achieved. This analysis is pessimistic, because we are assuming that tasks encounter a critical instant in all CPUs and networks, but this may not be possible in practice because the execution of tasks is not independent.

Using task offsets:

- This is a much more complex analysis, that takes into account the interactions among tasks through the mechanism of considering task offsets. The results of this analysis are much less pessimistic than with the assumption of independent tasks.

9.3. Holistic analysis technique

Mainly developed at the University of York

Each resource is analyzed separately (CPU's and communication networks):

- **all activations after the first have "*jitter*"**
- ***jitter* in one action is considered equal to the worst-case response time of the previous actions**
- **analysis in one resource affects *response times* in other resources**
- **the analysis is repeated *iteratively*, until a stable solution is achieved**
- **the method converges because of the *monotonic* relation between jitters and response times**

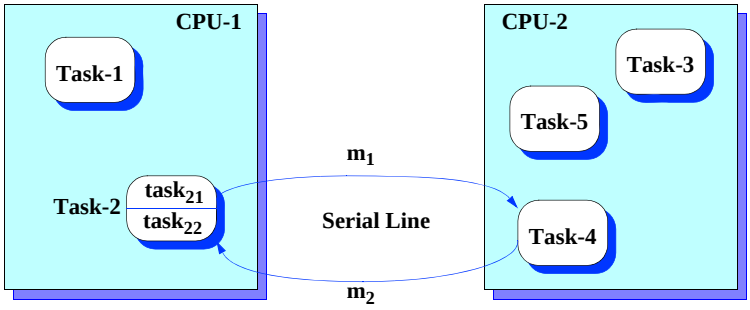
Analysis in the distributed system

```
algorithm WCRT is
begin
  initialize jitter terms to zero
  loop
    calculate worst-case response times;
    calculate new jitters, equal to response
      times of preceding actions
    exit when not schedulable or
      no variation since last iteration;
  end loop;
end WCRT
```

Assumption: $J_i = R_i - R^b_i$, R^b_i = best-case response time=0

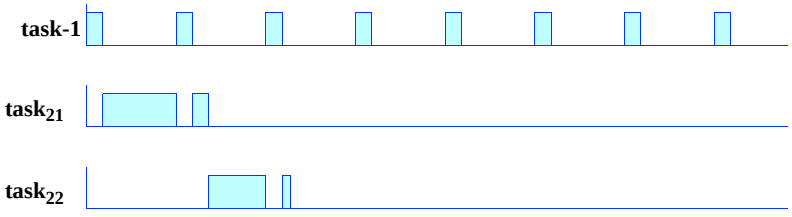
Pessimism in holistic analysis

Holistic analysis technique assumes independent task activations in each resource



Independent task analysis

Execution timeline for $task_{22}$ in previous example:



Response time for task-2:

- includes times for $task_{21}$, m_1, task-4, m_2 and $task_{22}$
- total is 270

9.4. Analysis with offsets

Developed by Tindell, at the University of York:

- The exact analysis is intractable
- An upper-bound approximation provides good results (equal to exact analysis in 93% of tested cases)

Main limitations:

- Offsets are static:
  - not applicable to general distributed transactions
- Offsets are less than the task periods:
  - for distributed systems, deadlines would need to be smaller than or equal to the task periods

Extended by Palencia (1998) to address these limitations

Offset-based techniques

The objetive is reduce the pessimism of the worst-case analysis in multiprocessor and distributed systems:

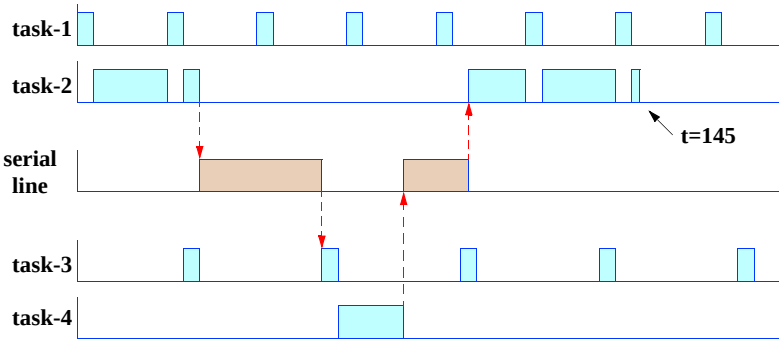
- by considering offsets in the analysis,
- offsets can be larger than task periods;
  - this is important if deadlines > task periods
- offsets can be static or dynamic:
  - offsets are dynamic in distributed systems
  - also in tasks that suspend themselves

This enhancement comes “for free”, as there is no change to the application,

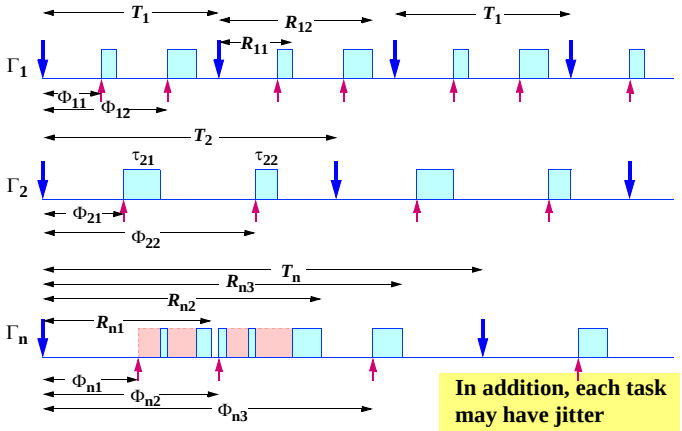
- although better results can be obtained if best-case execution times are measured

Using offsets to reduce pessimism

Execution timeline for analysis of task-2:



System model with offsets



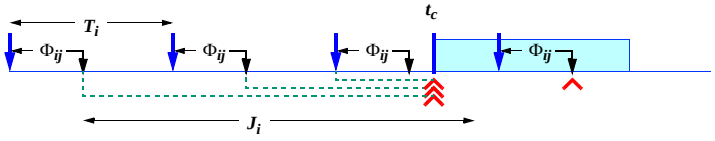
Exact analysis with static offsets

Contribution of task τ_{ij} to the response time of lower priority tasks:

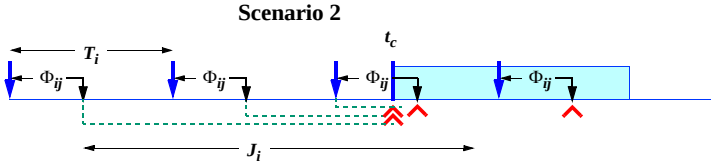
- ***Set 0***: activations that occur before the critical instant and that cannot occur inside the busy period
- ***Set 1***: activations that occur before or at the critical instant, and that may occur inside the busy period
  - **Theorem 1**: worst-case when they all occur at the critical instant
  - **Theorem 2**: worst-case when the first one experienced its maximum jitter
- ***Set 2***:activations that occur after the critical instant
  - **Theorem 1**: worst-case when they have zero jitter

Scenarios for calculating the worst-case contribution of τ_{ij}

Scenario 1



Scenario 2



Upper bound approximation for worst-case analysis

Exact analysis is intractable:

- The task that generates the worst-case contribution of a given transaction is unknown.
- The analysis has to check all possible combinations of tasks

Tindell developed an upper bound approximation:

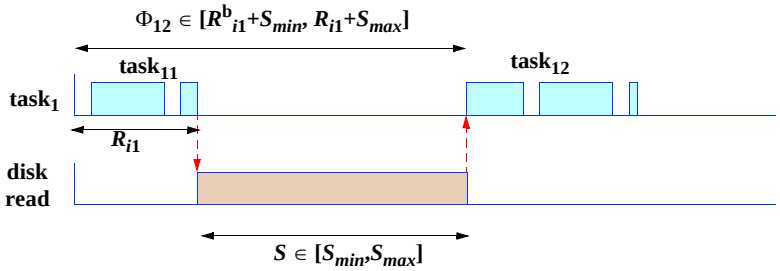
- For each transaction we consider a function that is the maximum of all the worst-case contributions considering each of the tasks of the transaction to be initiating the critical instant
- This technique is pessimistic, but polynomial
- In 93% of the tested cases, the response times were exact

Analysis with dynamic offsets

In many systems the offset may be dynamic:

$$\Phi_{ij} \in [\Phi_{ij,min}, \Phi_{ij,max}]$$

Example: task with a suspending operation



Analysis with dynamic offsets (cont'd)

Dynamic offsets can be modeled with static offsets and jitter:

- Equivalent offset:

$$\Phi'_{ij} = \Phi_{ij,min}$$

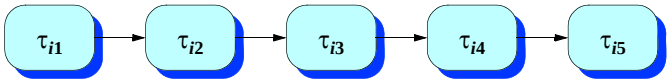
- Equivalent jitter:

$$J'_{ij} = J_{ij} + (\Phi_{ij,max} - \Phi_{ij,min})$$

The problem is that now offsets depend on response times, and response times depend on offsets

- The solution is to apply the analysis iteratively, starting with response times = zero, until a stable solution is achieved
- We call this algorithm WCDO

Distributed transaction Γ_i



Dynamic offsets in distributed transactions can also be modeled with static offsets and jitter:

- **Equivalent offset:**

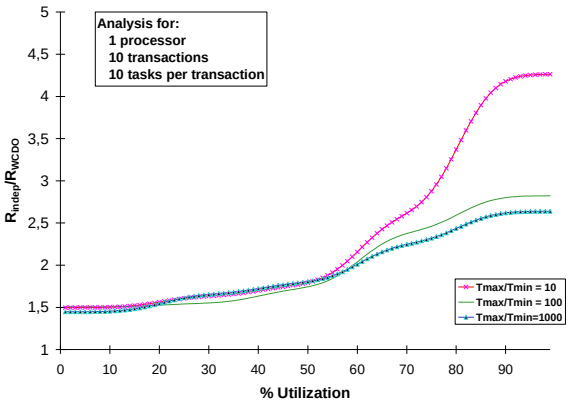
$$\Phi'_{ij} = \Phi_{ij,min} = R^b_{ij-1}$$

- **Equivalent jitter:**

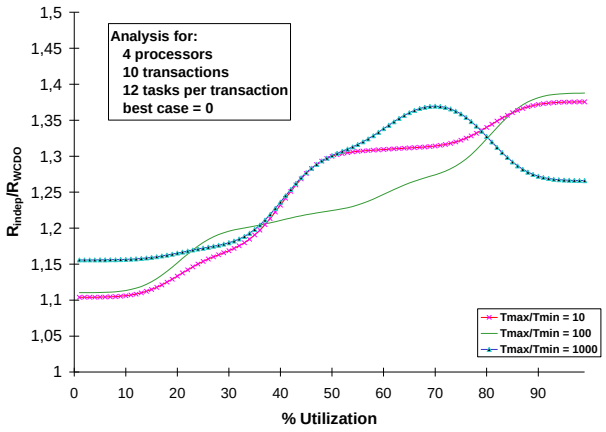
$$J'_{ij} = J_{ij} + (\Phi_{ij,max} - \Phi_{ij,min}) = R_{ij-1} - R^b_{ij-1}$$

If we apply the analysis with offsets to the suspending task of the example in slice 5, we get the following results:

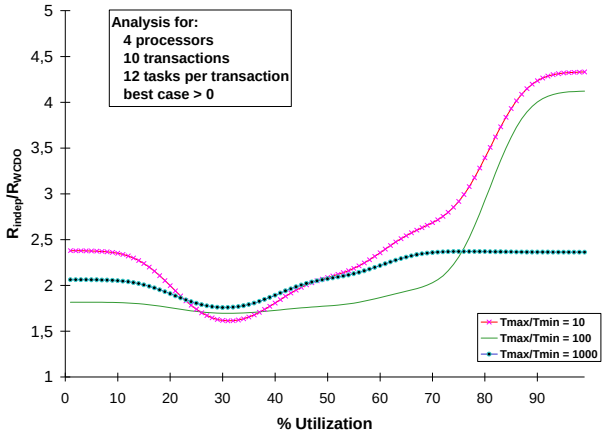
Task	D_i	R_i
1	20	4
2	150	145
3	30	5
4	-	-
5	200	140



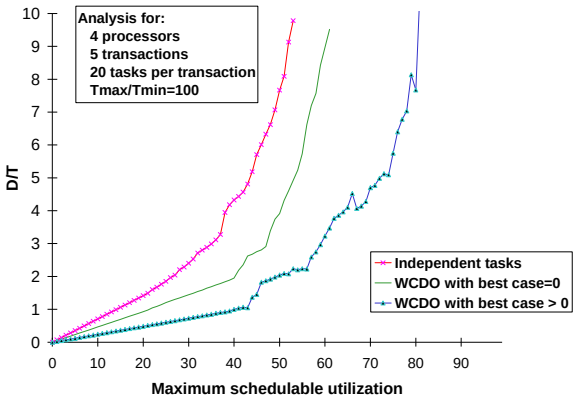
Comparison with four processors, and best-case=0



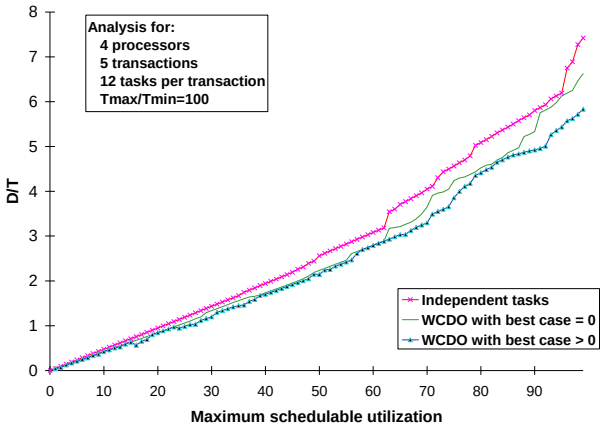
Comparison with four processors and best-case>0



Maximum utilization with 20 tasks per transaction



Maximum utilization with 12 tasks per transaction

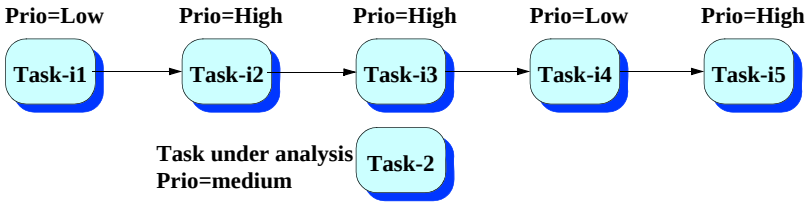


9.5. Optimized offset-based analysis

Offset-based analysis produces results that are much less pessimistic than other methods (i.e., holistic analysis)

But offset-based analysis still has room for improvement

- **a high priority task that is preceded by a low priority task may not be able to execute before a medium priority task under analysis**



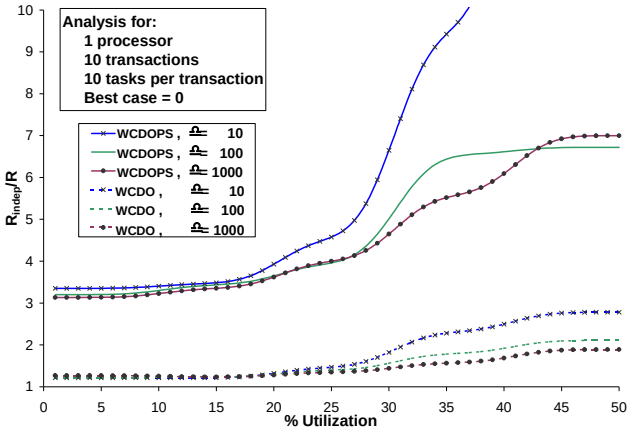
Objectives of optimized offset-based analysis

To enhance the offset-based schedulability analysis by:

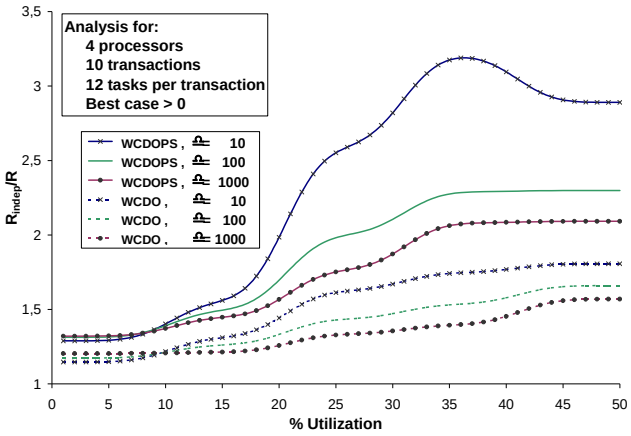
- **Eliminating from the analysis the effects of higher priority tasks that cannot execute due to precedence constraints**
- **Eliminating the effects of the tasks that are preceded by the task under analysis**

These enhancements reduce much of the pessimism in the analysis of distributed systems

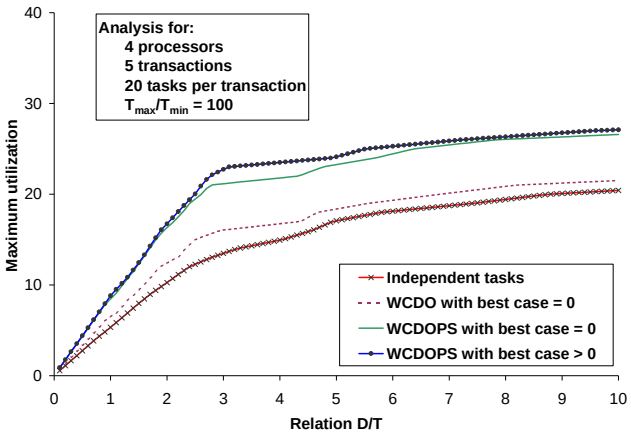
Simulation results: response times



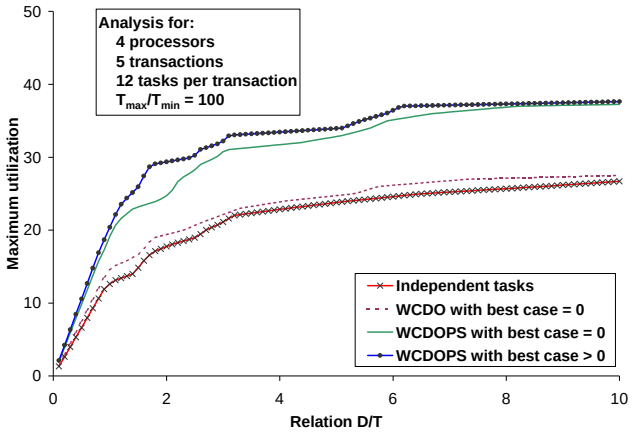
Simulation results: response times



Simulation results: utilization



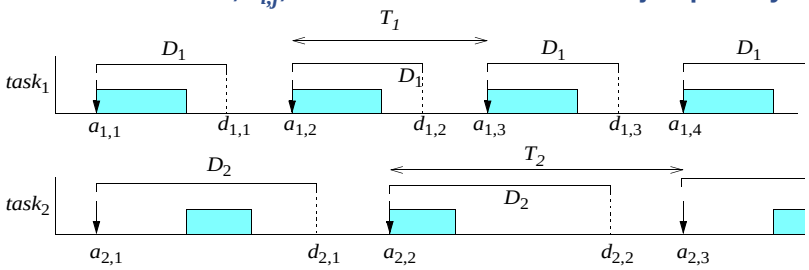
Simulation results: utilization



9.6. EDF Scheduling Policy

Each task *i* has a relative period, T_i, and a relative deadline assigned: D_i

Each task-*i* job *j* has an absolute activation time, $a_{i,j}$, and an absolute deadline, $d_{i,j}$, used as the inverse of the job priority



Optimality of EDF in monoprocessor

In a system with only periodic independent tasks, with a preemptive scheduler, executing in a single processor

- **The EDF scheduling policy is optimal (Liu & Layland, 1973)**
- **100% utilization may be achieved**

Response Time Analysis for EDF in Monoprocessor Systems

Worst case *response time* of a task: found in a busy period in which all *other* tasks:

- **are released at the beginning of the busy period,**
- **and have experienced their maximum jitter**

Differences with fixed priorities:

- **The task under analysis does not necessarily start with the busy period**
- **The busy period is longer, because it involves all tasks**

Response Time Analysis for EDF

Worst contribution of task τ_i to the busy period at time t, when the deadline of the analyzed task, τ_a, is D:

$$W_i(t, D) = min\left(\left\lceil \frac{t + J_i}{T_i} \right\rceil, \left\lfloor \frac{J_i + D - d_i}{T_i} \right\rfloor + 1\right)_0 \cdot C_i$$

Worst completion time of activation p, if first activation at A:

$$w_a^A(p) = pC_a + \sum_{\forall i \neq a} W_i(w_a^A(p), D^A(p))$$

Worst response time if first activation is A:

$$R^A(p) = w_a^A(p) - A + J_a - (p-1)T_a$$

Response time analysis in a single resource (cont'd)

Set of potential critical instants; L is the longest busy period:

$$\Psi = \cup\{(p-1)T_i - J_i + d_i\} \qquad \forall p = 1 \ldots \left\lceil \frac{L - J_a}{T_a} \right\rceil, \forall i \neq a$$

Values of A to check:

$$\Psi^* = \{\Psi_x \in \Psi \,|\, (p-1)T_a - J_a + d_a \leq \Psi_x < pT_a - J_a + d_a\}$$

$$A = \Psi_x - [(p-1)T_a - J_a + d_a]$$

Worst-case response time

$$R_a = max[R^A(p)] \qquad \forall p = 1 \ldots \left\lceil \frac{L - J_a}{T_a} \right\rceil, \forall A \in \Psi^*$$

EDF schedulers in distributed systems

Global EDF scheduling:

- each task is assigned a global scheduling deadline that is referenced to the arrival of the event that releases the end-to-end flow, possibly in a different processing resource
- it requires clock synchronization among all the processing resources involved

Local EDF scheduling:

- each task is assigned a local scheduling deadline that is referenced to the release time of its associated event in its own processing resource
- it uses the local clock of each processing resource and clock synchronization is not necessary

Holistic analysis for EDF in distributed systems

Developed by *Spuri* and suitable for global EDF schedulers

- similar to the holistic analysis developed for fixed priorities at the University of York

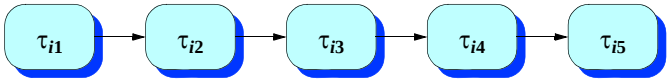
Each resource is analyzed separately (CPU's and networks):

- all activations after the first present "***jitter***"
- ***jitter*** in one action is considered equal to the worst-case response time of the previous actions
- analysis in one resource affects response times in other resources
- the analysis is repeated, until a stable solution is achieved
- the solution is pessimistic

A holistic analysis for local EDF schedulers was proposed in Rivas et al 2010

Offset-based analysis for global EDF (Palencia, 2003)

Distributed transaction Γ_i



Dynamic offsets in distributed transactions can also be modeled with static offsets and jitter:

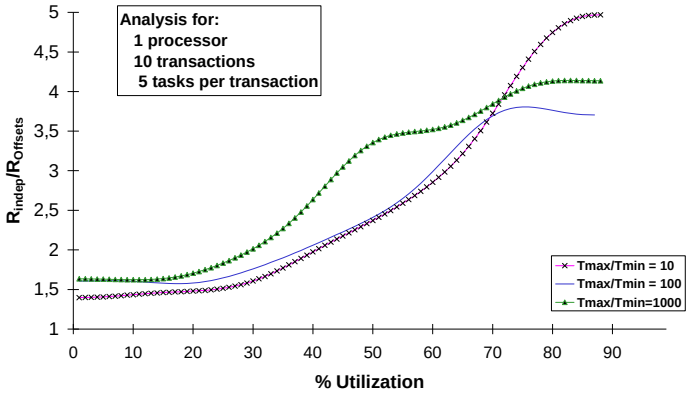
- **Equivalent offset:**

$$\Phi'_{ij} = \Phi_{ij,min} = R^b_{ij-1}$$

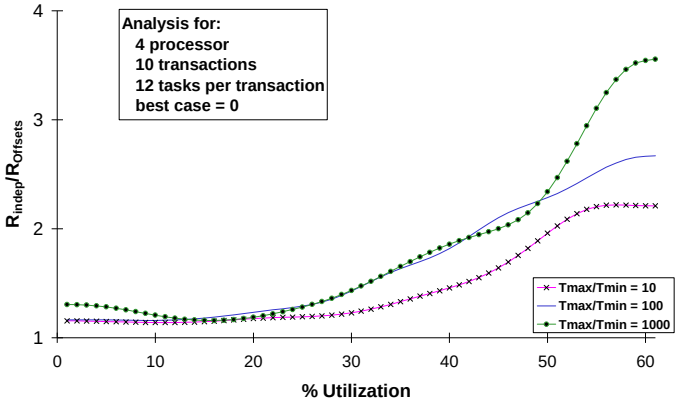
- **Equivalent jitter:**

$$J'_{ij} = J_{ij} + (\Phi_{ij,max} - \Phi_{ij,min}) = R_{ij-1} - R^b_{ij-1}$$

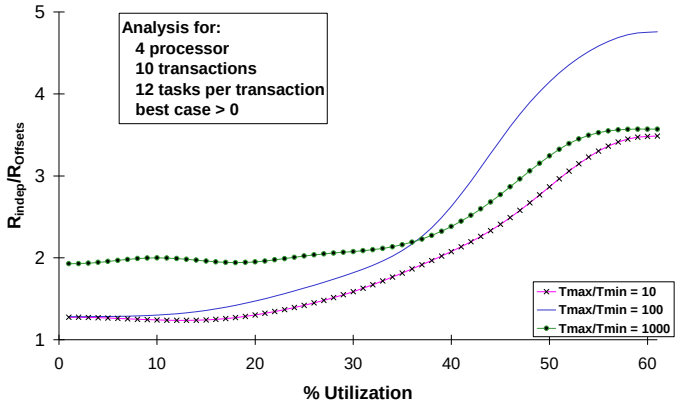
Comparison with holistic analysis: 1 processor



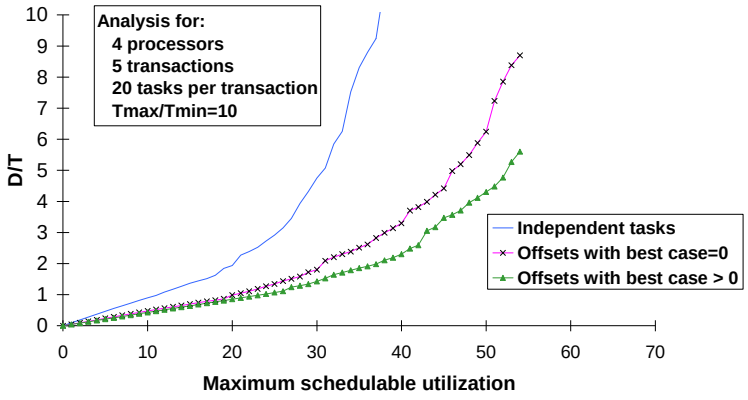
Comparison with four processors, and best-case=0



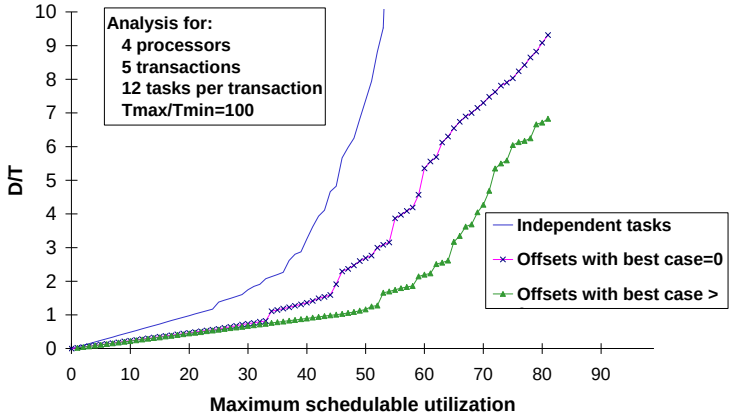
Comparison with four processors and best-case>0



Maximum utilization with 20 tasks per transaction



Maximum utilization with 12 tasks per transaction



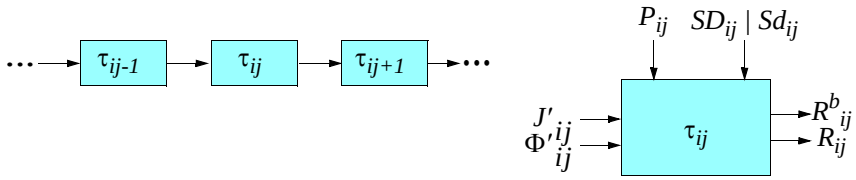
Summary of Analysis Techniques

Kind of Analysis	Deadlines	Number of processors	Fixed Priorities	Local EDF	Global EDF
`Utilization test`	D=T	1	☑		☑
`Response Time Analysis`	Arbitrary	1	☑		☑
`Holistic Analysis`	Arbitrary	Many	☑	☑	☑
`Offset-Based Analysis`	Arbitrary	Many	☑		☑

Distributed model also applicable to:

- signal & wait synchronization
- activities that suspend themselves (i.e., delays, I/O, ...)

9.7 Analysis of heterogeneous FP and EDF scheduling

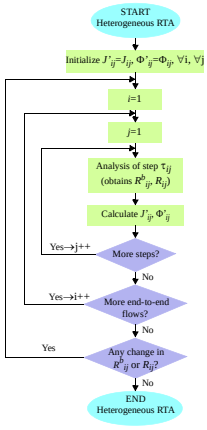


Schedulability analysis techniques for FP and EDF can be combined using equivalent jitters and offsets:

$$J'_{ij} = J_{ij} + max(R_{ij-1}, \Phi_{ij}) - max\left(R^b_{ij-1}, \Phi_{ij}\right)$$

$$\Phi'_{ij} = max\left(R^b_{ij-1}, \Phi_{ij}\right)$$

Heterogeneous RTA algorithm



9.8 Scheduling parameters assignment

PD (Proportional Deadline)

- distributes deadlines proportionally to the worst-case execution times of the steps in the end-to-end flow

NPD (Normalized Proportional Deadline)

- similar to PD but normalizing it by the utilization of the processing resources

Simulated Annealing: an optimization technique first used by Tindell, Burns, and Wellings for assigning priorities

HOSPA (Heuristic Optimized Scheduling Parameters Assignment)

- provides better and faster results for heterogeneous distributed systems, based on HOPA for fixed priotities and HOSDA for EDF

HOSPA algorithm

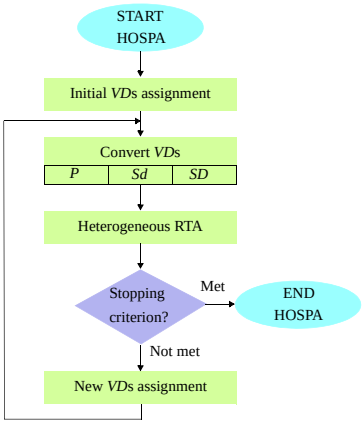
It is based on

- the distribution of the global deadlines of each end-to-end flow among the different steps that compose it, and
- the iteration over the results of RTA to redistribute these deadlines

Once each step is assigned what we will call a ***virtual deadline*** (VD), it is converted into

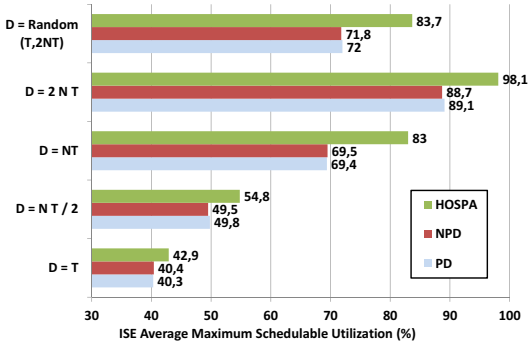
- a priority (P)
- a local scheduling deadline (Sd), or
- a global scheduling deadline (SD)

HOSPA algorithm (cont'd)



Comparison of scheduling parameters assignment techniques

Results for an example with 5 processors and 8 end-to-end flows (variable length between 1 and 5 tasks)



Fixed Priority Assignment Techniques in MAST

Technique	Single-Processor FP	Single-Processor EDF	Multi-Processor FP	Multi-Processor EDF	Multi-Processor FP/EDF
Monoprocessor	☑	☑			
PD	☑	☑	☑	☑	☑
NPD	☑	☑	☑	☑	☑
Simulated Annealing	☑		☑		
HOSPA	☑	☑	☑	☑	☑

- **Monoprocessor FP:**
  - **If deadlines within periods, DM**
  - **Otherwise, Audsley's algorithm: iteratively apply analysis, successively ordering tasks by priority: $O(n^2)$ times the analysis**