

Probabilistic Network Loads with Dependencies and the Effect on Queue Sojourn Times

Matthias Ivers and Rolf Ernst

TU Braunschweig, Institute of Computer and Communication Network Engineering,
Hans-Sommer-Strasse 66, 38106 Braunschweig
ivers@ida.ing.tu-bs.de

Abstract. For the dimensioning of shared resources, the latency and utilization of the service is a vital design characteristic. The throughput and latency is as important for e.g. network streaming applications as in e.g. (small-scale) distributed embedded systems interacting with physical processes.

Calculating latencies of a system involves the analysis of the queue sojourn times. The analysis of queue sojourn time depends on the model of the load. While for fixed and known load, natural and deterministic worst-case models are a good choice, highly variable loads are more appropriately modeled in a stochastic fashion.

For the analysis of stochastic load models, the load is often assumed to be stochastic independent and time-invariant. Analysis of loads with auto-correlation or modelling of different streams that are correlated (or dependent in general) requires a highly tuned and specialized model to capture all effects.

In this work we apply a queuing sojourn time analysis of streams with stochastic load models with upper and lower bounds guaranteed under any stochastic dependency. The experimental results show how big the effect of dependencies really is and that stochastic load dependency is vital to the calculation of resource utilization and response times (or transmission delays). We propose the use of Fréchet bounds and probability boxes to allow real-time analysis of stochastic models with unknown dependencies.

1 Introduction

Safety critical systems rely on the timely processing of all signals and guarantee a reaction before a deadline, a minimum throughput or other performance characteristics. The duration of processing is as important as the correctness of the result under hard real-time requirements. The violation of a single deadline is considered fatal. To verify the correct operation, a real-time analysis that calculates upper bounds for the system response time has to be performed. In order to allow for tractable analysis, the components of the systems are abstracted. Deterministic real-time analysis characterizes the load by an upper bound of the processing time. In uniprocessor scheduling analysis this is called the worst-case execution time (WCET). Variable execution time due to input data or state

dependency or performance variation due to unpredictable systems (e.g. out-of-order processors and caches) are not captured. Thus deterministic analysis can lead to severe overestimations as the WCET can be much higher than the average execution time.

Stochastic uniprocessor performance analysis, which were pioneered in soft or firm real time systems, tackle this pessimism by modelling the execution time of tasks more precise: instead of a single worst case execution time, a task is characterized by a series of execution times together with a probability.

The precision of the load descriptions comes with a higher computational cost and, more important, further restrictions that are imposed on the analysed system: The mathematical foundation of the proposed stochastic analyses requires the explicit knowledge of every stochastic load dependency.

We want to *assess the effects of (uncaptured) dependencies between tasks*. The dependency information between distributions is of critical importance to the correctness of the results. For researchers in risk management the effect of dependency in stochastic models is well known. Initial work by Bernat et al. recasts methods presented for financial research into the realm of worst-case execution time analysis. Our work is based on the findings presented in [2]. We published parts of the presented work also in [9].

The presented work is based on scheduling analysis for uniprocessors. The central aspect of our work, the *dependency aware analysis of stochastic models* is not at all limited to the domain of uniprocessor scheduling. The central results can be transferred quite straightforward to other domains of stochastic analysis, as our model assumes a system with a single prioritized queue, a single (preemptive) server, a deterministic arrival model and a general load model *with possible stochastic dependencies*.

2 Related Work

Timing analysis of real-time systems traditionally characterizes tasks by their period and worst case execution time (WCET). The WCET of tasks is determined by different analytical methods. Methods like [15] decompose the analysis of tasks into an analysis of the runtime of code segments without (or more advanced with a limited number of) conditional branches and the synthesis of an upper bound WCET of all possible analyzed code segments that constitute the task under analysis.

These deterministic analysis yields a single value as WCET which is true under all circumstances. This WCET is then reused in a scheduling and system analysis to calculate end-to-end reponse times. Eventually end-to-end response times are compared against the application defined deadlines. Due to the high lever of abstraction, deterministic models, that characterize system load by a single WCET value, yield poor results for applications with highly variable system loads. E.g. multimedia streaming is a well-studied application domain where the consideration of the load variance improves analysis results considerably.

Specialized load models have been suggested for the analysis of multimedia streaming applications. These stochastic models try to capture a detailed presentation of the variance. Some models are so detailed, that they are fitted to the *content* and must be adapted, when video or audio content is modified. A survey [10] lists 19 different models proposed for VBR streaming applications. The models' statistical features can vary when exchanging application content. [17] identifies critical parameters in the dependency structure of streaming video application content that statistic models traditionally used fail to model. [12] concentrates on the autocorrelation of VBR video streams and gives examples for the effects on buffer sizing. These load models are specifically tailored to match the exact application *and* analysis method. Some of these models also demand lengthy computations or excessive memory. The proposed models do consider the intra stream dependencies of the load and also consider multiple levels of auto-correlation within the stream load. On the other hand, detailed analysis of the specific load is required and even more important, the computation with these models is highly resource consuming. We aim for a compositional system level analysis and the amount of specialization required for these models does not seem suit our goals, as some parts (or the environment) of the system might not be fully known during analysis (see below for effects of unknown statistical dependencies).

Concerning the stochastic scheduling analysis for systems, frameworks such as [1, 5, 14] execution time distributions. The distribution is due to changing input data requiring different processing, due to machine state (caches, branch predictors etc) or due to error and exception handling. The model is more precise, but the proposed analysis algorithms are only applicable if the tasks are stochastic independent [3]. The requirement of stochastic independence inhibits the applicability to a bigger class of systems.

Calculations with stochastic variables with unknown dependencies have been studied extensively and are a common tool in risk theory [4]. Bernat et al. use in their approach [3] only a single possible distribution which they assume as the worst case convolution, the comonotonous convolution. Later Bernat et al. [2] state that neither the assumption of independence, nor the comonotonicity are safe approximations and propose the use of the supremal convolution, which is an upper bound on all possible convolutions. Further they use copulas to separate the modelling of stochastic behaviour of a single task and the stochastic dependency between tasks.

In this work we focus on the cause and effect of stochastic dependencies in systems of multiple, concurrent, potentially interacting tasks. We integrate the safe calculations with stochastic variables under unknown dependencies with the stochastic scheduling analysis proposed by Diaz, Kim et al. Our methods give more insight into the system reponse not only by giving a reliable upper bound on the stochastic response times, but as well a lower bound. The difference between lower and upper bound gives direct feedback about the potential impact of dependencies.

3 Motivating Example

Consider the near-empty queue displayed in figure 1 which is part of a real-time systems with deadline requirements. The queue contains two jobs $j_{0,0}$ and $j_{0,1}$ that just arrived. We want to calculate the finish time $\mathcal{F}_{0,1}$ of the second job in the queue.

Both jobs belong to a task τ_0 with the observed execution time distribution $F_{\mathcal{C}'_0}$ shown in figure 1. The finish time $\mathcal{F}_{0,1}$ is easily found to be the sum of the individual execution times $\mathcal{F}_{0,1} = \mathcal{C}_{0,0} + \mathcal{C}_{0,1}$.

As the distribution of $\mathcal{C}_{0,0}$ and $\mathcal{C}_{0,1}$ is given, we can calculate response time distribution as the sum of the execution times *assuming mutual independence*.

If, however, the execution times are not mutually independent, the assumption can lead to wrong results. We will construct two corner-cases to explain the potential errors. For simplicity both examples rely on the fact that the jobs are colored and that the color changes the odds of the execution times.

We assume *red* takes 10 time units to process and *blue* takes 2 time units to process. Further we assume that the source feeding the queue acts in a color-keeping burst mode: it emits two events of the same color and pauses for a long time.

Thus the queue can only contain two *red* or two *blue* jobs, the resulting joint distribution is shown in the middle graph of figure 2. Comparing the left and the middle distributions, we can see that the initial assumption has a 0.25 underestimation of the probability that the joint processing takes at least 20 time units.

Assuming the same setup, but with the source constantly changing color and keeping the burst length of two jobs, we get a queue which can only contain one *red* and one *blue* job. This source will have a response time distribution given on the right of figure 2. The graph shows just a single possible execution time with probability 1.0; queues which take 2 time units on the first job, take 10 time units on the second job (and vice versa). This leads to a constant sum of 12 time units. Again we can see that the initial independency assumption underestimates the

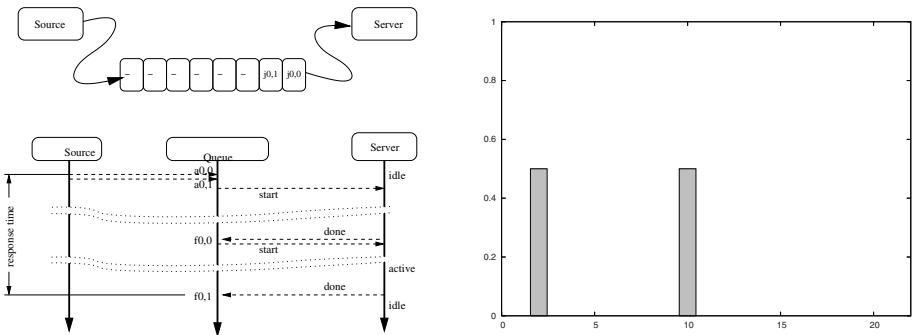


Fig. 1. Simple queue with a single task & the observed exec. time distribution

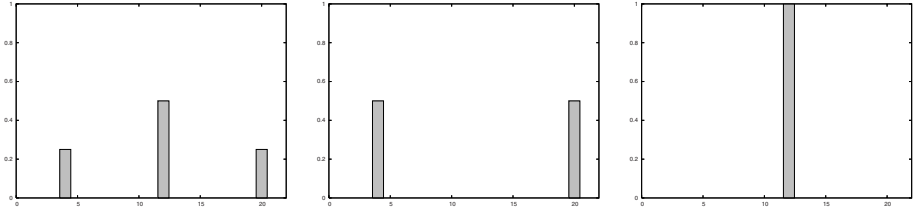


Fig. 2. $c_{0,0} + c_{0,1}$: (a) ind. (b) comonotonic (c) countermonotonic

probability: this time for the deadline of 12 time units. This difference translates directly into unsafe predictions. Assuming the system has a deadline requirement that $\mathcal{F}_{0,1}$ is below 20 time units in at least 70% of all cases ($P(\mathcal{F}_{0,1} < 20) \geq 0.70$). The first and the last graph of figure 2 do fulfill this requirement. The middle graph shows that the system would not meet the requirement $P(\mathcal{F}_{0,1} < 20) = 0.50$. The middle graph is, obviously, most pessimistic for the stated requirement. It is, however, not always most pessimistic, and thus cannot be used as *the worst case convolution*: Changing the deadline requirement to $P(\mathcal{F}_{0,1} < 10) \geq 0.25$ we can see that the first and the middle graph fulfill the requirement, but the third graph does predict system failure.

All these different scenarios are based on the same individual execution time distributions $F_{\mathcal{C}_0}$. This error in the analysis is due to the *uncaptured dependencies* between the random execution times. These dependencies have to be excluded or they have to be considered by the analysis to guarantee the correctness of the result.

4 Problem Statement

Given a set of tasks $S = \{\tau_1, \tau_2, \dots, \tau_n\}$. $\tau_i = (\mathcal{C}_i, d_i, M)$. Where \mathcal{C}_i is the execution time and d_i is the relative deadline of task τ_i and M is real number between 0 and 1. \mathcal{C}_i is a discrete random variable with probability mass function $f_{\mathcal{C}_i}(c) = P(\mathcal{C}_i = c)$ and cumulative probability function $F_{\mathcal{C}_i}(c) = P(\mathcal{C}_i \leq c)$. (Where $P(x)$ is the probability that in the specific system the event x is observed.) Furthermore a series of jobs $\tau_{i,j} = (\tau_i, a_{i,j})$ with task τ_i and an arrival time $a_{i,j}$ is given.

We assume a static priority preemptive scheduling. The tasks τ_i $i \in \mathbf{N}$ are w.l.o.g. ordered by priority. Jobs violating their deadlines are instantly killed.

For each job $\tau_{i,j}$ the response time is given by the random variable $\mathcal{R}_{i,j}$. Job $\tau_{i,j}$ is said to *fulfill its QoS requirement* (d_i, M) if $F_{\mathcal{R}_{i,j}}(d_i) \geq M$. We are seeking to find for all jobs $\tau_{i,j}$ bounds $(\overline{\mathcal{R}_{i,j}}, \underline{\mathcal{R}_{i,j}})$ to the distributions of the response time $\mathcal{R}_{i,j}$ fulfilling $\forall r \in \mathbf{N}. \overline{F_{\mathcal{R}_{i,j}}}(r) \geq F_{\mathcal{R}_{i,j}}(r) \geq \underline{F_{\mathcal{R}_{i,j}}}(r)$. These bounds should be valid *under arbitrary dependencies*.

5 Independent Jobs

First, we reproduce the results for the response time analysis of independent tasks. In the following we will extend the theory to handle unknown dependencies.

One main differentiator of the works is the definition how two random variables are added: Given two independent random variables \mathcal{X} and \mathcal{Y} the distribution of the sum $\mathcal{Z} = \mathcal{X} + \mathcal{Y}$ can be calculated as:

$$F_{\mathcal{Z}}(z) = \sum_{z=x+y} F_{\mathcal{X}}(x) * F_{\mathcal{Y}}(y)$$

This is the basic convolution that is assumed for the following works. We will later extend this addition to handle unknown dependencies.

[16] reinterpret the classical scheduling formula for probabilistic systems. For a job $\tau_{i,j} = (\tau_i, a_{i,j})$ and time interval $[a_{i,j}, a_{i,j} + t]$ let $\tau_{\chi_0}, \tau_{\chi_1}, \dots, \tau_{\chi_r}$ be the finite series of higher priority job arrivals from $a_{i,j}$ until $a_{i,j} + t$. τ_{χ_r} is the last job started before $a_{i,j} + t$. First an upper bound to the response time distribution at time t is defined:

$$\mathcal{R}_{i,j}^t = \mathcal{C}_i + \mathcal{C}_{\chi_0} + \mathcal{C}_{\chi_1} + \dots + \mathcal{C}_{\chi_r} \tag{1}$$

With \mathcal{C}_{χ_n} being the execution time distribution of τ_{χ_n} .

The probability for completion before the deadline is

$$\max\{F_{\mathcal{R}_{i,j}^t}(t) | t \in E\} \tag{2}$$

Where E is the set containing the $d_{i,j}$, the deadline of task $\tau_{i,j}$, and all arrival times of higher priority tasks before $d_{i,j}$. The authors remark the potential problems of the assumed independency in their approach.

Eq 1 is a straightforward extension of the deterministic case. It does not differentiate 'when' the interference happens. Consider a task with random execution time executing for 2 time units and then being interrupted (by a deterministic task) for 4 time units. The left part of figure 3 follows the assumption implicitly made in eq 1. The right side of the figure shows how taking the offset between the preemption and the activation into account improves the response time analysis. In an analysis of deterministic tasks this 'interference-offset' does not have to be taken into account. This can be easily seen when the distribution of the task in fig 3 is exchanged by the distribution of a deterministic task with a single step of height 1.0 at $t \geq 3$ (see fig 4).

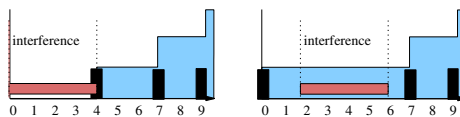


Fig. 3. Stochastic task interrupted - approximate and exact solution

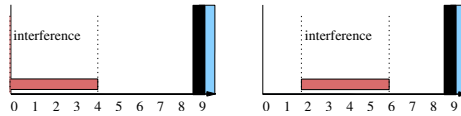


Fig. 4. Task with one possible execution time interrupted - interference offset does not matter

[5] propose the operation “convolve from r ” that respects the condition of the interference: only instances that take longer than r are interfered.

The random variable \mathcal{Y} is added if and only if \mathcal{X} exceeds r . To achieve this, $F_{\mathcal{X}}$ is split into $F_{\mathcal{X}}^{[0,r]}$ and $F_{\mathcal{X}}^{(r,\infty)}$ where $x \in I \Rightarrow F_{\mathcal{X}}^I(x) = F_{\mathcal{X}}(x)$ otherwise $F_{\mathcal{X}}^I(x) = 0$.

The sum \mathcal{Z} of \mathcal{X} from r and \mathcal{Y} under independence (written $\mathcal{Z} = \mathcal{X} +_r \mathcal{Y}$) has the distribution:

$$F_{\mathcal{Z}}(z) = F_{\mathcal{X}}^{[0,r]}(z) + \sum_{z=x+y} F_{\mathcal{X}}^{(r,\infty)}(x) * F_{\mathcal{Y}}(y)$$

The operation $+_r$ is non-associative. We define the operation $+_r$ to be left-associative. For better readability parentheses are omitted.

The response time distribution is given by

$$\mathcal{R}_{i,j} = \mathcal{C}_i +_{\delta_{\mathcal{X}_0}} \mathcal{C}_{\mathcal{X}_0} +_{\delta_{\mathcal{X}_1}} \mathcal{C}_{\mathcal{X}_1} \dots +_{\delta_{\mathcal{X}_r}} \mathcal{C}_{\mathcal{X}_r} \tag{3}$$

$\delta_{\mathcal{X}_n} := a_{\mathcal{X}_n} - a_i$ is the arrival time difference between the analyzed job $\tau_{i,j}$ and $\tau_{\mathcal{X}_n}$.

Interference which occurs only under the condition that the execution took longer than a specific time now affects only that part of the distribution. The probability for completion within the deadline is thus $F_{\mathcal{R}_{i,j}}(d_i)$.

6 Dependencies in Execution Times

The previous analysis assumed for all random variables independency. The results are only valid if we can assure for any two jobs $\tau_{i,j}, \tau_{i',j'}$ in the system that the execution time distribution of one job does not change when another job has a certain execution time ($\mathcal{C}_{i',j'} = c$).

$$\forall (i, j) \neq (i', j'). P(\mathcal{C}_{i,j} = c) = P(\mathcal{C}_{i,j} = c | \mathcal{C}_{i',j'} = c') \tag{4}$$

The term dependency is also commonly used in the sense of “one task waiting for another task”. In this work “dependency” signifies a relationship between execution times only and not a precedence relationship.

Here *stochastic dependencies* describe all remaining influences on the response time distributions ($F_{\mathcal{R}}$) once the execution time distributions ($F_{\mathcal{C}}$) and activation times are fixed. We will depict two different types of dependency. First a dependency between the jobs of a single source and then a dependency between the jobs of different sources.

6.1 Sources with History

A *intra source dependency* is a dependency between two jobs of the same 'source'. This kind of dependency typically occurs in streams of jobs which have an inherent recurring regularity. An intra source dependency exists if

$$\exists n > 0. P(\mathcal{C}_{i,j} = x \mid \mathcal{C}_{i,j-n} = y) \neq P(\mathcal{C}_{i,j} = x) \quad (5)$$

An intra source dependency exists whenever the execution time of a job depends on the execution time of previous jobs.

An example for this scenario is the processing of MPEG streams. MPEG streams consist of a series of so-called I-, B-, and P-Frames. An I-Frame will typically be followed by a number of B- and P-Frames. Conversely the occurrence of two consecutive I-Frames is rare. As the different types of frames have very different transmission lengths and execution times associated with them, this qualifies as an intra source dependency.

In deterministic performance analysis for MPEG stream processors, these streams are modelled in a context-aware fashion. This means that the analysis is 'application-aware' and can handle this situation [8].

Another example for an intra source dependency, which is not due to the application model, are systems with caches: Consider two jobs executed back-to-back which require the same processing. The first invocation can suffer a cache miss, executes a long time while loading data into the cache. The second invocation, accessing the same data, will *not* suffer this cache miss and thus will execute faster. In this situation, the long execution of the first run will not be followed by another long execution.

6.2 Synchronized Sources

We speak of an *inter source dependency* if two or more sources change the load characteristic at the same time. An inter source dependency exists if there are two jobs $(\tau_{i,j}, \tau_{i',j'})$ with

$$P(\mathcal{C}_{i,j} = x \mid \mathcal{C}_{i',j'} = y) \neq P(\mathcal{C}_{i,j} = x) \text{ with } (i, j) \neq (i', j') \quad (6)$$

As an example consider variable bitrate video streams with isolated processing of audio and video. Streaming rates depend on the 'level of action' inside a stream. Calm scenes are probably accompanied by calm sounds and scenes with higher activity typically have high activity at both levels, audio and video. If one task is 'nearly idle', the other is probably as well. This is an inter source dependency.

Another example is a system with exclusive-or style load balancing between two tasks. It is not uncommon for two system functions to be designed in such a way that only one task is highly loaded at a time, while the other task (sharing the same processing resource) is nearly idle. This happens for example if in a multimedia stream processing different codecs are implemented in different tasks. As only one codec is active at a time, only this will generate high load. The unused codecs will generate no load.

6.3 Modeling of Dependency

A colored task $\tau_i = (\mathcal{C}_{k_0}, \mathcal{C}_{k_1}, \mathcal{C}_{k_2}, \dots, d_i)$ is a deadline d_i , a set of colors k_0, k_1, \dots and for each color a random response time \mathcal{C}_{k_0} . A colored job $\tau_{i,j} = (\tau_i, \alpha_{i,j}, k_{i,j})$ is a task with an activation time $\alpha_{i,j} \in \mathbf{N}$ and colored token $k_{i,j} \in \mathbf{K}$. The execution time $\mathcal{C}_{i,j}$ of task $\tau_{i,j}$ is chosen as determined by the color $k_{i,j}$. No other data than the color changes the odds of the random execution time $\mathcal{C}_{i,j}$. More formally

$$P(\mathcal{C}_{i,j} = c \mid k_{i,j} = k) = P(\mathcal{C}_{i,j} = c \mid k_{i,j} = k \vee \phi) \tag{7}$$

where ϕ is any formula which does *not* contain $\mathcal{C}_{i,j} = c'$. Using this model, we consider the job source as the generator of stochastic dependencies within the system. This model holds for a big class of systems and resembles multi-modal tasks in deterministic models.

6.4 Simulation of Effect

A simple example of a series of task activations with response time dependency will clarify the use of the model:

A single task t_1 with two possible execution times is periodically activated. The task has an arbitrary high deadline, and yet unprocessed tasks are stored in an unbounded queue. The execution time $c_{1,n}$ of job $t_{1,n}$ is determined by the following formula: $c_{1,n} = t_{1,off} + t_{1,mul} * x$. Where $t_{1,off}$ and $t_{1,mul}$ are non-negative task parameters and $x \in \{0, 1\}$ is a random variable (the color). Furthermore $t_{1,off} \geq t_{1,mul}$. The random variable x is determined by a source with one of the following strategies:

- independence: x changes from 0 to 1 (or vice versa) with a probability of 0.5
- positive dependence: x changes with a probability of 0.1
- negative dependence: x changes with a probability of 0.9

The tasks' execution time distribution $F_{\mathcal{C}_n}$ is identical for all dependencies. The resulting response-time has been plotted in figure 5. The figure shows the measured CDFs for the three scenarios. Comparing the three scenarios from left to right, you can identify that the end-to-end response time is dominated by the dependency of the tasks. The example demonstrates that a dependency has a significant effect on the response time distribution and should be taken into account.

The task's execution time distribution $F_{\mathcal{C}}$ is identical for all cases. The resulting response-time of the first 500 simulated tasks has been plotted in the bottom part of figure 5. The three lower graphs show the the execution time on the y-axis and the invocation number on the x-axis. The top part of the figure shows the measured CDFs for the three scenarios.

Comparing the three scenarios from left to right, it can be seen, that the backlog of the queues dominate the response time. The example demonstrates that an intra source dependency has a significant effect on the response time distribution and should be taken into account.

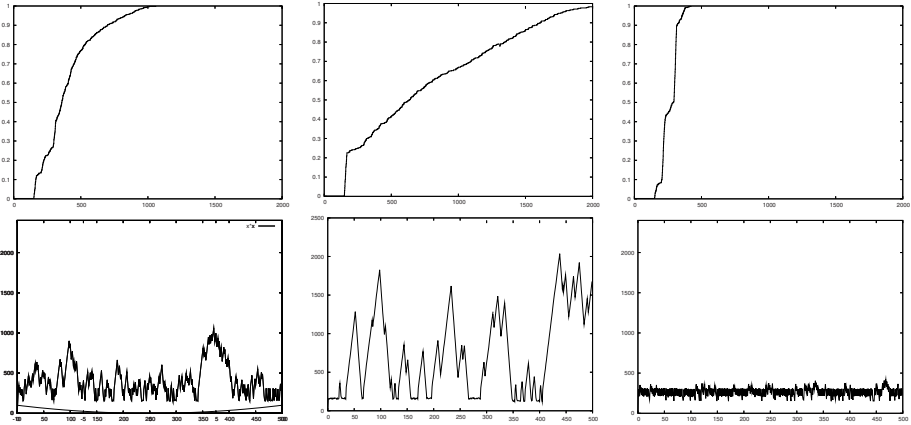


Fig. 5. Simulated response-time with (a) independent (b) + dependent and (c) - dependent tasks

7 Taking Dependencies into Account

Analyzing a job $\tau_{i,j}$, we have to consider all intra source dependencies for jobs of *higher priority* which can be activated multiple times before the deadline d_i (i.e. jobs of smaller period than τ_i). Additionally for all concurrently running higher priority jobs, we have to consider a potential inter source dependency for the sum of jobs of different priorities.

For this we have to use distribution functions that are able to represent uncertainty (i.e. distribution functions which can bound the probability for the value to be within a certain *interval*). To reason about interval bounds of stochastic variables, we will introduce so-called probability boxes.

7.1 Probability Bounds and Probability Boxes

A probability box (p-box) [6] is the generalisation of interval arithmetic in the realm of distribution functions. Given a cumulative distribution $F_X : \mathbf{N} \rightarrow [0, 1]$, a p-box is a pair of functions $\underline{F}_X, \overline{F}_X : \mathbf{N} \rightarrow [0, 1]$ with

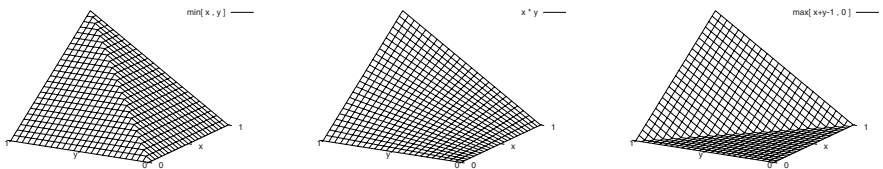


Fig. 6. M , H and W copulas

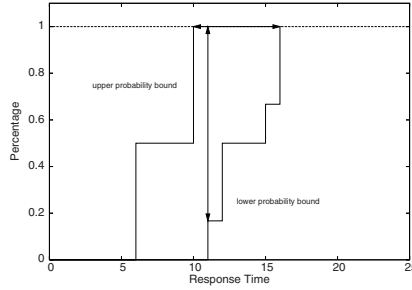


Fig. 7. A Probability Box

$$\underline{F}_{\mathcal{X}}(x) \leq F_{\mathcal{X}}(x) \leq \overline{F}_{\mathcal{X}}(x) \tag{8}$$

For any $F_{\mathcal{X}}$, $(\overline{F}_{\mathcal{X}}, \underline{F}_{\mathcal{X}}) := (F_{\mathcal{X}}, F_{\mathcal{X}})$ is a bounding p-box. Execution time distributions $F_{\mathcal{X}}$ map a time t_x to the probability that the execution time is less than or exactly t_x time units. Execution time p-boxes $(\overline{F}_{\mathcal{X}}, \underline{F}_{\mathcal{X}})$ map a time t_x to a minimum and maximum probability that the execution time is less than or exactly t_x time units (the interval for $t_x = 11$ is marked by the vertical arrow in figure 7). Formally, a p-box satisfies

$$\underline{F}_{\mathcal{X}}(x) \leq P(\text{task finished before } t_x) \leq \overline{F}_{\mathcal{X}}(x) \tag{9}$$

Where $P(x)$ is the probability that in the specific system the event x is observed.

Probability boxes can also be used to read an execution time interval for a given probability. E.g. the horizontal arrow in figure 7 shows that the WCRT (the response time with an accumulated probability of 100%) ranges from 10 to 16 time units. These values say nothing about the *best-case* response time; the meaning is that there may exist a system with a *worst-case* response time of only 10 time units. The *best-case* response time is found in the graph at the points where the probability bounds leave the 0%-line. The *best-case* is guaranteed to be between 6 and 11 time units.

Probability bounds can be efficiently described with probability boxes. The remaining question is how the sum of stochastic variables should be safely calculated. [7, 13, 18] study arithmetic on stochastic variables with unknown dependencies.

Copulas are used to formalize dependencies between stochastic variables. Copulas model the relation between (typically available) marginal distributions and their joint distribution. i.e.: Given a two-dimensional distribution function $F_{\mathcal{F}\mathcal{G}}(x, y)$ with (one-dimensional) marginals $F_{\mathcal{F}}(x)$ and $F_{\mathcal{G}}(y)$. Then there exists a copula C such that

$$\mathcal{H}(x, y) = C(\mathcal{F}(x), \mathcal{G}(y)) \tag{10}$$

In our situation, 2 marginals $F_{\mathcal{F}}$ and $F_{\mathcal{G}}$ for different tasks/jobs are given. The unknown dependency is modeled only by C .

Obviously C is a function $C : [0, 1] \times [0, 1] \rightarrow [0, 1]$. Furthermore it has been proven that all copulas satisfy

- $C(a, 0) = C(0, a)$ and $C(a, 1) = C(1, a)$ for all $a \in [0, 1]$
- they are 2-increasing: i.e. $C(a_2, b_2) - C(a_1, b_2) - C(a_2, b_1) + C(a_1, b_1) \geq 0$ for all $a_1 \leq a_2, b_1 \leq b_2$.

Given these constraints, there exists a unique smallest and a unique largest copula. Namely $W(a, b) := \max(a + b - 1, 0)$ and $M(a, b) := \min(a, b)$. These copulas are handy as all copulas satisfy

$$W(a, b) \leq C(a, b) \leq M(a, b) \tag{11}$$

This observation lead to the Fréchet bounds, that give upper and lower bounds on the effect the dependency between marginals can have on the joint distribution:

$$\max(F_{\mathcal{F}}(x) + F_{\mathcal{G}}(y) - 1, 0) \leq F_{\mathcal{H}}(x, y) \leq \min(F_{\mathcal{F}}(x), F_{\mathcal{G}}(y)) \tag{12}$$

Another commonly used copula is $\Pi(x, y) := x * y$ which models stochastic independency of two random variables. M, W and Π are shown in figure 6.

Copulas and especially the W copulas give lower bounds for the probability $P(\mathcal{X} \leq x_1 \wedge \mathcal{X} \leq y_1)$ given the probability for $P(\mathcal{X} \leq x_1)$ and $P(\mathcal{X} \leq y_1)$. This is closely related to the sum of random variables, yet a little extension is necessary as we are not interested in the probability of $P(\mathcal{X} \leq x_1 \wedge \mathcal{X} \leq y_1)$, but instead we search for any given z_1 the probability $P(\mathcal{X} + \mathcal{Y} \leq z_1)$.

[18] gives probability bounds for the sum of two stochastic variables $\mathcal{Z} = \mathcal{C}_1 + \mathcal{C}_2$.

$$\overline{F}_{\mathcal{Z}}(t) = \inf_{c_1+c_2=t} \{W^d(F_{\mathcal{C}_1}(c_1), F_{\mathcal{C}_2}(c_2))\} \tag{13}$$

$$\underline{F}_{\mathcal{Z}}(t) = \sup_{c_1+c_2=t} \{W(F_{\mathcal{C}_1}(c_1), F_{\mathcal{C}_2}(c_2))\} \tag{14}$$

Where $W^d(x, y) := x + y - W(x, y) = \min(u + v, 1)$ is the dual of W . In order to consider sums of p-boxes (instead of distribution functions as above) the function is extended to p-boxes. The sum of two stochastic variables described by probability boxes $(\overline{F}_{\mathcal{Z}}, \underline{F}_{\mathcal{Z}}) = (\overline{F}_{\mathcal{C}_1}, \underline{F}_{\mathcal{C}_1}) + (\overline{F}_{\mathcal{C}_2}, \underline{F}_{\mathcal{C}_2})$ is calculated as follows:

$$\overline{F}_{\mathcal{Z}}(t) = \inf_{c_1+c_2=t} \{\min(\overline{F}_{\mathcal{C}_1}(c_1) + \overline{F}_{\mathcal{C}_2}(c_2), 1)\} \tag{15}$$

$$\underline{F}_{\mathcal{Z}}(t) = \sup_{c_1+c_2=t} \{\max(\underline{F}_{\mathcal{C}_1}(c_1) + \underline{F}_{\mathcal{C}_2}(c_2) - 1, 0)\} \tag{16}$$

The proposed operation for the addition of two random variables has been proven to be safe and furthermore pointwise best-possible [18]. That means for any bound tighter than $(\overline{F}_{\mathcal{Z}}, \underline{F}_{\mathcal{Z}})$, one can find a dependency between the underlying random processes, that would lead to a violation of these (tighter) bounds.

7.2 Adaptions to Scheduling Analysis

We will now integrate Fréchet bounds and probability boxes into the scheduling analysis formula using the “convolve from” operation. In order to do that, we basically have to lift the “convolve from” operation from simple distributions to probability boxes. I.e. the inputs of the convolve from operation are now probability boxes, and the output of the convolve from operation are probability boxes as well.

As this lifting from distributions to probability boxes is intriguingly conclusive, we show directly the “convolve from” operation using probability boxes and the Fréchet bounds:

The sum of $(\overline{F_X}, \underline{F_X})$ from r and $(\overline{F_Y}, \underline{F_Y})$ under any dependence is bounded by $(\overline{F_Z}, \underline{F_Z}) = (\overline{F_X}, \underline{F_X}) +_r (\overline{F_Y}, \underline{F_Y})$ is defined:

$$\overline{F_Z}(z) = \overline{F_X}^{[0,r]}(z) + \inf_{z=x+y} \{ \min(\overline{F_X}^{(r,\infty)}(x) + \overline{F_Y}(y), 1) \} \quad (17)$$

$$\underline{F_Z}(z) = \overline{F_X}^{[0,r]}(z) + \sup_{z=x+y} \{ \max(\underline{F_X}^{(r,\infty)}(x) + \underline{F_Y}(y) - 1, 0) \} \quad (18)$$

The operation $+_r$ is non-associative. We define the operation $+_r$ to be left-associative. For better readability parentheses are omitted.

The response time distribution is given by

$$(\overline{F_{\mathcal{R}_{i,j}}}, \underline{F_{\mathcal{R}_{i,j}}}) = F_{c_i} +_{\delta_{x_0}} F_{c_{x_0}} +_{\delta_{x_1}} F_{c_{x_1}} \dots +_{\delta_{x_r}} F_{c_{x_r}} \quad (19)$$

δ_{t_n} is the arrival time difference between the analyzed task and t_n .

The equation is well defined, as we can write F_{c_i} for the bounding p-box $(\overline{F_{c_i}}, \underline{F_{c_i}})$ with $\overline{F_{c_i}} = \underline{F_{c_i}} = F_{c_i}$. Following our previous reasonings about the fréchet bounds we can see that

$$\forall t. \overline{F_{\mathcal{R}_{i,j}}}(t) \geq F_{\mathcal{R}_{i,j}}(t) \geq \underline{F_{\mathcal{R}_{i,j}}}(t) \quad (20)$$

Using this function for our scheduling analysis, we calculate safe and sharp bounds to the distributions. Additionally, we also get a notion of how much the missing dependency information is affecting the system, as we also calculate a lower bound.

8 Comparison

Kim, Diaz, et. al. present an scheduling analysis for priority-driven periodic real-time systems [11]. They give an example for the calculation of the response time distribution of a job based on a given interarrival pattern of different jobs and the job’s execution time given as a distribution. The jobs are assumed to be mutually independent in their execution time. The initial workload is assumed to be zero. Figure 9 presents a timeline with task activations and the ETPDFs of t_1 , t_2 and t_3 . Six graphs below the timeline show the remaining workload distribution at different times.

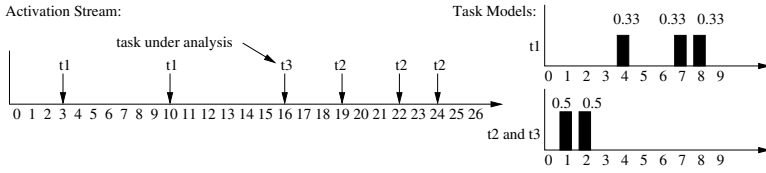


Fig. 8. Activation Diagram and Task Models

By time unit (tu) 3, task t_1 is activated: as the current workload is zero, the workload becomes exactly the task profile of t_1 . All graphs show results of Diaz’ original analysis and the proposed p-box based analysis. Diaz’ result is always within the p-box (the proposed method). In the first graph only a single line is visible, as both approaches are equal at 3 tu.

Until the next activation at 10 tu, no task is activated. To model the passing of time, the graph is *shrunked* i.e. shifted to the left and all probabilities for response times below zero are accumulated at 0. For details on *shrinking* see [11] or the examples below. Now, at 10tu, a second job starts. The task execution time distribution is added to the workload distribution. The result is presented in the second graph.

The third graph shows the workload distribution by the activation time of the task under analysis (16 tu). The workload distribution is composed of t_3 ’s own execution time distribution and the backlog distribution of the previous invocations.

The next three graphs show the workload distribution of the task t_3 at 16 time units considering the preemption at 16+3 (fourth graph), 16+6 (fifth graph) and 16+8 (sixth graph). The graphs are no longer shrunked, instead the activation time of the interrupting tasks is added starting from 3, 6 or 8 time units. The last graph shows the actual response time distribution of this invocation of t_3 (including all possible preemptions). This last graph is *the job response time*.

The big height of the p-box demonstrates how sensitive the system is to execution time dependencies. Remarkably the result assuming independence is quite far away from the ‘worst-case’ p-box bound. As the frechet bounds used to calculate the p-box are known to be sharp, we can assure that a wrongly assumed independency can practically lead to misleading optimistic results.

Two specific examples give insight how the dependencies might look like, that produce this big difference.

8.1 $x=3$ $y=1.0$ Example

Figure 10 shows the changing workload distribution on the left and arriving tasks on the right. All distributions are aggregated by colors ‘a’, ‘b’, ‘c’ showing the dependencies between the different task activations. As t_3 finishes within 3 timeunits, we do not give graphs for the succeeding invocations of t_2 .

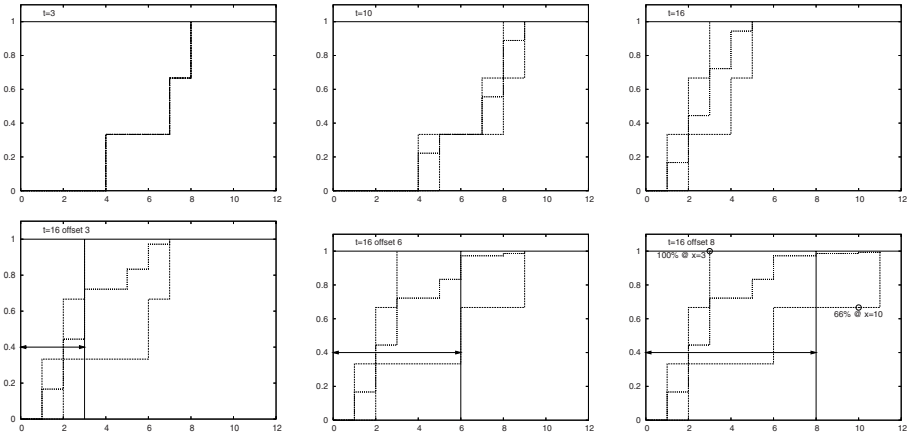


Fig. 9. First steps of Kim and Diaz' example (at time: 3, 10, 16, 16+3, 16+6, 16+8)

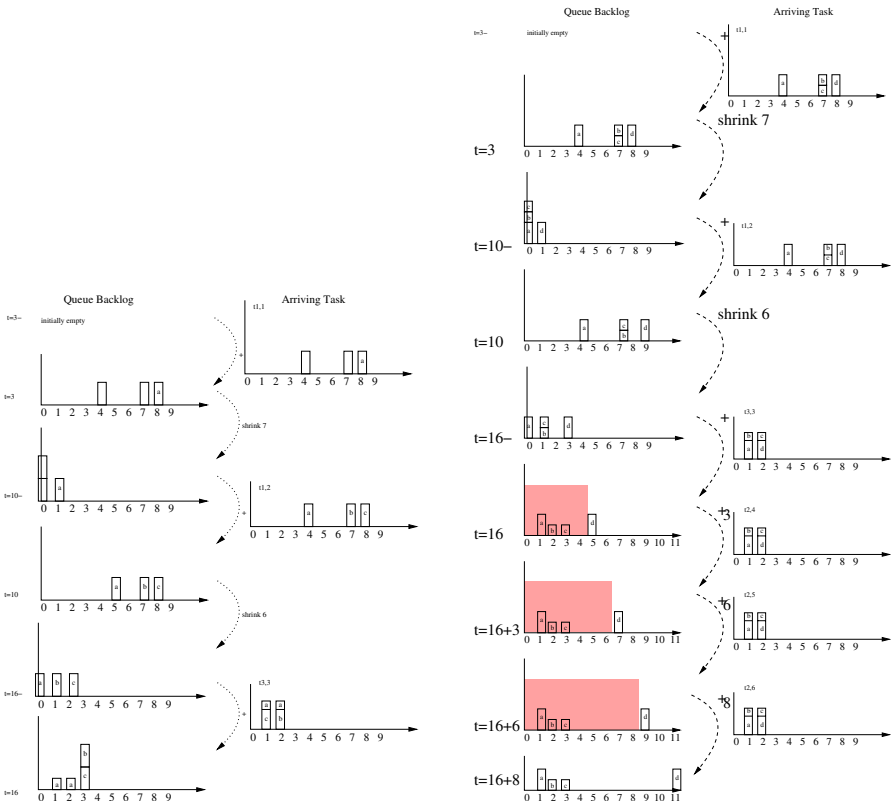


Fig. 10. Dependency pattern for $x=3, y=1.0$ (left) and for $x=10, y=0.66$ (right)

- Color 'a': if $t_{1,1}$ takes 8 time units, $t_{1,2}$ takes 4 time units. 15 tus after the activation of $t_{1,1}$, at the arrival time of $t_{3,1}$, no backlog is in the run-queue. $t_{3,1}$ is guaranteed to complete within 2 time units.
- N.B.: if $t_{1,1}$ takes less than 8 time units, the run-queue is empty at the arrival time of $t_{2,2}$. Thus $t_{1,1}$ only has to be considered if it executes 8 time units.
- Color 'b': if $t_{1,2}$ executes for 7 time units, $t_{3,3}$ executes for 2 time units. At the arrival time of $t_{3,3}$ the backlog will be one time unit leading to a total execution time of 3 time units.
- Color 'c': if $t_{1,2}$ executes for 8 time units, $t_{3,3}$ executes for 1 time units. At the arrival time of $t_{3,3}$ the backlog will be two time units leading to a total execution time of 3 time units.

The given dependency leads to $\mathcal{R}(3) = 1.0$. Under independence the guarantee is worse with $\mathcal{R}(3) \approx 0.722$.

8.2 $x=10$ $y=0.66$ Example

Figure 10 shows the queue distribution with the dependency to the new activations. For the distributions at $t = 16 + 3$ and beyond, only the case 'd' of the graph is affected, as this is the only case which is inside the manipulated part of the queue distribution ($+_n$ only affects the part of the distribution which is $> n$).

The given dependency leads to $\mathcal{R}(10) \approx 0.667$. Under independence the guarantee is far more optimistic with $\mathcal{R}(10) \approx 0.993$.

9 Conclusion

We demonstrated the effect of uncaptured dependencies in stochastic system analysis and have introduced probability boxes to describe uncertain probabilities instead of resorting to comonotonicity as a worst-case measure. Our examples demonstrate the magnitude of the effect and that neither the assumption of independence, nor the assumption of comonotonicity leads to safe estimations of the system behaviour.

Using our frechet-bound based 'convolve from' operation, we generate safe and sharp bounds for the analysis results. Using these methods, we can construct a compositional system analysis. Furthermore by the calculation of the upper and lower bound, we give a measure of the expected variation due to unknown dependencies.

References

1. Atlas, A.K., Bestavros, A.: Statistical rate monotonic scheduling, pp. 123–132. Boston University, Computer Science Department (1998)
2. Bernat, G., Burns, A., Newby, M.: Probabilistic timing analysis: An approach using copulas. *J. Embedded Comput.* 1(2), 179–194 (2005)

3. Bernat, G., Colin, A., Petters, S.M.: Wcet analysis of probabilistic hard real-time systems. In: Proceedings of the 23rd Real-Time Systems Symposium, RTSS 2002, pp. 279–288 (2002)
4. Chebyshev, P.: Sur les valeurs limites des integrales. *Journal de Mathematiques Pures Appliques* (1874)
5. Díaz, J.L., García, D.F., Kim, K., Lee, C.-G., Bello, L.L., López, J.M., Min, S.L., Mirabella, O.: Stochastic analysis of periodic real-time systems. In: RTSS 2002: Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS 2002), Washington, DC, USA, p. 289. IEEE Computer Society, Los Alamitos (2002)
6. Ferson, S., Kreinovich, V., Ginzburg, L., Myers, D.S., Sentz, K.: Constructing probability boxes and dempster-shafer structures. Technical report, Sandia National Laboratories (2002)
7. Frank, M., Nelsen, R., Schweizer, B.: Best-possible bounds for the distribution of a sum- a problem of kolmogorov. *Prob. Theory Related Fields* (1987)
8. Henia, R., Hamann, A., Jersak, M., Racu, R., Richter, K., Ernst, R.: System level performance analysis - the symta/s approach. In: IEE Proceedings Computers and Digital Techniques (2005)
9. Ivers, M., Ernst, R.: Effect of Stochastic Load Dependencies on Queue Sojourn Times. In: Proceedings of 4th International Workshop on Real-Time Software (2009)
10. Izquierdo, M.R., Reeves, D.S.: A survey of statistical source models for variable bit-rate compressed video. *Multimedia Systems* 7, 199–213 (1999)
11. Kim, K., Diaz, J.L., Lopez, J.M., Lo Bello, L., Lee, C.-G., Min, S.L.: An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Trans. Comput.* 54(11), 1460–1466 (2005)
12. Krunz, M.: The correlation structure for a class of scene-based video models and its impacts on the dimensioning of video buffers. *IEEE Trans. Multimedia* 2, 27–36 (2000)
13. Makarov, G.: Estimates for the distribution function of a sum of two random variables when the marginal distributions are fixed. *Theory Probab. Appli.* (1981)
14. Manolache, S.: Analysis and optimisation of real-time systems with stochastic behaviour. PhD thesis, Linköpings universitet – Institute of Technology (2005)
15. Staschulat, J., Ernst, R., Schulze, A., Wolf, F.: Context sensitive performance analysis of automotive applications. In: DATE 2005: Proceedings of the conference on Design, Automation and Test in Europe, Washington, DC, USA, pp. 165–170. IEEE Computer Society, Los Alamitos (2005)
16. Tia, T.-S., Deng, Z., Shankar, M., Storch, M., Sun, J., Wu, L.-C., Liu, J.-S.: Probabilistic performance guarantee for real-time tasks with varying computation times. In: Real-Time and Embedded Technology and Applications Symposium, p. 164. IEEE, Los Alamitos (1995)
17. Varatkar, G., Marculescu, R.: Traffic analysis for on-chip networks design of multimedia applications. In: DAC 2002: Proceedings of the 39th conference on Design automation, pp. 795–800. ACM, New York (2002)
18. Williamson, R.C., Downs, T.: Probabilistic arithmetic. i. numerical methods for calculating convolutions and dependency bounds. *Int. J. Approx. Reasoning* 4(2), 89–158 (1990)