

Don't Just Go With the Flow: Cautionary Tales of Fluid Flow Approximation

Alireza Pourranjbar¹, Jane Hillston¹, and Luca Bortolussi^{2,3}

¹ LFCS, School of Informatics, University of Edinburgh, UK.

² DMG, University of Trieste, Italy.

³ CNR/ISTI, Pisa, Italy.

Abstract. Fluid flow approximation allows efficient analysis of large scale PEPA models. Given a model, this method outputs how the mean, variance, and any other moment of the model's stochastic behaviour evolves as a function of time. We investigate whether the method's results, i.e. moments of the behaviour, are *sufficient* to capture system's actual dynamics.

We ran a series of experiments on a client-server model. For some parametrizations of the model, the model's behaviour can accurately be characterized by the fluid flow approximations of its moments. However, the experiments show that for some other parametrizations, these moments are not sufficient to capture the model's behaviour, highlighting a pitfall of relying only on the results of fluid flow analysis. The results suggest that the sufficiency of the fluid flow method for the analysis of a model depends on the model's concrete parametrization. They also make it clear that the existing criteria for deciding on the sufficiency of the fluid flow method are not robust.

1 Introduction

One of the features of Performance Evaluation Process Algebra, or PEPA, is that the concise set of formal primitives it provides is rich enough to be used to model a wide range of systems [1–4]. In general, analysis of a PEPA model involves deriving the model's underlying Continuous Time Markov Chain (CTMC) and applying Markovian analysis. When PEPA is used to model a large scale system, i.e. a system with large populations of entities, the size of the underlying CTMC becomes so large that Markovian analysis becomes expensive, time consuming or even, due to the memory constraints, practically infeasible; a famous problem referred to in the literature as the problem of *state space explosion*.

For the analysis of large scale PEPA models, methods have been developed which provide us with *approximate* results. One such method is Monte Carlo stochastic simulation. Here, given a large scale PEPA model, the modeller runs a number of simulations over the underlying CTMC, sampling the state of the system at the time t of interest. The obtained set of samples are then used to characterize the system performance metric's approximate distribution. The approximate distribution, found by applying the Monte Carlo method, tends to

the exact one as the number of simulation runs tends to infinity. Depending on the size of the system, running the stochastic simulations can be computationally expensive. An alternative approach for the analysis of large scale models is fluid flow approximation [5]. Using this method, one derives from a PEPA model, a set of ordinary differential equations (ODEs) whose solutions approximate the moments of the underlying CTMC to a given order. When the fluid flow method is used to analyse a model, the modeller characterizes the system's stochastic behaviour by the moments of its associated CTMC.

Compared with Monte Carlo simulation, the fluid flow method is orders of magnitude more efficient. However, it only captures the first few moments of the underlying CTMC and, under some conditions, it can suffer from a significant amount of error. These two observations raise the question of *sufficiency* of the fluid flow method: having a large scale PEPA model, is it *sufficient* to use *only* the fluid flow method for its analysis and are the results of this method enough to characterize the system's dynamics?

Our focus is on these two questions: we present the results of our experiments, in which we investigated if the fluid flow method is a sufficient tool for the analysis of variants of a simple client-server system. These experiments showed that, whilst for some models the results of fluid flow method can be used to accurately characterize the system's behaviour, for other models the results contain a large amount of error or are even misleading about the system's real executions.

The issue of sufficiency and related questions have previously been considered in the literature. In [6] Hayden and Bradley show that the state space of each PEPA model can be partitioned into regions. The suitability of the fluid flow method depends on the region that the system's execution resides in; the accuracy of the results of the fluid flow method decreases as the dynamics of the system gets closer to the *unsafe* regions. In [7], Stefanek et al. suggest that for periods of time when the system is performing in unsafe regions, one needs to run a large number of stochastic simulations, use such trajectories to calculate the moments (mean, variance, etc.) of the system's behaviour, and use them in place of the result of the fluid flow method. Our experiment results show the same phenomenon with respect to the quality of the fluid flow approximation as well as the importance of calculating accurate moments. Nevertheless, they also show that some systems have a particular type of dynamics which cannot be captured only by looking at the moments of its behaviour.

The work presented in [8] provides the basis of this paper. In [8], Bortolussi et al. suggest that one requirement for applying the fluid flow method when analysing a model, is for the all of the model's transitions to have *continuous* effect on model's dynamics. It is argued in [8] that this requirement might not be fully respected by all models. Such non-conforming models suggested in [8] are: models where a small population of components are interacting with a large population or models in which there is an interplay of slow and fast transitions. For these models, the authors suggest that the appropriate analysis of the model is performing *hybrid stochastic simulation*. In this analysis method, the *continuous* and *discrete* transitions, both necessary for capturing the model's dynamics,

are taken into account. In [8] Bortolussi et al. looked at the necessary conditions for sufficiency of the fluid flow method and provided some heuristics or criteria for checking, by looking at a model, if the fluid flow approximation is sufficient for its analysis. Our experiments showed that, given a PEPA model, more robust criteria than the ones presented in [8] are needed for one to decide on sufficiency of the fluid flow method.

Structure of this paper: Section 2 describes the model which we considered in our experiments. Section 3 shows the results of analysis of our model, where different methods have been applied. Section 4 describes the results of our experiment with respect to the sufficiency of the fluid flow method. In Section 5 we present our concluding remarks about sufficiency results and elaborate on future work.

2 The Model

To illustrate our argument throughout the paper we use variants of a simple client-server system. The following is the PEPA model of this system, where n_c is the number of client and n_s is the number of servers:

$$\begin{aligned}
 C_{thinking} &\stackrel{def}{=} (think, r_t).C_{requesting} \\
 C_{requesting} &\stackrel{def}{=} (req, r_c).C_{thinking} \\
 S_{idle} &\stackrel{def}{=} (req, r_s).S_{logging} \\
 S_{logging} &\stackrel{def}{=} (log, r_l).S_{idle} \\
 CS &\stackrel{def}{=} S_{idle}[n_s] \bowtie_{\{req\}} C_{thinking}[n_c]
 \end{aligned}$$

Model 1: PEPA model of a simple client-server system.

A server's initial state is S_{idle} when it is waiting for a client to synchronize with it on the action req ⁴. A client's initial state is $Client_{thinking}$. Each client, initially, performs an action $think$ and when her thinking is finished, she undertakes the shared action req synchronizing with an idle server. When req is done, the client goes to her initial state and the server goes to the state $Server_{logging}$. Here the server undertakes the action log and then becomes $idle$ again. In PEPA each action has a *rate* which is the parameter of an *exponentially distributed random variable* that governs the *delay* associated with performing that action. For instance, the rate of the action $think$ is r_t .

In this paper we assume that the objective of the analysis of this model is to find the *distribution* of the number of clients who are in the state $Client_{requesting}$ at the equilibrium. These are the clients being queued, waiting to get the service. We also want to know how the length of this queue changes as we reconfigure

⁴ req stands for "request".

the system: adding servers, considering more complicated behaviour for servers, changing the service rate of the existing servers, or when the population of clients increases.

Preliminaries

We use the notion of *numerical vectors* to build the underlying state space of our client-server system [5]. Each state of this system is represented as a vector, consisting of four *state variables*: $\langle S_i, S_l, C_t, C_r \rangle$, where in the current state, S_i represents the number of *idle* servers, S_l is the number of *logging* servers, C_t is the number of *thinking* clients and finally C_r is the number of clients who are requesting. At any given time, $\langle S_i, S_l, C_t, C_r \rangle \in \mathbb{Z}^{+4}$.

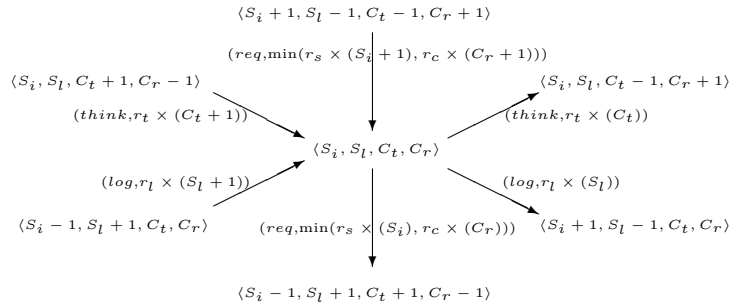


Fig. 1: Transitions into and out of a typical state $\langle S_i, S_l, C_t, C_r \rangle$.

Figure 1 shows a typical state of this system and how the system might transition into or out of it. Each transition's rate is a function of the rate of the transition's action and the population count which enables that transition. For instance, in state $\langle S_i, S_l, C_t, C_r \rangle$ the rate of the action *log* is $r_l \times S_l$. According to [9] when two components synchronize on an action, the rate of the shared action is defined to be the minimum of the rates at which the synchronizing components can perform that action. This notion can be lifted to *populations* of components [10]. When components of two populations synchronize on an action, the rate at which the action takes place is the minimum of the rates which each of those populations *offer* for the shared action. In Fig. 1, when the system is in state $\langle S_i, S_l, C_t, C_r \rangle$, the server population can perform action *req* with rate $S_i \times r_s$ and client population with the rate $C_r \times r_c$. Consequently, for state $\langle S_i, S_l, C_t, C_r \rangle$, the action *req* happens with the rate $\min(S_i \times r_s, C_r \times r_c)$.

Given concrete values for parameters of the client-server system (activity rates and the initial populations), one can use the pattern shown in Fig. 1 to build the complete state space \mathbb{D} ($\mathbb{D} \subseteq \mathbb{Z}^{+4}$) underlying the model. This state space can be treated as a Continuous Time Markov Chain on the space \mathbb{D} and be used as the basis of performance evaluation [5]. An important aspect of a dependable client-server system is to have short queues for the clients and serving

them in a timely manner. Hence, in the analysis of this model, we want to find the behaviour of state variable C_r .

3 Analysis of the Model

Typically, a client-server system involves a large population of clients communicating with a relatively smaller population of servers. Let us make a concrete client-server model based on Model 1 with the parameter values shown in Table 1. Our concrete model describes a system with 10000 clients and 10 fast servers. The rates are chosen in a way to make the length of the client waiting queue be sensitive to the number of servers available in the system and enable us to study an interesting behaviour exhibited by an extension of this model which will be shown later in Sec. 4.2.

Table 1: Parameter values considered for PEPA Model 1.

Parameter	Value	Description
r_s	500	On average, it takes 1/500th of an hour for a server to initiate a communication link with a client.
r_l	120	On average, it takes 1/120th of an hour for a server to process a request.
c_r	2	On average, it takes 1/2 of an hour for a client to initiate a communication link with a server.
c_t	0.06	On average, it takes 1/0.06th of a hours for a client to think.
n_s	10	Total population of servers.
n_c	10000	Total population of clients.

In order to find C_r 's equilibrium distribution one can construct the model's underlying CTMC, use Markovian analysis to solve it and find C_r 's *exact* distribution. However, when the client population is large, even in the range of a few thousands, the size of the CTMC becomes so large that solving it incurs a large computational cost. Therefore, a more feasible way to find C_r 's behaviour is to use alternative approaches, such as the Monte Carlo method or the fluid flow method [5], to find more efficiently *approximations* to the *exact* solutions.

The Monte Carlo method was first applied to find C_r 's distribution at the equilibrium. We performed 20000 simulations over the model's underlying CTMC, collected C_r 's value at $t = 15000$ (hours) and built a histogram, shown in Fig. 2.1. One can use this histogram to derive information such as C_r 's mean ($\mathbb{E}_{M.C.}[C_r]$) and standard deviation ($\sigma_{M.C.}[C_r]$)⁵. Running 20000 simulations guarantees that the mean *self distance* [11] of resulting histogram is bounded by $\log 2\sqrt{\pi}/20000$; an indication that the histogram is reasonably stable.

Another way to find C_r 's behaviour, is to use the fluid flow method. For a PEPA model, this method constructs a set of ordinary differential equations (ODEs) which approximates how the *mean*, *variance* (and higher order moments) of state variables evolve over the course of time [6, 10]. Tools such as the PEPA Eclipse Plugin [12] and the Grouped PEPA Analyser [13] can be used to

⁵ The subscript M.C. expresses that the measure is a result of applying the Monte Carlo method.

automatically derive a PEPA model's underlying ODEs. Here in this paper, we will use Chapman-Kolmogorov forward equations, related to the model's underlying CTMC, to derive its corresponding ODEs [6] to show how exactly ODEs are constructed and what they describe about our model's CTMC. Assuming

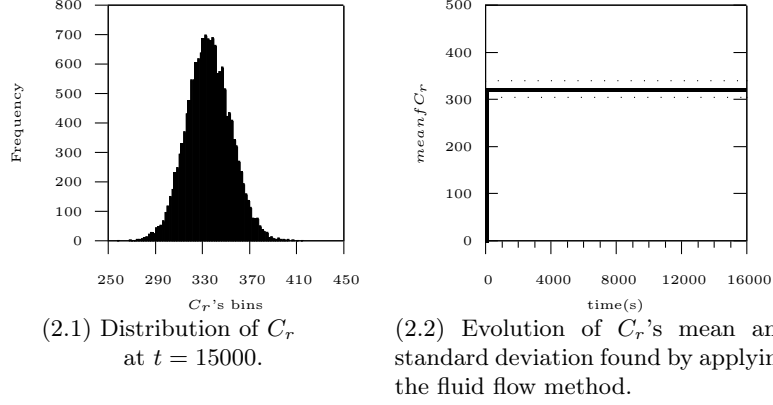


Fig. 2: C_r 's behavior, found by applying the Monte Carlo method and fluid flow method.

that the model's underlying CTMC is completely defined by the pattern shown in Fig. 1, one can write the Chapman-Kolmogorov forward equations which express, as a function of time, how the probability of being in any given state $\mathbf{v} = \langle S_i, S_l, C_i, C_t \rangle \in \mathbb{D} \subset \mathbb{Z}^4$ evolves. Let $p_{\langle S_i, S_l, C_t, C_r \rangle}(t)$ be the probability of being in state $\langle S_i, S_l, C_i, C_t \rangle$ at time t . Then we have:

$$\begin{aligned}
 \frac{d p_{\langle S_i, S_l, C_t, C_r \rangle}(t)}{d t} = & \quad (1) \\
 & + (C_t + 1) \times r_t \times p_{\langle S_i, S_l, C_t+1, C_r-1 \rangle}(t) \\
 & + (S_l + 1) \times r_l \times p_{\langle S_i-1, S_l+1, C_t, C_r \rangle}(t) \\
 & + \min((S_i + 1) \times r_s, (C_r + 1) \times r_c) \times p_{\langle S_i+1, S_l-1, C_t-1, C_r+1 \rangle}(t) \\
 & - \min(S_i \times r_s, C_r \times r_c) \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t) \\
 & - S_l \times r_l \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t) \\
 & - C_t \times r_t \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t).
 \end{aligned}$$

This ODE system has one equation for each state $\mathbf{v} \in \mathbb{D}$. Having a large number of states means one cannot form and solve equations for all states of the model's CTMC. However, we reconfigure the equations and for each state variable, find one equation describing how the mean of that state variable evolves.

For state variable C_r :

$$\begin{aligned}
 \frac{d \mathbb{E}[C_r](t)}{d t} &= \sum_{\langle S_i, S_l, C_t, C_l \rangle \in \mathbb{D}} \frac{C_r \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t)}{d t} \\
 &= + \sum_{\langle S_i, S_l, C_t, C_l \rangle \in \mathbb{D}} C_t \times r_t \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t) \\
 &\quad - \sum_{\langle S_i, S_l, C_t, C_l \rangle \in \mathbb{D}} \min(S_i \times r_s, C_r \times r_c) \times p_{\langle S_i, S_l, C_t, C_r \rangle}(t) \\
 &= + r_t \times \mathbb{E}[C_t](t) - \mathbb{E}[\min(S_i \times r_s, C_r \times r_c)](t).
 \end{aligned} \tag{2}$$

Note the term $\mathbb{E}[\min(S_i \times r_s, C_r \times r_c)]$ in the above equations. In order to *close* the system of ODEs and make them solvable, the approximation $\mathbb{E}[\min(S_i \times r_s, C_r \times r_c)] \approx \min(\mathbb{E}[S_i \times r_s], \mathbb{E}[C_r \times r_c])$ is applied. If we repeat this process for C_t , S_i and S_l and apply the same approximation, we will have the following equations⁶.

$$\begin{aligned}
 \frac{d \mathbb{E}_{F.F.}[C_t](t)}{d t} &= -r_t \times \mathbb{E}_{F.F.}[C_t](t) + \min(r_c \times \mathbb{E}_{F.F.}[C_r](t), r_s \times \mathbb{E}_{F.F.}[S_i](t)) \\
 \frac{d \mathbb{E}_{F.F.}[C_r](t)}{d t} &= -\min(r_c \times \mathbb{E}_{F.F.}[C_r](t), r_s \times \mathbb{E}_{F.F.}[S_i](t)) + r_t \times \mathbb{E}_{F.F.}[C_t](t) \\
 \frac{d \mathbb{E}_{F.F.}[S_i](t)}{d t} &= -\min(r_c \times \mathbb{E}_{F.F.}[C_r](t), r_s \times \mathbb{E}_{F.F.}[S_i](t)) + r_l \times \mathbb{E}_{F.F.}[S_i](t) \\
 \frac{d \mathbb{E}_{F.F.}[S_l](t)}{d t} &= +\min(r_c \times \mathbb{E}_{F.F.}[C_r](t), r_s \times \mathbb{E}_{F.F.}[S_i](t)) - r_l \times \mathbb{E}_{F.F.}[S_l](t)
 \end{aligned} \tag{3}$$

Due to the approximation step, the solutions $\mathbb{E}_{F.F.}[C_t]$, $\mathbb{E}_{F.F.}[C_r]$, $\mathbb{E}_{F.F.}[S_i]$ and $\mathbb{E}_{F.F.}[S_l]$ are approximations to $\mathbb{E}[C_t]$, $\mathbb{E}[C_r]$, $\mathbb{E}[S_i]$ and $\mathbb{E}[S_l]$ which, could have been derived by solving the model's underlying CTMC. Here, in order to save space, we did not include the ODEs related to variance of the state variables. Figure 2.2 shows $\mathbb{E}_{F.F.}[C_r]$ found by the fluid flow analysis of our model. The dotted lines above and below $\mathbb{E}_{F.F.}[C_r]$'s trajectory show $\mathbb{E}_{F.F.}[C_r] + \sqrt{\text{Var}_{F.F.}[C_r]}$ and $\mathbb{E}[C_r] - \sqrt{\text{Var}_{F.F.}[C_r]}$ respectively. Note that the system reaches its equilibrium within seven seconds. This means that $t = 15000$ (hours), which was the sampling time when running the Monte Carlo method, is a time when the system has reached its equilibrium.

4 Sufficiency of Fluid Flow Analysis

When analysing a large scale PEPA model, the decision has to be made about an appropriate analysis method. The fluid flow approximation is an efficient way to analyse the model. However, its results might be insufficient to capture the actual dynamics of the system. Moreover, the approximation:

$$\mathbb{E}[\min(g(x), h(x))] \approx \min(\mathbb{E}[g(x)], \mathbb{E}[h(x)])$$

⁶ The subscript F.F. expresses that the measure is the result of applying the fluid flow approximation.

introduces a certain amount of error in the fluid flow approximation's results, which is difficult to control. These issues raise the question of *sufficiency* of the fluid flow method as an analysis tool when analysing a PEPA model. In the following, we provide evidence that whilst the fluid flow approximation is a sufficient method for analysis of some models, applying this method might not be appropriate (and may even be misleading) for others.

4.1 The Client-Server System

Let us consider again Model 1. Considering the equation set (3), we know that in the equilibrium:

$$\frac{d C_t(t)}{dt} = \frac{d C_r(t)}{dt} = \frac{d S_i(t)}{dt} = \frac{d S_l(t)}{dt} = 0 \quad (4)$$

In the equilibrium, depending on the rates and populations of the clients and servers, one of the following cases can happen:

- A: No contention case: $\min(r_s \times S_i, r_c \times C_r) = r_c \times C_r$. In this case, there is enough service capacity to satisfy the requirements of the clients. Simplifying the equations we have:

$$C_r = \frac{r_t}{r_t+r_c} n_c, \quad C_t = \frac{r_c}{r_c+r_t} n_c, \quad S_i = n_s - \frac{r_c}{r_t} \frac{r_t}{r_t+r_c} n_c, \quad S_l = \frac{r_c}{r_t} \frac{r_t}{r_t+r_c} n_c. \quad (5)$$

- B: Contention case: $\min(r_s \times S_i, r_c \times C_r) = r_s \times S_i$. In this case, clients become blocked, because there is not enough service capacity to satisfy their needs. In this case:

$$C_r = n_c - \frac{r_s}{r_t} \frac{r_l}{r_s+r_l} n_s, \quad C_t = \frac{r_s}{r_t} \frac{r_l}{r_s+r_l} n_s, \quad S_i = \frac{r_l}{r_s+r_l} n_s, \quad S_l = \frac{r_s}{r_s+r_l} n_s. \quad (6)$$

For the parameter values of Table 1, the system settles in case A.

We wished to check whether it is sufficient to use *only* the fluid flow method when analysing this model and how the sufficiency depends on the concrete population levels of clients and servers. For this purpose, initially, we ran experiments where we considered various populations for the servers (n_s varies from three to 18) and for each population level, the analysis results derived by the fluid flow method were compared against those derived from the Monte Carlo method. The client population was kept constant ($n_c = 10000$). Table 2 summarises the results of these experiments. In Table 2, $\mathbb{E}_{M.C.}[C_r]$ and $\sigma_{M.C.}[C_r]$ are respectively, the mean value of C_r and C_r 's standard deviation found by the Monte Carlo method. Similarly $\mathbb{E}_{F.F.}[C_r]$ and $\sigma_{F.F.}[C_r]$ are C_r 's mean and the associated standard deviation found by the fluid flow method. As the results of the Monte Carlo method are closer to C_r 's exact behaviour, we consider such results as the basis of checking the validity of the results of the fluid flow method. Hence, the error associated with $\mathbb{E}_{F.F.}[C_r]$ (similarly for other measures) is defined as:

$$Err(\mathbb{E}_{M.C.}[C_r], \mathbb{E}_{F.F.}[C_r]) = \left| \frac{\mathbb{E}_{M.C.}[C_r] - \mathbb{E}_{F.F.}[C_r]}{\mathbb{E}_{M.C.}[C_r]} \right| \times 100.$$

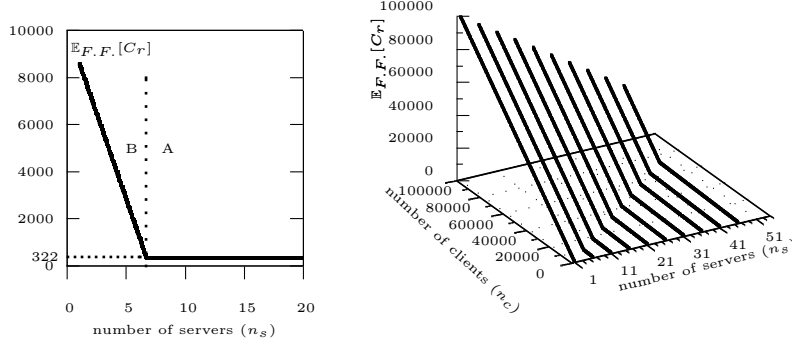
Table 2: Comparing results of the fluid flow method and Monte Carlo simulation.

	n_s	3	4	5	6	7	8	9	10	12	14	16	18
$\mathbb{E}[C_r]$	F.F.A.	5645	4193	2741	1290.3	322	322	322	322	322	322	322	322
	M.C.	5644	4192	2740	1290	490	384	349	335	325	323	322.6	322.3
	Err.(%)	0.01	0.01	0.03	0.04	34	16	7.7	3.8	0.9	0.3	0.18	0.12
$\sigma[C_r]$	F.F.A.	60.62	70	78.26	85.73	17.66	17.66	17.66	17.66	17.66	17.66	17.66	17.66
	M.C.	60.45	69.45	78.79	86.5	36.90	23.49	19.84	18.72	17.99	17.72	17.76	17.74
	Err.(%)	0.26	0.25	0.67	0.9	52.3	25.20	11.44	5.6	1.8	0.3	0.5	0.4

Considering $Err(\mathbb{E}_{M.C.}[C_r], \mathbb{E}_{F.F.}[C_r])$ in Table 2, we observed that whilst in some configurations of the client-server system, $\mathbb{E}_{F.F.}[C_r]$ reasonably approximates $\mathbb{E}_{M.C.}[C_r]$ ($n_s \leq 6$ or $n_s \geq 9$, $Err(\mathbb{E}_{M.C.}[C_r], \mathbb{E}_{F.F.}[C_r]) < 5\%$), in other configurations, the error is relatively high ($n_s = 7$ or $n_s = 8$). Based on the notion of *switching points* defined in [6], one explanation for this phenomenon is that when the system’s dynamics keeps *switching* between case A and case B, then the quality of the approximation $\mathbb{E}[\min(r_s \times S_i, r_c \times n_c)] = \min(\mathbb{E}[r_s \times S_i], \mathbb{E}[r_c \times C_r])$ is poorer and the mean behaviour shown by the fluid flow method has more error than the case when the system’s dynamics settles down in one mode. Looking at the results of the Monte Carlo method, the relatively large value of the *coefficient of variation* ($\frac{\sigma_{M.C.}[C_r]}{\mathbb{E}_{M.C.}[C_r]}$) associated with $\mathbb{E}_{M.C.}[C_r]$ when $n_s = 7$ or $n_s = 8$, shows that C_r ’s distribution is relatively wide and hence the system’s dynamics is likely to switch *more often* between the case of having a small number of clients requesting (A) and the case of having a large number of them (B).

Figure 3.1 summarises another aspect of the fluid flow approximation for this experiment. It shows that for this particular example, if $n_s > 6.6 + \epsilon$ the system converges to the behaviour associated with case A, i.e. $C_r = 322$ and when $n_s < 6.6 + \epsilon$ the system’s dynamics converges to case B. The values of Table 2 suggest that the maximum error associated with $\mathbb{E}_{F.F.}[C_r]$ happens when n_s is chosen to be in the interval (6,8).

We extended the previous experiments to also consider different populations of clients. Here, n_c takes values in [1000, 100000], n_s in [1, 50] and the rate values were again chosen based on Table 1. For each pair of (n_s, n_c) we plot $\mathbb{E}_{F.F.}[C_r]$. Figure 3.2 shows the result. As we can see, for each n_c , there is a n_s which defines the border between cases A and B for that n_c . For instance, when $n_c = 30000$, if $n_s > 21$ then in the equilibrium, $\min(r_s \times S_i, r_c \times C_r) = r_c \times C_r$ (the system resides in case A) and if $n_s < 21$ then in the equilibrium, the system resides in case B. The same phenomenon happens when $n_s = 70000$ and $n_s = 40$. The results in Table 2 and the fact that the mode change occurs in the given configurations of this particular model, suggests that the accuracy of C_r ’s moments, found by the fluid flow method, increases, i.e. the error associated with the moments decreases, if the system’s populations are changed in a way that reduces the probability that the system settles near the switching points. The experiments showed that in cases where the error associated with $\mathbb{E}_{F.F.}[C_f]$, $\sigma_{F.F.}[C_r]$ (or higher moments) is small, the fluid flow method can be considered



(3.1) $\mathbb{E}_{F.F.}[C_r]$ at the equilibrium. Here n_s varies and $n_c = 10000$. (3.2) $\mathbb{E}_{F.F.}[C_r]$ at the equilibrium. Both n_s and n_c vary.

Fig. 3: $\mathbb{E}_{F.F.}[C_r]$ at the equilibrium for various populations considered for clients and servers.

as an efficient way to characterize C_r 's behaviour. In such cases, the modeller can efficiently find accurate moments such as mean, variance, skewness and kurtosis and then construct an *approximate histogram*, similar to the one that can be derived by the application of the Monte Carlo method. The knowledge that C_r 's exact distribution is uni-modal (see the histogram of Fig. 2.1) makes it easier to construct C_r 's approximate histogram from the moments. Here, the fluid flow approach is sufficient to form a reasonable characterization of C_r 's behaviour.

4.2 An Extension of Client-Server System

In this subsection we consider Model 2 which is an extension of Model 1. The structure of the Model 2 is the same as Model 1 except that in Model 2 each server might occasionally *break down* and stop serving the clients:

$$\begin{aligned} S_{idle} &\stackrel{def}{=} (req, r_s).S_{logging} + (brk, r_b).S_{broken} \\ S_{logging} &\stackrel{def}{=} (log, r_l).S_{idle} \\ S_{broken} &\stackrel{def}{=} (fix, r_f).S_{idle} \end{aligned}$$

Table 3 shows the chosen values for the new rates r_s , r_b and r_f . The rest of the parameters in Model 2 are as in Table 1. For illustration purposes, we consider a system where the average delay associated with fixing a server is longer than the average time it takes for a server to break down (i.e. $r_f < r_b$). This helps us to study a system where there is a higher probability to see more servers breaking down while one has already broken down and is being fixed.

The results of the analysis of Model 2, through the Monte Carlo technique as well as the fluid flow method is depicted in Fig. 4. Figure 4.1 shows that this

Table 3: Parameter values used for Model 2.

Parameter	Value	Description
r_s	200	Compared to Model 1, we have made the servers slower in this model.
r_b	0.0006	It takes 70 days, on average, for a server to break down.
r_f	0.0004	It takes 100 days, on average, for a server to get fixed.
r_t	0.05	Compared to Model 1, we have made a clients' thinking longer.

model, with the current parameter values, exhibits a *multi-modal* behaviour. The first *mode* corresponds to the situation where there is no broken server ($S_b = 0$). The experiment showed that in 7100 trajectories out of 20000 simulation trajectories, at $t = 15000$, the system was observed to be in this mode. When a server breaks down, the system changes its *mode*. Due to the decrease in S_i , the overall service rate offered by the servers decreases and C_r 's values cluster around a higher level in the new mode.

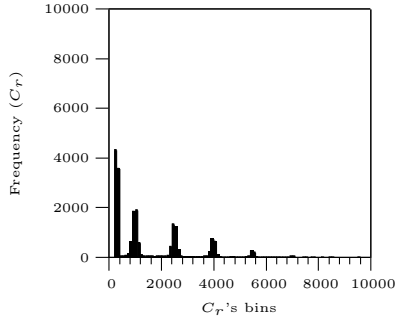
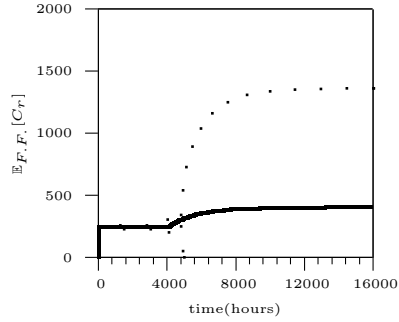
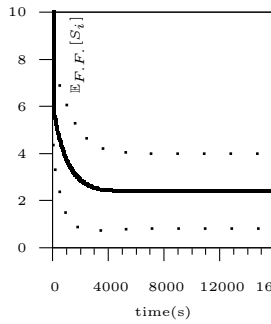
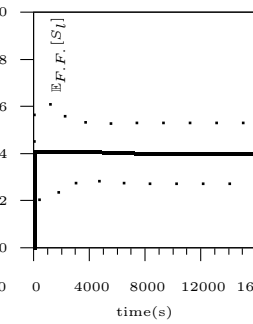
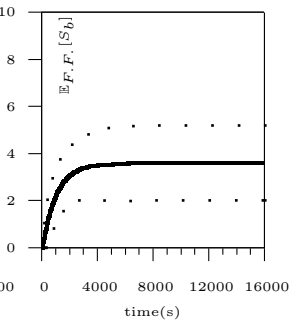
Figures 4.2, 4.3, 4.4 and 4.5. show that until $t \approx 4000$, $\mathbb{E}[C_r]$ remains constant, $\mathbb{E}[S_i]$ is *decreasing* and $\mathbb{E}[S_b]$ is increasing. This shows a phase of execution where in spite of the fact that $\mathbb{E}[S_i]$, experienced by the clients, is decreasing, still, the overall service rate offered by the servers ($r_s \times S_i$) is large enough to keep $\mathbb{E}[C_r]$ at the same level. Interestingly, at $t \approx 4000$, $\mathbb{E}[S_b]$ becomes large enough, i.e. $\mathbb{E}[S_i]$ becomes small enough, so that $\mathbb{E}[C_r]$ starts to increase. This is the second phase of the system's execution. During this period of time, $\mathbb{E}[S_i]$ keeps decreasing and $\mathbb{E}[C_r]$ increasing. Eventually, at $t = 10000$ the system reaches its equilibrium.

In order to check the sufficiency of the fluid flow analysis for analysing this model we ran a similar experiment to the one described in Section 4.1. The population of clients was kept constant at $n_c = 10000$ and different populations of servers in the range $[3, 32]$ were considered. Table 4 summarizes the results. Checking the sufficiency property for this model was more interesting as it is capable of showing a multi-modal behaviour.

Table 4: comparison between results of the fluid flow method and the Monte Carlo method. $n_c = 10000$ and n_s varies.

	n_s	3	4	5	6	7	8	9	10	12	14	16	18	24	32
$\mathbb{E}[C_r]$	F.F.A.	7119	6159	5199	4239	3279	2319	1359	399	243	243	243	243	243	243
	M. C.	7156	6177	5236	4295	3387	2599	1975	1460	843	533	378	309	251	244
	Err.(%)	0.6	0.2	0.7	1.3	3.18	10.77	31.1	72.6	71.1	54.4	35.7	21	3	0.1
$\sigma[C_r]$	F.F.A.	1240	1432	1601	1753	1894	2025	2148	959	15.42	15.42	15.42	15.42	15.42	15.42
	M. C.	1245	1420	1609	1758	1808	1792	1656	1456	1048	713	470	314	76	17.80
	Err.(%)	0.42	0.79	0.52	0.25	4.7	13	29	34.1	98	97	96	95	79	13

Considering the error $Err(\mathbb{E}_{M.C.}[C_r], \mathbb{E}_{F.F.}[C_r])$, one can see the same trend as in the results of Model 1: as we gradually increase the number of servers, the error between the results of the fluid flow analysis and the Monte Carlo method

(4.1) C_r 's histogram.(4.2) $\mathbb{E}_{F.F.}[C_r]$ (4.3) $\mathbb{E}_{F.F.}[S_i]$ (4.4) $\mathbb{E}_{F.F.}[S_l]$ (4.5) $\mathbb{E}_{F.F.}[S_b]$ Fig. 4: State variables' behaviour in Model 2 when $n_c = 10000$ and $n_s = 10$.

with respect to $\mathbb{E}[C_r]$ first rises and then decreases. In order to explain this trend, we use Fig. 5, which shows the results of Monte Carlo simulations for each case.

We observed that again the quality of the approximation $\mathbb{E}[\min(r_s \times S_i, r_c \times C_r)] \approx \min(r_s \times \mathbb{E}[S_i], r_c \times \mathbb{E}[C_r])$ plays an important role in the accuracy of the results of the fluid flow method. In a model where n_s is relatively small (e.g. $n_s = 4$), despite the fact that system's dynamics has *different modes*, still, in *any* of those modes, the above approximation is exact:

$$\mathbb{E}[\min(r_s \times S_i, r_c \times C_r)] \approx \min(r_s \times \mathbb{E}[S_i], r_c \times \mathbb{E}[C_r]) = r_s \times S_i.$$

As n_s increases, we see models whose dynamics contain mode(s) where $\min(r_s \times \mathbb{E}[S_i], r_c \times \mathbb{E}[C_r]) = r_s \times S_i$ and mode(s) where $\min(r_s \times \mathbb{E}[S_i], r_c \times \mathbb{E}[C_r]) = r_c \times C_r$. For example, when $n_s = 6$ or $n_s = 10$, the system keeps switching between a *series of modes* where there are not enough servers to keep C_r small and *the mode* where there are enough servers to satisfy client requests. In models with such a behaviour, the quality of approximation is poorer and the error associated with the results of the fluid flow method increases. For relatively larger values of n_s (e.g. $n_s > 12$), it becomes more probable that when a server fails, the overall

service rate offered by other active servers remain large enough to satisfy client requesting rate. This means, in spite of having different modes of behaviour, less switches take place in the dynamics of the system with respect to minimum approximation (see Fig. 5.4) and hence, the approximation is more accurate. Consequently, as we see in Table 4, the error associated with $\mathbb{E}_{F.F.}[C_r]$ decreases as we consider relatively larger populations of servers.

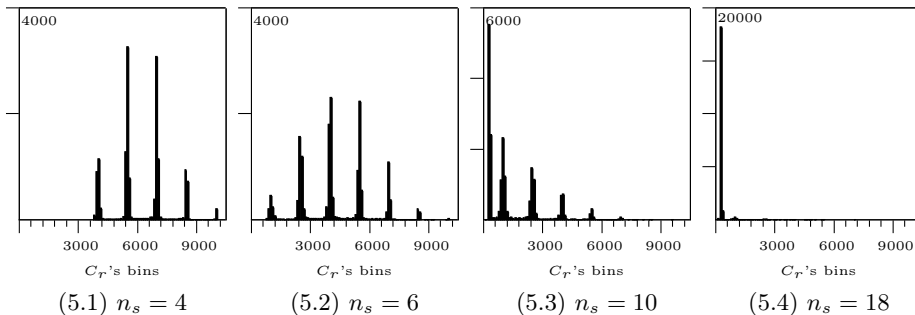
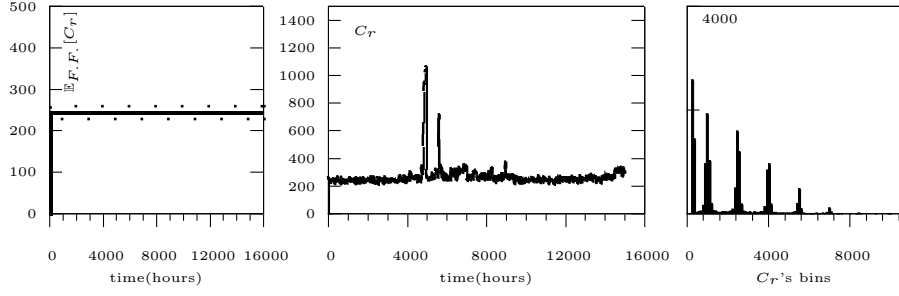


Fig. 5: the histogram of $\mathbb{E}[C_r]$ for some of the cases, $n_c = 10000$ and n_s varies.

Another aspect of checking the sufficiency of the fluid flow method is to take into account the fact that this method is finding *only moments* (mean, variance, etc.) of the state variables. We checked, for the client-server system of Model 2, if these moments alone are providing enough information about the system’s actual execution.

Figure 6.1 describes the solution of the ODEs for C_r , when $n_s = 18, n_c = 10000$. Figure 6.2 shows one trajectory of C_r considering the same populations. In this particular trajectory, in a large portion of the time, C_r ’s value fluctuates in close proximity of $\mathbb{E}_{F.F.}[C_r]$. In this part of the simulation $\mathbb{E}_{F.F.}[C_r]$ and $\sigma_{F.F.}[C_r]$ can accurately represent C_r ’s evolution. Figure 6.2 also shows that there can also be some *spikes* which occur due to a combination of contention and one (or more) servers being broken. In this period of time, $\mathbb{E}_{F.F.}[C_r]$ and $\sigma_{F.F.}[C_r]$ cannot represent C_r ’s behaviour. One might logically argue that these spikes have happened only in this particular trajectory and can be ignored. However, Fig. 5.4 indicates that in 800 out of 20000 simulation runs, we observe $C_r > \mathbb{E}_{F.F.}[C_r] + 20 \times \sigma_{F.F.}[F]$. Therefore, having occasional long queues of the clients, requesting the service, is one of the intrinsic characteristics of the system with the assumed populations. We conclude that characterising the dynamics of the system by only the results of the fluid flow method, i.e. $\mathbb{E}[C_r]$ and $\sigma_{F.F.}[C_r]$, does not account for such occasional spikes. These spikes can be especially important from the performance modelling point of view, for instance when the designers of a client-server system are going through the capacity planning phase.

The client-server systems where C_r has a multi-modal distribution, show another sufficiency issue with respect to the fluid flow method. In such cases, C_r ’s mean and standard deviation are too *crude* to reflect C_r ’s actual behaviour.



(6.1) $\mathbb{E}_{F.F.}[C_r]$ $n_s = 18$ (6.2) A trajectory of C_r , $n_s = 18$ (6.3) C_r 's distribution, and $n_c = 10000$. and $n_c = 10000$ $n_s = 9$ and $n_c = 10000$.

Fig. 6: Comparing an actual trajectory of C_r with $\mathbb{E}_{F.F.}[C_r]$.

For instance, consider the model where $n_c = 10000, n_s = 9$. Figure 6.3 shows C_r 's distribution. Table 4 shows that for this case $\mathbb{E}_{F.F.}[C_r] = 1359$ and $\sigma_{F.F.}[C_r] = 2148$. By looking at the histogram, we observed that only in 5500 out of 20000 simulations,

$$\mathbb{E}_{F.F.}[C_r] - 0.25 \times \sigma_{F.F.}[C_r] < C_r < \mathbb{E}_{F.F.}[C_r] + 0.25 \times \sigma_{F.F.}[C_r].$$

In other words, in contrast with what one might expect from a *mean* measure, $\mathbb{E}_{F.F.}[C_r]$ is showing a value that the system's dynamics is more likely to be away from.

Figure 6 shows that in the analysis of a model, any given approach which only outputs the moments of behaviour, might not be sufficient to analyse that model. For instance, running stochastic simulations and *only* extracting the average, which is a common analysis method, might keep hidden some of the important aspects of the system's execution.

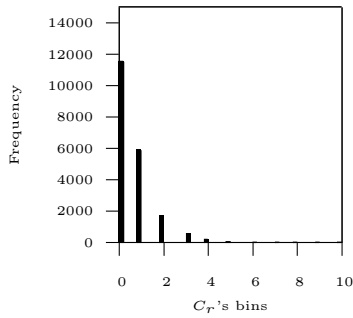


Fig. 7: C_r 's histogram when $r_t = 0.01, r_c = 200$.

As we have seen, some ranges of values for n_s and n_c give rise to instantiations of Model 2 for which the results of fluid flow analysis alone, are too *abstract* to characterize the system’s dynamics. However, experiments showed that for the same model but with different parameters, the multi-modality disappears, and for these instantiations of Model 2, $\mathbb{E}_{F.F.}[C_r]$ and $\sigma_{F.F.}[C_r]$ can *accurately* exhibit the system’s executions. For instance, if we consider larger populations for the servers ($n_s > 32$), even when one or more servers are broken, the remaining service capacity is large enough to prevent a sudden increase in C_r . Or as another example, consider $n_s = 14, n_c = 10000$ (the condition as seen in Fig. 5.1, but decrease the thinking rate from $r_t = 0.05$ to $r_t = 0.01$ and make clients communication with servers faster: increase $r_c = 2$ up to $r_c = 200$. Effectively, each client now spends more time thinking independently and once she enters the request queue, because she communicates with a servers more efficiently, she leaves the queue more quickly. This stops high contention in the system (see Fig. 7) and consequently server’s breakdowns will not have any effect on C_r .

5 Conclusion and Future Work

Our experiments led us to the following conclusions:

1. In performance evaluation studies, there are models which exhibit multi-modal behaviour. For such systems, using analysis approaches which only capture the behaviour’s moments might not be sufficient to reveal the actual dynamics of the system. This insufficiency is not exclusively related to the fluid flow method. Even with stochastic simulations, it might not be sufficient to look only at the *average* of the simulation trajectories — *averaging* might hide some aspects of the system’s performance.
2. It was shown in [6] that the accuracy of the fluid flow method depends on the quality of the approximation:

$$\mathbb{E}[\min(g(x), h(x)) \approx \min(\mathbb{E}[g(x)], \mathbb{E}[h(x)])].$$

The results of the fluid flow approximation are more accurate when it is less probable for the system to be in regions of the state space where the quality of this approximation is poor. In the context of our experiments we showed that this accuracy criterion can also be applied for multi-modal systems: if the dynamics of the system does not enter the unsafe regions, even though the system observes different modes, the moments of the behaviour can still be accurately calculated by applying the fluid flow method.

3. For a PEPA model which is capable of exhibiting multi-modal behaviour, the sufficiency of the fluid flow method depends on the concrete values one considers for the model’s parameters. Moreover, the multi-modal behaviour is very sensitive to parameter values and model structure.
4. In a PEPA model, having a small population of components interacting with a larger one, or having a combination of slow and fast transitions, does not necessarily imply the insufficiency of the fluid flow method for the analysis of that model. However, for models in these classes [8], the modeller must run a rigorous validation of the results of the fluid flow method.

5.1 Future work

Our experiments showed some models for which the fluid flow method was concluded to be *insufficient*. However, such conclusions could only be made after running numerous computationally expensive stochastic simulations. Our future work will focus on finding algorithms or analytical results which, for a given PEPA model, can decide whether the fluid flow method is a sufficient analysis tool. Specifically we aim to build a method which can statically analyse a completely parametrized PEPA model and reveal if the model has the potential to show multi-modal behaviour. This involves improvements on the sufficiency criteria offered in [8].

References

1. J. T. Bradley, S. Gilmore, and J. Hillston. Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models. *J. Comput. Syst. Sci.*, 74(6):1013–1032, September 2008.
2. M. D. Harrison, M. Massink, and D. Latella. Engineering crowd interaction within smart environments. In *Proc. of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '09, pages 117–122, USA, 2009. ACM.
3. H. Wang, D. Laurenson, and J. Hillston. PEPA analysis of MAP effects in hierarchical mobile IPv6. In *Proc. 15th Int. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '07, pages 337–342, 2007. IEEE Computer Society.
4. J. Ding, J. Hillston, and D. Laurenson. Performance modelling of content adaptation for a personal distributed environment. *Wirel. Pers. Commun.*, 48(1):93–112, January 2009.
5. J. Hillston. Fluid flow approximation of PEPA models. In *Quantitative Evaluation of Systems, 2005. 2nd Int. Conf. on the*, pages 33 – 42, sept. 2005.
6. R. Hayden and Bradley J. T. A fluid analysis framework for a Markovian process algebra. *Theoretical Computer Science*, 411(22–24):2260 – 2297, 2010.
7. A. Stefanek, R. Hayden, and J. T. Bradley. Hybrid analysis of large scale PEPA models. In *9th Workshop on Process Algebra and Stochastically Timed Activities (PASTA 2010)*, September 2010.
8. L. Bortolussi, V. Galpin, J. Hillston, and M. Tribastone. Hybrid semantics for PEPA. In *Quantitative Evaluation of Systems (QEST), 7th Int. Conf. on the*, pages 181–190, sept. 2010.
9. J. Hillston. *A compositional approach to performance modelling*. Cambridge University Press, 1996.
10. M. Tribastone, S. Gilmore, and J. Hillston. Scalable differential analysis of process algebra models. *Software Engineering, IEEE Trans. on*, 38(1):205–219, jan.-feb. 2012.
11. Y. Cao and L. Petzold. Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems. *Journal of Computational Physics*, 212(1):6–24, 2006.
12. M. Tribastone, A. Duguid, and S. Gilmore. The PEPA eclipse plugin. *SIGMET-RICS Perform. Eval. Rev.*, 36(4):28–33, March 2009.
13. A. Stefanek, R. Hayden, and J. T. Bradley. GPA - a tool for fluid scalability analysis of massively parallel systems. In *Proc. 8th Int. Conf. on Quantitative Evaluation of SysTems*, QEST '11, pages 147–148, 2011. IEEE Computer Society.