ODE-based general moment approximations for PEPA

Richard A. Hayden Jeremy T. Bradley

Dept. of Computing, Imperial College London
180 Queen's Gate, London SW7 2BZ, UK

{rh,jb}@doc.ic.ac.uk

August 1, 2008

Abstract

In this paper we show how the powerful ODE-based fluid-analysis technique for the stochastic process algebra PEPA is an approximation to the first moments of the counting processes in question. For a large class of models this approximation has a particularly simple form and it is possible to make qualitative statements regarding how the quality of the approximation varies for different parameters.

Furthermore, this particular point of view facilitates a natural generalisation to higher order moments. This allows modellers to approximate, for instance, the variance of the component counts. In particular, we show how systems of ODEs facilitating the approximation of arbitrary moments of the component counting processes can be naturally defined. The effectiveness of this generalisation is illustrated by comparing the results with those obtained through stochastic simulation for a particular case study.

1 Introduction

Fluid-analysis of performance models offers the exciting potential of analysing massive state-spaces at small computational cost. In the case of stochastic process algebra models, fluid-analysis involves approximating the underlying discrete state-space with continuous real-valued variables and describing the time-evolution of those variables with ordinary differential equations (ODEs). This approach was first applied to a subset of the stochastic process algebra PEPA [1] in [2] and has subsequently been extended in [3] and [4].

Despite the successful and widespread application of these techniques, see e.g. [5; 3; 6; 7], limited effort has been expended in formally relating the analysis to the underlying continuous time Markov chain (CTMC). In [8], we showed how these techniques have an exact interpretation for a very basic subset of PEPA[1]. In this paper, we build on this and exhibit the nature of the approximation for a much larger and more useful class of PEPA models. Through the insight gained as a result, we are also able to define for the first time similar ODE-based analyses which provide computationally inexpensive access to higher order stochastic features of models, such as the variance or skewness of the corresponding distributions.

In the following section, Section 1.1, we introduce the stochastic process algebra PEPA and in Section 1.2, we introduce the existing fluid semantics by means of a simple example for the sake of brevity. In Section 2, we discuss the nature of the existing fluid-analysis as an approximation to the first moments of certain stochastic processes associated with the model and in Section 3, we show how this may be extended to higher order moments.

[1] Models involving no synchronisation, i.e. only purely parallel concurrency.

1.1 PEPA

PEPA [9] as a performance modelling formalism has been used to study a wide variety of systems: multimedia applications [10], mobile phone usage [11], GRID scheduling [12], production cell efficiency [13] and web-server clusters [14] amongst others. The definitive reference for the language is [9].

As in all process algebras, systems are represented in PEPA as the composition of *components* which undertake *actions*. In PEPA the actions are assumed to have a duration, or delay. Thus the expression $(\alpha, r).P$ denotes a component which can undertake an α action at rate r to evolve into a component P. Here $\alpha \in \mathcal{A}$ where $\mathcal{A}$ is the set of action types. The rate r is interpreted as a random delay which samples from an exponential random variable with parameter, r.

PEPA has a small set of combinators, allowing system descriptions to be built up as the concurrent execution and interaction of simple sequential components. The syntax of the type of PEPA model considered in this paper may be formally specified using the following grammar:

$$\begin{aligned} S &::= (\alpha, r).S \mid S + S \mid C_S \\ P &::= P \underset{L}{\bowtie} P \mid P/L \mid C \end{aligned}$$

where S denotes a *sequential component* and P denotes a *model component* which executes in parallel. C stands for a constant which denotes either a sequential component or a model component as introduced by a definition. C_S stands for constants which denote sequential components. The effect of the syntactic separation between these types of constants is to constrain legal PEPA components to be cooperations of sequential processes.

More information and structured operational semantics on PEPA can be found in [9]. A brief discussion of the basic PEPA operators is given below:

Prefix The basic mechanism for describing the behaviour of a system with a PEPA model is to give a component a designated first action using the prefix combinator, denoted by a full stop, which was introduced above. As explained, $(\alpha, r).P$ carries out an α action with rate r, and it subsequently behaves as P.

Choice The component $P + Q$ represents a system which may behave either as P or as Q. The activities of both P and Q are enabled. The first activity to complete distinguishes one of them: the other is discarded. The system will behave as the derivative resulting from the evolution of the chosen component.

Constant It is convenient to be able to assign names to patterns of behaviour associated with components. Constants are components whose meaning is given by a defining equation. The notation for this is $X \stackrel{def}{=} E$. The name X is in scope in the expression on the right hand side meaning that, for example, $X \stackrel{def}{=} (\alpha, r).X$ performs α at rate r forever.

Hiding The possibility to abstract away some aspects of a component's behaviour is provided by the hiding operator, denoted P/L. Here, the set L identifies those activities which are to be considered internal or private to the component and which will appear as the unknown type τ.

Cooperation We write $P \underset{L}{\bowtie} Q$ to denote cooperation between P and Q over L. The set which is used as the subscript to the cooperation symbol, the *cooperation set* L, determines those activities on which the components are forced to synchronise. For action types not in L, the components proceed independently and concurrently with their enabled activities. We write $P \parallel Q$ as an abbreviation for $P \underset{L}{\bowtie} Q$ when L is empty. Furthermore, $P[n]$ is shorthand for the parallel cooperation of n P-components, $\underbrace{P \mid\mid \cdots \mid\mid P}_{n}$.

In process cooperation, if a component enables an activity whose action type is in the cooperation set it will not be able to proceed with that activity until the other component also enables an activity of that type. The

two components then proceed together to complete the *shared activity*. Once enabled, the rate of a shared activity has to be altered to reflect the slower component in a cooperation.

In some cases, when a shared activity is known to be completely dependent only on one component in the cooperation, then the other component will be made *passive* with respect to that activity. This means that the rate of the activity is left unspecified (denoted $\top$) and is determined upon cooperation, by the rate of the activity in the other component. All passive actions must be synchronised in the final model.

Within the cooperation framework, PEPA respects the definition of *bounded capacity*: that is, a component cannot be made to perform an activity faster by cooperation, so the rate of a shared activity is the minimum of the apparent rates of the activity in the cooperating components.

1.2 Fluid-analysis

For the sake of brevity, we will not formally present here the fluid semantics for PEPA. It can be found in different degrees of generality in the literature [2; 3; 4]. Instead, we will introduce the techniques by considering a simple case study.

In the PEPA model $System$ below, we have a population of N_C Clients and a population of N_S Servers. The system uses a 2-stage fetch mechanism: a client requests data from the pool of servers; one of the servers receives the request, another server may then fetch the data for the client. At any stage, a server in the pool may fail.

$$
\begin{aligned}
Client &\stackrel{def}{=} (request, r_{req}).Client\_waiting \\
Client\_waiting &\stackrel{def}{=} (data, r_{data}).Client\_think \\
Client\_think &\stackrel{def}{=} (think, r_{think}).Client \\
\\
Server &\stackrel{def}{=} (request, r_{req}).Server\_get + (break, r_{break}).Server\_broken \\
Server\_get &\stackrel{def}{=} (data, r_{data}).Server + (break, r_{break}).Server\_broken \\
Server\_broken &\stackrel{def}{=} (reset, r_{reset}).Server \\
\\
System &\stackrel{def}{=} Client[N_C] \underset{L}{\bowtie} Server[N_S]
\end{aligned}
$$

where $L = \{request, data\}$.

Since each client and server can be in one of three derivative states, it is clear that this model has $3^{N_C+N_S}$ states in its underlying CTMC, and thus it is quickly intractable to traditional analysis methods. Consider the three integer-valued stochastic processes which count the number of the N_C clients in each of the three possible derivative states of $Client$. Let these be $n_C(t)$, $n_{C_w}(t)$ and $n_{C_t}(t)$ respectively. Similarly, define for the servers, $n_S(t)$, $n_{S_g}(t)$ and $n_{S_b}(t)$. Using *strong equivalence* it is straightforward to show that the partition of the state-space into mutually exclusive subsets, such that all of these stochastic processes take on the same value in each subset, is a *lumpable* partition, see [1][Chapter 8]. This allows these states to be combined and the rates aggregated, resulting in a smaller CTMC, for which each state is specified uniquely by the values of the six stochastic processes defined above. Unfortunately, this simplification does not, in general, solve the state-space explosion problem. However, it is a necessary first step before constructing a fluid-analysis.

The idea of the fluid-analysis is to define deterministic, real-valued fluid approximations $v_.(t)$ (defined by ODEs) to the integer stochastic processes $n_.(t)$, in some sense. In order to construct the ordinary differential equation which governs the evolution of $v_C(t)$, for example, we consider the aggregate CTMC rate at which $Client$ components are lost in the model and the rate at which they are gained, balancing the two quantities in terms of the fluid approximations $v_.(t)$:

$$
\frac{dv_C(t)}{dt} = -\min(v_C(t),\, v_S(t))r_{req} + v_{C_t}(t)r_{think} \tag{1.1}
$$

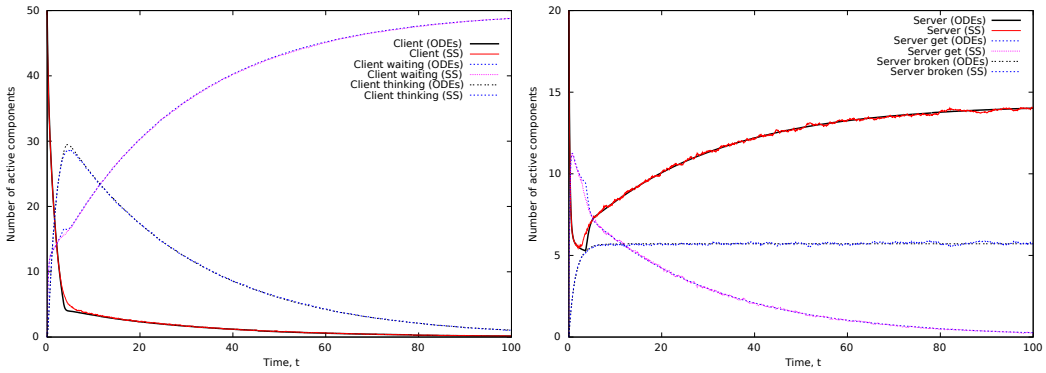

Fig. 1. Comparison of ODE approximation with expectations obtained via stochastic simulation. Rates used are $r_{req} = 3.0$, $r_{think} = 0.4$, $r_{break} = 0.2$, $r_{data} = 1.5$ and $r_{reset} = 0.5$. Initial conditions are 50 $Client$ and 20 $Server$ components.

That is, $Client$ components are lost only through evolving into $Client\_waiting$ components. This happens by virtue of completing a $request$ shared action with a $Server$ component, at the aggregate CTMC rate $\min(n_C(t), n_S(t))r_{req}$. $Client$ components are gained only through $Client\_think$ components completing their $think$ action at aggregate CTMC rate $n_{C_t}(t)r_{think}$. Similar considerations for the other client and server components lead to a complete set of six ODEs. These can then be inexpensively integrated to obtain the $v_\cdot(t)$ as deterministic, real-valued functions. The most natural interpretation of $v_\cdot(t)$ is as an approximation to the (deterministic) expectation $\mathbb{E}[n_\cdot(t)]$, and indeed, as Figure 1 shows, the correspondence is often very impressive.

2 First moment approximation

Despite the fact that the fluid-analysis introduced in the last section yields very impressive results in a lot of cases, there are certainly instances where it is not as accurate. In this section, we exhibit the nature of the approximation by deriving the system of ODEs directly from the CTMC via an approximation to the underlying Chapman-Kolmogorov equations.

We consider again the model introduced in the previous section. Write $p_{(C, C_w, C_t, S, S_g, S_b)}(t)$ as the transient probability of being in the unique aggregated state where $n_C(t) = C$ and $n_{C_w}(t) = C_w$ etc. at time t, then the Chapman-Kolmogorov equations which govern the evolution of the underlying aggregated CTMC

are:

$$\begin{aligned}\dot{p}_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t) = {} & \min(C+1,\,S+1)r_{req}\cdot p_{(C+1,\,C_w-1,\,C_t,\,S+1,\,S_g-1,\,S_b)}(t) \\ & + \min(C_w+1,\,S_g+1)r_{data}\cdot p_{(C,\,C_w+1,\,C_t-1,\,S-1,\,S_g+1,\,S_b)}(t) \\ & + (C_t+1)r_{think}\cdot p_{(C-1,\,C_w,\,C_t+1,\,S,\,S_g,\,S_b)}(t) \\ & + (S+1)r_{break}\cdot p_{(C,\,C_w,\,C_t,\,S+1,\,S_g,\,S_b-1)}(t) \\ & + (S_g+1)r_{break}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g+1,\,S_b-1)}(t) \\ & + (S_b+1)r_{reset}\cdot p_{(C,\,C_w,\,C_t,\,S-1,\,S_g,\,S_b+1)}(t) \\ & - \min(C,\,S)r_{req}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t) \\ & - \min(C_w,\,S_g)r_{data}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t) \\ & - C_t\cdot r_{think}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t) \\ & - S\cdot r_{break}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t) \\ & - S_g\cdot r_{break}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t) \\ & - S_b\cdot r_{reset}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t)\end{aligned} \tag{2.1}$$

where each of the first six summands appears only when it is valid in the sense that the subscript of $p_{\cdot}(t)$ is within the aggregated state-space, say S. Now for all states $s = (C,\,C_w,\,C_t,\,S,\,S_g,\,S_b) \in S$, multiplying $p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t)$ by C and summing, we obtain:

$$\begin{aligned}\sum_{s\in S}\dot{p}_s(t)C = \sum_{s\in S}\Big[& (C-1)\min(C,\,S)r_{req}\cdot p_s(t) + C\min(C_w,\,S_g)r_{data}\cdot p_s(t) + (C+1)C_t r_{think}\cdot p_s(t) \\ & + CSr_{break}\cdot p_s(t) + CS_g r_{break}\cdot p_s(t) + CS_b r_{reset}\cdot p_s(t) \\ & - C\min(C,\,S)r_{req}\cdot p_s(t) - C\min(C_w,\,S_g)r_{data}\cdot p_s(t) - CC_t\cdot r_{think}\cdot p_s(t) \\ & - CS\cdot r_{break}\cdot p_s(t) - CS_g\cdot r_{break}\cdot p_s(t) - CS_b\cdot r_{reset}\cdot p_s(t)\Big]\end{aligned} \tag{2.2}$$

Notice in particular that, for example:

$$\begin{aligned}\sum_{s\in S} C\min(C+1,\,S+1)r_{req}\cdot p_{(C+1,\,C_w-1,\,C_t,\,S+1,\,S_g-1,\,S_b)}(t) = \\ \sum_{s\in S}(C-1)\min(C,\,S)r_{req}\cdot p_{(C,\,C_w,\,C_t,\,S,\,S_g,\,S_b)}(t)\end{aligned}$$

since all $s \in S$ can be expressed as $s = (C+1,\,C_w-1,\,C_t,\,S+1,\,S_g-1,\,S_b)$, except for maybe those whose contribution to the sum would be zero anyway (e.g. those for which $C_w = N_C$, in which case $\min(C,\,S) = \min(0,\,S) = 0$).

Expanding and cancelling Equation (2.2) then yields:

$$\frac{d\mathbb{E}[n_C(t)]}{dt} = -\mathbb{E}[\min(n_C(t),\,n_S(t))]r_{req} + \mathbb{E}[n_{C_t}(t)]r_{think} \tag{2.3}$$

We thus obtain Equation (1.1) if we write $v_{\cdot}(t)$ as the approximation to $\mathbb{E}[n_{\cdot}(t)]$ that is obtained on application of the following approximation to Equation (2.3):

$$\mathbb{E}[\min(\cdot,\,\cdot)] \approx \min(\mathbb{E}[\cdot],\,\mathbb{E}[\cdot])$$

It is easily seen that in general, however:

$$\mathbb{E}[\min(\cdot,\,\cdot)] \leq \min(\mathbb{E}[\cdot],\,\mathbb{E}[\cdot])$$

This programme can be completed similarly for each of the other five client and server derivative states.

More generally, we can show that for a very large class of PEPA models[2], the fluid-analysis technique when viewed as an approximation to the expected value of the component counts relies only on (potentially repeated) application of the above approximation.

2.1 Nature of the approximation

Having identified the quantitative nature of the fluid-analysis, as relying on the approximation $\mathbb{E}[\min(\cdot,\,\cdot)] \approx \min(\mathbb{E}[\cdot],\,\mathbb{E}[\cdot])$, we may identify two key properties of a given PEPA model which should significantly affect its accuracy. The first of these has to do with the variability of the component counting stochastic processes $n.(t)$ and the second is a more structurally explicit aspect of the model itself.

2.1.1 Variability at 'switch points'

A *switch point* is defined to be a point in the aggregated state-space at which the dominant side of a $\min(\cdot,\,\cdot)$ term in the system of ODEs for a given PEPA model changes.

It is easy to see that far away from switch points, we would expect the fluid approximation to remain good (as long as it is not already poor) unless the variability (i.e. spread of the distribution) is very high. Consider the term $\mathbb{E}[\min(n_C(t),\,n_S(t))]$. If we are very far away from a switch point, say all states with non-negligible probability have $C > S$, then $\mathbb{E}[\min(n_C(t),\,n_S(t))]$ will be very well approximated by $\mathbb{E}[n_S(t)]$. Around switch points, a non-negligible proportion of the probability distribution may be in states for which $C < S$ and also in states for which $S < C$, so both $\mathbb{E}[n_C(t)]$ and $\mathbb{E}[n_S(t)]$ would be likely to be much less accurate approximations to $\mathbb{E}[\min(n_C(t),\,n_S(t))]$.

The extent to which this phenomenon affects the quality of the approximation depends on how far the distribution is spread either side of switch points, i.e. is determined by the variability of the stochastic processes $n.(t)$.

2.1.2 Heterogeneous cooperation rates

Consider modifying the original model so that instead of both the Client and Server components completing the $\mathit{request}$ action at rate r_{req}, the Client component does it at rate r_1 and the Server at rate r_2. Then, the term $\mathbb{E}[\min(n_C(t),\,n_S(t))]r_{req}$ in Equation (2.3) becomes $\mathbb{E}[\min(n_C(t)r_1,\,n_S(t)r_2)]$. If instead of $r_1 = r_2$, say $r_1 \ll r_2$, the same amount of variability in the component counts could clearly translate to a much larger relative quantitative error under the:

$$\mathbb{E}[\min(n_C(t)r_1,\,n_S(t)r_2)] \approx \min(\mathbb{E}[n_C(t)]r_1,\,\mathbb{E}[n_S(t)]r_2)$$

approximation.

In [4], an improved fluid semantics for passive cooperation between component groups was presented. This involved replacing the passive action by an active action with a rate that is fast enough to ensure that the action is effectively passive within its encompassing model structure. The fluid semantics for active cooperation can then be applied directly. Such cooperations will of course involve rates, one of which is much greater than the other by their very nature. As a passive resource runs out and the corresponding switch point approaches, the error in the above approximation may be very large indeed since a large part of the probability distribution corresponding to states with zero resources will not affect the approximation as early as it should, resulting in an underestimation of the corresponding blocking effect. This can be seen in Figure 2, which shows a version of the earlier example with heterogeneous cooperation (created as described above, by using different rates for the $\mathit{request}$ action for the Client and Server components; in this case, r_2 is picked so that the Server is effectively passive for the $\mathit{request}$ cooperation). The blocking

[2] Specifically, those whose aggregate CTMC rates do not involve rational functions of the component counts. Such aggregate CTMC rates occur when more than one component derivative state in a component group enables the same shared action.

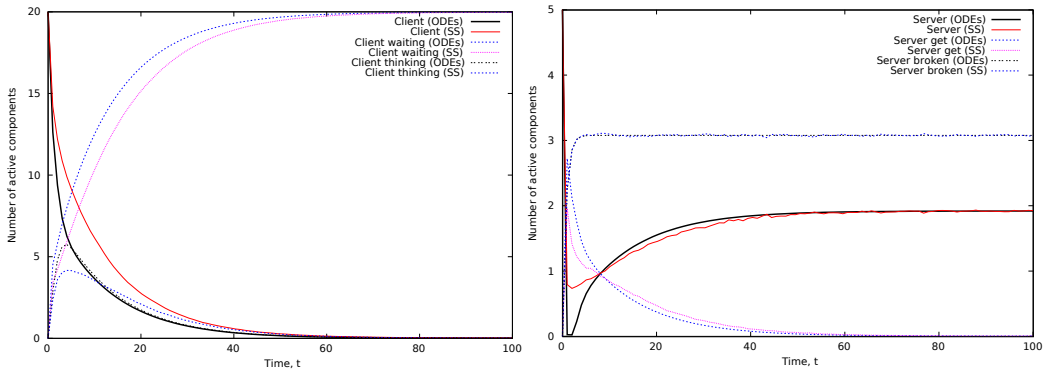

Fig. 2. Comparison of ODE approximations with expectations obtained via stochastic simulation for massively heterogeneous (effectively passive) *request* cooperation. Rates used are $r_1 = 3.0$, $r_2 = 500$, $r_{think} = 0.4$, $r_{break} = 0.8$, $r_{data} = 1.5$ and $r_{reset} = 0.5$. Initial conditions are 20 *Client* and 5 *Server* components.

effect on the *Client* components due to a lack of *Server* components to service their *request* actions is clearly seen to have been underestimated by the ODE approximation.

3 Higher order moment approximations

In this section we show how the insight gained in Section 2 can be used to naturally define similar approximations for higher order moments of the component counting stochastic processes for a PEPA model.

Consider again the model of the previous sections. Recall the Chapman-Kolmogorov equations of its underlying aggregated CTMC (Equation (2.1)). Proceed as in Section 2, but instead of multiplying by C and summing, multiply by C^2 and sum:

$$\begin{aligned}\sum_{s\in S}\dot{p}_s(t)C^2 = \sum_{s\in S}\Big[&(C-1)^2\min(C,\,S)r_{req}\cdot p_s(t) + C^2\min(C_w,\,S_g)r_{data}\cdot p_s(t)\\ &+ (C+1)^2C_t r_{think}\cdot p_s(t)\\ &+ C^2 S r_{break}\cdot p_s(t) + C^2 S_g r_{break}\cdot p_s(t) + C^2 S_b r_{reset}\cdot p_s(t)\\ &- C^2\min(C,\,S)r_{req}\cdot p_s(t) - C^2\min(C_w,\,S_g)r_{data}\cdot p_s(t) - C^2 C_t\cdot r_{think}\cdot p_s(t)\\ &- C^2 S\cdot r_{break}\cdot p_s(t) - C^2 S_g\cdot r_{break}\cdot p_s(t) - C^2 S_b\cdot r_{reset}\cdot p_s(t)\Big]\end{aligned}$$

Expanding and cancelling now yields:

$$\begin{aligned}\frac{d\mathbb{E}[n_C^2(t)]}{dt} = &\,\mathbb{E}[\min(n_C(t),\,n_S(t))]r_{req} - 2\mathbb{E}[\min(n_C^2(t),\,n_C(t)n_S(t))]r_{req}\\ &+ \mathbb{E}[n_{C_t}(t)]r_{think} + 2\mathbb{E}[n_C(t)n_{C_t}(t)]r_{think}\end{aligned}$$

Write $v_{C^2}(t)$ for the approximation to the second moment of the *Client* counting process and $v_{C\cdot S}(t)$ for the approximation to the joint moment of the *Client* and *Server* counting processes etc. Applying again, $\mathbb{E}[\min(\cdot,\,\cdot)] \approx \min(\mathbb{E}[\cdot],\,\mathbb{E}[\cdot])$, then gives:

$$\begin{aligned}\frac{dv_{C^2}(t)}{dt} = &\,\min(v_C(t),\,v_S(t))r_{req} - 2\min(v_{C^2}(t),\,v_{C\cdot S}(t))r_{req}\\ &+ v_{C_t}(t)r_{think} + 2v_{C\cdot C_t}(t)r_{think}\end{aligned}$$

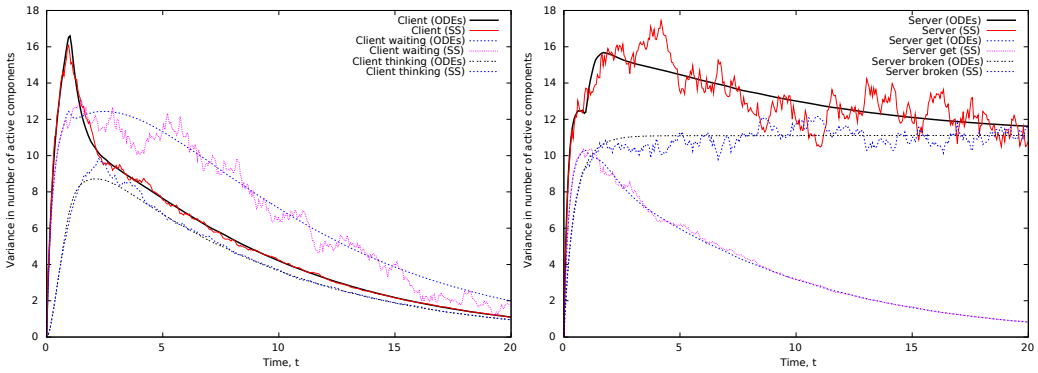

Fig. 3. Comparison of ODE-derived variance approximation with that obtained via stochastic simulation. Rates used are $r_{req} = 1.0$, $r_{think} = 1.0$, $r_{break} = 0.5$, $r_{data} = 1.0$ and $r_{reset} = 1.0$. Initial conditions are 50 $Client$ and 50 $Server$ components.

If we repeat this programme for all second order (joint) moments, we obtain 27 such ODEs, which uniquely determine the approximation. Figure 3 shows a comparison between the variance of the component counting processes obtained by computing the first and second order moments using the ODE-based approximation defined above, with that obtained through stochastic simulation.

For moments of any order, the same idea works (multiplication by C^n for example) and the accuracy again relies solely on the above approximation for the same large class of PEPA models as in the first order case.

4 Conclusion and future work

Through this work, we have a more precise understanding of what fluid-analysis of PEPA models means in terms of the underlying CTMC, and a much better idea of what affects the quality of the approximation for given models and parameters.

We have also shown how this style of approximation may be naturally extended to allow access to previously inaccessible features of models with massive state spaces, including, for example, the variance and skewness of the component counts. Before now, the only computationally feasible method of obtaining such measures was through stochastic simulation using, for example, the Gillespie algorithm [15]. Furthermore, due to the small magnitude of, for example, the variance of a component count, it is much more expensive to obtain via stochastic simulation for the same relative error than the expected values of component counts. Indeed, to obtain the stochastic simulations in the graphs of Figure 3, 100,000 independent replications were required, and even still, there are visible fluctuations.

One interesting direction for future work is the possibility of using the higher order approximations introduced in this work to improve the first order approximation to the expected component counts. Approximate knowledge of the variability of the distribution of component counts around switch points may provide a natural route to improve the underlying $\mathbb{E}[\min(\cdot, \cdot)] \approx \min(\mathbb{E}[\cdot], \mathbb{E}[\cdot])$ approximation. In particular, if $\min(\cdot, \cdot)$ were smoother[3], a Taylor expansion of $\min(\cdot, \cdot)$ in terms of higher order moments of the component counts in Equation (2.2) would be one possible route. Since it is not at all smooth, we might consider using smoother functions in the ODE approximation, which take on the same discrete values as $\min(\cdot\,\cdot)$, i.e. do not change the underlying CTMC, only differing on the values in between the integer component counts.

[3] That is, differentiable at least a few times.

References

[1] J. Hillston, *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[2] J. Hillston, "Fluid flow approximation of PEPA models," in *QEST'05, Proceedings of the 2nd International Conference on Quantitative Evaluation of Systems*, (Torino), pp. 33–42, IEEE Computer Society Press, September 2005.

[3] J. T. Bradley, S. T. Gilmore, and J. Hillston, "Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models," *Journal of Computer and System Sciences*, July 2007. (in press).

[4] R. A. Hayden and J. T. Bradley, "Fluid semantics for passive stochastic process algebra cooperation," in *VALUETOOLS'08, Third International Conference on Performance Evaluation Methodologies and Tools*, (Athens), 2008.

[5] A. Duguid, "Coping with the parallelism of BitTorrent: Conversion of PEPA to ODEs in dealing with state space explosion," in *Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS 2006, Paris, France, September 25-27, 2006, Proceedings* (E. Asarin and P. Bouyer, eds.), vol. 4202 of *Lecture Notes in Computer Science*, pp. 156–170, Springer, 2006.

[6] S. Gilmore and M. Tribastone, "Evaluating the scalability of a web service-based distributed e-learning and course management system," in *Third International Workshop on Web Services and Formal Methods (WS-FM'06)* (M. T. N. n. Mario Bravetti and G. Zavattaro, eds.), vol. 4184 of *Lecture Notes in Computer Science*, (Vienna, Austria), pp. 156–170, Springer, 2006.

[7] M. Bravetti, S. Gilmore, C. Guidi, and M. Tribastone, "Replicating web services for scalability," in *Proceedings of the Third International Conference on Trustworthy Global Computing (TGC'07)* (G. Barthe and C. Fournet, eds.), vol. 4912 of *LNCS*, pp. 222204–221, Springer-Verlag, 2008.

[8] R. A. Hayden and J. T. Bradley, "Fluid-flow solutions in pepa to the state space explosion problem," in *PASTA 2007, 6th Workshop on Process Algebra and Stochastically Timed Activities*, (London), 2007.

[9] J. Hillston, *A Compositional Approach to Performance Modelling*, vol. 12 of *Distinguished Dissertations in Computer Science*. Cambridge University Press, 1996.

[10] H. Bowman, J. W. Bryans, and J. Derrick, "Analysis of a multimedia stream using stochastic process algebras," *The Computer Journal*, vol. 44, no. 4, pp. 230–245, 2001.

[11] J. M. Fourneau, L. Kloul, and F. Valois, "Performance modelling of hierarchical cellular networks using PEPA," *Performance Evaluation*, vol. 50, pp. 83–99, November 2002.

[12] N. Thomas, J. T. Bradley, and W. J. Knottenbelt, "Stochastic analysis of scheduling strategies in a GRID-based resource model," *IEE Software Engineering*, vol. 151, pp. 232–239, September 2004.

[13] D. R. W. Holton, "A PEPA specification of an industrial production cell," in *Process Algebra and Performance Modelling Workshop* (S. Gilmore and J. Hillston, eds.), vol. 38(7) of *Special Issue: The Computer Journal*, pp. 542–551, CEPIS, Edinburgh, June 1995.

[14] J. T. Bradley, N. J. Dingle, S. T. Gilmore, and W. J. Knottenbelt, "Derivation of passage-time densities in PEPA models using ipc: the Imperial PEPA Compiler," in *MASCOTS'03, Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems* (G. Kotsis, ed.), (University of Central Florida), pp. 344–351, IEEE Computer Society Press, October 2003.

[15] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.