Using ODEs from PEPA models to derive asymptotic solutions for a class of closed queueing networks

Nigel Thomas
School of Computing Science, Newcastle University, UK
Email: Nigel.Thomas@ncl.ac.uk

***Abstract*—In this paper a class of closed queueing network is modelled in the Markovian process algebra PEPA. It is shown that a fluid flow approximation using ordinary differential equations (ODEs) gives rise to well known asymptotic results. This result gives context to the use of a fluid flow approximation and is potentially useful in cases where the model is not obviously a closed queueing network. The approach is illustrated using examples of a secure key distribution centre and a multi-user query processing system.**

I. Introduction

Since the introduction of stochastic process algebra, there have been many attempts to tackle the state space explosion problem caused by the composition of many parallel components (see [10] for example). One of the more recent approaches to the issue has been the introduction of fluid flow approximations from systems biology to tackle models where there is a large number of instances of a particular component. Such an approach gives rise to a system of ordinary differential equations, which are generally solved by simulation. Following original work by Hillston [11] on biological systems, this style of fluid approximation has also been applied to more traditional computer applications e.g. [5], [17]. However, this approach has met with some scepticism amongst some computer scientists for a number of reasons. The main criticism is that the approximation maps a stochastic model specification on to a deterministic representation, thus the intrinsic randomness of the system is lost. In addition it can be difficult to derive important computer performance measures such as utilisation (because the fluid is always flowing) and interpreting the behaviour of a continuous fragment of a component does not always make sense. For a full description of the application of fluid approximations to PEPA models, see [2], [7], [8], [11].

In this paper we use the ODE approach to derive an analytical solution to a class of model, specified using the Markovian process algebra PEPA [9]. It is shown that this solution is identical to that used for many years as an asymptotic solution to the mean value analysis of closed queueing networks. This has two clear benefits.

- By relating the fluid flow approximation to established results in queueing theory, we gain greater confidence in the use of ODEs as a solution method.
- By deriving the ODE solution directly from the PEPA model specification, the asymptotic results become easily available in the analysis of models which are not obviously closed queueing networks. Thus the applicability of the asymptotic solution is practically extended without the need for specialist knowledge or insight on the part of the modeller.

The paper is organised as follows. In the next section the model and its asymptotic solution are introduced, followed by the PEPA specification of the class of model under investigation. We show how the ODEs can be derived from the PEPA specification and solved analytically to give the asymptotic solution. The process is illustrated by means of two examples; a secure key exchange protocol and a multi-user query processing system. Finally, some conclusions are drawn and potential future work is discussed.

II. The model and its asymptotic solution

Consider a model of a closed queueing network of N jobs circulating around M service stations, denoted 1 to M; each station is either a queueing station or an infinite server station. There are M_q queueing stations. At each queueing station, i, there is an associated queue (bounded at N) operating a FCFS policy and K_i servers which serve jobs at rate r_i. At each infinite server station, j, jobs experience a random delay with mean $1/r_j$. All services are negative exponentially distributed. Let $\mathcal{M} = \{1, 2, \ldots, M\}$ be the set of all queueing stations.

Queueing models of this form have traditionally been solved using mean value analysis [14]. However, this solution becomes costly when N is large and so a number of approximations have been proposed. The simplest amongst these is the asymptotic bound (see Haverkort [6] pp. 245-247).

Define V_i to be the visit count, the ratio of visits made to station i relative to station 1 (hence $V_1 = 1$). Now consider the smallest possible population size, $N = 1$. This solitary job would find each queue empty and experience a delay of $1/r_i$ at each station i at each visit. Hence, the average number of jobs at station i when $N = 1$, $L_i(1)$, is given by the proportion of time a job spends there. Similarly, when $N > 1$ but still small, a job entering a station has a high probability that there will be at least one idle server, hence the delay at station i is still approximately $1/r_i$. The sum of all average queue lengths

must be N, hence,

$$L_i(N) \geq \frac{NV_i}{r_i \sum_{j=1}^{M} \frac{V_j}{r_j}} \quad (1)$$

Now consider the case when N is very large. Assume that there is one queueing station with less service capacity than all the other queueing stations, denoted by $i = 1$. Obviously as $N \to \infty$ the utilisation of this bottleneck station will approach 1 and its throughput will tend to $K_1 r_1$, i.e. it becomes saturated. Obviously, the throughput must balance across all stations, hence, $K_1 r_1 = V_i L_i r_i \ \forall i \geq 2$. Thus,

$$L_i(N) = \frac{K_1 r_1}{V_i r_i} \ , \ i \geq 2 \quad (2)$$

The sum of all queues must equal N, hence

$$L_1(N) \leq N - \sum_{i=2}^{M} \frac{K_1 r_1}{V_i r_i} \quad (3)$$

Thus, for station 1 the approximate average queue length when the population size is N is given by

$$Max \left[\frac{N}{r_1 \sum_{i=1}^{M} \frac{V_i}{r_i}}, N - \sum_{i=2}^{M} \frac{K_1 r_1}{V_i r_i} \right]$$

And for all other stations, i, the approximate average queue length when the population size is N is given by,

$$Max \left[\frac{NV_i}{r_i \sum_{j=1}^{M} \frac{V_j}{r_j}}, \frac{K_1 r_1}{V_i r_i} \right]$$

Clearly a very simple approximation for the average response time has been used to make this calculation for average queue length. However, if our goal is to find the average response time when the population size is N, $W(N)$, a much better estimate can be found by combining this approximation for $L_i(N-1)$ with the mean value analysis estimation to give,

$$W_1(N) = \frac{1}{r_1} + \frac{1}{r_1} Max \left[\frac{N-1}{r_1 \sum_{i=1}^{M} \frac{V_i}{r_i}}, N - 1 - \sum_{i=2}^{M} \frac{K_1 r_1}{V_i r_i} \right] \quad (4)$$

and,

$$W_i(N) = \frac{1}{r_i} + \frac{1}{r_i} Max \left[\frac{(N-1)V_i}{r_i \sum_{j=1}^{M} \frac{V_j}{r_j}}, \frac{K_1 r_1}{V_i r_i} \right] \ , \ 2 \leq i \leq M \quad (5)$$

This is essential a single step of the mean value analysis algorithm (the Nth step) with all previous steps replaced by the asymptotic approximation. The total average respone time, that is the average time from leaving node 1 to subsequently completing another service at node 1, can be found by summing $W_i(N)$ over all nodes, i.

It is a simple matter to consider the case where there is more than one bottleneck, although this is not a concern in this paper. Clearly these two sets of asymptotic results are most accurate at their extremes, i.e. when $N = 1$ or as $N \to \infty$. Thus the asymptotic solution is least accurate when the two curves for L_i meet. There are a number of other approximations and enhancements which seek to improve accuracy and applicability without the additional computational cost associated with mean value analysis.

This form of simple approximation is generally applied across the entire network to derive measures such as system throughput and response time. However, it may also be applied to single nodes as outlined above. In such situations the accuracy of the approximation is greatly variable. The approximation generally works well when queueing only has a significant effect at one station. This situation arises when there is only one queueing station (the remainder being infinite service stations) or where one queueing station has much less service capacity than the others, relative to load.

III. A CLASS OF CLOSED QUEUEING NETWORKS IN PEPA

The model introduced in Section II is now modelled in PEPA. Clearly there are many possible ways to model this system and the particular form of the PEPA model here is a design choice. In particular, the model has been specified in such a way that the ODEs can be derived easily (without further transformation) and all behaviours are named, to improve clarity.

In PEPA a queue station can be modelled as

$$QStation_i \stackrel{def}{=} (service_i, r_i).QStation_i \ , \ \forall i \in \mathcal{M}$$

The infinite server stations are not represented explicitly.

Each job will receive service from a sequence of stations determined by a set of routing probabilities,

$$Job_i \stackrel{def}{=} \sum_{j=1}^{M} (service_i, P_{ij}(i) r_i).Job_j \ , \ 1 \leq i \leq M$$

Where,

$$\sum_{j=1}^{M} P_{ij}(i) = 1 \ , \ 1 \leq i \leq M$$

The entire system can then be represented as follows:

$$\left(\prod_{\forall i \in \mathcal{M}} QStation_i[K_i] \right) \underset{\mathcal{L}}{\bowtie} Job_1[N]$$

Where $\mathcal{L}$ is the set of all action types $service_i$ where $i \in \mathcal{M}$.

The ODEs for such a system are relatively simple to derive directly:

$$\begin{aligned} \frac{d}{dt} Job_i &= \sum_{\forall j \notin \mathcal{M}} P_{ji}(j) r_j Job_j(t) \\ &\quad + \sum_{\forall j \in \mathcal{M}} P_{ji}(j) r_j \ min[K_j, Job_j(t)] \\ &\quad - r_i Job_i(t) \ , \ \forall i \notin \mathcal{M} \end{aligned}$$

$$\begin{aligned} \frac{d}{dt} Job_i &= \sum_{\forall j \notin \mathcal{M}} P_{ji}(j) r_j Job_j(t) \\ &\quad + \sum_{\forall j \in \mathcal{M}} P_{ji}(j) r_j \ min[K_j, Job_j(t)] \\ &\quad - min[K_i, Job_i(t)] r_i \ , \ \forall i \in \mathcal{M} \end{aligned}$$

ODEs such as these can be solved in a number of ways. Most commonly they would be simulated with a suitably small time step to find $Job_i(t)$. In general, this quantity tends to a constant value as $t \to \infty$, i.e. it has a steady state solution.

An alternative method for finding $\lim_{t\to\infty} Job_i(t)$ is to solve the ODEs analytically. Such a solution is based on the assumption that the system of ODEs will eventually reach a steady state. Thus the derivatives will tend to zero as t tends to ∞, i.e.

$$\lim_{t\to\infty} \frac{d}{dt} Job_i \to 0 \ , \ 1 \leq i \leq M$$

This gives rise to the following set of simple simultaneous equations:

$$\begin{aligned}
&\lim_{t\to\infty} \sum_{\forall j \notin \mathcal{M}} P_{ji}(j) r_j Job_j(t) \\
&+ \sum_{\forall j \in \mathcal{M}} P_{ji}(j) r_j \ min[K_j, Job_j(t)] \\
&= \lim_{t\to\infty} r_i Job_i(t) \ , \ \forall i \notin \mathcal{M}
\end{aligned}$$

$$\begin{aligned}
&\lim_{t\to\infty} \sum_{\forall j \notin \mathcal{M}} P_{ji}(j) r_j Job_j(t) \\
&+ \sum_{\forall j \in \mathcal{M}} P_{ji}(j) r_j \ min[K_j, Job_j(t)] \\
&= \lim_{t\to\infty} r_i \ min[K_i, Job_i(t)] \ , \ \forall i \in \mathcal{M}
\end{aligned}$$

There are M equations, but each equation can be expanded (with respect to min function) in up to 2^{M_q} ways.

Define the probability that a component will evolve from Job_i to Job_j, without revisiting Job_i, as follows:

$$P_{ij}(i) = p_{ij} + \sum_{\forall k \notin \sigma} p_{ik} P_{kj}(i)$$

Clearly the system is irreducible if

$$P_{ij}(i) > 0 \ \forall i, j \ , \ i \neq j$$

Define L_i to be the steady state average number of components behaving as Job_i, given by

$$L_i = \lim_{t\to\infty} Job_i$$

Now select $i \in \mathcal{M}$ such that[1]

$$P_{ij}(i) r_i K_i < P_{ji}(j) r_j K_j \ \forall j \in \mathcal{M} \ , \ j \neq i \qquad (6)$$

Hence,

- If $K_i \leq L_i$ then

$$P_{ij}(i) r_i K_i = P_{ji}(j) r_j L_j \ , \ \forall j \qquad (7)$$

- If $K_i > L_i$ then

$$P_{ij}(i) r_i Job_i = P_{ji}(j) r_j L_j \ , \ \forall j \qquad (8)$$

[1]If (6) does not hold then there is no unique solution for this fluid system except when N is very small. Instead the value of L_i will depend on the initial values $Job_i(0)$ $\forall i$.

Thus, if $K_i \leq L_i$ then

$$\begin{aligned}
L_j &= \frac{P_{ij}(i) r_i K_i}{P_{ji}(j) r_j} \ , \ \forall j \neq i & (9) \\
L_i &= N - \sum_{\forall j \neq i} L_j = N - \sum_{\forall j \neq i} \frac{P_{ij}(i) r_i K_i}{P_{ji}(j) r_j} & (10)
\end{aligned}$$

Otherwise, if $K_i \geq L_i$ then

$$\begin{aligned}
L_j &= \frac{P_{ij}(i) r_i}{P_{ji}(j) r_j} L_i \ , \ \forall j \neq i & (11) \\
L_i &= N - \sum_{\forall j \neq i} L_j = \frac{N}{1 + \sum_{\forall j \neq i} \frac{P_{ij}(i) r_i}{P_{ji}(j) r_j}} & (12)
\end{aligned}$$

Clearly (10) and (12) meet when $K_i = L_i$. This point is given by the population size $N = N^*$, given by

$$N^* = K_i \left(1 + \sum_{\forall j \neq i} \frac{P_{ij}(i) r_i}{P_{ji}(j) r_j} \right)$$

Thus, if $N \leq N^*$ then

$$\begin{aligned}
L_j &= \frac{P_{ij}(i) r_i}{P_{ji}(j) r_j} L_i \ , \ \forall j \neq i & (13) \\
L_i &= N - \sum_{\forall j \neq i} L_j = \frac{N}{1 + \sum_{\forall j \neq i} \frac{P_{ij}(i) r_i}{P_{ji}(j) r_j}} & (14)
\end{aligned}$$

Otherwise, if $N \geq N^*$ then

$$\begin{aligned}
L_j &= \frac{P_{ij}(i) r_i K_i}{P_{ji}(j) r_j} \ , \ \forall j \neq i & (15) \\
L_i &= N - \sum_{\forall j \neq i} L_j = N - r_i K_i \sum_{\forall j \neq i} \frac{P_{ij}(i)}{P_{ji}(j) r_j} & (16)
\end{aligned}$$

Clearly, the visit count is given by $V_i = P_{ij}(i)/P_{ji}(j)$. Hence (13), (14), (15) and (16), are equivalent to (1), (2) and (3). Observe also that (13) and (14) hold, with arbitrary i, regardless of the existence of (6) as long as $L_j \leq K_j \ \forall j$.

From these expressions for L_j we can derive the average response time at station j when the population size is N, $W_j(N)$ in the same was as 4 and 5.

$$\begin{aligned}
W_j(N) &= \frac{1}{r_j} \ , \ L_j(N-1) + 1 \leq K_j \\
W_j(N) &= \frac{L_j(N-1) + 1}{K_j r_j} \ , \ L_j(N-1) + 1 > K_j
\end{aligned}$$

Where $L_j(N)$ is the average number of jobs at station j when the population size is N. This computation for $W_j(N)$ is based on the queueing theory result of an arrival as random observer, see Mitrani [12] page 141 for example. If the random observer sees a free server, then the average response time will be the average service time. However, if the random observer sees all the servers busy, then the average response time will be the average service time plus the time it takes for one server to become available (including scheduling the other jobs waiting ahead of the random observer).

In addition we can derive an expression for utilisation at the bottleneck queueing station i, U_i, based on the flow into the station being equal to the available service.

$$U_i = \sum_{\forall j \neq i} \frac{P_{ji}(j) L_j}{r_j K_i r_i}$$

IV. Example 1: A Secure Key Distribution Centre

Consider a model of the classic Needham-Schroeder key distribution protocol (taken from [19]) specified as follows:

$$\begin{aligned}
KDC &\stackrel{def}{=} (response, r_p).KDC \\
\\
Alice_0 &\stackrel{def}{=} (request, r_q).Alice_1 \\
Alice_1 &\stackrel{def}{=} (response, r_p).Alice_2 \\
Alice_2 &\stackrel{def}{=} (sendBob, r_B).Alice_3 \\
Alice_3 &\stackrel{def}{=} (sendAlice, r_A).Alice_4 \\
Alice_4 &\stackrel{def}{=} (confirm, r_c).Alice_5 \\
Alice_5 &\stackrel{def}{=} (usekey, r_u).Alice_0
\end{aligned}$$

The system is then defined as:

$$KDC[K] \underset{response}{\bowtie} Alice_0[N]$$

Where, K is the number of *KDC*'s and N is the number of client pairs (*Alices*'s).

It is a simple matter to write down the ODEs for this system as follows.

$$\begin{aligned}
\frac{d}{dt}Alice_0 &= r_u Alice_5(t) - r_q Alice_0(t) \\
\frac{d}{dt}Alice_1 &= r_q Alice_0(t) - r_p\ min(KDC(t), Alice_1(t)) \\
\frac{d}{dt}Alice_2 &= r_p\ min(KDC(t), Alice_1(t)) - r_B Alice_2(t) \\
\frac{d}{dt}Alice_3 &= r_B Alice_2(t) - r_A Alice_3(t) \\
\frac{d}{dt}Alice_4 &= r_A Alice_3(t) - r_c Alice_4(t) \\
\frac{d}{dt}Alice_5 &= r_c Alice_4(t) - r_u Alice_5(t) \\
\frac{d}{dt}KDC &= 0
\end{aligned}$$

In this analysis we are interested primarily in the number of client pairs awaiting a response from the *KDC* (or *KDC*'s) from a population of size N, which we denote as $L(N)$ This is represented in the model by the number of $Alice_1$'s; $L(N) = \lim_{t\to\infty} Alice_1(t)$ when there are N client pairs (*Alice*'s) in the population.

If the system reaches a steady state then all the derivatives will tend to zero as t tends to ∞, i.e.

$$\lim_{t\to\infty} \frac{d}{dt} Alice_i \to 0\ ,\ 0 \leq i \leq 5$$

Hence,

$$\begin{aligned}
\lim_{t\to\infty} r_p\ min(KDC(t), Alice_1(t)) &= \lim_{t\to\infty} r_B Alice_2(t) \\
&= \lim_{t\to\infty} r_A Alice_3(t) \\
&= \lim_{t\to\infty} r_c Alice_4(t) \\
&= \lim_{t\to\infty} r_u Alice_5(t) \\
&= \lim_{t\to\infty} r_q Alice_0(t)
\end{aligned}$$

Thus we only need to solve this set of simple parallel equations to find $L(N)$. If $KDC(t) \geq Alice_1(t)$ then the ODEs give rise to

$$L(N) = \lim_{t\to\infty} Alice_1 = \frac{N r_x}{r_x + r_p} \tag{17}$$

If $KDC(t) \leq Alice_1(t)$ then the ODEs give rise to

$$L(N) = \lim_{t\to\infty} Alice_1 = \frac{N r_x - K r_p}{r_x} \tag{18}$$

Where r_x is given by

$$r_x = \left(\frac{1}{r_q} + \frac{1}{r_B} + \frac{1}{r_A} + \frac{1}{r_c} + \frac{1}{r_u} \right)^{-1} \tag{19}$$

(17) and (18) meet when $KDC(t) = Alice_1(t)$ for a given population size N^*, hence, with (19) we get,

$$N^* = K + \frac{K r_p}{r_x} = K + K r_p \left(\frac{1}{r_q} + \frac{1}{r_B} + \frac{1}{r_A} + \frac{1}{r_c} + \frac{1}{r_u} \right)$$

Figure 1 shows the average response time of the *KDC*, found approximately using (17) and (18) and computed exactly using mean value analysis [16]. Clearly, when the service rate is smaller, the response time is larger and its rate of increase is larger. As noted above, there is a difference between the two solutions around N^*, which is clearly evident.

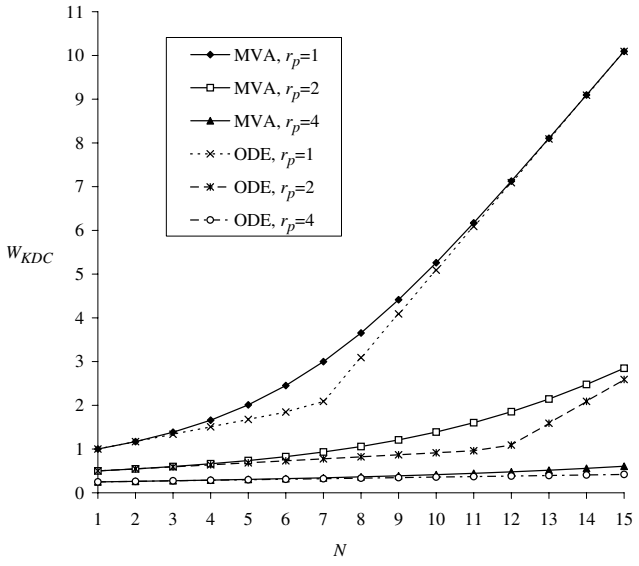

Fig. 1. Average response time at the *KDC* varied with population size ($r_q = r_B = r_A = r_c = 1$, $r_u = 1.1$, $K = 1$)

Figure 2 shows the average queue length at the *KDC*, L_{KDC} for this system when there is either one fast server or K slower servers. When the population size is large ($N > 30$ in this case) the *KDC* becomes saturated and there is consequently no difference in the service rate offered between the two cases shown. However, when N is smaller, there will be periods where one or more of the K servers will be idle, thus reducing the overall service capacity offered. Hence, for smaller N, a single fast server will out perform multiple slower servers with the same overall capacity. Once again, there is a clear divergence around N^*.

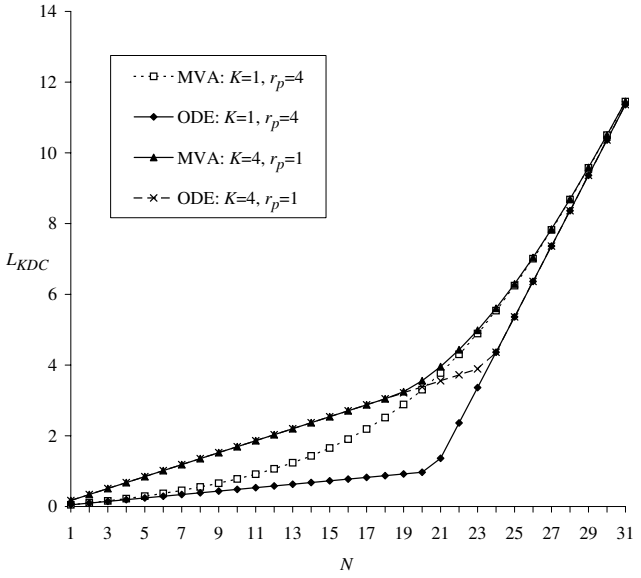

Fig. 2. Average queue length at *KDC* varied with population size ($r_q = r_B = r_A = r_c = 1$, $r_u = 1.1$)

V. Example 2: A Multi-User Query Processing System

Consider the following PEPA specification of a classic model taken from Lazowska et al [13].

$$\begin{aligned} Proc &\stackrel{def}{=} (service, \mu).Proc \\ Disk &\stackrel{def}{=} (write, \eta).Disk \end{aligned}$$

$$\begin{aligned} User_1 &\stackrel{def}{=} (think, \xi).User_2 \\ User_2 &\stackrel{def}{=} (service, p\mu).User_1 \\ &\quad +(service, (1-p)\mu).User_3 \\ User_3 &\stackrel{def}{=} (write, \eta).User_1 \end{aligned}$$

The entire system is then specified as

$$(Proc||Disk[K]) \underset{\{service,\ write\}}{\bowtie} User_1[N]$$

This system depicts a processor and an array of K independent disks. Users request a service from the processor. After this they either think for a while, before making another request, or their result requires writing to a disk before thinking and then another request.

The ODEs are given as

$$\begin{aligned} \frac{d}{dt}User_1(t) &= p\mu\ min[1, User_2(t)] \\ &\quad +\eta\ min[K, User_3(t)] \\ &\quad -\xi User_1(t) \\ \frac{d}{dt}User_2(t) &= \xi User_1(t) - \mu\ min[1, User_2] \\ \frac{d}{dt}User_3(t) &= (1-p)\mu\ min[1, User_2(t)] \\ &\quad -\eta\ min[K, User_3(t)] \end{aligned}$$

If the system reaches a steady state then all the derivatives will tend to zero as t tends to ∞, i.e.

$$\lim_{t \longrightarrow \infty} \frac{d}{dt} User_i \longrightarrow 0\ ,\ 0 \leq i \leq 5$$

Define $L_i = \lim_{t \longrightarrow \infty} User_i$ to be the steady state average number of users at each point in the system. Hence,

$$\begin{aligned} p\mu\ min[1, L_2] + \eta\ min[K, L_3] &= \xi L_1 \\ \xi L_1 &= \mu\ min[1, L_2] \\ (1-p)\mu\ min[1, L_2] &= \eta\ min[K, L_3] \end{aligned}$$

There are two possible bottlenecks in this system If $(1-p)\mu < K\eta$ then the bottleneck is the processor.

- If $1 \leq L_2$ then

$$\begin{aligned} L_1 &= \frac{\mu}{\xi} \\ L_3 &= \frac{(1-p)\mu}{\eta} \\ L_2 &= N - \frac{\mu}{\xi} - \frac{(1-p)\mu}{\eta} \end{aligned}$$

- If $1 \geq L_2$ then

$$\begin{aligned} L_1 &= \frac{\mu}{\xi}L_2 \\ L_3 &= \frac{(1-p)\mu}{\eta}L_2 \\ L_2 &= N - \frac{\mu}{\xi}L_2 - \frac{(1-p)\mu}{\eta}L_2 \\ &= \frac{N\xi\eta}{\xi\eta + \mu\eta + \mu\xi(1-p)} \end{aligned}$$

In this case,

$$N^* = \frac{\xi\eta + \mu\eta + \mu\xi(1-p)}{\xi\eta}$$

Alternatively, if $(1-p)\mu > K\eta$ then the bottleneck is writing to the disks.

- If $K \leq L_3$ then

$$\begin{aligned} L_1 &= \frac{\eta K}{(1-p)\xi} \\ L_2 &= \frac{\eta K}{(1-p)\mu} \\ L_3 &= N - \frac{\eta K}{\xi(1-p)} - \frac{\eta K}{\mu(1-p)} \end{aligned}$$

- If $K \geq L_3$ then

$$\begin{aligned} L_1 &= \frac{\eta}{(1-p)\xi} L_3 \\ L_2 &= \frac{\eta}{(1-p)\mu} L_3 \\ L_3 &= N - \frac{\eta}{(1-p)\xi} L_3 - \frac{\eta}{(1-p)\mu} L_3 \\ &= \frac{N\xi\mu(1-p)}{\xi\mu(1-p) + \eta\mu + \eta\xi} \end{aligned}$$

In this case,

$$N^* = K + \frac{\eta K}{\xi(1-p)} + \frac{\eta K}{\mu(1-p)}$$

If $(1-p)\mu = K\eta$ then the solution will depend on the initial values of $User_1(0)$, $User_2(0)$ and $User_3(0)$, unless,

$$\frac{N\xi\eta}{\xi\eta + \mu\eta + \mu\xi} \leq 1$$

and,

$$\frac{N\xi\mu}{\xi\mu + K\eta\mu + K\eta\xi} \leq K$$

In which case L_i is given by

$$\begin{aligned} L_1 &= \frac{N\mu\eta}{\xi\eta + \mu\eta + \mu\xi} \\ L_2 &= \frac{N\xi\eta}{\xi\eta + \mu\eta + \mu\xi} \\ L_3 &= \frac{N(1-p)\mu\xi}{\xi\eta + \mu\eta + \mu\xi} \end{aligned}$$

Figures 3 and 4 show the average queue lengths at the processor and the disk array for various values of p, where the processor is the bottleneck (Figure 3) and where the disk array is the bottleneck (Figure 4). In both cases results are shown as calculated by the ODE method in this paper and the mean value analysis method from [16].

It can be seen that when p is relatively large, the approximation works well (except around N^*). Whereas when p is smaller, particularly when p is close to 0.5, it is much poorer, and even diverging with N. It might perhaps be surprising that the ODE and MVA results are not closer when $p = 0.1$. After all, in this scenario, most jobs will visit the disk array and experience a long delay there. However, even when $p = 0.1$ queueing effects still have an effect at the processor and this causes a difference between the two methods.

Clearly, the accuracy of the ODE approximation of average queue length is sensitive to p. However, as stated earlier, the asymptotic solution is generally applied across the entire network, and not at an individual station. Therefore it is interesting to observe the accuracy of system wide metrics. Figure 5 shows the average response time for the entire system, computed as

$$W = \sum_{i=1}^{M} V_i W_i$$

Where V_i is the visit count and $min[V_1, \ldots, V_M] = 1$.

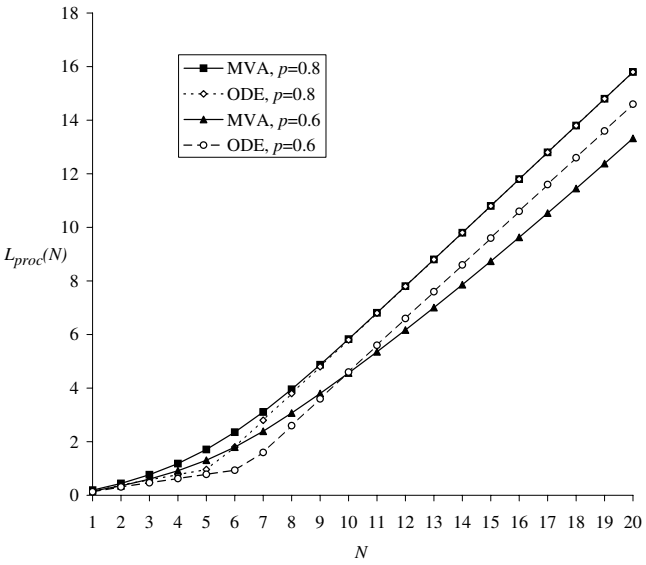

Fig. 3. Average queue length at processor and disk array varied with population size ($\xi = 10\mu = 30, \eta = 5,\ K = 3$)

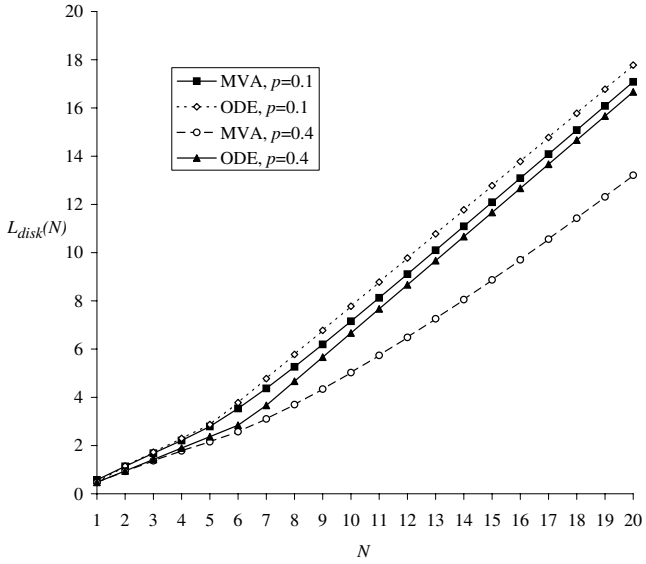

Fig. 4. Average queue length at processor and disk array varied with population size ($\xi = 10\mu = 30, \eta = 5,\ K = 3$)

Clearly the system response time is much less sensitive to the errors in the average number of jobs in each queue than we might naively expect. Indeed, Figure 5 shows only a very small divergence between MVA and ODE calculations, even when $p = 0.4$ (the worst case in the earlier graphs). The explanation for this is relatively simple, in that the maximum error in predicting the queue lengths is caused when the service capacity at each queueing station are relatively similar. Hence, when computing the average response time, we replace a delay at one station with a very similar delay at the other. As such, the errors, to an extent, disappear when aggregated across the

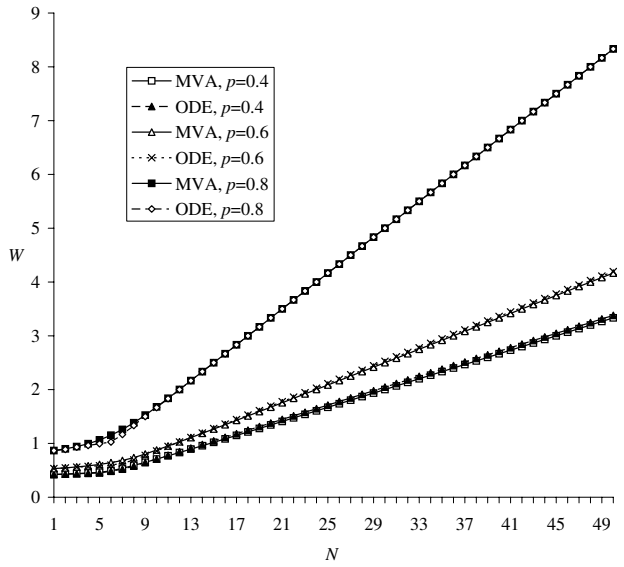

Fig. 5. Average system response time varied with population size ($\xi = 10\mu = 30, \eta = 5$, $K = 3$)

whole network in this way.

VI. Conclusions and further work

It has been demonstrated that the fluid approximation for a class of PEPA models coincides with the well known asymptotic approximation for a corresponding class of closed queueing network. This result is potentially useful as an alternative means for characterising models of queueing networks specified using PEPA, particularly when the model specified is not obviously a queueing model. This is the first class of fluid PEPA model for which there is an explicit expression for where the fluid solution is least accurate with respect to the exact solution of the stochastic model. The derivation of ODEs used in this paper is based on the work of Hillston [11] and is incorporated into the PEPA Eclipse Plug-in [18], facilitating easy numerical solution.

The result in this paper is limited to a class of cyclic queueing model where each station can perform just one action type. However, the asymptotic approximation applies to a much wider class of model. Thus it should be possible to extend this result to consider PEPA models with multiple competing action types at each Job_i derivative, each occurring at different rates but with the overall rate capped by a $QStation_j$ component. In addition, the asymptotic result holds for general service distributions, suggesting that the applicability of the fluid approximation in PEPA potentially extends beyond its conventional Markovian semantics. These investigations are left as ongoing work.

The result here is also limited to the case where there is a single bottleneck, unless the population is small enough that $L_i \leq K_i \ \forall i$. It is clearly possible to easily compute the average queue length for systems where there is more than one bottleneck, however it is not possible to find a unique solution directly from the ODEs, which is the aim of this paper. Furthermore, the approximation is shown to be accurate only when there are significant queueing effects at one station only. This gives some further insight as to the kind of model where ODE analysis is (in)appropriate.

Acknowledgements

The author is indebted to A. Clark, A. Duguid, S. Gilmore and M. Tribastone of the University of Edinburgh for invaluable comments on earlier work which contributed to this paper, in particular for clarifying aspects of the PEPA Eclipse Plug-in and the apparent rate. The example of the Key Distribution Centre is based on earlier work by Zhao and Thomas [19].

References

[1] J. Bradley, S. Gilmore and N. Thomas, Performance analysis of Stochastic Process Algebra models using Stochastic Simulation, in: *Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium*, IEEE Computer Society, 2006.

[2] J. Bradley, A ticking clock: Performance analysis of a Circadian rhythm with stochastic process algebra, in: C. Juiz and N. Thomas (eds.), *Computer Performance Evaluation: 5th European Performance Engineering Workshop*, LNCS 5261, Springer Verlag, 2008.

[3] A. Clark, A. Duguid, S. Gilmore and M. Tribastone, Partial evaluation of PEPA models for fluid-flow analysis, in: *Computer Performance Engineering: Proceedings of the 5th European Workshop on Performance Engineering (EPEW)*, LNCS 5261, Springer-Verlag, 2008.

[4] G. Clark and S. Gilmore and J. Hillston and N. Thomas, *Experiences with the PEPA Performance Modelling Tools*, IEE Proceedings - Software, pp. 11-19, 146(1), 1999.

[5] A. Duguid, Coping with the Parallelism of BitTorrent: Conversion of PEPA to ODEs in dealing with State Space Explosion. in: E. Asarin and P. Bouyer (eds.), *Formal Modeling and Analysis of Timed Systems*, LNCS 4202, Springer-Verlag, 2006.

[6] B. Haverkort, *Performance of Computer Communication Systems: A model Based Approach*, Wiley, 1998.

[7] Richard Hayden, Addressing the state space explosion problem for PEPA models through fluid-flow approximation, Undergraduate Project Dissertation, Imperial College London, 2007.

[8] R. Hayden and J. Bradley, Fluid-flow solutions in PEPA to the state space explosion problem, ValueTools 2008.

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

[10] J. Hillston, Exploiting Structure in Solution: Decomposing Compositional Models, in: E. Brinksma *et al*, *Lectures on Formal Methods and Performance Analysis*, LNCS 2090, Springer-Verlag, 2003.

[11] J. Hillston, Fluid flow approximation of PEPA models, in: *Proceedings of QEST'05*, pp. 33-43, IEEE Computer Society, 2005.

[12] I. Mitrani, *Probabilistic Modelling*, Cambridge University Press, 1998.

[13] E. Lazowska, J. Zahorjan, S. Graham and K. Sevcik, *Quantitative System Performance*, Prentice-Hall, 1984.

[14] M. Reiser and S. Lavenberg, Mean value analysis of closed multichain queueing networks, *JACM*, **22**(4), pp. 313-322, 1980.

[15] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice Hall, 1999.

[16] N. Thomas and Y. Zhao, Mean value analysis for a class of PEPA models, in: Proceedings of 6th European Performance Engineering Workshop, LNCS 5652, Springer-Verlag, 2009.

[17] N. Thomas and Y. Zhao, Fluid flow analysis of a model of a secure key distribution centre, in: *Proceedings 24th Annual UK Performance Engineering Workshop*, Imperial College London, 2008.

[18] M. Tribastone, The PEPA plug-in project, in: *Proceedings of 4th International Conference on the Quantitative Evaluation of Systems (QEST)*, pp. 53-54, IEEE Computer Society, 2007.

[19] Y. Zhao and N. Thomas, Approximate solution of a PEPA model of a key distribution centre, in: *Performance Evaluation - Metrics, Models and Benchmarks: SPEC International Performance Evaluation Workshop*, pp. 44-57, LNCS 5119, Springer-Verlag, 2008.