Mean Value Analysis for a Class of PEPA Models

Nigel Thomas and Yishi Zhao

School of Computing Science, Newcastle University, UK
{Nigel.Thomas,Yishi.Zhao}@ncl.ac.uk

Abstract. In this paper a class of closed queueing network is modelled in the Markovian process algebra PEPA and solved using the classical Mean Value Analysis (MVA). This approach is attractive as it negates the need to derive the entire state space, and so certain metrics from large models can be obtained with little computational effort. The class of model considered includes models which are not obviously classical closed queueing models. The approach is illustrated with three examples.

1 Introduction

There have been many attempts to find efficient solutions to large stochastic process algebra (SPA) models. SPA models suffer from the well known problem of state space explosion, where each additional component cause a multiplicative increase in the size of the global state space. This problem is particularly significant when there are many instances of the same type of component (so-called *massively parallel systems*). Such models may be extremely concise to specify, but even when the state space is folded or lumped, it may still far exceed the capacity available for solution.

Many of the approaches to efficiently solving SPA models have been based on concepts of decomposition originally derived for queueing networks [4]. Applying such approaches to stochastic process algebra allows the concepts to be understood in a more general modelling framework and applied to non-queueing models. Hillston [5] took an alternative, inspired by systems biology, approach by deriving a fluid approximation based on ordinary differential equations. Recently, Thomas [12] showed that such a fluid approximation is equivalent to a well known asymptotic solution for a class of closed queueing network (similar to the class of model considered in this paper). Traditionally this asymptotic solution was used as a computationally cheap alternative to mean value analysis [8] for very large populations. As such, it is clear that the class of model considered in [12] is also amenable to solution by mean value analysis. In this paper we make such an application and in doing so consider an extension to the class of model studied earlier [11,12].

Mean value analysis (MVA) is a method for deriving performance metrics based on steady state averages directly from the queueing network specification, without the need to derive any of the underlying Markov chain. As such it is relatively computationally efficient as long as the population size is not excessively large.

This paper is organised as follows. In the next section a brief overview of PEPA is given. The subsequent section then defines the class of model under consideration and gives the MVA solution of this class. Three examples are then used to illustrate the approach and to explore some numerical results. Finally some conclusions are drawn and some further work discussed.

2 PEPA

A formal presentation of PEPA is given in [3], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that occur with rates that are negative exponentially distributed. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the type of the activity, α, and the rate of the associated negative exponential distribution, r. This rate may be any positive real number, or given as unspecified using the symbol $\top$. It is important to note that in this paper the unspecified rate is not used.

The syntax for describing components is given as:

$$A \mid (\alpha, r).P \mid P + Q \mid P/L \mid P \underset{\mathcal{L}}{\bowtie} Q$$

$A \overset{def}{=} P$ gives the constant A the behaviour of the component P. The component $(\alpha, r).P$ performs the activity of type α at rate r and then behaves like P. The component $P + Q$ behaves either like P or like Q, the resultant behaviour being given by the first activity to complete.

Concurrent components can be synchronised, $P \underset{\mathcal{L}}{\bowtie} Q$, such that activities in the cooperation set $\mathcal{L}$ involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to activities of that type. The shorthand notation $P||Q$ is used to mean $P \underset{\emptyset}{\bowtie} Q$ and $\prod_{i=1}^{N} P_i$ is used to mean the parallel composition of the components P_i where i takes the values 1 through to N, i.e. $P_1|| \dots ||P_N$. In addition, we employ a further shorthand for synchronisation over many identical components, first introduced in [5], whereby $P[N]$ is taken to mean N copies of the component P, synchronised on the empty set, i.e. $P|| \dots ||P$ where there are N instances of component P.

The component P/L behaves exactly like P except that the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ. In this paper we do not make use of hiding, although non-shared actions could be hidden in a model if that was desirable. The action set $\mathcal{A}(P)$ is defined as the set of sections which are currently enabled in the derivative P.

In this paper we consider only models which are cyclic, that is, every derivative of components P and Q are reachable in the model description $P \underset{\mathcal{L}}{\bowtie} Q$. Necessary conditions for a cyclic model may be defined on the component and model definitions without recourse to the entire state space of the model.

3 A Class of Closed Queueing Networks in PEPA

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

There are J job types. Each job type can be served at most one station. When a job of type j completes a service of at a given station, it will proceed to service at a station (possibly the same station) as a job of type k according to some routing probability p_{jk}.

In PEPA a queue station can be modelled as

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

Note that r_i is always specified as finite, and not $\top$. This is because passive actions are subject to the *apparent rate* in PEPA. 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,

$$Job_i \overset{def}{=} \sum_{k=1}^{J} (service_j, p_{jk} r_j).Job_k \ , \ 1 \leq i, j \leq J$$

Where, $0 \leq p_{jk} \leq 1$ and

$$\sum_{k=1}^{J} p_{jk} = 1 \ , \ 1 \leq j \leq J$$

Denote $\mathcal{S}_i$ to be the set of all job types which perform $service_i$ actions, i.e. $\mathcal{S}_i = j$ if $service_i \in \mathcal{A}(Job_j)$.

The entire system can then be represented as follows:

$$\left(\prod_{\forall i \in \mathcal{M}} (QStation_i[K_i]) \right) \underset{\mathcal{L}}{\bowtie} Job_1[N] \qquad (1)$$

Where

$$\mathcal{L} = \bigcup_{\forall i \in \mathcal{M}} \{service_i\}$$

3.1 Mean Value Analysis

We now consider the *arrival theorem*, first derived independently by Sevcik and Mitrani [9] and Lavenberg and Reiser [7], applied to this class of PEPA model.

Theorem 1 ***Arrival Theorem.*** *Consider a component Job_i evolving into its successor derivative, Job_j in a system given by (1). The steady state distribution of the number of components behaving as any component Job_k at that moment is equal to the steady state distribution of the number of components behaving as Job_k in a system without the evolving job.*

The arrival theorem is as profound as it is simple and seemingly intuitive. It consequently gives rise to the well known *mean value analysis*, whereby the average behaviour of a system of N components may be derived from the average behaviour of a system of $N-1$ components. Therefore it is never necessary to derive a solution to the full CTMC if we are only concerned with the average behaviour of systems of this kind. This follows from the following set of relationships, derived following the pattern of Haverkort [2] pp. 241-245.

Theorem 1 implies that the average time a component spends in behaviour Job_j, denoted $W_j(N)$, where $\mathcal{A}(Job_j) = service_i$ and $i \in \mathcal{M}$, is given by the average number of Job_k ($\forall k \in \mathcal{S}_i$) components in a system with one fewer Job_l, $\forall i$, components in total. Denote $L_j(N)$ to be the steady state average number of components behaving as Job_j in a system with N jobs in total. If $\sum_{\forall i \in \mathcal{S}_j} L_i(N-1) \leq K_j - 1$ and $j \in \mathcal{M}$ then

$$W_i = \frac{1}{r_j}\ , \ \forall i \in \mathcal{S}_j \tag{2}$$

Otherwise, if $\sum_{\forall i \in \mathcal{S}_j} L_i(N-1) > K_j - 1$ and $j \in \mathcal{M}$ then

$$W_i = \frac{1 + \sum_{\forall i \in \mathcal{S}_j} L_i(N-1)}{K_j r_i} \tag{3}$$

Clearly, if $j \notin \bigcup_{\forall i \in \mathcal{M}} \mathcal{S}_\rangle$, then $W_j(N)$ is a constant, given as $W_j(N) = 1/r_j$.

We now need to compute a quantity generally referred to as the *visit count*, and denoted V_i. The visit count is the number of times derivative Job_i is visited, relative to the number of times some reference derivative Job_I is visited, where $1 \leq I \leq J$. The actual value of V_i is not crucial, rather its value relative to the value of V_I. As such the choice of I is strictly arbitrary.

We can compute the visit count from the routing probabilities p_{ij}. Define the probability that a component will evolve from Job_i to Job_j, without revisiting Job_i, as follows:

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

The set σ here contains only the starting and ending behaviours of interest, in this case i and j, i.e. it is used to tell us if we reach Job_j or first return to Job_i. For convenience define the shorthand,

$$P_{ij} = P_{ij}(\{i, j\})$$

By definition, $P_{ii} = 1$. Clearly the system is irreducible if

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

Now we choose some reference point I, such that,

$$V_i = \frac{P_{Ii}}{P_{iI}} \; , \; \forall i \neq I$$

and $V_I = 1$. Thus, V_i gives the number of times a component assumes the behaviour Job_i, relative to the number times it assumes the behaviour Job_I.

Given the quantity V_j, we can now compute the average response time per passage for a component behaving as Job_j.

$$\hat{W}_j(N) = V_j W_j(N) \tag{4}$$

From Little's theorem we know that

$$L_j(N) = X_j(N)W_j(N) = X(N)V_jW_j(N) = X(N)\hat{W}_j(N) \tag{5}$$

Where $X_j(N)$ is the observed rate of activity $service_j$ when the population size is N, and $X(N)$ is the sum of all possible $X_j(N)'s$.

Summing (5) over all behaviours Job_i, $i = 1, 2, \ldots, J$ gives,

$$\sum_{j=1}^{J} L_j(N) = X(N) \sum_{j=1}^{J} \hat{W}_j(N) = X(N)\hat{W}(N) = N$$

where $\hat{W}(N) = \sum_{j=1}^{J} \hat{W}_j(N)$. Thus,

$$X(N) = \frac{N}{\hat{W}(N)}$$

Hence, with Little's law applied for a given behaviour Job_j,

$$L_j(N) = X_j(N)W_j(N) = X(N)V_jW_j(N) = \frac{N}{\hat{W}(N)}\hat{W}_j(N) \tag{6}$$

We are now in a position to calculate $L_j(N)$ for any value of N if we can calculate $L_j(1)$. A solitary Job_i component will never compete for cooperation over the actions in $\mathcal{L}$, and so will experience a delay of $1/r_i$ in each derivative Job_i. Hence, the average number of components behaving as Job_j when $N = 1$, $L_j(1)$ is given by the proportion of time a component spends in that behaviour.

$$L_j(1) = \frac{V_j}{r_j \sum_{i=1}^{J} \frac{V_i}{r_i}} \tag{7}$$

We now apply the following iterative solution.

1. Calculate $L_j(1)$ for $j = 1, 2, \ldots J$, using (7).
2. $n = 2$
3. Compute $\hat{W}_j(n)$ for $j = 1, 2, \ldots J$, using (2), (3) and (4) and $L_j(n-1)$ from 1 above.
4. Compute $\hat{W}(n) = \sum_{j=1}^{J} \hat{W}_j(n)$.
5. Compute $L_j(n)$ for $j = 1, 2, \ldots J$, using (6) and $\hat{W}(n)$ from 4 above.
6. Increment n.
7. If $n \leq N$ return to step 3 else end.

Clearly this solution is not complicated to implement. For a system of J job types and N jobs it is necessary to compute $(2J+1)N$ distinct quantities. Hence, this will generally only be costly when N is extremely large.

4 Examples

In this section we explore the class of models introduced above, through three example PEPA models. Each example depicts a different aspect of this class. The first model is an abstract queueing model, with probabilistic branching on completion of service at one of the stations. The other two examples are practical models drawn from an ongoing area of study into performance modelling of security protocols. The first of these is a model of the classic Needham–Schroeder key distribution protocol. This model has no branching and so mean value analysis is applied easily. The second practical model is of a non-repudiation protocol. This model involves a single queueing station processing separate requests from two participants in an exchange.

4.1 Example 1: A Three Node Closed Queueing Network

Consider the following PEPA specification of a simple closed queueing network

$$Node1 \overset{def}{=} (service_1, \xi).Node_1$$
$$Node_2 \overset{def}{=} (service_2, \mu).Node_2$$
$$Node_3 \overset{def}{=} (service_3, \eta).Node_3$$

$$Request_1 \overset{def}{=} (service_1, \xi).Request_2$$
$$Request_2 \overset{def}{=} (service_2, (1-p)\mu).Request_1 + (service_2, p\mu).Request_3$$
$$Request_3 \overset{def}{=} (service_3, \eta).Request_1$$

The entire system is then specified as

$$(Node_1[K_1] || Node_2[K_2] || Node_3[K_3]) \underset{\{service_1, service_2, service_3\}}{\bowtie} Request_1[N]$$

This system depicts a three node closed queueing network where all three nodes are queueing stations. After completing service at node 1, all requests proceed to node 2. Following service at node 2, a proportion of requests, p, will return to node 1, whilst the remainder will be directed to node 3. All requests completing service at node 3 will return to node 1.

In this example it is a simple matter to compute the visit count for each node.

$$V_1 = 1$$
$$V_2 = 1$$
$$V_3 = p$$

There are clearly many possible approaches to implementing the iterative solution given above. For convenience this model has been solved in an Excel spreadsheet. Solutions with population sizes of over 10000 have been derived without any problems, although clearly a more efficient implementation is desirable for larger N when a range of parameter values are being considered.

Figure 1 shows the average queue size at node 3 varied with population size N for various values of p.

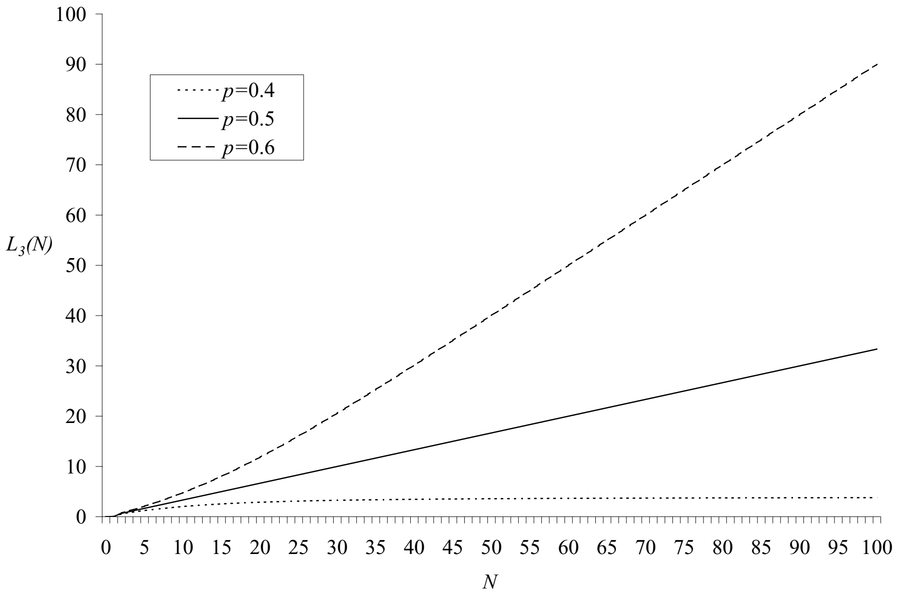

Fig. 1. Average queue length at node 3 varied with population size ($\xi = \mu = 10, \eta = 5$)

When $p = 0.5$ all three nodes have the same load, hence their queue sizes will be equal, i.e. $L_i = N/3$. Obviously, if p is less than 0.5 then node 3 will have a lower load than the other two nodes, hence it will have a smaller average queue length. In fact, the average queue length at node 3 will tend to a fixed value ($L_3(N) \rightarrow 4$ as $N \rightarrow \infty$ when $p = 0.4$). Conversely, if $p > 0.5$ then the third node will become the bottleneck of the system, and the majority of jobs will be queueing there. In the case $p = 0.6$, the average queue length at node 3 will tend to $N - 10$.

4.2 Example 2: A Secure Key Distribution Centre

Consider a model of the classic Needham-Schroeder key distribution protocol (taken from [14]) specified as follows:

$$KDC \overset{def}{=} (response, r_p).KDC$$

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

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).

In this model the component $Alice_i$ represents the actions of a pair of clients (normally referred to as *Alice* and *Bob*). The sequence of actions includes *Alice* requesting a session key from a secure server, known as the *key distribution centre* (KDC). This results in competition for the resources of the KDC amongst the various client pairs. Once the key has been issued to *Alice*, *Alice* and *Bob* exchange messages to confirm their mutual trust (established by shared trust of the KDC), before using the provided session key.

Clearly there is no branching, and so $V_i = 1$, $\forall i$. Furthermore there is only one queueing station, so this is always the bottleneck of the system unless K is large relative to N.

Figure 2 shows the average response time at the *KDC*, W_{KDC} for this system when there is one server for various service rates. Clearly, when the service rate is smaller, the response time is larger and its rate of increase is larger.

Figure 3 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 outperform multiple slower servers with the same overall capacity, as is well known from queueing theory.

4.3 Example 3: A Non-repudiation Protocol

Non-repudiation protocols are used to prevent participants in a communication from later falsely denying that they took part in that communication. There

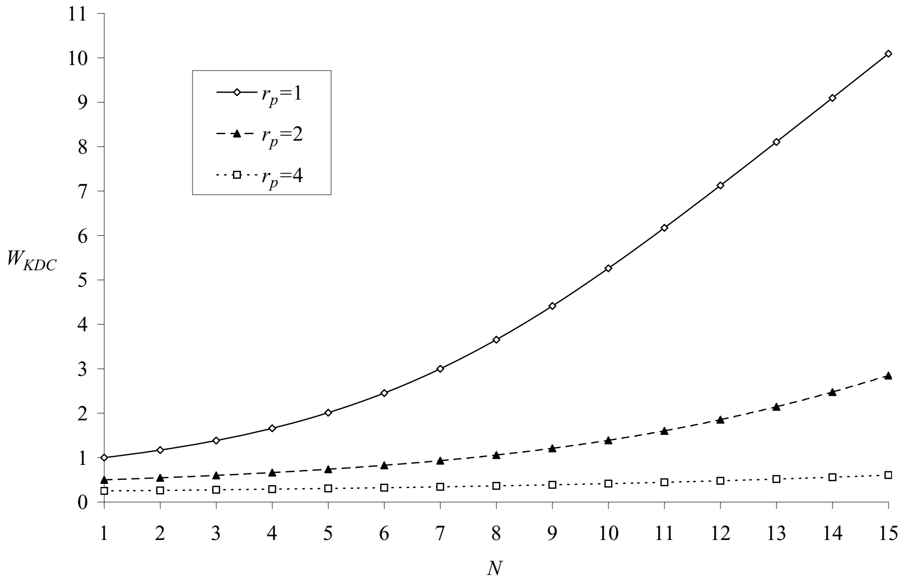

Fig. 2. Average response time at the *KDC* varied with population size ($r_q = r_B = r_A = r_c = 1$, $r_u = 1.1$, $K = 1$)

are many such protocols, with different properties. The one depicted here, first proposed by Zhou and Gollmann [15], utilizes a secure server, known as a *Trusted Third Party*, or TTP. Consider the following PEPA specification.

$$\begin{aligned}
TTP &\overset{def}{=} (publish, rp).TTP \\
AB_0 &\overset{def}{=} (request, rq).AB_1 \\
AB_1 &\overset{def}{=} (publish, rp).AB_2 \\
AB_2 &\overset{def}{=} (getByA_1, rga_1).AB_3 \\
AB_3 &\overset{def}{=} (sendB, rb).AB_4 \\
AB_4 &\overset{def}{=} (sendTTP, rttp).AB_5 \\
AB_5 &\overset{def}{=} (publish, rp).AB_6 \\
AB_6 &\overset{def}{=} (get, rgb).AB_7 + (get, rga_2).AB_8 \\
AB_7 &\overset{def}{=} (getByA_2, rga_2).AB_9 \\
AB_8 &\overset{def}{=} (getByB, rgb).AB_9 \\
AB_9 &\overset{def}{=} (work, rw).AB_0 \\
System &= TTP \underset{\{publish\}}{\bowtie} AB_0[N]
\end{aligned}$$

The model depicts a single TTP with N client pairs (*Alice* and *Bob*, denoted AB). The protocol utilises publishing in a public space (e.g. a bulletin board) by the TTP, from where the clients each download. In the specification

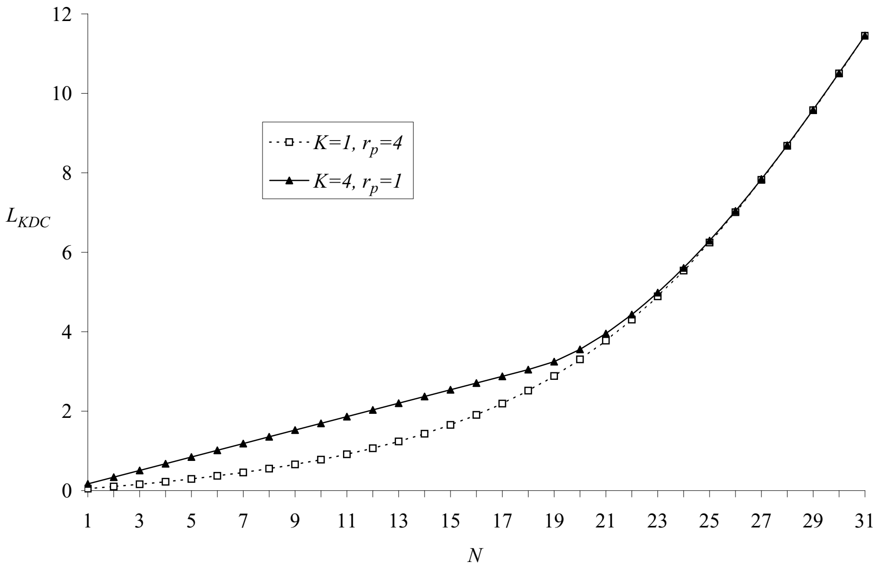

Fig. 3. Average queue length at *KDC* varied with population size ($r_q = r_B = r_A = r_c = 1$, $r_u = 1.1$)

above we are only concerned with contention on the publication by the TTP, although it would also be possible to specify an additional component to act as a web server.

In this model there is branching through a race on the *get* action in AB_6, and on service at the TTP (depending on which job type is served). Thus, the average number of components behaving as AB_1 when there are N client pairs, $L_1(N)$, depends the number of AB_1 and AB_5 when there are $N-1$ client pairs, $L_1(N-1)$ and $L_5(N-1)$. Likewise, the average number of components behaving as AB_6. The branching in AB_6 is specified over two *get* actions with different rates. These actions depict a race condition on *Alice* and *Bob* independently downloading from the bulletin board. In theory these could be given different names (they should ideally be called *getByB* and $getByA_2$ respectively) but that is not possible in the current characterisation of the class of model given in Section 3.

The visit count is identical for each behaviour AB_i, $V_i = 1$, except V_7 and V_8, given by,

$$V_7 = \frac{rgb}{rgb + rga_2}$$
$$V_8 = \frac{rga_2}{rgb + rga_2}$$

Note that the visit count for behaviours AB_1 and AB_5, V_1 and V_5, are 1. Since *publish* actions from both AB_1 and AB_5 are served by the TTP, the

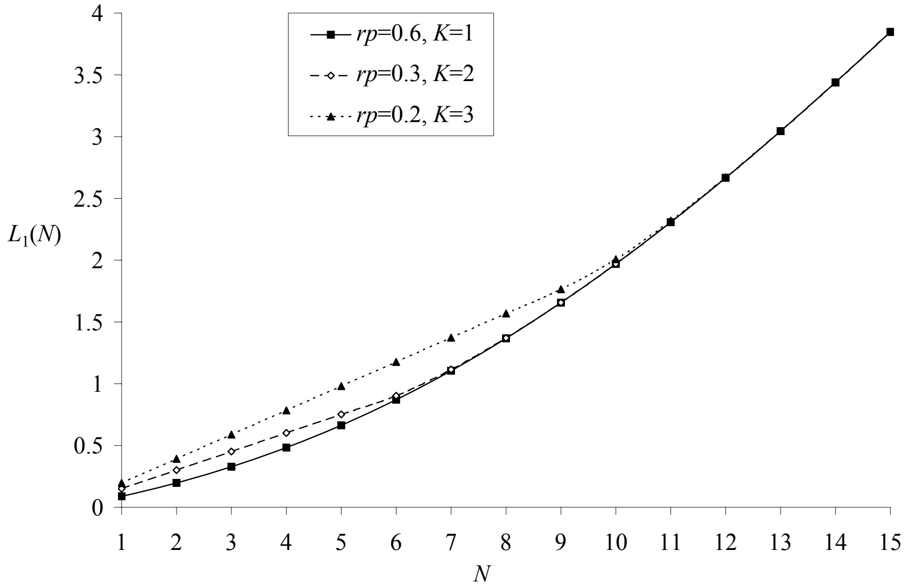

Fig. 4. Average number of AB_1 components varied with population size ($rq = rga_1 = rb = rttp = rga_2 = rgb = 1$, $rw = 0.01$)

'apparent visit count' of the TTP is effectively double that of the other stations, although we are not concerned with this quantity in our derivation of mean value analysis.

Figure 4 shows the average number of AB_1 components awaiting service at the TTP, for different values of service (publishing) rate, *rp*, varied with population size. Two parameter sets are used, identical except for the values of *rp*, although in each case the average service time is the same. Clearly, the values of $L_1(N)$ and $L_5(N)$ are the same, and so only $L_1(N)$ is shown. Clearly, the more jobs that are waiting in the queue, the longer an arriving job will have to wait. Thus, we find that the queue becomes longer at the TTP and that *proportionally* less time is spent performing the other actions, and so the throughput decreases. The figure shows a comparison between a single server and multiple servers with the same total service capacity. When the queue size is small then not all servers will be utilised; the more servers there are the more likely they are to be idle. Hence, there is linear growth of the queue when $K = 3$ and $N < 10$. In contrast, initially queue grows less quickly when there is only one (faster) server. However, once the population grows sufficiently for all servers to be highly utilised, then the three cases will show identical performance.

This situation is more clearly illustrated when looking at the response time for AB_1 components in Figure 5. When $K > 1$ the response time is static

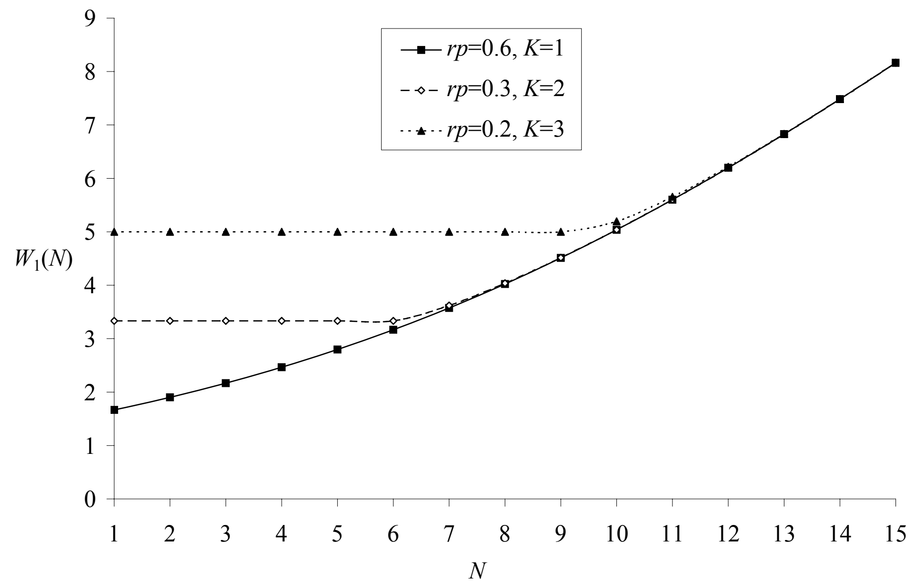

Fig. 5. Average response time for AB_1 components varied with population size ($rq = rga_1 = rb = rttp = rga_2 = rgb = 1$, $rw = 0.01$)

with N when the population is small, but once the population is large enough, the performance is once again shown to be the same in all cases.

5 Conclusions and Further Work

This paper demonstrates the solution of a class of PEPA models using classical Mean Value Analysis [8]. This gives a relatively computationally cheap method for solving large models without need to derive the state space of the underlying Markov chain. The CTMC for this class of model, even when lumped using standard techniques, is still large and generally grows at an exponential rate. Thus, when the number of instances of the Job_i components is extremely large, a CTMC solution is not going to be a feasible proposition. In contrast we have derived solutions to the example models here with over 10000 Job_i components, using a very simple (and not very efficient) spreadsheet based implementation on a relatively old laptop PC, in a fraction of a second. In the case of example 1 (Section 4.1) the lumped CTMC would have in excess of 50 million states when there are 10000 $Request_i$ components.

Clearly, the approach is limited in both the metrics that can be derived and also the class of model that is considered. The former limitation is a feature of mean value analysis (hence the name). However, the class of model could be extended in a number of ways. Mean value analysis applies to multiple classes of jobs in closed queueing network. Therefore it should be straightforward to define a class of model with different groups of components, each with potentially different action rates and routing probabilities. This would be a relatively simple extension of the current class, but would involve careful use of notation to distinguish classes in a meaningful way.

The final example considered in this paper, a non-repudiation protocol, highlights a situation where we may wish an action type to appear in more than one *job* component, and potentially different action types in the same *job* component. Such a limitation is largely cosmetic, as an equivalent model can be specified in the existing class, as is done here. However, it should be possible to incorporate this option with a further adjustment to the notation used in this paper.

Finally, it should be noted that there can only be one service action type at a station and that must be given the same rate in any job type where it is enabled. Although intuitively it is possible to model the case where there are more job types enabled at a queueing station, doing so potential introduces race conditions and therefore distorts the effective service rate. We are still considering how such a situation might be best specified and it may be more feasible to consider approximate solutions in this scenario.

References

1. Clark, G., Gilmore, S., Hillston, J., Thomas, N.: Experiences with the PEPA Performance Modelling Tools. IEE Proceedings–Software 146(1), 11–19 (1999)
2. Haverkort, B.: Performance of Computer Communication Systems: A Model-based Approach. Wiley, Chichester (1998)
3. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, Cambridge (1996)
4. Hillston, J.: Exploiting Structure in Solution: Decomposing Compositional Models. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.) EEF School 2000 and FMPA 2000. LNCS, vol. 2090, p. 278. Springer, Heidelberg (2001)
5. Hillston, J.: Fluid-flow approximation of PEPA models. In: Proceedings of QEST 2005, pp. 33–43. IEEE Computer Society Press, Los Alamitos (2005)
6. Mitrani, I.: Probabilistic Modelling. Cambridge University Press, Cambridge (1998)
7. Lavenberg, S., Reiser, M.: Stationary state space probabilities at arrival instants for closed queueing networks with multiple types of customers. Journal of Applied Probability 17(4), 1048–1061 (1980)
8. Reiser, M., Lavenberg, S.: Mean value analysis of closed multichain queueing networks. JACM 22(4), 313–322 (1980)
9. Sevcik, K., Mitrani, I.: The distribution of queueing network states at input and output instants. JACM 28(2), 358–371 (1981)
10. Stallings, W.: Cryptography and Network Security: Principles and Practice. Prentice-Hall, Englewood Cliffs (1999)
11. Thomas, N.: Mean value analysis for a class of PEPA models, Technical Report CS-TR-1128, School of Computing Science, Newcastle University (2008)
12. Thomas, N.: Using ODEs from PEPA models to derive asymptotic solutions for a class of closed queueing networks, Technical Report CS-TR-1129, School of Computing Science, Newcastle University (2008)

13. Thomas, N., Zhao, Y.: Fluid flow analysis of a model of a secure key distribution centre. In: Proceedings 24th Annual UK Performance Engineering Workshop, Imperial College London (2008)
14. Zhao, Y., Thomas, N.: Approximate solution of a PEPA model of a key distribution centre. In: Kounev, S., Gorton, I., Sachs, K. (eds.) SIPEW 2008. LNCS, vol. 5119, pp. 44–57. Springer, Heidelberg (2008)
15. Zhou, J., Gollmann, D.: An efficient non-repudiation protocol. In: Proceedings of the l0th Computer Security Foundations Workshop (CSFW 1997). IEEE Computer Society Press, Los Alamitos (1997)