Process-Algebraic Modelling of Priority Queueing Networks

Giuliano Casale
Department of Computing
Imperial College London, U.K.
gcasale@doc.ic.ac.uk

Mirco Tribastone
Institut für Informatik
Ludwig-Maximilians-Universität Munich, Germany
tribastone@pst.ifi.lmu.de

We consider a closed multiclass queueing network model in which each class receives a different priority level and jobs with lower priority are served only if there are no higher-priority jobs in the queue. Such systems do not enjoy a product form solution, thus their analysis is typically carried out through approximate mean value analysis (AMVA) techniques. We formalise the problem in PEPA in a way amenable to differential analysis. Experimental results show that our approach is competitive with simulation and AMVA methods.

1 Introduction

Closed multiclass queueing networks with priorities have been intensively studied since the early days of performance evaluation to assess the impact of priorities in I/O-bound workloads and in computing systems with background processes. Although these models are relatively well-understood, the recent diffusion of virtualised server architectures that allow to dynamically change the priority of different virtual machines and their I/O workloads raises the challenge of devising new solutions approaches that could match the needs of modern performance analyses, such as estimating response time distributions or predict the impact of generally-distributed service times. In this paper, we propose an initial study on the applicability and accuracy of fluid methods for PEPA models [5] to the problem of estimating performance indexes of closed queueing networks with priorities, with a focus on non-preemptive priority scheduling which is known to be harder to approximate than preemptive disciplines [3].

Related Work Existing techniques for the analysis of closed queueing networks with priorities include Sevcik's shadow server method [6] and approximations based on the arrival theorem used in mean value analysis (MVA) [6]. In the shadow server method, the service time of a low priority job is inflated to account for the capacity used by high priority classes. The inflation factor of a class depends on the steady-state utilisation of the classes with higher priorities and can be efficiently determined by iterative techniques [6, 3]. The shadow server method is effective for preemptive-resume (PR) priority scheduling, but it usually suffers larger errors under non-preemptive disciplines. The MVA priority approximation in [1] significantly improves over the shadow server method by developing estimates for the mean queue-length seen on arrival by a priority class and for inter-dependencies between throughputs of different priority classes in closed models. A comprehensive discussion of such results is given in [3] together with an extension to the approximate MVA (AMVA) framework [6] of both the shadow server technique (AMVA-SHA) and the MVA priority approximation (AMVA-CL). AMVA techniques leverage on fixed-point approximations of the MVA arrival theorem [6] and are generally more accurate than recursive methods since they avoid error propagation that affects the latter. Additionally, AMVA methods tend to be cheaper computationally, especially on models with large job populations. However, their main limitation is that they can be hard to generalise outside the initial assumptions. This is because the AMVA framework requires to develop accurate approximation for queue-lengths seen on arrival, residual

Submitted to:
PASTA 2010

service times, and interference effects between classes. In what follows, we try to address this limitation of approximation techniques for priority queueing networks by applying fluid analysis techniques.

2 PEPA Models

Problem Statement We consider a closed multiclass queueing network with J stations and I workload classes indexed by $j = 1,\ldots,J$ and $i = 1,\ldots,I$, respectively. Service rates of class j jobs at station i are exponential with rate $\mu_{i,j}$; the probability that a class i job visits station k immediately after completing service at station j is indicated with p_{ijk}. The total job population of class i is denoted by N_i. Stations are partitioned into two groups: labels $1,\ldots,D$ denote delay (infinite server) stations, labels $D+1,\ldots,J$ indicate priority queues with $M_j \geq 1$ servers. We focus on non-preemptive priority scheduling. We assume that jobs of class i have higher priority than jobs of class $i' > i$. However, the arrival of a class-i job at a station does not cause the removal of a class-i' job currently in service. This scheduling is known to be harder to model than preemptive-resume (PR) priority scheduling due to inter-dependencies between closed classes [3]. We now develop a PEPA model for a single-server queue and later integrate this definition into a PEPA queueing network specification.

PEPA Model of a Priority Station Without loss of generality, we initially discuss a workload with $I = 2$ classes and one job for each class. The server is approximated in PEPA as a two-state sequential component which admits a job of either class and then serves it (with exponentially-distributed rates μ_1 and μ_2), i.e.,

$$Admit \stackrel{def}{=} (admit, \nu).Serve, \qquad Serve \stackrel{def}{=} (serve_1, \mu_1).Admit + (serve_2, \mu_2).Admit. \tag{1}$$

where ν denotes the rate at which the empty server is replenished with a new job if one is available for admission. Intuitively, for $\nu \to +\infty$ the above approximation converges to the exact behaviour of a non-idling server.

We propose a separate PEPA model to describe jobs visiting the station. For a job of class i, we define the following states

$$Wait_i \stackrel{def}{=} (admit, w_i\nu).Run_i, \qquad Run_i \stackrel{def}{=} (serve_i, \mu_i).Wait_i, \qquad i \in \{1,2\},$$

where the weight w_i acts in the final model as the relative probability that the next admitted job to the queue is of class i. Based on the above definitions, the system equation

$$Sched \stackrel{def}{=} (Wait_1 \parallel Wait_2) \underset{L}{\bowtie} Admit, \qquad L = \{admit, serve_1, serve_2\},$$

captures the non-preemptive access to the server as follows. If the class-1 job is in the Run_1 state then the class-2 job must be in the $Wait_2$ state. The service completion of a job is also modelled correctly as

$$(Run_1 \parallel Wait_2) \underset{L}{\bowtie} Serve \xrightarrow{(serve_1,\mu_1)} (Wait_1 \parallel Wait_2) \underset{L}{\bowtie} Admit$$
$$(Wait_1 \parallel Run_2) \underset{L}{\bowtie} Serve \xrightarrow{(serve_2,\mu_2)} (Wait_1 \parallel Wait_2) \underset{L}{\bowtie} Admit$$

The *Sched* model can approximate a priority queue if a careful selection of the parameters w_1, w_2, and ν is made. Observe that the following two transitions are enabled from *Sched*:

$$(Wait_1 \parallel Wait_2) \underset{L}{\bowtie} Admit \xrightarrow{(admit, \frac{w_1}{w_1+w_2}\min((w_1+w_2)\nu,\nu))} (Run_1 \parallel Wait_2) \underset{L}{\bowtie} Serve$$
$$(Wait_1 \parallel Wait_2) \underset{L}{\bowtie} Admit \xrightarrow{(admit, \frac{w_2}{w_1+w_2}\min((w_1+w_2)\nu,\nu))} (Wait_1 \parallel Run_2) \underset{L}{\bowtie} Serve$$

which imply that the probability that a job is admitted into service are $w_1/(w_1+w_2)$ for class 1 and $w_2/(w_1+w_2)$ for class 2. It then follows that for sufficient large $w_1 \gg w_2$ one can accurately approximate that class-1 jobs have priority over class-2 jobs. Furthermore, if $w_1+w_2 \geq 1$, their actual values do not have any impact on the mean holding time of state *Sched*, which is $1/\nu$.

We naturally extend the above approximation approach to a station with many jobs and M identical servers with the following description

$$(Wait_1[N_1] \parallel Wait_2[N_2]) \underset{L}{\bowtie} Admit[M].$$

The state associated with this component has the following two transitions:

$$(Wait_1[N_1] \parallel Wait_2[N_2]) \underset{L}{\bowtie} Admit[M] \xrightarrow{\left(admit, \frac{N_1 w_1}{N_1 w_1 + N_2 w_2} \min(N_1 w_1 + N_2 w_2)\nu, M\nu\right)}$$
$$(Wait_1[N_1-1] \parallel Run_1 \parallel Wait_2[N_2]) \underset{L}{\bowtie} (Admit[M-1] \parallel Serve), \quad (2)$$

$$(Wait_1[N_1] \parallel Wait_2[N_2]) \underset{L}{\bowtie} Admit[M] \xrightarrow{\left(admit, \frac{N_2 w_2}{N_1 w_1 + N_2 w_2} \min(N_1 w_1 + N_2 w_2)\nu, M\nu\right)}$$
$$(Wait_1[N_1] \parallel Wait_2[N_2-1] \parallel Run_2) \underset{L}{\bowtie} (Admit[M-1] \parallel Serve). \quad (3)$$

In this case, to model the priority of class 1 over class 2, the weights must be chosen such that, $N_1 w_1 \gg N_2 w_2$ for any N_1 and N_2 which can be done easily since N_1 and N_2 are bounded quantities. It is worth noting that this PEPA model does not keep track of the arrival order of jobs of the same class. That is, the admitted job within a class is selected randomly. Following the principles of the shadow server approximation, all existing analysis techniques for priority networks take this assumption which blurs the differences between scheduling policies such as random, first-come first-served, last-come first-served. In our validation study presented in the next section, the PEPA model is compared against stations with first-come first-served priority scheduling, also called head-of-line (HOL) scheduling.

Multiserver stations cause a further source of approximation. Suppose, for instance, that $w_1 = 1000$, $w_2 = 1$, $N_1 = 2$, $N_2 = 1$ and $M = 4$. Then, from (2) and (3) follows that the mean holding time in state $(Wait_1[N_1] \parallel Wait_2[N_2]) \underset{L}{\bowtie} Admit[M]$ is $1/\min((N_1 w_1 + N_2 w_2)\nu, M\nu) = 1/(4\nu)$. However, this holding time is shorter than $1/(3\nu)$, which is the correct value under approximation assumptions when $N_1 + N_2 = 3$ jobs in the queue compete for access to the server. Nevertheless, using a shorter holding time is only beneficial to approximation accuracy, since in the real queueing network admission is in fact instantaneous.

Queueing Network Model The PEPA model for the queueing network considers each workload class and each station, either delay or with priority, as a distinct component. A class i job in a delay station j is associated with a single derivative Run_{ij}, whereas a job in a priority station is modelled with two derivatives which are visited in sequence, i.e., $Wait_{ij}$ and Run_{ij}:

$$Run_{ij} \stackrel{def}{=} \sum_{k=1}^{D} \left(serve_{ij}, p_{ijk}\mu_{ij}\right).Run_{ik} + \sum_{k=D+1}^{J} \left(serve_{ij}, p_{ijk}\mu_{ij}\right).Wait_{ik}, 1 \leq i \leq I \text{ and } 1 \leq j \leq J$$
$$Wait_{ij} \stackrel{def}{=} (admit_j, w_i\nu).Run_{ij}, \qquad 1 \leq i \leq I \text{ and } D < j \leq J$$

The following PEPA definitions are the generalisation of (1) for the behaviour of the priority stations

$$Wait_j \stackrel{def}{=} (admit_j, \nu).Run_j, \qquad Run_j \stackrel{def}{=} \sum_{i=1}^{I} (serve_{ij}, \mu_{ij}).Wait_j, \qquad \text{for } D < j \leq J.$$

Based on the above definitions, we conclude the specification of the priority queueing network by the synchronisation

$$\big(Run_{11}[N_1] \parallel Run_{21}[N_2] \parallel \cdots \parallel Run_{I1}[N_I]\big) \underset{L}{\bowtie} \big(Wait_{D+1}[M_{D+1}] \parallel \cdots Wait_J[M_J]\big),$$
$$\text{where } L = \{admit_j : D < j \le J\} \cup \{serve_{ij} : 1 \le i \le I, D < j \le J\}, \quad (4)$$

which initialises the system with all jobs in the delay station with index 1.

Performance Metrics Estimates of queue lengths and utilisations can be readily obtained from this model through fluid approximation. Let $\mathscr{N}(S,t)$ denote the solution of the ordinary differential equation which gives the number of sequential components in state S at time t ($t = \infty$ for steady-state measures). For instance, (4) implies that $\mathscr{N}(Wait_j, 0) = M_j$ for $D < j \le J$. The steady-state queue length of class-i jobs at station j, for all i and j, is denoted by Q_{ij} and is defined as

$$Q_{ij} = \begin{cases} \mathscr{N}(Run_{ij}, \infty) & \text{for } 1 \le j \le D, \\ \mathscr{N}(Wait_{ij}, \infty) & \text{for } D < j \le J. \end{cases}$$

For a delay station j, Q_{ij} indicates the number of jobs of class i in the station. For a priority station, Q_{ij} does not include the number of jobs in service. Instead, the fraction of servers in a priority station j utilised by class-i jobs, denoted by U_{ij}, is defined as

$$U_{ij} = \mathscr{N}(Run_{ij}, \infty)/M_j, \qquad \text{for all } i \text{ and } D < j \le J.$$

3 Validation

Two validation data sets have been used to assess the accuracy of the differential priority queueing model. The first data set consists of models with $R = 2$ workload classes, whereas the second data set has models with $R = 3$ workload classes. Each data set contains networks with $D = 1$ delay station and 2, 4, or 10 priority queues. Routing probabilities and service demand are drawn randomly from uniform distributions. Service demands range in $[0,1]$ for the priority queues and in $[0,1000]$ and $[0,5000]$ for the delay server respectively when $R = 2$ and $R = 3$, which allowed to span in a balanced manner the full range of queue utilizations. Each model generated in this manner was then analysed for all possible class population mixes such that each class had $N_i = 5, 10, \ldots, 120$ jobs and the sum of the jobs across all classes was equal to $N = 120$. Each of such models was also analysed with different server capacities $K = 1, 2, 5$ at all priority queues.

Differential analysis was performed by setting $\nu = 50000$ with admission weights w_i separated by three orders of magnitude, i.e., $w_{i-1}/w_i = 1000$ and $w_I = 1$. A comparison against discrete-event simulation with the software library JINQS [4] was employed to assess the accuracy. Let $E_s(x_{ij})$ and $E_d(x_{ij})$ be the estimation of the performance metric x_{ij} as computed through simulation and through differential analysis, respectively. The notions of approximation errors for utilisations and queue lengths, taken from [2], are defined as

$$\Delta^{U_{ij}} = |E_s(U_{ij}) - E_d(U_{ij})| \qquad \text{and} \qquad \Delta^{Q_{ij}} = |E_s(Q_{ij}) - E_d(Q_{ij})|/N_i,$$

respectively. For each model, the following four statistics were computed: $\Delta^U_{avg}, \Delta^U_{max}, \Delta^Q_{avg}$, and Δ^Q_{max}, where the subscripts *avg* and *max* stand for the average and the maximum errors, respectively, across $1 \le i \le I$ and $D < j \le J$, of utilisations and queue lengths.

		Two Classes				*Three Classes*			
J	K	Δ^U_{avg}	Δ^U_{max}	Δ^Q_{avg}	Δ^Q_{max}	Δ^U_{avg}	Δ^U_{max}	Δ^Q_{avg}	Δ^Q_{max}
3	1	0.0070	0.0164	0.0121	0.0344	0.0019	0.0077	0.0072	0.0270
3	2	0.0024	0.0061	0.0060	0.0168	0.0010	0.0047	0.0043	0.0170
3	5	0.0014	0.0041	0.0023	0.0055	0.0002	0.0012	0.0015	0.0061
3	*Avg*	0.0035	0.0088	0.0068	0.0188	0.0011	0.0045	0.0044	0.0168
5	1	0.0096	0.0266	0.0116	0.0515	0.0030	0.0167	0.0058	0.0410
5	2	0.0044	0.0135	0.0052	0.0267	0.0009	0.0056	0.0024	0.0193
5	5	0.0014	0.0043	0.0020	0.0112	0.0004	0.0017	0.0003	0.0019
5	*Avg*	0.0050	0.0145	0.0062	0.0293	0.0014	0.0079	0.0030	0.0222
11	1	0.0089	0.0480	0.0085	0.0785	0.0042	0.0387	0.0059	0.0833
11	2	0.0058	0.0316	0.0048	0.0540	0.0016	0.0173	0.0029	0.0478
11	5	0.0020	0.0151	0.0021	0.0261	0.0003	0.0044	0.0009	0.0175
11	*Avg*	0.0054	0.0308	0.0050	0.0516	0.0020	0.0195	0.0031	0.0483
All models		0.0046	0.0177	0.0060	0.0327	0.0015	0.0104	0.0035	0.0286

Table 1: Average approximation errors of fluid approximation for the two data sets, aggregated by the number of stations and server capacity.

Table 1 shows the approximation errors. The error statistics across all model instances (last row) show excellent agreement in general. For a given number of stations, the accuracy of the approximation increases with K. This can be explained by the fact that, for fixed network parameters, distinct values of K give rise to the same system of ODEs with different initial conditions. Thus, larger K corresponds to more replicas of independent sequential components in the model, which makes the differential approach more and more accurate [7]. The approximation is found to be largely independent from the network size, with error statistics of the same order of magnitude for 3, 5, and 11 stations. We also found that the accuracy is better for lightly loaded networks — saturated networks have errors up to two orders of magnitude higher on average than networks with low utilisation. Numerical results showing this trend are however not reported here due to space constraints.

Table 2 compares the above results with the accuracy of AMVA methods. Results show that the random models cannot be approximated in first place by product-form networks. The approximation error of AMVA methods for product-form models (AMVA-PF), here computed using the AQL method [3], shows maximum errors of up to 17% for utilisations and 30% for queue lengths, respectively. Conversely, methods that explicitly account for priorities have errors that are generally below 5%. In all cases, the fluid method behaves comparably in terms of accuracy.

4 Conclusion and Future Work

We have found that PEPA fluid approximation techniques successfully apply to closed queueing networks with priorities and non-preemptive scheduling. Results indicate that the fluid approach is competitive with established AMVA approximations, but we expect it to generalise more easily to networks with multiple non-product-form features, such as priorities and general service times. In addition, the fluid framework will allow us to evaluate other performance indices than steady-state expected values, such as

			Two Classes				*Three Classes*			
Method	J	K	Δ^U_{avg}	Δ^U_{max}	Δ^Q_{avg}	Δ^Q_{max}	Δ^U_{avg}	Δ^U_{max}	Δ^Q_{avg}	Δ^Q_{max}
Fluid	3	*All*	0.0035	0.0088	0.0068	0.0188	0.0011	0.0045	0.0044	0.0168
AMVA-CL	3	*All*	0.0017	0.0043	0.0052	0.0120	0.0006	0.0025	0.0045	0.0168
AMVA-SHA	3	*All*	0.0025	0.0067	0.0069	0.0182	0.0007	0.0026	0.0049	0.0189
AMVA-PF	3	*All*	0.0141	0.0311	0.0238	0.0642	0.0022	0.0069	0.0127	0.0627
Fluid	5	*All*	0.0050	0.0145	0.0062	0.0293	0.0014	0.0079	0.0030	0.0222
AMVA-CL	5	*All*	0.0030	0.0092	0.0050	0.0201	0.0008	0.0047	0.0030	0.0201
AMVA-SHA	5	*All*	0.0037	0.0116	0.0062	0.0267	0.0010	0.0054	0.0036	0.0246
AMVA-PF	5	*All*	0.0265	0.0638	0.0254	0.1488	0.0021	0.0106	0.0076	0.0681
Fluid	11	*All*	0.0054	0.0308	0.0050	0.0516	0.0020	0.0195	0.0031	0.0483
AMVA-CL	11	*All*	0.0036	0.0221	0.0044	0.0356	0.0014	0.0153	0.0029	0.0388
AMVA-SHA	11	*All*	0.0047	0.0278	0.0051	0.0446	0.0019	0.0202	0.0038	0.0552
AMVA-PF	11	*All*	0.0322	0.1748	0.0230	0.3001	0.0047	0.0430	0.0085	0.1660
Fluid	*All models*		0.0046	0.0177	0.0060	0.0327	0.0015	0.0104	0.0035	0.0286
AMVA-CL	*All models*		0.0027	0.0117	0.0049	0.0224	0.0009	0.0074	0.0035	0.0252
AMVA-SHA	*All models*		0.0036	0.0152	0.0061	0.0296	0.0012	0.0093	0.0041	0.0328
AMVA-PF	*All models*		0.0241	0.0885	0.0241	0.1687	0.0030	0.0201	0.0096	0.0985

Table 2: Average approximation errors for differential and AMVA techniques.

transient characteristics and passage-time distributions. Future work will focus on such generalisations and on the natural extension of our ideas to the preemptive scheduling case.

Acknowledgement

The authors would like to thank Tony Field for his help with JINQS. The work of Giuliano Casale was supported by the Imperial College Junior Research Fellowship.

References

[1] R. M. Bryant, A. E. Krzesinski & P. Teunissen (1984): *The MVA Priority Approximation*. *ACM TOCS* 2, pp. 335–359.

[2] K. M. Chandy & D. Neuse (1982): *Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems*. *Commun. ACM* 25(2), pp. 126–134.

[3] D. L. Eager & J. N. Lipscomb (1988): *The AMVA Priority Approximation*. *Perf. Eval.* (8), pp. 173–193.

[4] A. J. Field (2006). *JINQS: An Extensible Library for Simulating Multiclass Queueing Networks*. Available at `http://www.doc.ic.ac.uk/~ajf/Research/manual.pdf`.

[5] J. Hillston (2005): *Fluid Flow Approximation of PEPA Models*. In: *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems*, IEEE Computer Society Press, Torino, Italy, pp. 33–43.

[6] E. D. Lazowska, J. Zahorjan, G. S. Graham & K. C. Sevcik (1984): *Quantitative System Performance*. Prentice-Hall.

[7] M. Tribastone, S. Gilmore & J. Hillston (2010). *Scalable Differential Analysis of Process Algebra Models*. *Transactions on Software Engineering*, in press.