Higher response time moments for M/M/1 discriminatory processor sharing queues

Tiberiu Chis
Imperial College London
London, UK, SW7 2RH
tiberiu.chis07@imperial.ac.uk

Peter G. Harrison
Imperial College London
London, UK, SW7 2RH
p.harrison@imperial.ac.uk

ABSTRACT

Obtaining response time moments in processor sharing (PS) queues is difficult due to serving of multiple jobs. Egalitarian PS (EPS) queues are limited to one class of arriving jobs. Discriminatory PS (DPS) assigns weights to different job classes and offers more diverse modeling capabilities than EPS. It is known that response time is the representative metric for delay as specified in service level agreements (SLAs), which consider higher moments important. Hence, we build an automated numerical algorithm for calculating higher moments of response time in M/M/1-DPS queues for multiple job classes and test two different case studies.

CCS Concepts

•Mathematics of computing → Queueing theory; Numerical analysis; •Networks → Network performance evaluation;

Keywords

Response time; M/M/1; discriminatory processor sharing

1. INTRODUCTION

The processor sharing (PS) discipline has important applications in web server design [4] and bandwidth-sharing protocols in packet-switched networks [11] with delay as the key measure. In a PS system with service rate 1 and n jobs, each job is served at $1/n$ times the speed of the processor. PS has the following useful properties: an implicit fairness where expected job response time is directly proportional to its size; arriving jobs access server resources without queueing; handling heavy-tailed service times, which may arise as short jobs are allowed to overtake long jobs. Often, the egality of PS omits applications with priority jobs and, hence, we consider variants of PS. Discriminatory PS (DPS) [16], where each job j in the system receives its own percentage of the server, extends PS by supporting multiple job classes. In DPS, K job types are served by a vector of weights $(\alpha_j > 0, j = 1, \ldots, K)$ and, assuming there are n_i class i jobs $(i = 1, \ldots, K)$ in the system, each class j job is served at rate:

$$r_j(n_1, \ldots, n_K) = \frac{\alpha_j}{\sum_{i=1}^{K} \alpha_i n_i}, \; j = 1, \ldots, K \qquad (1)$$

Hence, the share of a job class increases with the number of jobs, which prevents classes with smaller weights from starving. DPS becomes PS if $\alpha_i = \alpha_j$, for $i, j = 1, \ldots, K$. By varying DPS weights, the choice of instantaneous service weights of different job classes enables differentiated quality of service among specific type of jobs. For example, ADSL subscribers are offered different payment rates in return for corresponding shares of available bandwidth. Existing work in the literature proves, via experiments, that the expected unconditional response time of PS systems is reduced by 33% with DPS [15]. With such work focusing primarily on reducing mean response times, it is important to consider higher moments to understand the effects of variance and skewness on response time distributions. Similarly, existing work for approximating response times in M/M/1-DPS queues [3, 7, 12, 17] is limited by the number of moments obtained or the number of job classes considered. Hence, this paper provides an automated algorithm to iteratively calculate higher response time moments (i.e. more than two) for multiple job classes in M/M/1-DPS queues. Our contributions are as follows:

- Extend the two-moment equations introduced by Kim and Kim [7] (utilized in an algorithm by Chis and Harrison [17]) via an automated algorithm.
- Provide an explicit formula for the third moment of response time in M/M/1-DPS queues for two classes.
- Obtain arbitrary (up to any order) response time moments numerically for multi-class M/M/1-DPS queues.

This paper is organized as follows: section 2 defines response time; section 3 offers related work on response time

VALUETOOLS 2015, December 14-16, Berlin, Germany

DOI 10.4108/eai.14-12-2015.2262730

in DPS queues; section 4 presents the moment-generating algorithm for M/M/1-DPS queues; section 5 summarizes results on two case studies; section 6 concludes the paper.

2. BACKGROUND

Response time (or sojourn time) is the time needed, after a job enters a system, to attain the required service and depart the system. In queueing systems, response time T is the sum of queueing time (or time spent waiting to start service) and service time. Let λ be the arrival rate, μ be the service rate, and $\rho = \lambda/\mu < 1$ be the steady state system utilization. For PS systems, the mean unconditional response time at equilibrium ($\mathbb{E}[T]$) can be computed using Little's law: $\mathbb{E}[T] = L/\lambda = \rho/\lambda(1-\rho) = 1/\mu(1-\rho)$, where L is the mean number of jobs in the system. When jobs require x units of service time, the mean conditional response time is $\mathbb{E}[T(x)] = x/(1-\rho)$. Calculating higher moments of response time under PS requires layered branching of arriving jobs [8]. Further, DPS scheduling adds complexity with multiple job classes. The next section summarizes existing methods for analyzing response time in DPS queues.

3. RELATED WORK

One of the earliest significant works on DPS queues was Kleinrock [16] in 1967. Subsequently, Fayolle *et al* [12] summarized DPS results of Kleinrock and Mitrani in 1980 and obtained expected conditional response times for M/M/1-DPS queues as a solution of integro-differential equations. Further, Laplace transforms were used to obtain mean response times for multiple classes, but there were no results on higher response time moments. Yashkov [8] abstracted PS scheduling as a layered branching of arriving jobs in 1987, but only references DPS queues via Fayolle.

In a PS queue with utilization ρ, it is known [1, 3] that the response time T of an arriving job requiring x units of service has a distribution function with Laplace transform:

$$T^*(s \mid x) = \frac{(1-\rho)(1-\rho r^2)e^{-[\rho\mu(1-r)+s]x}}{(1-\rho r)^2 - \rho(1-r)^2 e^{-[1/r-\rho r]\mu x}} \tag{2}$$

where r is the smaller root of the equation $\rho r^2 - (\rho+1+s/\mu)r + 1 = 0$. This result is derived by solving a partial differential equation (PDE) for a generating function $G(z, s, x)$, namely:

$$(\mu z^2 - (\rho\mu + \mu + s)z + \rho\mu)\frac{\partial G}{\partial z} - \frac{\partial G}{\partial x} = (\rho\mu + s - \mu z)G \tag{3}$$

Hence, this yields $T^*(s \mid x) = (1-\rho)G(\rho, s, x)$. In 2004, Kim and Kim [7] offered a joint transform to obtain response time moments for M/M/1-DPS queues with K job classes. Response time moments were derived by differentiating a PDE that governed a joint transform of elapsed response time and number of customers in the system. Thus, Kim and Kim solved $(K+1)(K+2)/2$ independent equations to obtain conditional response time moments. When $K = 1$, substituting equal α_i weights into the aforementioned PDE from [7] and simplifying duly yields equation (3). For $K = 2$, a mechanized algorithm to solve the Kims' equations for the second moments was obtained for multiple job types [17] that allowed arbitrary weighted α_i, mean service requirements $1/\mu_i$ and arrival rates λ_i. Explicit expressions for the first two moments of class 1 jobs, which assume equal weights $\alpha_1 = \alpha_2 = 0.5$, were found to be:

$$\mathbb{E}[T_1] = \frac{1}{\mu_1(1-\rho_1-\rho_2)} \tag{4}$$

and

$$\mathbb{E}[T_1^2] = \frac{4(\mu_1(1+\rho_2)+\mu_2(1-\rho_2))}{\mu_1^2(1-\rho_1-\rho_2)^2(\mu_1(2-\rho_1)+\mu_2(2-\rho_1-2\rho_2))} \tag{5}$$

where the class-utilizations are $\rho_i = \lambda_i/\mu_i$, for $i = 1, 2$, and $\rho_1 + \rho_2 < 1$. The moments for class 2 are symmetrical, interchanging the subscripts 1 and 2 in the expressions.

4. M/M/1-DPS RESPONSE TIME

To obtain the general k^{th} moment in M/M/1-DPS queues, we extend the work of [17] by forming a novel automated algorithm, implemented in Wolfram's Mathematica. The numerical algorithm is based on the direct approach of solving the moment-equations obtained by differentiating the Kim's joint transform PDE [7] repeatedly – k times for the k^{th} moment; the details are given in figure 1. This example screenshot of the algorithm initializes different weights and rates for two job classes and outputs four response time moments (sufficient for the purpose of this work), but easily extends to higher moments as it is fast in its numerical iterations. We display the third moment $\mathbb{E}[T_1^3]$ in equation (6) symbolically for job class 1 (with corresponding class 2 moment $\mathbb{E}[T_2^3]$ obtained by inverting subscripts 1 and 2). Note that we set $\alpha_1 = \alpha_2 = 0.5$ for presentation purposes, but the weight ratio can take any value:

$$\begin{aligned} d_1\mathbb{E}[T_1^3] = {} & \mu_1^2\left(\rho_1(\rho_2-1) - 2\left(\rho_2^2+4\rho_2+1\right)\right) \\ & - 2\mu_1\mu_2(\rho_1 - 2\rho_2 + 2)(\rho_2+1) \\ & + \mu_2^2(\rho_1 - 2\rho_2 + 2)(\rho_2 - 1) \end{aligned} \tag{6}$$

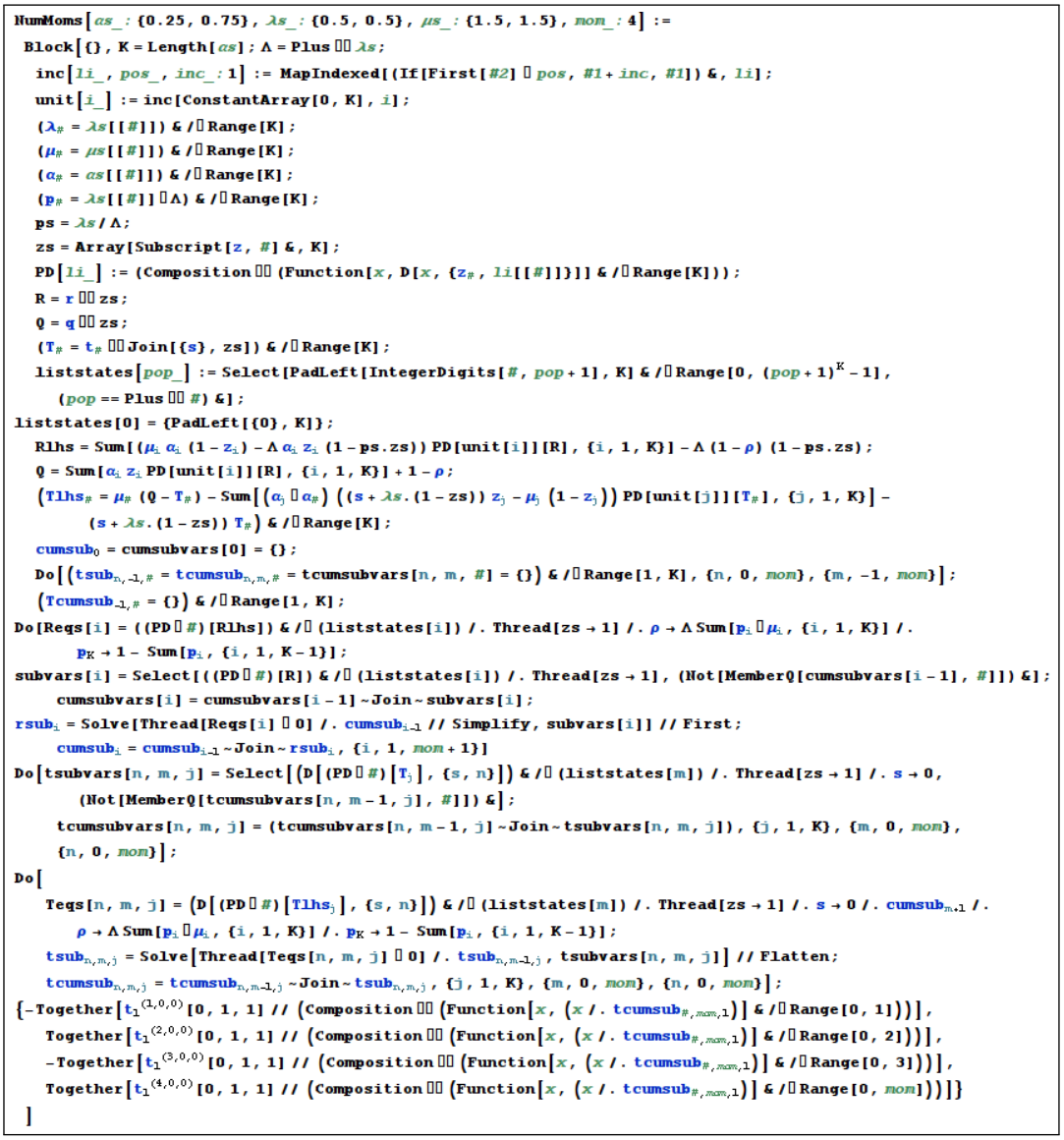
where $12d_1 = \mu_1^3(\rho_1+\rho_2-1)^3(\mu_1(\rho_1-2) + \mu_2(\rho_1+2\rho_2-2))^2$.

4.1 Response time distribution

Response time distribution can be approximated from moments using, for example, the generalized lambda distribution (GLD) [5, 9]. The GLD is a distribution-fitting approximation, which inputs the first four response time moments to match four "lambda" parameters and outputs density and distribution functions. The GLD "quality-of-fit" is ascertained only through a goodness-of-fit test as closed-form solutions do not exist for the lambda parameters. With response time moments obtained from variations of equations (4), (5), (6) and the algorithm from figure 1 (with different values for α priority weights), one can easily parametrize the GLD and obtain distributions. For our results, we compare response time moments from our analytical approximations with simulations on two different case studies.

5. RESULTS

We calculate results from two workloads: first, we collect inter-arrival times of TCP traffic from experiments on



```
NumMoms[as_ : {0.25, 0.75}, λs_ : {0.5, 0.5}, μs_ : {1.5, 1.5}, mom_ : 4] :=
 Block[{}, K = Length[as]; Λ = Plus @@ λs;
  inc[li_, pos_, inc_ : 1] := MapIndexed[(If[First[#2] ∈ pos, #1 + inc, #1]) &, li];
  unit[i_] := inc[ConstantArray[0, K], i];
  (λ# = λs[[#]]) & /@ Range[K];
  (μ# = μs[[#]]) & /@ Range[K];
  (α# = as[[#]]) & /@ Range[K];
  (p# = λs[[#]] ∈ Λ) & /@ Range[K];
  ps = λs / Λ;
  zs = Array[Subscript[z, #] &, K];
  PD[li_] := (Composition @@ (Function[x, D[x, {z#, li[[#]]}]] & /@ Range[K]));
  R = r @@ zs;
  Q = q @@ zs;
  (T# = t# @@ Join[{s}, zs]) & /@ Range[K];
  liststates[pop_] := Select[PadLeft[IntegerDigits[#, pop + 1], K] & /@ Range[0, (pop + 1)^K - 1],
    (pop == Plus @@ #) &];
liststates[0] = {PadLeft[{0}, K]};
  Rlhs = Sum[(μi αi (1 - zi) - Λ αi zi (1 - ps.zs)) PD[unit[i]][R], {i, 1, K}] - Λ (1 - ρ) (1 - ps.zs);
  Q = Sum[αi zi PD[unit[i]][R], {i, 1, K}] + 1 - ρ;
  (Tlhs# = μ# (Q - T#) - Sum[(αj ∈ α#) ((s + λs.(1 - zs)) zj - μj (1 - zj)) PD[unit[j]][T#], {j, 1, K}] -
      (s + λs.(1 - zs)) T#) & /@ Range[K];
  cumsub0 = cumsubvars[0] = {};
  Do[(tsubn,-1,# = tcumsubn,m,# = tcumsubvars[n, m, #] = {}) & /@ Range[1, K], {n, 0, mom}, {m, -1, mom}];
  (Tcumsub-1,# = {}) & /@ Range[1, K];
Do[Reqs[i] = ((PD ∈ #)[Rlhs]) & /@ (liststates[i]) /. Thread[zs → 1] /. ρ → Λ Sum[pi ∈ μi, {i, 1, K}] /.
     pK → 1 - Sum[pi, {i, 1, K - 1}];
subvars[i] = Select[((PD ∈ #)[R]) & /@ (liststates[i]) /. Thread[zs → 1], (Not[MemberQ[cumsubvars[i - 1], #]]) &];
   cumsubvars[i] = cumsubvars[i - 1] ~Join~ subvars[i];
rsubi = Solve[Thread[Reqs[i] ∈ 0] /. cumsubi-1 // Simplify, subvars[i]] // First;
   cumsubi = cumsubi-1 ~Join~ rsubi, {i, 1, mom + 1}]
Do[tsubvars[n, m, j] = Select[(D[(PD ∈ #)[Tj], {s, n}]) & /@ (liststates[m]) /. Thread[zs → 1] /. s → 0,
     (Not[MemberQ[tcumsubvars[n, m - 1, j], #]]) &];
   tcumsubvars[n, m, j] = (tcumsubvars[n, m - 1, j] ~Join~ tsubvars[n, m, j]), {j, 1, K}, {m, 0, mom},
   {n, 0, mom}];
Do[
  Teqs[n, m, j] = (D[(PD ∈ #)[Tlhsj], {s, n}]) & /@ (liststates[m]) /. Thread[zs → 1] /. s → 0 /. cumsubm-1 /.
     ρ → Λ Sum[pi ∈ μi, {i, 1, K}] /. pK → 1 - Sum[pi, {i, 1, K - 1}];
  tsubn,m,j = Solve[Thread[Teqs[n, m, j] ∈ 0] /. tsubn,m-1,j, tsubvars[n, m, j]] // Flatten;
  tcumsubn,m,j = tcumsubn,m-1,j ~Join~ tsubn,m,j, {j, 1, K}, {m, 0, mom}, {n, 0, mom}];
{-Together[t1^(1,0,0)[0, 1, 1] // (Composition @@ (Function[x, (x /. tcumsub#,mom,1)] & /@ Range[0, 1]))],
  Together[t1^(2,0,0)[0, 1, 1] // (Composition @@ (Function[x, (x /. tcumsub#,mom,1)] & /@ Range[0, 2]))],
  -Together[t1^(3,0,0)[0, 1, 1] // (Composition @@ (Function[x, (x /. tcumsub#,mom,1)] & /@ Range[0, 3]))],
  Together[t1^(4,0,0)[0, 1, 1] // (Composition @@ (Function[x, (x /. tcumsub#,mom,1)] & /@ Range[0, mom]))]}
 ]
```

Figure 1: Mathematica code of the automated algorithm with two job classes and four moments.

Intel Core i7-2600 CPU @ 3.40GHz machines, which form the TCP dataset; secondly, we use parameters from a GRID network application [13]. Both datasets exhibit multiple job classes with different access priorities and, hence, are suitable for our M/M/1-DPS queue. Class priorities are calculated using packet type and size. We approximate analytical response time moments using the numerical algorithm in figure 1 . To obtain simulated moments, we execute 10000 runs of one million observations using a standard MATLAB package and provide 95% confidence intervals.

5.1 TCP datasets

Data is collected from servers running data applications with TCP packet delivery. Using inter-arrival times from a TCP dataset, we parametrize our M/M/1-DPS queue as follows: $\lambda_1 = 0.2$, $\lambda_2 = 0.3$, $\mu_1 = 1$, $\mu_2 = 2$, $\alpha_1 = \alpha_2 = 0.5$. Hence, we obtain analytical (**A**) and simulated (**S**) response time moments for class 1 in table 1 with corresponding class 2 moments in table 2. Analytical moments are calculated from the numerical algorithm in figure 1 . Further, all moments are centralized to reveal the spread from the mean.

Table 1: Class 1 response time moments from TCP dataset 1.

	$\mathbb{E}[T]$	$\mathbb{E}[(T-\mathbb{E}[T])^2]$	$\mathbb{E}[(T-\mathbb{E}[T])^3]$	$\mathbb{E}[(T-\mathbb{E}[T])^4]$
A	**1.54**	**3.25**	**16.64**	**174.62**
S	**1.54±9e-04**	**3.29±2e-03**	**16.86±0.01**	**177.15±0.10**

Table 2: Class 2 response time moments from TCP dataset 1.

	$\mathbb{E}[T]$	$\mathbb{E}[(T-\mathbb{E}[T])^2]$	$\mathbb{E}[(T-\mathbb{E}[T])^3]$	$\mathbb{E}[(T-\mathbb{E}[T])^4]$
A	**0.77**	**0.88**	**2.67**	**16.71**
S	**0.78±4e-04**	**0.89±5e-04**	**2.69±2e-03**	**16.95±0.01**

Further, we set different priority weights for each job class, which is typical of TCP packets arriving with different-sized

requests. For example, one class i may have higher share of the server (i.e. larger α_i value), but have lower mean service rate (μ_i). We collect mean arrival and service rates from a second TCP dataset and parametrize our queue with $\lambda_1 = 0.2$, $\lambda_2 = 0.6$, $\mu_1 = 2$, $\mu_2 = 1.2$, $\alpha_1 = 0.33$ and $\alpha_2 = 0.67$. The corresponding multi-class response time moments are summarized in tables 3 and 4. The applicability of modeling multiple arrivals and the flexibility of setting arbitrary weights allows the automated algorithm in figure 1 to improve existing work; previously, either all α_i weights were equal, as with Chis and Harrison's work [17], or different weights were used to approximate response time for up to two moments only, as with Kim and Kim's work [7].

Table 3: Class 1 response time moments from TCP dataset 2.

	$\mathbb{E}[T]$	$\mathbb{E}[(T-\mathbb{E}[T])^2]$	$\mathbb{E}[(T-\mathbb{E}[T])^3]$	$\mathbb{E}[(T-\mathbb{E}[T])^4]$
A	**1.68**	**6.30**	**64.88**	**1286.8**
S	**1.69±1e-03**	**6.37±4e-03**	**65.73±0.04**	**1305.5±0.74**

Table 4: Class 2 response time moments from TCP dataset 2.

	$\mathbb{E}[T]$	$\mathbb{E}[(T-\mathbb{E}[T])^2]$	$\mathbb{E}[(T-\mathbb{E}[T])^3]$	$\mathbb{E}[(T-\mathbb{E}[T])^4]$
A	**2.00**	**7.27**	**71.40**	**1394.6**
S	**2.02±1e-03**	**7.35±4e-03**	**72.34±0.04**	**1414.8±0.80**

5.2 GRID network

For the second experiment, we parametrize an M/M/1-DPS queue with values obtained directly from a GRID network application [13] as follows: $\lambda_1 = 22.1$, $\lambda_2 = 7.16$, $\mu_1 = 50, \mu_2 = 20$, $\alpha_1 = 0.25$ and $\alpha_2 = 0.75$. We obtain response time moments for two job classes in M/M/1-DPS queues presented in tables 5 and 6 , respectively.

Table 5: Class 1 response time moments from GRID dataset.

	$\mathbb{E}[T]$	$\mathbb{E}[(T-\mathbb{E}[T])^2]$	$\mathbb{E}[(T-\mathbb{E}[T])^3]$	$\mathbb{E}[(T-\mathbb{E}[T])^4]$
A	**0.15**	**0.06**	**0.06**	**0.14**
S	**0.15±8e-05**	**0.06±3e-05**	**0.07±4e-05**	**0.14±8e-05**

Table 6: Class 2 response time moments from GRID dataset.

	$\mathbb{E}[T]$	$\mathbb{E}[(T-\mathbb{E}[T])^2]$	$\mathbb{E}[(T-\mathbb{E}[T])^3]$	$\mathbb{E}[(T-\mathbb{E}[T])^4]$
A	**0.18**	**0.07**	**0.07**	**0.14**
S	**0.18±1e-04**	**0.07±4e-05**	**0.07±4e-05**	**0.14±8e-05**

6. CONCLUSION

We provide an automated algorithm to yield higher response time moments in M/M/1-DPS queues with multiple job classes. The results reveal that analytical approximations match simulated moments well. Applications include modeling multi-class Internet traffic, where delay addresses SLA constraints, and spatiotemporal resource allocation in networks. Extensions of this work include generalizing job arrivals to MAPs and, hence, finding response time moments in MAP/M/1-DPS queues. Alternatively, multiple streams of TCP packets can be modeled as job arrival processes using variations of the Markov-modulated Poisson process (MMPP), which is a special case of MAP. Discretized MMPPs (or hidden Markov models) replicate the burstiness of TCP packet traces, which can be clustered in groups, and, hence, allow model parameters to converge on multiple traces simultaneously at reduced computational complexity [10]. Further, arrival parameters of queueing models can be updated incrementally via online EM learning algorithms [2, 6, 14], which are suitable for live systems.

7. REFERENCES

[1] E. G. Coffman Jr., R. R. Muntz, H. Trotter: Waiting Time Distributions for Processor-Sharing Systems, In *Journal ACM*, **17**, pp. 123-130 (1970)

[2] T. Chis: Sliding Hidden Markov Model for Evaluating Discrete Data, In *Proc. EPEW* (2013)

[3] P. G. Harrison, N. M. Patel: Performance Modelling of Communication Networks and Computer Architectures, *Addison-Wesley*, **1** (1993)

[4] M. A. Kjaer, M. Kihl, A. Robertsson: Response-Time Control of a Processor-Sharing System using Virtualised Server Environments, In *Proc. IFAC* (2008)

[5] S. W. M. Au-Yeung, N. J. Dingle, W. J. Knottenbelt: Efficient Approximation of Response time Densities and Quantiles in Stochastic Models, In *Proc. ACM WOSP* (2004)

[6] T. Chis, P. G. Harrison: Adapting Hidden Markov Models for Online Learning, In *Elsevier ENTCS*, **318**, pp. 109-127 (2015)

[7] J. Kim, B. Kim: Sojourn time distribution in the M/M/1 queue with discriminatory processor-sharing, In *Perform. Eval.*, **58**, pp. 341-365 (2004)

[8] S. F. Yashkov: Processor-Sharing Queues: Some Progress In Analysis, In *Queue. Sys.*, **2**, pp. 1-17 (1987)

[9] A. Lakhany, H. Mausser: Estimating the parameters of the General Lambda Distribution, In *Algo. Research Quarterly*, **3**, pp. 47-58 (2000)

[10] T. Chis, P. G. Harrison: Modeling Multi-User Behaviour in Social Networks, In *Proc. IEEE MASCOTS* (2014)

[11] J. W. Roberts: A survey on statistical bandwidth sharing, In *Computer Networks*, **45**, pp. 319-332 (2004)

[12] G. Fayolle, R. Iasnogorodski, I. Mitrani: Sharing a Processor Among Many Job Classes, In *Journal ACM*, **27**(3), pp. 519-532 (1980)

[13] H. Li, M. Muskulus, L. Wolters: Modeling Job Arrivals in a data-intensive Grid, In *Proc. JSSPP Workshop* (2006)

[14] T. Chis, P. G. Harrison: iSWoM: The incremental Storage Workload Model using Hidden Markov Models, In *Proc. ASMTA* (2013)

[15] K. Avrachenkov, U. Ayesta, P. Brown, R. Nunez-Queija: Discriminatory Processor Sharing Revisited, In *Proc. IEEE Infocom* (2005)

[16] L. Kleinrock: Time-shared systems: A theoretical treatment, In *Journal ACM*, **14**(2), pp. 242-261 (1967)

[17] T. Chis, P. G. Harrison: Moment-Generating Algorithm for Response Time in Processor Sharing Queueing Systems, In *Proc. EPEW* (2015)