WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. (2016)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.2689

SPECIAL ISSUE PAPER

Task admission control policy in cloud server pools based on task arrival dynamics

Haleh Khojasteh and Jelena Mišić*

Ryerson University, Toronto, ON, Canada

ABSTRACT

In this paper, we propose two task admission control algorithms that utilize random task filtering: a lightweight algorithm based on long-term estimates of average utilization and offered load, and a more complex algorithm based on instantaneous utilization. Detailed performance evaluation confirms that both algorithms are able to ensure that the system remains in the stable operating region. We have also found that more aggressive filtering tends to decrease task blocking rate and delay. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS

cloud computing; task admission control; resource allocation; utilization-based filtering

***Correspondence**

Jelena Mišić, Computer Science, Ryerson University, Toronto, ON, Canada.
E-mail: jmisic@scs.ryerson.ca

1. INTRODUCTION

In a cloud server pool, resources are composed of virtual machines (VMs), which are deployed on physical machines (PMs). Obviously, the cloud operator is interested in accepting and servicing as many incoming task requests as possible in order to improve its utilization of resources and maximizes return on investment. The number of accepted tasks is limited by the size of the server pool and also by the terms of its service level agreement. However, random nature of task request arrivals makes the decision whether to admit a service request or reject it a non-trivial one. Through utilizing appropriate admission control and resource allocation mechanisms, it is possible to provide a satisfactory level of utilization whilst maintaining the system in a stable operating region defined by the cloud operator.

In this paper, we propose two task admission algorithms aimed to achieve this goal. We utilize two controlling parameters, full rate task acceptance threshold and filtering coefficient, to guide task admission. The criteria to choose one of the algorithms are the relative stability of the task arrival rate. The first task admission algorithm is based on the long-term estimation of average utilization and offered load upon the arrival of tasks; it is more computationally lightweight. Estimates will be more accurate, and quality of control will be better when mean task arrival rate is changing slowly.

The second algorithm is based on instantaneous utilization upon the arrival of tasks and exhaustive search to find optimal value of full rate task acceptance threshold and filtering coefficient. While this algorithm is more computationally intensive, it is better suited for systems that experience sudden changes in their task arrival rate.

Using the developed queuing model of the proposed task admission algorithms, we have analyzed the short-term variability of the system, as well as the effect of the size of tasks' waiting queue on the system performance in our simulations. We have also utilized a different calculation scheme for filtering coefficient in the lightweight task admission control algorithm, which is more aggressive toward rejecting the incoming tasks. We have evaluated the effect of the different filtering coefficients on the system performance.

The remainder of the paper is organized as follows: in Section 2, we describe some approaches to cloud resource allocation, in particular, those pertaining to task admission control, proposed in the literature. Section 3 presents the admission control policy and the proposed algorithms and describes the analytical model of the proposed scheme. Section 4 discusses our experiments and results. Section 5 concludes the paper and discusses some directions for future work.

2. RELATED WORK

Admission control and resource allocation in cloud systems have received a great deal of attention during recent years. Queueing theory was often used to obtain the performance indicators necessary to make online control decisions [1,2]. This information may then be used to drive Lyapunov optimization to design an online admission control, routing, and resource allocation algorithm [3]. An adaptive feedback control scheme alongside with a queue model of the application was employed in [4]. Authors in [5] modeled cloud services using queuing theory and controlled them through adaptive proactive controllers that estimate whether services need some of the resources in the near future or not.

The Markov decision process framework proposed in [6] modeled admission control in cloud, while approximate dynamic programming paradigm was utilized to devise optimized admission policies. Authors in [7] formulated an optimization problem for dynamic resource sharing of mobile users in mobile cloud computing hotspot with a cloudlet as a semi-Markov decision process. The process was transformed into a linear programming model to obtain an optimal solution. Set-theoretic control techniques were used to solve admission control and resource allocation problems in [8].

Authors in [9] developed a technique to determine the effective bandwidth for aggregated flow to make admission decisions using network calculus. The admission control problem in the cloud was also modeled using the general algebraic modeling system [10].

A session-based adaptive admission control approach for virtualized application servers was presented in [11]. Also, a session deferment mechanism was implemented to reduce the number of rejected sessions. Scheduling and admission control algorithms proposed in [12] incorporated resource overbooking to improve utilization, and a combination of modeling, monitoring, and prediction techniques was used to avoid exceeding the total infrastructure capacity.

With mobile devices increasingly able to connect to cloud servers from anywhere, resource-constrained devices can potentially perform offloading of computational tasks to either improve resource usage or improve performance.

However, the characteristics of mobile devices and wireless network make the resource allocation of mobile clouds more complicated than stationary clouds. Offloading requests from a mobile device usually require quick response, may be infrequent, and are subject to variable network connectivity, whereas stationary clouds incur relatively long setup times, are leased for long-time periods, and experience uninterrupted network connectivity [13]. Also, the volume of workload to be offloaded may not be known in advance because many of the offload requests are the consequence of decisions made by the (generally unpredictable) human user of the device. Offloading requests from a mobile device can lead to forking of new tasks in on-demand manner. In [14], we have addressed this problem by proposing flexible resource allocation mechanisms in which forked tasks are given higher priority over newly arrived tasks.

Authors in [15] address the problems of task dependency in mobile cloud computing, and an optimization problem was formulated to minimize the latency while considering prescribed resource utilization constraints. We have presented the analytical model of our task admission control solution in [20] and we have evaluated the system performance using this model. In this paper, we have added substantial simulation experiments to support our claim.

3. ADMISSION CONTROL POLICY

Let us begin by stating our assumptions. Figure 1 illustrates the proposed task admission and resource allocation scheme in a server pool. Incoming task requests are admitted to the input queue from which they are provisioned

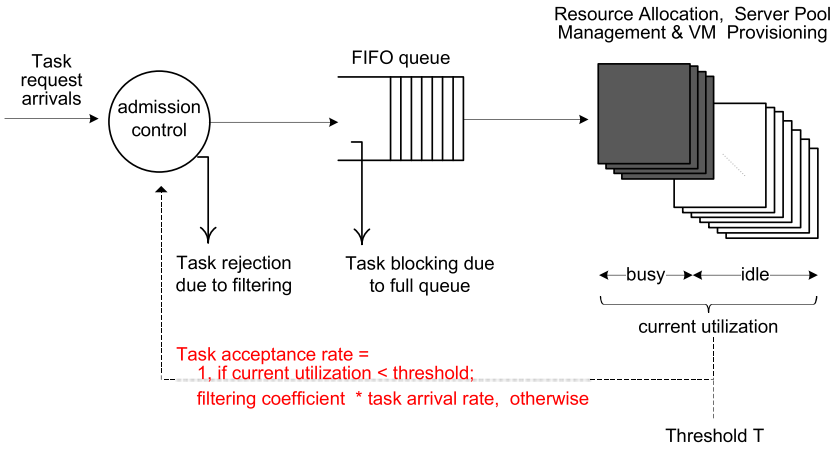


Figure 1. Task admission control and resource allocation scheme.

on VMs by the resource allocation process. System performance is continuously monitored to provide guidance to the task admission process which can reject incoming tasks if the admission would compromise stated performance limits (typically, a utilization threshold). In the following, we assume that each task is serviced by a single VM, but extension to multiple VMs in the manner similar to [16] is also possible.

3.1. Controlling parameters and related policies

As stated earlier, the goal is to keep the system in the stable operating region of the utilization threshold, U_{thr}. To achieve this goal, we have utilized two controlling parameters. First parameter is the threshold of full rate task acceptance, T, which is defined as the threshold position in the total number of VMs in the server pool or system capacity. The second parameter is the filtering coefficient, F_{cf}, which is defined as the proportion of accepted tasks in the server pool; note that tasks to be accepted are randomly chosen among the incoming tasks. The cloud operator can define different policies according to these two parameters; for example, it can set the threshold position and adjust the filtering coefficient according to the threshold position. Alternatively, it can preset the filtering coefficient and then find the resulting threshold position. In this work, we have selected the second option.

The use of a filtering coefficient results in selective task acceptance. The system's current resource availability provides the necessary feedback to the admission control mechanism. The resource allocation mechanism has been developed as three interactive stochastic sub-models including resource assigning module, pool management module, and VM provisioning module; the overall solution is obtained by iteration over individual sub-model solutions. The resource allocation mechanism and related provisioning processes are presented in detail in [2,17].

Filtering policy can be further by adjusting the actual values of the admission threshold and filtering coefficient. At lower values of the admission threshold, system will reject incoming tasks in a wider range but at a low rate; alternatively, the system can be set to drop tasks in a narrow range of values but at a higher rate.

3.2. Task admission scheme based on offered load

The first admission control scheme is presented in Algorithm 1, where parameters are re-calculated whenever a task arrives in the system or departs from it. First, average offered load ρ_{av} is calculated using the current task arrival rate. We have used exponentially weighted moving average (EWMA) technique [18] to smoothen the effect of sudden changes on the system. ρ_{av} is computed using EWMA smoothing factor (α), the previous average offered load (ρ_{avp}), and the system's current offered load (ρ_c). Estimated average utilization u'_{av} is calculated according to EWMA smoothing factor β, the previous estimated average utilization, u'_{avp}, and the cloud system's current utilization, u_s. In case of task arrival, if u'_{av} is less than the utilization threshold, U_{thr}, the system accepts the incoming task; otherwise, the system calculates the filtering coefficient as $F_{cf} = 1-(\rho_{av}-\rho_{thr})$ and then rejects the incoming task with the probability of $1 - F_{cf}$.

Algorithm 1 Admission mechanism based on offered load.

```
Upon task departure:
  Re-calculate ρ_av, u_s and u'_av;
Upon task arrival:
  Re-calculate ρ_av as ρ_av ⟵ α · ρ_c + (1 − α)ρ_avp;
  Estimate new average utilization (u'_av) as
  u'_av ⟵ β · u_s + (1 − β)u'_avp;
  if u'_av > U_thr then
    Calculate F_cf ⟵ 1 − (ρ_av − ρ_thr);
    Reject the incoming task with the probability of
    1 − F_cf;
  else
    Accept the incoming task;
  end if
```

3.3. Analytical model of admission control

Admission control scheme can be modeled as a birth-death process, which is a special case of continuous-time Markov chain. The task admission scheme can be analyzed using the Markov chain depicted in Figure 2. In this setup, T and R represent the thresholds of full rate task acceptance and full rate task rejection, respectively. Namely, when the mean blocking probability of the system is below the first threshold, admission control accepts all incoming tasks. As the result, T is the last state where all of the arrived tasks will be admitted. Beyond this threshold, admission control begins to drop some of the incoming tasks. R is the full rejection threshold state beyond that all incoming tasks will be rejected, and it is also known as the capacity of the system or the number of VMs in the model.

Transition rates of going from state i to state $i + 1$ (i.e., that the task is accepted by the admission control) are calculated in Equation (2), and the accepted rate of incoming tasks, λ_f, can be set as

$$\lambda_f = \begin{cases} \lambda & \text{state } i = 0..T \\ F_{cf} \cdot \lambda & \text{state } i = T + 1..R \\ 0 & \text{state } i > R \end{cases} \quad (1)$$

where λ is the task arrival rate and F_{cf} is filtering coefficient calculated as $F_{cf} = 1 - (\rho_i - U_{thr})$. Offered load in the state i presented as ρ_i and utilization threshold, U_{thr}, determine the value of filtering coefficient. The state probabilities of the Markov model can be calculated as

$$P_k = \begin{cases} \frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k \frac{1}{DD} & 0 \leqslant k \leqslant T \\ \frac{\lambda^T \prod_{i=1}^{k-T} \lambda_i}{k!\,\mu^k} \frac{1}{DD} & T < k \leqslant R \end{cases} \quad (2)$$

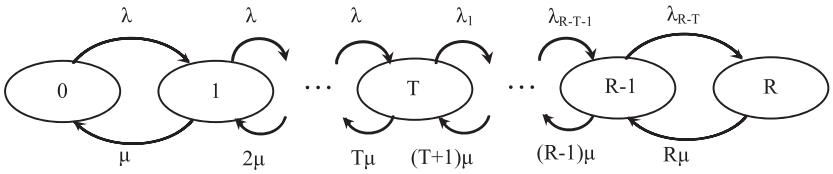


Figure 2. Markov chain model admission control.

where

$$DD = \sum_{0}^{T} \frac{1}{i!} \left(\frac{\lambda}{\mu}\right)^{i} + \left(\frac{\lambda}{\mu}\right)^{T} \sum_{T+1}^{R} \frac{\prod_{i=1}^{i-T} \lambda_i}{i!\ \mu^{i-T}} \tag{3}$$

and μ is the mean service time. The average utilization of the system, u_{av}, and the blocking probability of T threshold, P_{blk}, is given by

$$u_{av} = \frac{1}{R} \sum_{k=0}^{R} kP_k \tag{4}$$

$$P_{blk} = \sum_{k=T+1}^{R} P_k \tag{5}$$

We have obtained the value of threshold T using two different methods. In the first method, we have solved the system of equations (1) to (4) according to given utilization threshold, U_{thr}. In the second method, we have applied exhaustive search to find optimal value of T threshold which renders minimal difference between estimated utilization and utilization threshold, U_{thr}.

The task admission scheme described here supports one class of tasks only, but the extension to multiple classes is straightforward. One possible solution tailored to mobile clouds is proposed and analyzed in [14].

3.4. Task admission scheme based on current utilization

The impact of random fluctuations of task arrival rate is dampened by the EWMA technique, which means that large changes of mean task arrival rate take longer time to propagate due to exponential smoothing. Therefore, Algorithm 1 is appropriate for systems that experience steady values of mean task arrival rate. When task arrival rate changes rapidly and/or mean task arrival rate changes as well, a different algorithm must be used.

In this solution, the system tracks U_{thr} as closely as possible by exhaustive search over admission arrival rates to arrive at the instantaneous utilization value closest to the utilization threshold, U_{thr}. Algorithm 2 demonstrates this approach, inspired by the delta modulation technique used in pulse code modulation [19]. When a task arrives, offered load ρ is calculated using the current task arrival rate.

Algorithm 2 Admission mechanism based on current utilization.

```
flg_o ⟵ 0;
Upon task arrival:
  Consider the current utilization including the
  pending task (u'_c);
  if u'_c > U_thr then
    if |u'_c − u'_prev| > M_thr or flg_o = 0 then
      for T_i = U_thr · R to R do
          Adopt the instantaneous utilization (u_c)
          equation from the system of equations (1)
          to (4);
          Apply T_i threshold to u_c to obtain uc_i;
          Calculate Fc_i according to uc_i;
          Calculate δ_i ⟵ |uc_i − U_thr|;
      end for
      Find the minimum δ_i;
      Select the corresponding T_i and Fc_i as the
      optimal values of T and F_cf respectively;
      flg_o ⟵ 1;
    else
      Use the previous T as the current T;
      Apply T threshold to u_c and calculate F_cf
      according to u_c;
    end if
    Reject the incoming task with the probability of
    1 − F_cf;
  else
    Accept the incoming task;
  end if
```

Also, the current utilization including the pending task, u'_c, is calculated. If u'_c is less than the utilization threshold, U_{thr}, the system accepts the incoming task. Otherwise, if the difference between u'_c and the previous utilization before considering T threshold, u'_{prev}, is more than a threshold margin, M_{thr}, or if it is the first time occurrence of utilization more than U_{thr}, the system examines the threshold range of $U_{thr} \cdot R$ to R. In this case, T_i, the threshold position of full rate task acceptance will be replaced in the current utilization (u_c) equation, adopted from the system of equations (1) to (4). It then calculates the filtering coefficient (Fc_i) corresponding to T_i. Then, uc_i will be computed using the values of (Fc_i) and T_i. Also, the difference between uc_i and U_{thr}, δ_i, is calculated. The minimum value of δ_i represents the closest value of utilization to U_{thr}, and the corresponding T_i and Fc_i will be chosen as tar-

get T and F_{cf} values, respectively. Also, if the difference between u'_c and u'_{prev} is less than M_{thr} margin, system will use the previous T as the current T, and after applying this T to u_c, F_{cf} can be calculated. The system will reject the incoming task with the probability of $1 - F_{cf}$.

4. PERFORMANCE EVALUATION

To investigate the behavior of the admission control model, we have evaluated different performance parameters under varying levels of offered load, ρ, calculated as $\rho = \frac{\lambda_f}{v \cdot N \cdot \mu}$. In our experiments, $N = 100$ is the number of PMs, and $v = 10$ is the maximum number of VMs on each PM; therefore, the system's capacity is 1000 VMs. We have kept the task service time fixed and varied the task arrival rate to achieve the offered load in the range from 0.4 to 0.95. In order to investigate the effect of mean service time variation on the performance metrics separate from the effects of offered load and task arrival rate changes, we have used four different mean service times of 20, 40, 60, and 80 min for each parameter. The utilization threshold, U_{thr}, is set to 75%; in this range of value, U_{thr} is relatively high, and the system is operating well below saturation in which a significant number of tasks will get blocked due to the excessive traffic load.

Figure 3 shows the accepted rate of incoming tasks. At values of offered load under 0.75, an increase of arrival rate causes the accepted rate to increase as well. Beyond offered load of 0.75, accepted rate of incoming tasks essentially flattens so as to keep the average utilization in the defined threshold. Also, in order to have the same ranges of offered load in Figure 3, with increase of fixed service time, the accepted rate of incoming tasks moves to the lower ranges – from maximum value of 0.64 tasks per second (or 2300 tasks per hour) to 0.16 tasks per second (570 tasks per hour).

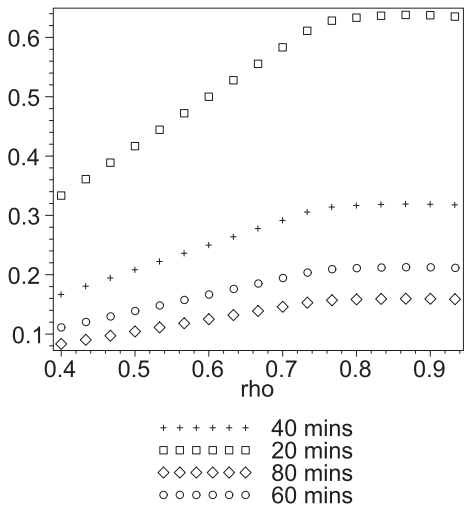


Figure 3. Accepted rate of incoming tasks at fixed service time and variable task arrival rate.

4.1. Performance of Algorithm 1

In this section, we investigate the performance of task admission scheme that uses Algorithm 1.

Figure 4 presents the threshold positions according to changing offered load for four fixed service times. As can be seen in Figure 4, the same offered load can result in the same threshold position. Also, with increasing of offered load or in other word, increasing of incoming task, threshold position moves to higher number of VMs in order to keep the average utilization around 75%. Boxes represent the obtained thresholds according to Equation (4), and crosses are the obtained thresholds of the exhaustive search method discussed in Section 3.3. As can be seen, the outcomes of two methods match quite well.

Figure 5 illustrates the blocking probability of admission control process. In Figure 5, when offered load is below 0.75, the blocking probability of admission control process is negligible, but when offered load goes beyond 0.75, the blocking probability increases when the offered load

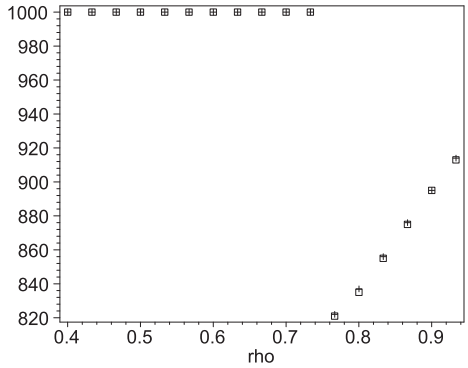


Figure 4. Threshold position at fixed service times of 20, 40, 60, and 80 min and variable task arrival rate using Algorithm 1.

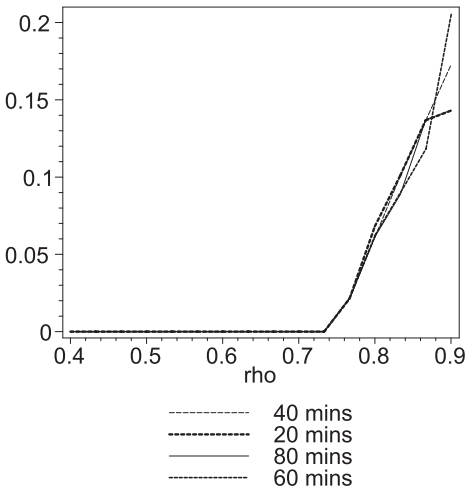


Figure 5. Blocking probability at fixed service time and variable task arrival rate using Algorithm 1.

increases. This is because the system is getting full, and it is blocking more tasks to maintain the desired level of utilization. Also, it can be seen that in different service times, the same offered load results in almost the same blocking probability, despite the slight discrepancies which arise from rounding errors in the computations. In Algorithm 1, we have obtained the value of threshold T by solving the system of equations (1) to (4) according to given utilization threshold, U_{thr}. Threshold is expected to be an integer value, and in its calculation, we have round the obtained value to floor. This integer value of threshold is used in the calculation of ρ_{av}, u'_{av}, F_{cf}, and so on; therefore, it can cause slight discrepancies in the calculations.

Figure 6 shows the utilization of the admission control scheme. This figure presents the outcome of applying the obtained threshold to the system. As expected, in all cases with choosing appropriate threshold position, utilization is not degrading, and with offered loads higher than 0.75, utilization gets into the stable condition of utilization threshold, U_{thr}. Therefore, our goal of keeping the utilization around a specific threshold is achieved.

In Figure 7, the value of filtering coefficient in four cases has been presented. As expected, in the same range of offered load, these coefficients are identical; this is because in Algorithm 1, filtering coefficient is calculated as a function of offered load, and therefore, the same offered load gives same filtering coefficient. Also, in all cases, when offered load is less than 0.75, the system does not filter the incoming tasks, but when offered load is getting higher than 0.75, system drops some of the arriving tasks. With increasing of offered load, dropping rate of arriving tasks increases linearly to compensate the target utilization threshold. In the worst case of $\rho = 95\%$, system only accepts 82% of the incoming tasks.

Figure 8 shows the total delay that includes the total resource provisioning time in a server pool. As can be seen, higher service time generally results in higher delay, but the admission control scheme not only prevents abrupt changes of total delay but also manages to maintain it at an almost steady value.

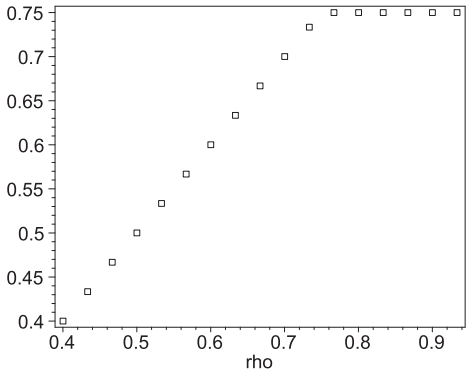


Figure 6. Utilization at fixed service times of 20, 40, 60, and 80 min and variable task arrival rate using Algorithm 1.

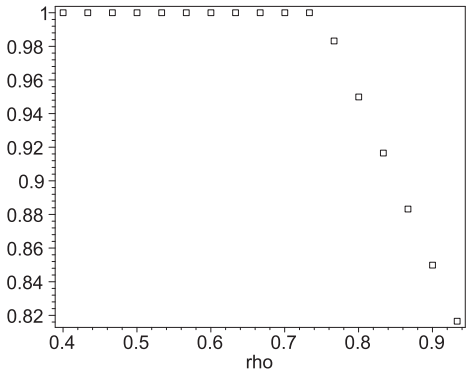


Figure 7. Filtering coefficient at fixed service times of 20, 40, 60 and 80 min and variable task arrival rate using Algorithm 1.

4.2. Performance of Algorithm 2

We now present the performance of task admission scheme that uses Algorithm 2. In Figure 9, when offered load goes beyond 0.75, threshold position approximately moves in

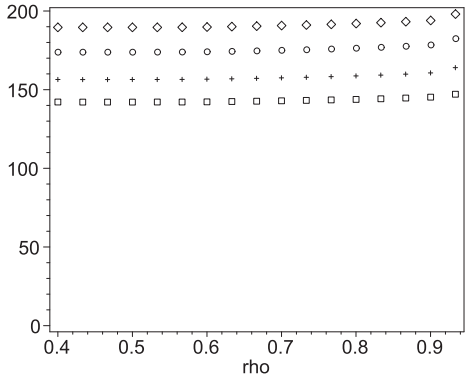


Figure 8. Total delay at fixed service times of 20, 40, 60, and 80 min and variable task arrival rate using Algorithm 1. Legends are the same as the legends in Figure 3.

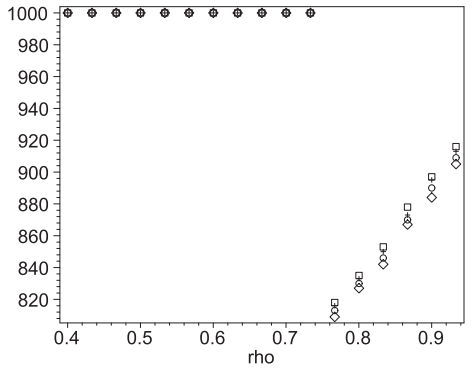


Figure 9. Threshold position at fixed service times of 20, 40, 60, and 80 mins and variable task arrival rate using Algorithm 2. Legends are the same as the legends in Figure 3.

the range from 805 to 915 VMs to keep the average utilization around 75%, similar to Figure 4; although, in this case, threshold position is found through exhaustive search. As can be seen, similar to Figure 4, the same offered load gives presents almost the same threshold position, and the slight differences are the result of rounding errors in the calculations. In Algorithm 2, the minimum value of δi is calculated that represents the closest value of instantaneous utilization to U_{thr}. As mentioned in Section 3.4, we have been inspired by delta modulation technique used in PCM to calculate the minimum value of δi and the obtained technique involves some approximation. The algorithm determines or calculates the optimal values of threshold and filtering coefficient according to the minimum value of δi.

The blocking probability of the alternative solution is shown in Figure 10. The blocking probability of this solution is slightly higher than the blocking probability presented in Figure 5; however, the range and pattern of changes of blocking probabilities are close. In the case of offered load larger than 75%, the system is getting full, and it is blocking more tasks to maintain the desired level of utilization. Also, it can be seen that in different service times, the same offered load results in almost the same blocking probability, despite the slight discrepancies which come from rounding errors in the calculations.

System utilization in Figure 11 matches the utilization presented in Figure 6; although in Figure 11, utilization is slightly changing over the different offered loads.

The pattern and the range of changes of filtering coefficient in Figure 12 are similar to the pattern and ranges of changes illustrated in Figure 7; although, in Figure 12, the filtering coefficient is decreasing slightly faster. Also, the slight discrepancies of filtering coefficients in Figure 12 are the result of rounding errors in the computations, and the same offered load presents almost the same filtering coefficient.

Also, mean task delay shown in Figure 13 matches the total delay in Figure 8.

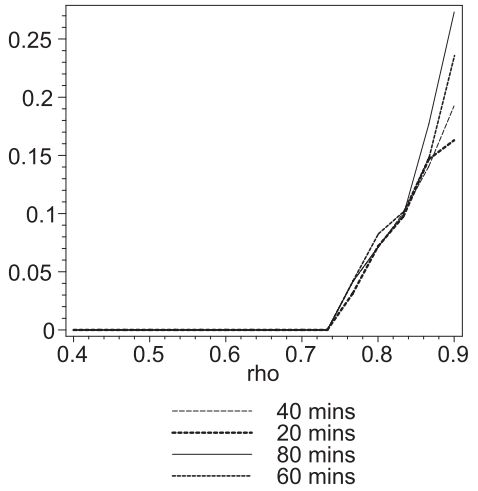


Figure 10. Blocking probability at fixed service time and variable task arrival rate using Algorithm 2.

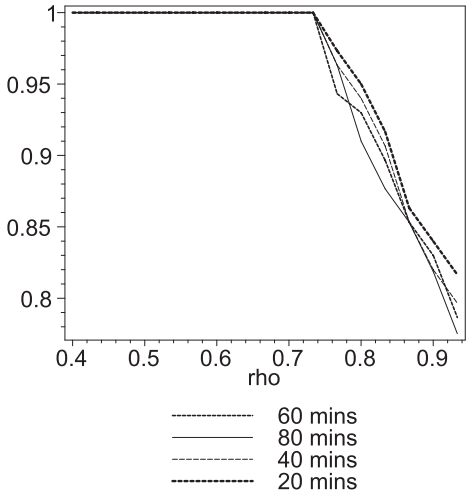


Figure 12. Filtering coefficient at fixed service time and variable task arrival rate using Algorithm 2.

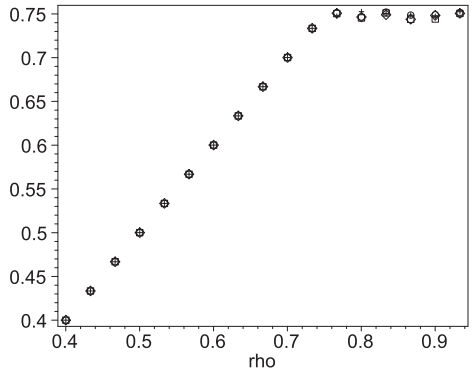


Figure 11. Utilization at fixed service times of 20, 40, 60, and 80 min and variable task arrival rate using Algorithm 2. Legends are the same as the legends in Figure 3.

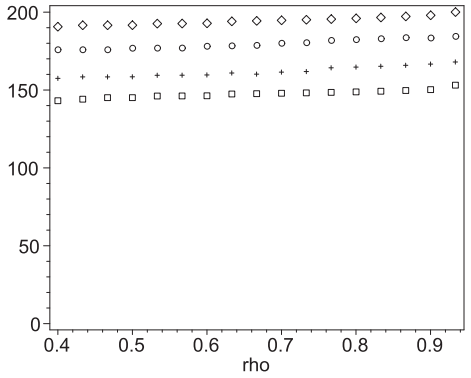
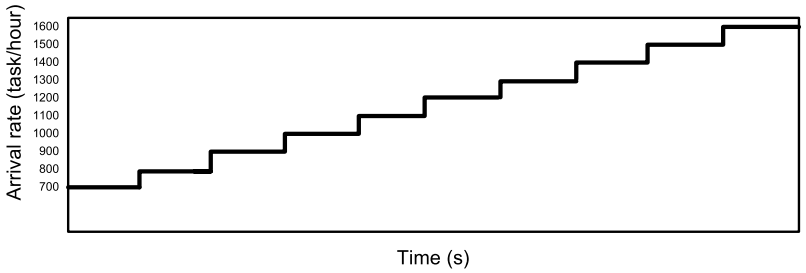
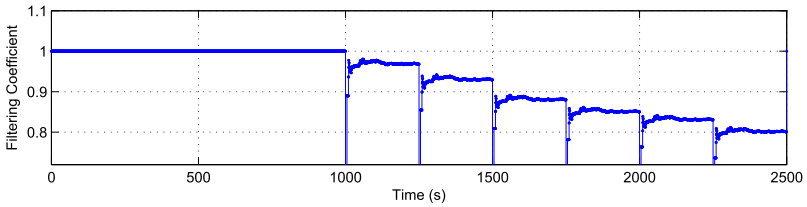


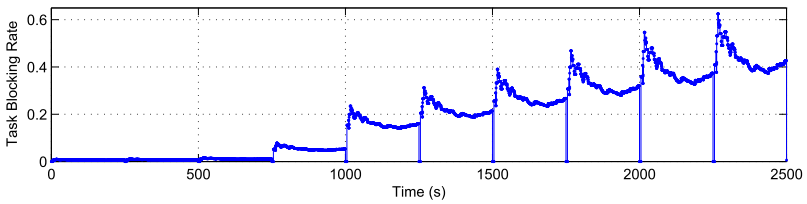
Figure 13. Total delay at fixed service times of 20, 40, 60, and 80 min and variable task arrival rate using Algorithm 2. Legends are the same as the legends in Figure 3.



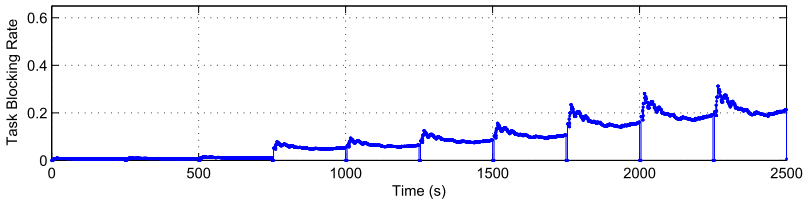
(a) Task arrival rate changes during the test time.



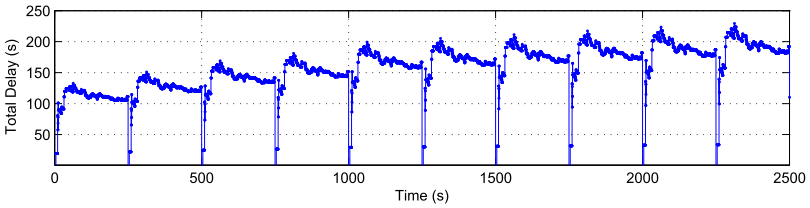
(b) Filtering coefficient with admission control (Algorithm 1).



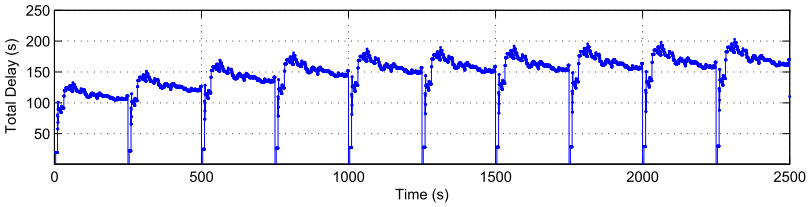
(c) Task blocking probability without admission control.



(d) Task blocking probability with admission control (Algorithm 1).

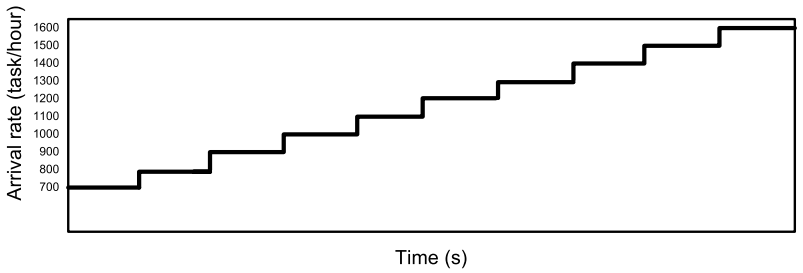


(e) Total delay without admission control.

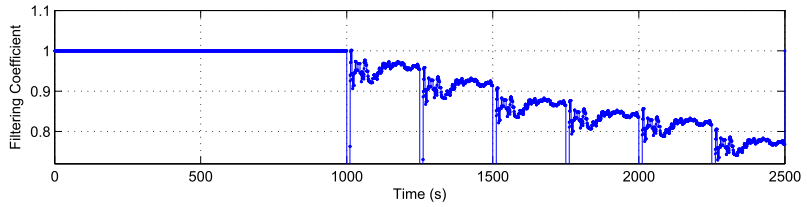


(f) Total delay with admission control (Algorithm 1).

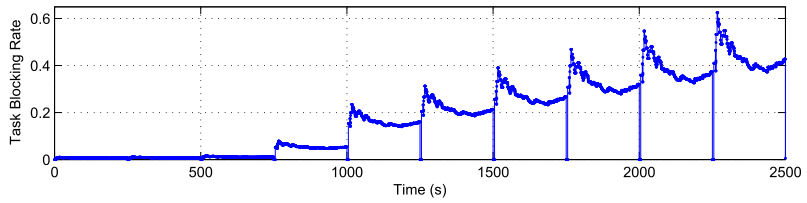
Figure 14. Test case of $Lq = 50$, task arrival rate variable from 700 to 1600 per hour, service time 40 min. $F_{cf} = 1 - (\rho_{av} - \rho_{thr})$.



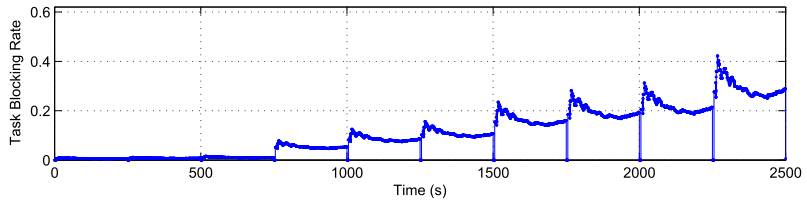
(a) Task arrival rate changes during the test time.



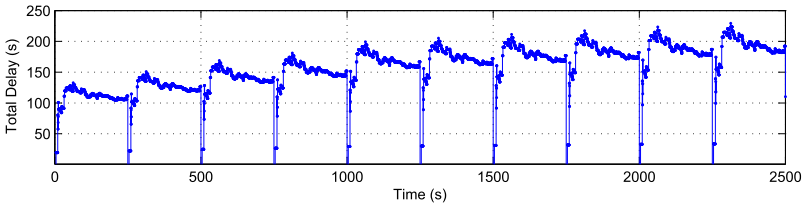
(b) Filtering coefficient with admission control (Algorithm 2).



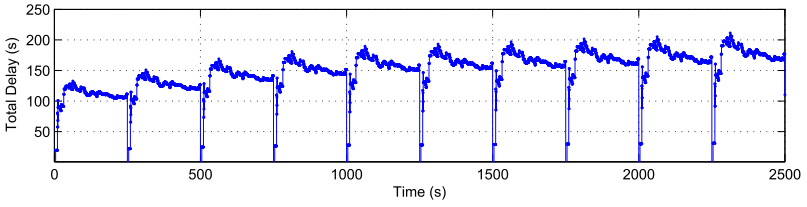
(c) Task blocking probability without admission control.



(d) Task blocking probability with admission control (Algorithm 2).

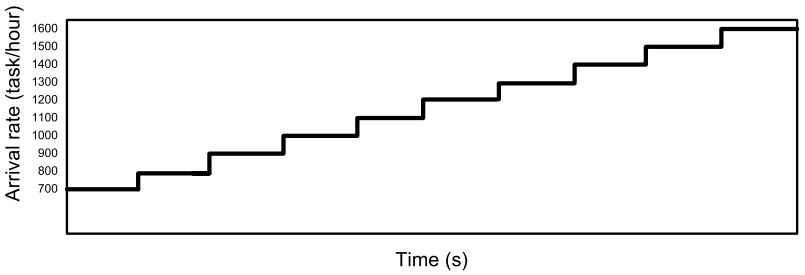


(e) Total delay without admission control.

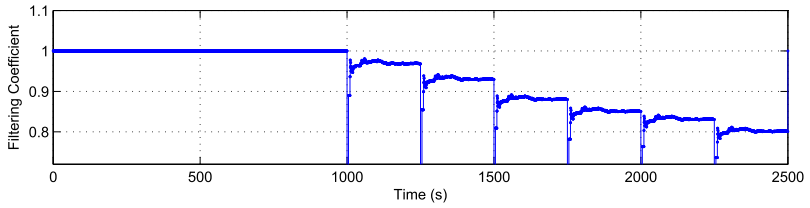


(f) Total delay with admission control (Algorithm 2).

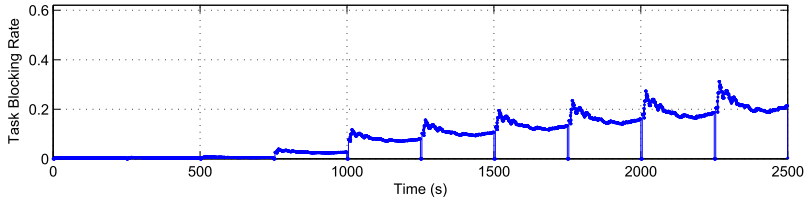
Figure 15. Test case of $Lq = 50$, task arrival rate variable from 700 to 1600 per hour, service time 40 min.



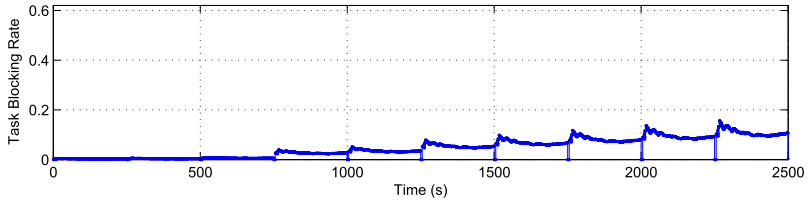
(a) Task arrival rate changes during the test time.



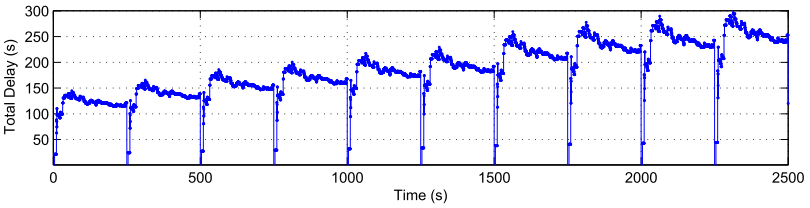
(b) Filtering coefficient with admission control (Algorithm 1).



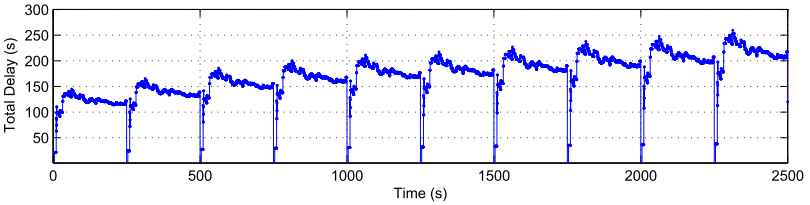
(c) Task blocking probability without admission control.



(d) Task blocking probability with admission control (Algorithm 1).

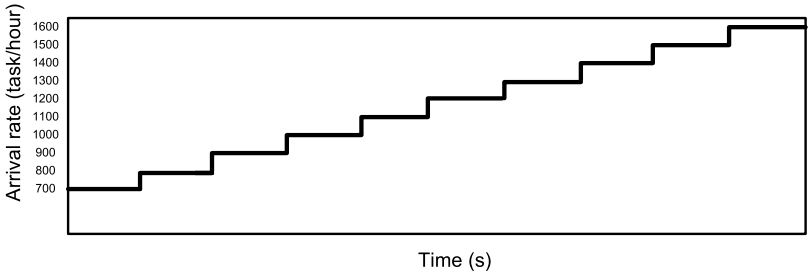


(e) Total delay without admission control.

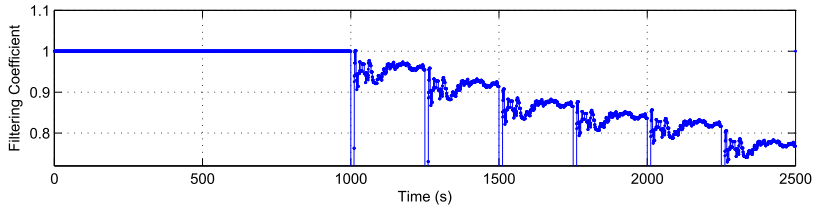


(f) Total delay with admission control (Algorithm 1).

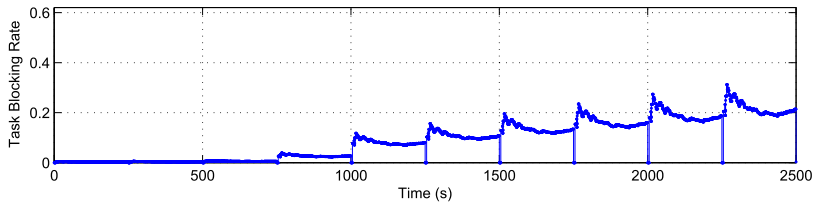
Figure 16. Test case of $Lq = 200$, task arrival rate variable from 700 to 1600 per hour, service time 40 min. $F_{cf} = 1 - (\rho_{av} - \rho_{thr})$.



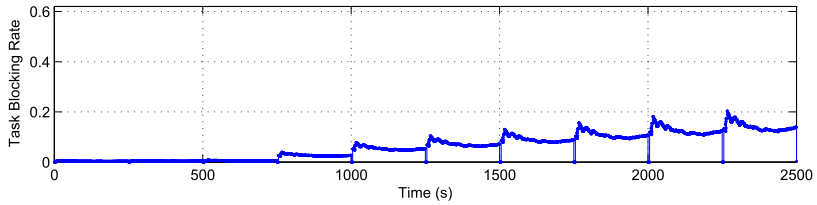
(a) Task arrival rate changes during the test time.



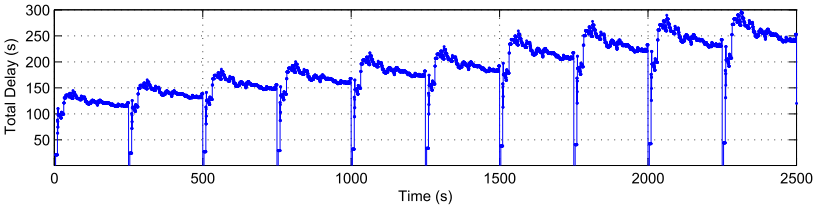
(b) Filtering coefficient with admission control (Algorithm 2).



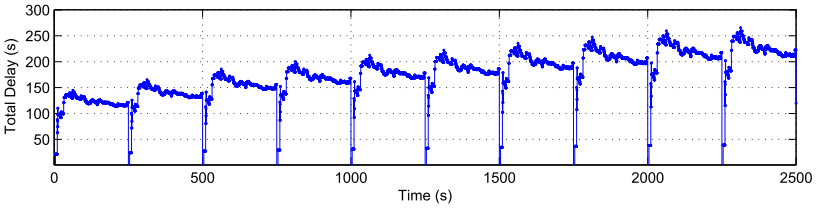
(c) Task blocking probability without admission control.



(d) Task blocking probability with admission control (Algorithm 2).

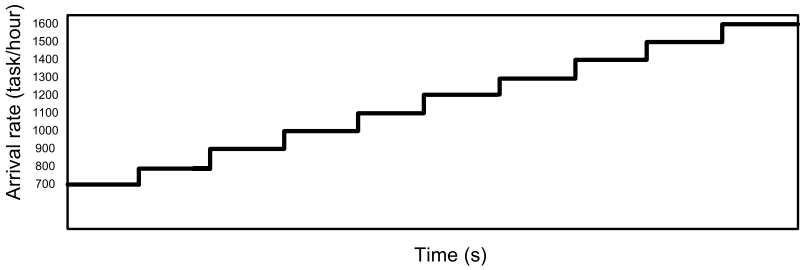


(e) Total delay without admission control.

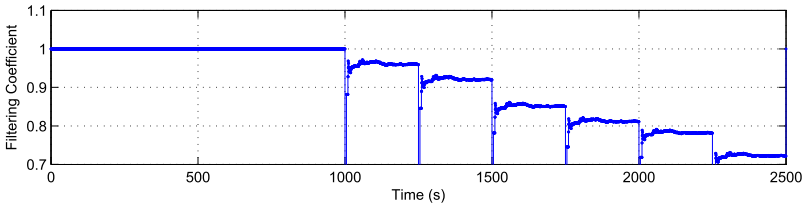


(f) Total delay with admission control (Algorithm 2).

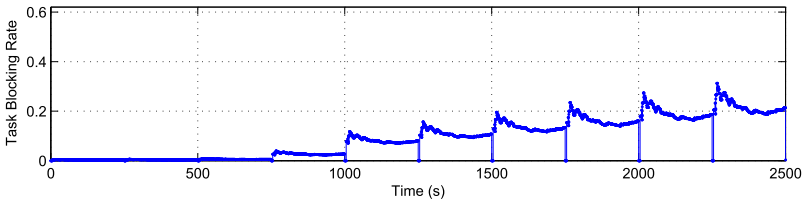
Figure 17. Test case of $Lq = 200$, task arrival rate variable from 700 to 1600 per hour, service time 40 min.



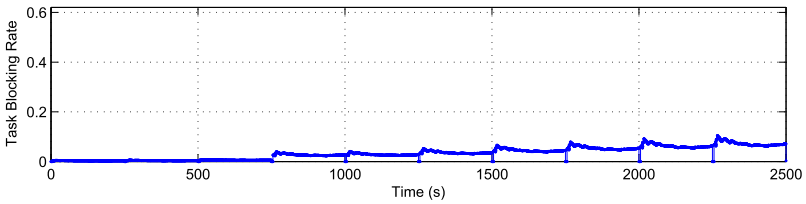
(a) Task arrival rate changes during the test time.



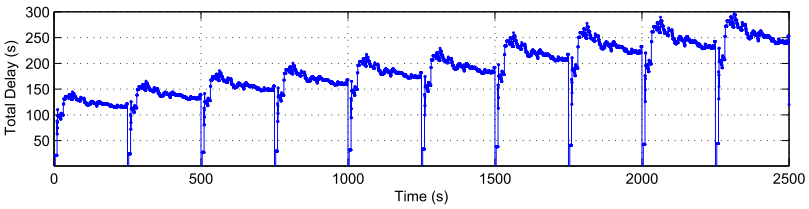
(b) Filtering coefficient with admission control (Algorithm 1).



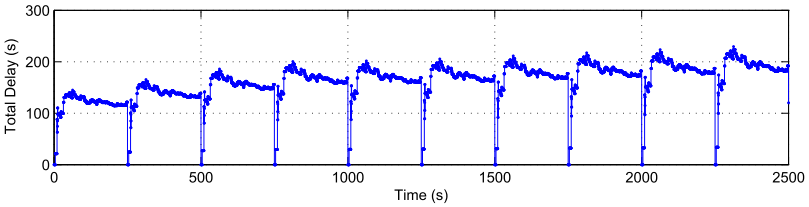
(c) Task blocking probability without admission control.



(d) Task blocking probability with admission control (Algorithm 1).



(e) Total delay without admission control.



(f) Total delay with admission control (Algorithm 1).

Figure 18. Test case of $Lq = 200$, task arrival rate variable from 700 to 1600 per hour, service time 40 min. $F_{cf} = 1-((\rho_{av}-\rho_{thr})/\rho_{thr})$

4.3. Analyzing the effect of the size of waiting queue and filtering coefficient on system performance

In this subsection, we analyze the effect of changing the size of FIFO waiting queue and filtering coefficient in Algorithm 1 on the system. In the results presented in Section 4.1 and 4.2, we have assumed that the size of waiting queue, Lq, is equal to 50. Also, filtering coefficient, F_{cf}, in Algorithm 1 is calculated as $F_{cf} = 1 - (\rho_{av} - \rho_{thr})$.

Figures 14 and 15 illustrate the simulation snapshots when the waiting queue has a smaller size of $Lq = 50$. Under Algorithm 1, filtering coefficient, F_{cf} is calculated as $F_{cf} = 1 - (\rho_{av} - \rho_{thr})$.

We change the task arrival rate from 700 to 1600 tasks per hour in order to investigate the behavior of both algorithms in different operation regions including linear, transition to saturation, and saturation. As can be seen in Figure 14(a) and (b), when task arrival rate is less than 1100 tasks per hour, or in other words, offered load is less than 0.75, system admits all arriving tasks; whereas, when offered load goes beyond 0.75, system starts rejecting some tasks and filtering coefficient decreases. According to task blocking rates presented in Figure 14(c) and (d), using Algorithm 1 results in almost 50% decreasing of the task blocking probability. Also, comparison of Figure 14(e) and (f) shows that using Algorithm 1 improves the total delay.

Figure 15 illustrates the snapshots of performance parameters when Algorithm 2 is used and no task admission control is applied. In Figure 15(b), similar to Figure 14(b), when offered load goes beyond 0.75, filtering coefficient decreases; although, in the case of using Algorithm 2, the dropping rate is slightly higher. Also, according to the task blocking rate presented in Figure 15(d), using Algorithm 2 improves the task blocking rate; although, the blocking rate improvement is not as high as the case of Algorithm 1. This is due to the heavy load of computation in Algorithm 2.

Figures 16 and 17 present the simulation snapshots when the waiting queue has a much larger size of $Lq = 200$. Under Algorithm 1, F_{cf} is calculated as $F_{cf} = 1 - (\rho_{av} - \rho_{thr})$.

If we compare Figures 16(c) and 14(c), we can see that even without using any admission control mechanism, increasing the size of waiting queue has a significant positive effect on task blocking rate; this is because more tasks have a chance to stay in the larger waiting queue, and they do not get rejected. As shown in Figure 16(d), using Algorithm 1 also improves the task blocking rate. However, Figure 16(e) and (f) indicates that using a larger waiting queue causes higher delay in the system; although, using Algorithm 1 still results in lower delay.

Similar to the case of Algorithm 1, using Algorithm 2 and increasing the size of waiting queue have a positive effect on task blocking rate (Figure 17(d)), and Algorithm 2 improves the total delay (Figure 17(f) compared with 17(e)); although, in case of using Algorithm 2, delay is slightly higher than case of Algorithm 1. Also, the larger waiting queue ($Lq = 200$) results in the higher delay than the short waiting queue ($Lq = 50$). This is because of experiencing longer waiting times in larger queues.

Furthermore, we have tried to achieve higher system performance in Algorithm 1 by using more aggressive filtering, where F_{cf} is calculated as $F_{cf} = 1 - ((\rho_{av} - \rho_{thr})/\rho_{thr})$. The results presented in Figure 18(d) and (f) demonstrate that this approach results in further improvement of system performance – *cf.* the decrease of task blocking rate and total delay compared with its counterpart case presented in Figure 16(d) and (f).

5. CONCLUSION

In this paper, we have proposed and analyzed two task admission schemes that try to maintain the desired level of utilization in cloud datacenters. Our performance analysis confirms that both schemes are capable of achieving the desired goal without compromising the stability of the cloud system. However, one of the schemes, based on exponential averages of system utilization, is better suited to systems with relatively stationary task request arrivals, while systems with varying mean task arrival rate are better serviced by the other scheme, which is based on instantaneous utilization and, consequently, computationally more demanding.

Our future work will focus on extending these schemes to cloud systems with two or more task request classes with different priority levels and, possibly, other requirements as well, and to further refinement of the proposed algorithms.

REFERENCES

1. Khazaei H, Mišić J, Mišić VB. Performance analysis of cloud computing centers using $M/G/m/m + r$ queueing systems. *IEEE Transactions on Parallel and Distributed Systems* 2012; **23**(5): 936–943.
2. Khazaei H., Mišić J, Mišić VB, Rashwand S. Analysis of a pool management scheme for cloud computing centers. *IEEE Transactions on Parallel and Distributed Systems* 2013; **24**(5): 849–861.
3. Urgaonkar R, Kozat UC, Igarashi K, Neely MJ. Dynamic resource allocation and power management in virtualized data centers. In *IEEE Network Operations and Management Symposium (NOMS)*, Osaka, Japan, April 2010; 479–486.
4. Leontiou N, Dechouniotis D, Denazis S. Adaptive admission control of distributed cloud services. In *International Conference on Network and Service Management (CNSM)*, Niagara Falls, ON, Canada, October 2010; 318–321.
5. Ali-Eldin A, Tordsson J, Elmroth E. An adaptive hybrid elasticity controller for cloud infrastructures. In *IEEE Network Operations and Management Symposium (NOMS)*, Maui, HI, April 2012; 204–212.

6. Feldman Z, Masin M, Tantawi AN, Arroyo D, Steinder M. Using approximate dynamic programming to optimize admission control in cloud computing environment. In *the Winter Simulation Conference 2011 (WSC)*, Phoenix, AZ, December 2011; 3153–3164.
7. Hoang DT, Niyato D, Wang P. Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In *IEEE Wireless Communications and Networking Conference (WCNC)*, Paris, France, April 2012; 3145–3149.
8. Dechouniotis D, Leontiou N, Athanasopoulos N, Bitsoris G, Denazis S. ACRA: a unified admission control and resource allocation framework for virtualized environments. In *8th International Conference on Network and Service Management (CNSM) workshops*, Las Vagas, NV, October 2012; 145–149.
9. He Y, Huang J, Duan Q, Xiong Z, Lv J, Liu Y. *A Novel Admission Control Model in Cloud Computing*, 2014. Available from: http://arxiv.org/abs/1401.4716v2 [accessed on September 1, 2014]; 1–14.
10. Konstanteli K, Cucinotta T, Psychas K, Varvarigou T. Elastic admission control for federated cloud services. *IEEE Transactions on Cloud Computing* 2014; **2**(3): 348–361.
11. Ashraf A, Byholm B, Porres I. A session-based adaptive admission control approach for virtualized application servers. In *Fifth IEEE International Conference on Utility and Cloud Computing (UCC)*, Chicago, IL, November 2012; 65–72.
12. Tomas L, Tordsson J. Improving cloud infrastructure utilization through overbooking. In *ACM Cloud and Autonomic Computing Conference (CAC)*, Miami, FL, August 2013; 1–10.
13. Shi C, Habak K, Pandurangan P, Ammar M, Naik M, Zegura E. COSMOS: computation offloading as a service for mobile devices. In *15th ACM International Symposium Mobile ad Hoc Networking and Computing (MobiHoc)*, Philadelphia, PA, August 2014; 287–296.
14. Khojasteh H, Mišić J, Mišić VB. Prioritization of overflow tasks to improve performance of mobile cloud. *IEEE Transactions on Cloud Computing* 2016: 1–11, DOI:10.1109/TCC.2016.2535240.
15. Kao Y, Krishnamachari B, Ra M, Bai F. Hermes: latency optimal task assignment for resource-constrained mobile computing. In *IEEE INFOCOM 2015*, Hong Kong, April 2015; 1894–1902.
16. Khazaei H, Mišić J, Mišić VB. Performance of cloud centers with high degree of virtualization under batch task arrivals. *IEEE Transactions on Parallel and Distributed Systems* 2013; **24**(12): 2429–2438.
17. Khojasteh H, Mišić J, Mišić VB. Characterizing energy consumption of IaaS clouds in non-saturated operation. In *IEEE INFOCOM 2014 Workshops*, Toronto, ON, Canada, April 2014; 398–403.
18. Brown RG. Statistical forecasting for inventory control. In *McGraw-Hill*, New York, 1959.
19. Steele R. Delta modulation systems. In *Pentech Press*, London, 1975; 55–81.
20. Khojasteh H, Mišić J, Mišić VB. Task filtering as a task admission control policy in cloud server pools. In *IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 2015)*, Dubrovnik, Croatia, August 2015; 1–6.

AUTHORS' BIOGRAPHIES

Haleh Khojasteh has received her PhD in Computer Science from Ryerson University in 2016. She received her MS (2011) degree in Computer Science from Ryerson University, and her BS degree in Electrical and Computer Engineering from Shahid Beheshti University, Tehran, Iran. Her research interests are cloud computing and wireless networks with emphasis on mathematical modeling and performance analysis.

Jelena Mišić is a Professor in the Department of Computer Science at Ryerson University, Canada. She received her PhD in Computer Engineering from University of Belgrade, Serbia, in 1993. She is an internationally recognized expert in the area of wireless networking, vehicular networks, and network security, where she has authored or co-authored four books, 115 journal papers, 24 book chapters, and 170 conference papers. She has chaired more than a dozen major international events and guest-edited more than a dozen special issues of various journals. She serves on the editorial boards of IEEE Transactions on Vehicular Technology, IEEE Network, Computer Networks and Ad Hoc Networks journals (both published by Elsevier), Wiley's Security and Communication Networks.