Annals of Telecommunications manuscript No.
DOI 10.1007/s12243-016-0527-6

Scheduling Internet of Things Applications in Cloud Computing

Husnu S. Narman · Md. Shohrab Hossain · Mohammed Atiquzzaman · Haiying Shen

Received: Date / Accepted: Date

Abstract Internet of Things (IoT) is one of the greatest technology revolutions in the history. Due to IoT potential, daily objects will be consciously worked in harmony with optimized performances. However, today's technology is not ready to fully bring its power to our daily life because of huge data analysis requirements in instant time. On the other hand, the powerful data management of cloud computing gives IoT an opportunity to make the revolution in our life. However, the traditional cloud computing server schedulers are not ready to provide services to IoT because IoT consists of a number of heterogeneous devices and applications which are far away from standardization. Therefore, to meet the expectations of users, the traditional cloud computing server schedulers should be improved to efficiently schedule and allocate IoT requests.

There are several proposed scheduling algorithms for cloud computing in the literature. However, these scheduling algorithms are limited because of considering neither heterogeneous servers nor dynamic scheduling approach for different priority requests. Our objective is to propose Dynamic Dedicated Server Scheduling

Husnu S. Narman
Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson SC, 29634
Tel.: 864-656-3190
E-mail: husnu@ou.edu

Md. Shohrab Hossain
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology
Tel.: 880 2 966 5650 / 6503
E-mail: mshohrabhossain@cse.buet.ac.bd

Mohammed Atiquzzaman
School of Computer Science, University of Oklahoma, Norman, OK 73019
Tel.: 405-325-8077
Fax: 405-325-4044
E-mail: atiq@ou.edu

Haiying Shen
Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson SC, 29634
Tel.: 864-656-3190
E-mail: shenh@clemson.edu

for heterogeneous and homogeneous systems to efficiently provide desired services by considering priorities of requests. Results show that the proposed scheduling algorithm improves throughput up to 40% in heterogeneous and homogeneous cloud computing systems for IoT requests. Our proposed scheduling algorithm and related analysis will help cloud service providers build efficient server schedulers which are adaptable to homogeneous and heterogeneous environments by considering system performance metrics, such as drop rate, throughput, and utilization in IoT.

Keywords Internet of Things · Cloud Computing · Analytical Model · Heterogeneous and Homogeneous Multi server · Multi-class · Queuing System · Performance · Priority.

1 Introduction

One of the greatest technology revolutions is undoubtedly Internet of Things (IoT). Due to the potential of IoT, daily objects can be programmable (called smart devices) and consciously work in harmony with optimized performances [1–5]. Therefore, IoT is changing the way how daily objects (such as phones to smartphones, TV to smart TV, etc.) work in our life. Regularly, the number of IoT capable devices is increasing. The number of smartphones and tablets for 2013 passed one billion and the expected number for 2017 is almost two billion [6]. However, the expected number of IoT device units in 2020 is 20 billion [4]. Although the number of Internet-connected devices increases, today's technology is not ready to fully bring the power of IoT in our daily life because a huge amount of data, which is generated by many Internet-connected devices, is required to be analyzed. On the other hand, the powerful data management of cloud computing gives IoT an opportunity to make the revolution in our life.

Simplicity of usage, flexibility of data access, ease of maintenance, time and energy efficiency, and pay as you go policy have already increased the usage of cloud computing over the traditional computing [7,8]. However, the traditional cloud computing server schedulers are not ready to provide services to IoT because it consists of a number of heterogeneous devices and applications which are far away from standardization. Therefore, to meet the expectations of users, the traditional cloud computing server schedulers should be improved to schedule efficiently and allocate IoT requests by considering the heterogeneity of servers and priorities (here requests can also be software, platform, infrastructure, etc.).

Cloud servers are generally assumed to be homogeneous servers (that have equal service rates and consist of the same type of devices) in the literature. However, cloud servers, in reality, are heterogeneous servers (such as having different service rates) because replacing failed or misbehaved servers with new and more powerful ones in a multi-server system makes the multi-server system heterogeneous [9]. Therefore, each server can support a limited number of Virtual Machines (VMs) according to its power. In addition, a server is capable of serving some types of requests rather than all types of requests in cloud computing [10]. Therefore, heterogeneous servers and their capabilities should be considered while designing server scheduling algorithms in cloud computing [11–14]. Because of heterogeneity, the distribution of requests

to servers (namely allocation policy) also needs to be taken into account. Moreover,

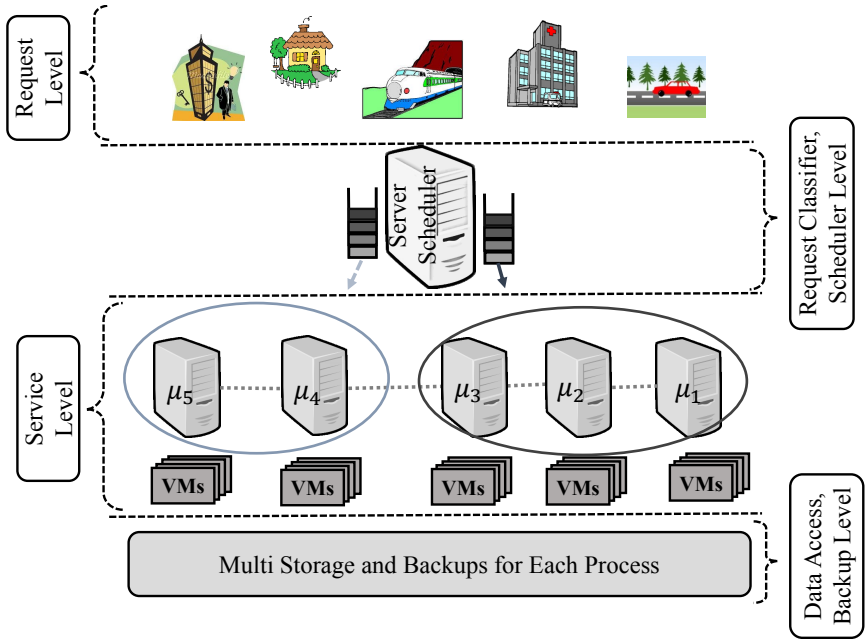


Fig. 1 Heterogeneous dedicated server groups for different types of IoT requests in cloud.

cloud computing systems must provide different levels of services to each application class such as health and shopping applications (see Fig. 1) [15,16,13]; the expectations of those applications can be different. For example, service expectations and requirements of a health application are naturally much higher than the shopping application; this means some classes of applications must have higher priorities than others. The definition of priority in cloud computing is, however, different from the general definition of priority in queuing systems. In cloud computing, priority can be used to decide the next request to be served and allocate the amount of resources for each request[17,18]. Fig. 2 shows high and low priorities with and without priority levels (Ψ_1 refers priority level of application 1 and Ψ_2 refers priority level of application 2).

As explained in our primary reports [17,18], the priority level term is used to quantify the difference between high and low priorities. In cloud computing, priority levels, which dynamically arranged according to service level agreement (SLA) and urgency of service, can be used to determine the amount of needed resources by each application. However, low and high priority terms without quantification do not mean anything in terms of resource allocation in cloud computing (see Fig. 2 column one). By using priority levels as shown in Fig. 2 (see second and third columns in Fig. 2) can help to allocate resources to each application. Moreover, server scheduling algorithms must use resources (such as servers) efficiently because of their limitations

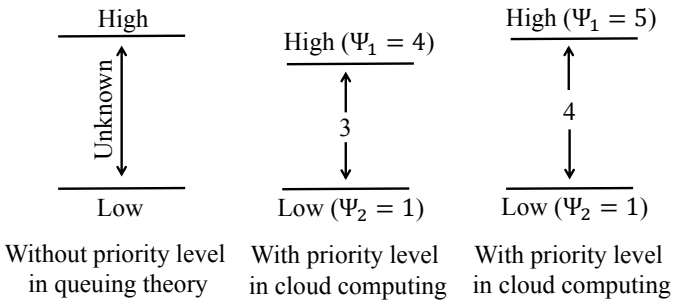


Fig. 2 High and low priorities without and with priority level.

and the excessive IoT data traffic flows. Therefore, server scheduling algorithms for cloud computing must satisfy expectations of each type of IoT requests in heterogeneous environments without wasting resources [7]. Therefore, the *aim* of this work is to propose a server scheduling algorithm by considering priority classes of IoT requests, and homogeneous and heterogeneous cloud servers. In this paper, we focus on more priority-based queuing analysis and priority-based scheduling algorithms in cloud server systems because the priority-based scheduling algorithm is more realistic for IoT due to the existence of different classes of non-standardization applications. It is important to note that our developed methods are also applicable to VM environments in cloud servers.

1.1 Objective and Contribution

The *objective* of this work is to improve and analyze the performance of cloud systems in terms of throughput, drop rate, and utilization by considering homogeneous and heterogeneous servers, and priority classes of IoT requests because some IoT requests such as a health application are delay sensitive. The key *contributions* of this work can be summarized as follows: (i) Homogeneous Dynamic Dedicated Server Scheduling (DDSS) and Heterogeneous Dynamic Dedicated Server Scheduling (h-DDSS) are proposed, and the scheduling procedure is explained. (ii) The upper and lower bound performance metrics (average occupancy, drop rate, average delay, and throughput for each class of application) of the proposed scheduling algorithm are analytically derived by using queuing theory. (iii) Derived expressions for performance metrics are verified by extensive simulations. (iv) The performance of DDSS and h-DDSS (here the performance of h-DDSS means as upper (Fastest Server First (FSF) which is integrated into our h-DDSS method) and lower bound performances of h-DDSS (Slowest Server First (SSF) which is integrated into our h-DDSS method)) are tested and compared with existing ones.

Results show that h-DDSS and DDSS significantly improve the drop rate and throughput by using proper priority levels for classes of IoT requests in cloud computing. Our proposed scheduling algorithm and related analysis will help cloud service providers build efficient server scheduling algorithms which are adaptable to homo-

geneous and heterogeneous server systems by considering the system performance metrics, such as drop rate, throughput, and utilization.

1.2 Organization

The rest of the paper is organized as follows: In Section 2, previous works are summarized and Section 3 has the details of DSS. The queuing models of the proposed DDSS and h-DDSS with scheduling algorithms are presented in Section 4. Section 5 includes analysis of the scheduling algorithms to derive upper and lower bounds performance of h-DDSS. In Section 7, we present the simulation and numerical results and compare the performances of DSS, DDSS, and h-DDSS. Finally, Section 8 has the concluding remarks with possible improvements in the future.

2 Related Works

Several research works [19–35] have been reported on performance analysis and scheduling algorithms for cloud computing.

Some of the research works [19–24] have analyzed performances of cloud computing systems. Authors in [19,20] proposed a queuing based analysis model for the performance evaluation of cloud systems using web applications as queues and VMs as service providers. Though creating unlimited VMs for each connection increases allocation rate, this dramatically decreases the performance of the system due to high response time [19] of each service request. Yang et al. [35] proposed a fault recovering system for cloud services and analyzed the system as an open queue problem. However, the result showed that the fault recovery system increases the average response time. Authors in [21,24] evaluated the performance of several cloud service providers for scientific computing tasks and found that the service providers are not ready to serve large data sets. Authors in [22,23] used a single class and a single queue model to analyze the performance of cloud computing and found several performance distributions. However, there are two limitations of the above works [19, 35,20–24]: (i) Different classes of applications (such as health and shopping applications) are not considered. (ii) Usage of heterogeneous servers is not taken into account in cloud networks. Therefore, a comprehensive analysis approach is not presented in cloud computing for IoT.

The scheduling algorithms can be grouped based on scheduling characteristics, such as dynamic or static, centralized or distributed, batch or immediate and co-operative or non-co-operative as explained in [36]. However, above mentioned scheduling groups do not consider distinct classes of requests. If classes of requests are considered, then scheduling algorithms can be grouped as pre-emptive or non-pre-emptive. Although pre-empitive scheduling algorithms can be used for tasks scheduling, it is not efficient in cloud computing because the class which has low priority can experience long delay and even dropping while getting service. Therefore, we have used a modified non-pre-empiteve model for our scheduling algorithm.

Several priority-based scheduling algorithms [29–34, 15, 16] are proposed for grid and cloud computing. In [29], Bansal et al. use priority in a static round robin scheduling algorithm to improve the performance. Authors in [32,33] dynamically schedule tasks by considering QoS and the performance metrics such as delay, deadline and execution times. In [34], Chtourou et al. consider resource constraints and apply two-phase (priority and threshold) to optimize the performance. All of the works in [29, 32–34] are priority-based scheduling in grid computing. However, adapting grid in cloud creates some efficiency problems, such as finding a server in grid to fully satisfy resource requirements for a type application while finding several servers in cloud to do so. In [31,30], authors develop priority-base scheduling algorithms in cloud computing. In both works [31,30], absolute priority is used; therefore, increasing the arrival rate of high priority requests leads to long delay for low priority requests. A threshold solves long delay issue for low priority class, but finding an optimum threshold is another problem.

Heterogeneity has recently been investigated in cloud computing [11–14, 10]. Mars et al. [11, 12] show the effects of the heterogeneity workloads on Google. Qi et al. [14] investigate the energy efficiency and workloads heterogeneity in cloud computing. Dastjerdi et al. [10], use different approaches by considering whole cloud systems such as discovery of resources, checking compatibility, selection of servers, and deployment to provide the required service to each user in heterogeneous environments. In [13], Delimitrou et al. proposed a QoS scheduling algorithm for heterogeneous servers with Paragon.

The features of the proposed scheduling algorithms in [15, 16, 13] have some aforementioned important requirements (such as addressing multiple classes of requests). In [16, 15, 13], authors have considered multiple classes of requests in cloud systems. Hu et al. [16] analyzed Shared Server Scheduling (SSS) and Dedicated Server Scheduling (DSS) for two priority classes of requests and obtained the minimum amount of sufficient service rates to satisfy each class. Ellens et al. [15] proposed a hybrid scheduling model for two priority classes of requests by reserving some servers for each class and sharing remaining servers. Delimitrou et al. [13] have used DSS for different type of classes. However, the recent works[15, 16] which are more related to our work, have considered neither priority levels of classes, nor heterogeneous server systems which require much harder complex analysis and [13] uses DSS which is based on policies of absolutely no sharing between different groups of servers for distinct classes of requests. The disadvantages of DSS are explained in Section 3. Moreover, our primary reports [18, 17] are the only existing works about priority level scheduling algorithm.

In [17], we have proposed DDSS for homogeneous server systems and compared the existing homogeneous Dedicated Servers Scheduling (DSS) with proposed DDSS scheduling by assuming DSS would be able to update the number of servers dynamically to see effects of priority level on the both scheduling. In [18], we have proposed h-DDSS for heterogeneous server systems and compared DDSS and h-DDSS to compare the performance of homogeneous and heterogeneous systems by using FSF and SSF allocation in h-DDSS to compare the priority level effects on both DDSS and h-DDSS. However, it is essential to analyze the performance of cloud computing by comparing proposed homogeneous and heterogeneous server systems with existing

DDS without any assumption to be able to understand dynamic scheduling effects on the systems.

In this paper, we have proposed a novel server scheduling algorithm for cloud computing by considering homogeneous and heterogeneous servers, priority level classes of IoT applications, arrival rates of IoT requests, and service rates of servers to answer mentioned requirements in IoT. The proposed server scheduling algorithm also considers both shared and dedicated server system and dynamically updates the assigned service rates (or the number of dedicated servers with VMs) for each request class. We have also shown the impacts of priority levels and the dynamic arrangement of dedicated servers (for each class) on the performance of the cloud computing systems. We have compared existing DSS, our proposed Dynamic Dedicated Server Scheduling for homogeneous servers (DDSS) and heterogeneous servers (h-DDSS). Consequently, the performance metrics of heterogeneous and homogeneous systems in cloud computing are analyzed and presented for IoT requests.

3 Dedicated Server Scheduling (DSS)

Fig. 3 shows DSS architecture for class 1 request (C_1) and class 2 request (C_2) traffic flows (such as requests of health and shopping applications). Here, some servers are used for C_1 and the others for C_2. The arrival rates of C_1 and C_2 are λ_1 and λ_2, respectively. Each request is enqueued in the corresponding buffers (Q_1 and Q_2). A

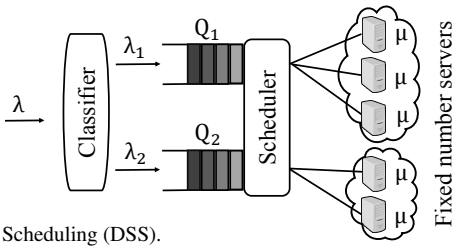


Fig. 3 Dedicated Server Scheduling (DSS).

new arriving request is dropped if the corresponding buffer is full. The service rate of each server is μ and all servers are capable of serving all types of applications (because of homogeneous server assumption) [16]. Each class of traffic is solely assigned to each dedicated server group as shown in Fig. 3. In the typical DSS [15, 16], there is absolutely no sharing of traffic among dedicated server groups, and the number of servers for each class is not updated dynamically. Therefore, increasing arrival rate of one type of requests (such as C_1) can decrease the throughput of the same type request although there may be available servers to provide services.

4 Dynamic Dedicated Server Scheduling

DDSS (see Fig. 4) frequently updates the number of dedicated servers with VMs for each class according to priority levels and arrival rates of C_1 and C_2. In contrast

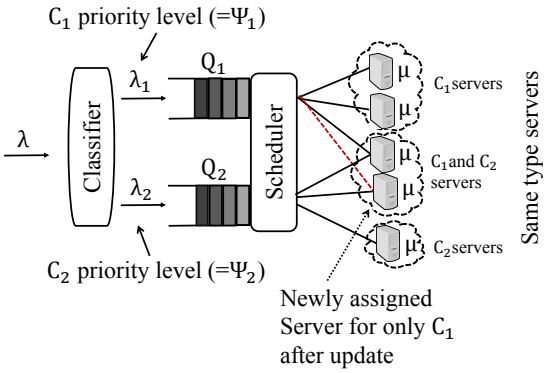


Fig. 4 Homogeneous Dynamic Dedicated Server Scheduling (DDSS).

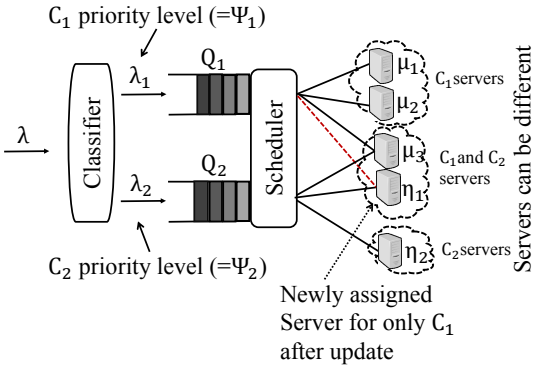


Fig. 5 Heterogeneous Dynamic Dedicated Server Scheduling (h-DDSS).

to DSS, capabilities of servers can be different. h-DDSS (see Fig. 5) also frequently updates the number of dedicated servers with VMs for each class according to priority levels and arrival rates of C_1 and C_2. In contrast to DSS and DDSS, service rates of servers can be different in h-DDSS. Capabilities of servers in h-DDSS can also vary as assumed for DDSS.

4.1 Notations

The notations used in the rest of the paper are listed in Table 1.

4.2 Scheduling Algorithm

In this section, the scheduling procedure and the measurement of the service rate for each class of request are presented.

Table 1 Notations

p_i	Probability of i number of C_1 requests in the system
λ	Arrival rate of total requests in the system
λ_1,λ_2	Arrival rates of C_1 and C_2
Ψ_1,Ψ_2	Priority levels of C_1 and C_2
μ_i, η_i	Service rate of i^{th} server for C_1 and C_2
K_{μ_i}, K_{η_i}	The number of VMs created in i^{th} server for C_1 and C_2
μ_{ti}, η_{ti}	Sum of service rates of dedicated servers for C_1 and C_2 until i^{th} server
μ_{total}	Total service rates of the system
m, k	Number of dedicated servers for C_1 and C_2
N	Size of Q_1 and Q_2
$\delta^{C_1}, \delta^{C_2}$	Average delays for C_1 and C_2
n^{C_1}, n^{C_2}	Average occupancies of C_1 and C_2
D^{C_1}, D^{C_2}	Drop probabilities of C_1 and C_2
$\gamma^{C_1}, \gamma^{C_2}$	Throughput for C_1 and C_2
δ	Average delay for a request in the system
n	Average occupancy in the system
D	Drop probability of a request from the system
γ	Throughput of the system

4.2.1 Service Rate Measurements

Our proposed server scheduling algorithm considers four crucial parameters that enable dynamic scheduling: (i) the arrival rates of C_1 and C_2 (λ_1, λ_2), (ii) the priority levels of C_1 and C_2 (Ψ_1, Ψ_2), (iii) the total service rate of servers in the system (μ_{total}) (iv) total service rates of servers which are only capable of serving one type of requests (μ_{C_1} for C_1 type of requests and μ_{C_2} for C_2 type of requests). Those four parameters can be used to derive the required service rates (μ_{tm} and η_{tk}) for each class traffic as follows:

$$\mu_{tm}^{'} = \left\lfloor \frac{\mu_{total}\Psi_1\lambda_1}{\Psi_1\lambda_1 + \Psi_2\lambda_2} \right\rfloor. \tag{1}$$

Indeed, $\mu_{tm}^{'}$ can be 0 because of floor function. Therefore, the scheduling algorithm reserves one server with at least one VM when $\mu_{tm}^{'} = 0$. Other than that, $\mu_{tm}^{'}$ is an ideal service rate for C_1 type of requests. However, it is hard to achieve equaling sum of service rates of heterogeneous servers to $\mu_{tm}^{'}$. Therefore, the faster servers (sum of service rates of these servers are approximately equal to μ_{tm}^{i}) are assigned to the higher priority class. Thus,

$$\mu_{tm} = \begin{cases} \sum_{i=1}^{m} \mu_i \approx \mu_{tm}^{'}, & \mu_{tm}^{'} > \mu_{C_1} \\ \mu_{C_1}, & \mu_{tm}^{'} \leq \mu_{C_1} \end{cases} \tag{2}$$

$$\eta_{tk}^{'} = \mu_{total} - \mu_{tm}. \tag{3}$$

$$\eta_{tk} = \begin{cases} \sum_{i=1}^{k} \mu_i \approx \eta_{tk}^{'}, & \eta_{tk}^{'} > \mu_{C_2} \\ \mu_{C_2}, & \eta_{tk}^{'} \leq \mu_{C_2} \end{cases} \tag{4}$$

$\Psi_2\lambda_2 \neq 0$ guarantees $\eta_{tk} \neq 0$. The above dynamic approach also works for homogeneous systems. Following example represents the server assignment process for such homogeneous systems. Assume there are five servers which are capable of serving all classes of requests and each server rate is 10. Therefore, $\mu_{total} = 50$. There are two classes of requests (C_1 and C_2) having priority levels, $\Psi_1 = 5$ and $\Psi_2 = 3$, and arrival rates λ_1 = 20 and λ_2 = 10. By substituting these values in Eqns. (1), (2) and (3), we get μ_{tm} = 30 and η_{tk} = 20 because each server rate is 10. That means that three servers are assigned to C_1 and two servers are assigned to C_2 if VMs are not used. However, if VMs are deployed in servers, then the number of VMs are arranged for each class. For example, if we assume, each virtual machine service rate is 1 then 38 VMs will be reserved for C_1 and 12 VMs will be used for C_2 (Here, we assume there is at least one server which is capable of serving both C_1 and C_2 requests).

Eqn. (1) can easily be extended to a multi-class system with r number of class types by assuming $\Psi_i\lambda_i \neq 0$ and $i = \{1, ..., r\}$ as follows:

$$\mu^{'}_{tm_1} = \left\lfloor \frac{\mu_{total}\Psi_1\lambda_1}{\Psi_1\lambda_1 + \Psi_2\lambda_2 + \ldots + \Psi_r\lambda_r} \right\rfloor . \tag{5}$$

Here, μ_{tm_1} (= $\sum_{i=1}^{m_1} \mu_i \approx \mu^{'}_{tm_1}$ or μ_{C_1}) is the total service rate assigned to C_1. After finding μ_{tm_1}, the remaining total service rate of servers are $\mu^{'}_{total} = \mu_{total} - \mu_{tm_1}$. Hence, the service rate assigned to C_2 can be obtained as follows:

$$\mu^{'}_{tm_2} = \left\lfloor \frac{\mu^{'}_{total}\Psi_2\lambda_2}{\Psi_2\lambda_2 + \Psi_3\lambda_3 + \ldots + \Psi_r\lambda_r} \right\rfloor . \tag{6}$$

Iteratively, following Eqns. (5) and (6), the dedicated number of servers for each class can be measured.

4.2.2 Scheduling Algorithm Procedure

The properties of the proposed scheduling algorithm are summarized: (i) classification of requests, (ii) assignment of different server groups to distinct request classes, (iii) creation of VMs in dedicated servers for requests if needed, (iv) regularly updating the number of assigned servers for each class based on the Eqns. (1), (2) and (3), (v) non-pre-empiteve model (Note: When the scheduling algorithm computes the new values of μ_{tm} and η_{tk}, some servers which were previously serving C_2 will be assigned to C_1 (see Figs. 5 and 4). The servers will continue to serve C_2 requests until they finish them. However, the scheduling algorithm will not assign any new C_2 requests to the servers which are recently assigned to C_1. This strategy protects the requests (in service) from experiencing large delay or drop.), (vi) creating a limited number of VMs in servers to avoid service degradation (creating at most K_{μ_i} and K_{η_i} number of VMs).

Two thread scheduling procedure is illustrated in Fig. 6 for two classes. The duties of the first thread are as follows: (i) The number of assigned servers to each class is updated regularly based on the Eqns. (1), (2) and (3) (Time to arrange servers means that it is time for periodic updating). (ii) The scheduling algorithm classifies an

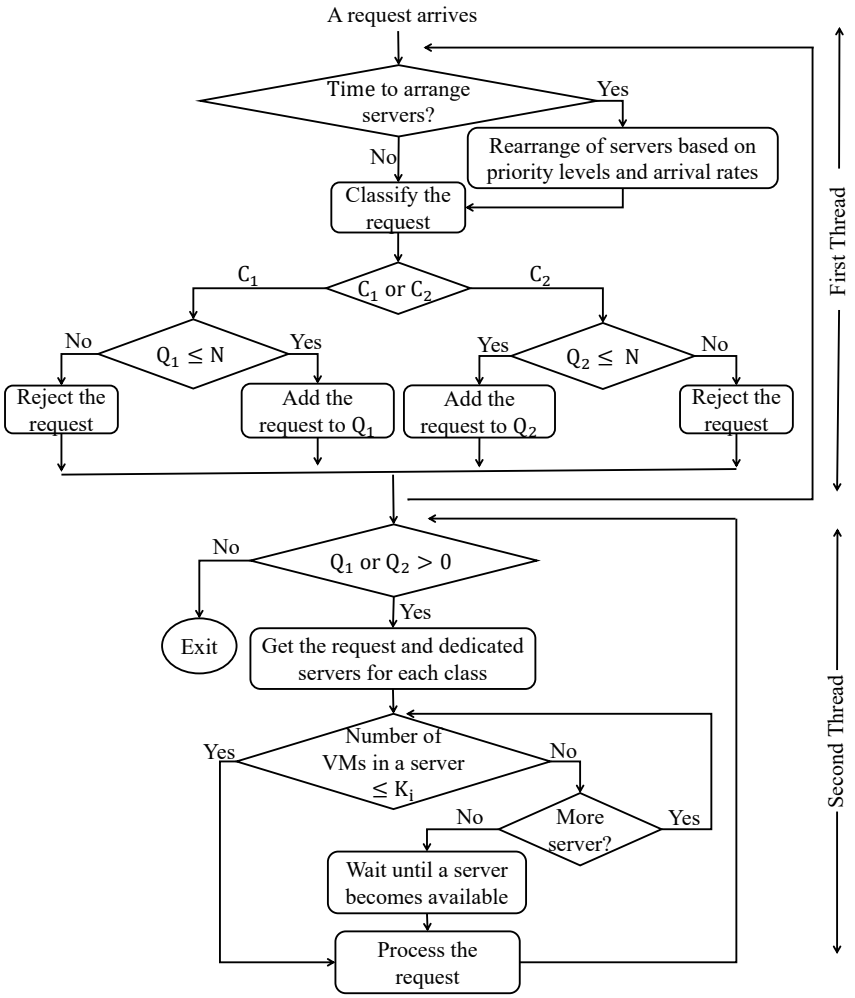


Fig. 6 Scheduling algorithm procedure.

arrived request (one of classification method such as the one in [13] can be used. Here, we assume that each request has a flag and the flag states that whether the request is C_1 or C_2) and enqueues the request to the corresponding queue of servers that are capable of serving to the request. (iii) Enqueuing process of requests continues while requests arrive.

The duty of second thread is that the scheduling algorithm dequeues a request from the queue then the dequeued request is forwarded to one of the available servers which have less than active K number of VMs according to allocation policies. If all active servers have already K number of active VMs, the scheduling algorithm wakes up a new server which has capability to serve the request (if there is an available sleeping server). It is important to note that the waking-up priority is given to servers

which are capable of serving both classes. The waking-up and sleeping process may increase the responding time of the request, but reduces the number of active servers to save energy. To decrease respond time, pre-waking up policy can be implemented. However, pre-waking up policy sacrifices the energy efficiency. The above dequeuing and allocation process continue until queues are empty. VM migration and resize VM are not considered in this paper for simplicity but it will be integrated into our developed method in the future.

5 Analytic Analysis

The various performance metric expressions of h-DDSS are analytically derived to be able to approximately estimate the performance of cloud systems. By using derived analytic expressions of the performance metrics, the performance of large sets of cloud servers for a huge data of IoT requests can roughly be measured.

5.1 Assumptions

To make the model analytically tractable, it is assumed that the queuing system is under heavy traffic flows, arrivals of requests follow Poisson distribution, and service times of requests are exponentially distributed. Type of queue discipline used in the analysis is FIFO. To make the model more realistic, we assume that the queue size is finite and equal to N, service rates of all servers and the number of created VMs in each server can be different (meaning the system is heterogeneous).

Because of heterogeneity, different allocation policies can be used. In our analysis, FSF and SSF allocation policies which are integrated into h-DDSS are used to analyze the best and worst performances of h-DDSS. The best performance (upper bound) of the heterogeneous system is obtained by using FSF allocation and the worst performance (lower bound) of the heterogeneous systems is obtained by using SSF allocation because an arrived request can be served faster at servers which have higher service rates and slower at servers which have lower service rates.

5.2 Derivation of Performance Metrics

The service rate of the system is a state dependent. When a request is in the system, the service rate is $\mu_{t1} = \mu_1$ and when two requests are in the system, the service rate is $\mu_{t2} = \mu_1 + \mu_2$. The service rate of the system increases until all servers are utilized (m servers for C_1 and k servers for C_2). Then the total service rate of the system is fixed at μ_{tm} and η_{tk} (using Eqns. (2) and (3)) for C_1 and C_2, respectively. Assuming $\mu_1 \leq \mu_2 \ldots \leq \mu_m$ gives the performance metrics of SSF allocation policy in h-DDSS. On the contrary, assuming $\mu_1 \geq \mu_2 \ldots \geq \mu_m$ gives the performance metrics of FSF allocation policy in h-DDSS. Thus, the lower and upper bounds of the performance metrics, average occupancy, average delay, drop rate, and throughput (n^{C_1}, δ^{C_1}, D^{C_1}, and γ^{C_1}) are computed for a heterogeneous multi-server system [37]. Only

the performance metrics of C_1 are derived by using the same approach with [38] because the performance metrics of C_2 can similarly be derived.

5.2.1 State probability

Fig. 7 shows the state transition diagram of the proposed model. p_i represents the probability of i number of C_1 requests in the system, N is size of the queue and λ_1 and μ_{ti} represent the probabilities of the state transitions where i = 1, 2, . . . , m.

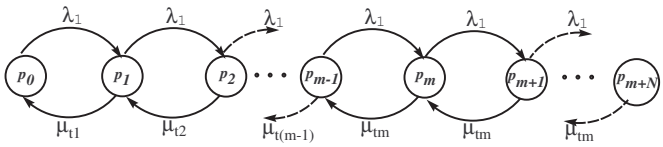


Fig. 7 State transition diagram of the proposed model.

State probability equations until m^{th} state can be written as follows by using the state transition diagram [39,37,38] in Fig. 7:

$$\begin{aligned}
\lambda p_0 = \mu_{t1} p_1 &\iff p_1 = p_0 \frac{\lambda}{\mu_{t1}}, \\
\lambda p_1 = \mu_{t2} p_2 &\iff p_2 = p_0 \frac{\lambda^2}{\mu_{t1}\mu_{t2}}, \\
\lambda p_2 = \mu_{t3} p_3 &\iff p_3 = p_0 \frac{\lambda^3}{\prod_{j=1}^{m} \mu_{tj}}, \\
&\vdots \\
\lambda p_{m-1} = \mu_{tn} p_m &\iff p_m = p_0 \frac{\lambda^m}{\prod_{j=1}^{m} \mu_{tj}}.
\end{aligned} \tag{7}$$

State probability equations after m^{th} state are different because the system has only m servers for C_1 requests. Thus, state probability equations can be written as follows:

$$\begin{aligned}
\lambda_1 p_m = \mu_{tm} p_{m+1} \iff p_{m+1} &= p_0 \frac{\lambda_1^{m+1}}{\mu_{tm} \prod_{j=1}^{m} \mu_{tj}} = p_0 \frac{\mu_{tm}^{m} \lambda_1^{m+1}}{\mu_{tm}^{m+1} \prod_{j=1}^{m} \mu_{tj}}, \\
\lambda_1 p_{m+1} = \mu_{tm} p_{m+2} \iff p_{m+2} &= p_0 \frac{\lambda_1^{m+2}}{\mu_{tm}^{2} \prod_{j=1}^{m} \mu_{tj}} = p_0 \frac{\mu_{tm}^{m} \lambda_1^{m+2}}{\mu_{tm}^{m+2} \prod_{j=1}^{m} \mu_{tj}}, \\
&\vdots \\
\lambda_1 p_{i-1} = \mu_{tm} p_i \iff p_i &= p_0 \frac{\lambda_1^{i}}{\mu_{tm}^{i-m} \prod_{j=1}^{m} \mu_{tj}} = p_0 \frac{\mu_{tm}^{m} \lambda_1^{i}}{\mu_{tm}^{i} \prod_{j=1}^{m} \mu_{tj}} \\
&\vdots \\
\lambda_1 p_{m+N-1} = \mu_{tm} p_{m+N} \iff p_{m+N} &= p_0 \frac{\lambda_1^{m+N}}{\mu_{tm}^{N} \prod_{j=1}^{m} \mu_{tj}} = p_0 \frac{\mu_{tm}^{m} \lambda_1^{m+N}}{\mu_{tm}^{N+m} \prod_{j=1}^{m} \mu_{tj}}.
\end{aligned} \tag{8}$$

where $m < i \leq m + N$. Shortly,

$$p_i = \begin{cases} p_0 \frac{\lambda_1^{i}}{\prod_{j=1}^{i} \mu_{tj}}, & i \leq m \\ p_0 \frac{\mu_{tm}^{m} \rho^{i}}{\prod_{j=1}^{m} \mu_{tj}}, & m < i \leq m + N \end{cases} \tag{9}$$

where $\rho = \lambda_1 / \mu_{tm}$. It is known that (because sum of all possibilities equals to 1.)

$$\sum_{i=0}^{m+N} p_i = 1. \tag{10}$$

To find state probabilities, we need to measure p_0 by using Eqn. (10).

$$
\begin{aligned}
1 &= \sum_{i=0}^{m+N} p_i \\
1 &= p_0 + \sum_{i=1}^{m} p_i + \sum_{i=m+1}^{m+N} p_i \\
1 &= p_0 + \sum_{i=1}^{m} p_0 \frac{\lambda_1^i}{\prod_{j=1}^{i} \mu_{tj}} + \sum_{i=m+1}^{m+N} p_0 \frac{\mu_{tm}^m \rho^i}{\prod_{j=1}^{m} \mu_{tj}} \\
1 &= p_0 \left[1 + \sum_{i=1}^{m} \frac{\lambda_1^i}{\prod_{j=1}^{i} \mu_{tj}} + \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} \sum_{i=m+1}^{m+N} \rho^i \right] \\
1 &= p_0 \left[1 + \sum_{i=1}^{m} \frac{\lambda_1^i}{\prod_{j=1}^{i} \mu_{tj}} + \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} (N + m - m)(if \rho = 1) \right]
\end{aligned}
\tag{11}
$$

Therefore, from Eqn. (11), p_0 can be written as follows:

$$
p_0^{-1} = \begin{cases} 1 + \sum_{i=1}^{m} \frac{\lambda_1^i}{\prod_{j=1}^{i} \mu_{tj}} + \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} \sum_{i=m+1}^{m+N} \rho^i, & \rho \neq 1 \\ 1 + \sum_{i=1}^{m} \frac{\lambda_1^i}{\prod_{j=1}^{i} \mu_{tj}} + N \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}}, & \rho = 1 \end{cases}
\tag{12}
$$

where $\rho = \lambda_1 / \mu_{tm}$.

5.2.2 Drop probability and throughput

The drop probability of a request from the system is the final state probability, p_{m+N}. Therefore, the drop rate and throughput can be obtained as follows:

$$
D^{C_1} = p_{m+N} = p_0 \frac{\mu_{tm}^m \rho^{m+N}}{\prod_{j=1}^{m} \mu_{tj}}.
\tag{13}
$$

Therefore, throughput is

$$
\gamma^{C_1} = \lambda_1 (1 - D^{C_1}).
\tag{14}
$$

5.2.3 Average class occupancy and delay

Average class occupancy and delay can be formulated by using state probabilities [37]. Average occupancy expression, $n_{M/M/1/N}$ for $M/M/1/N$ queue is

$$n_{M/M/1/N} = \sum_{i=1}^{N} ip_i. \tag{15}$$

However, $M/M_i/m/N$ queue system has m servers which means first m requests get service immediately. Therefore, from the above state probabilities (Eqn. (9)), n^{C_1} will be (state probability must be p_i where $m < i \leq m + N$ to have request in the queue. If the probability is p_{m+1}, there is only one request in the queue because there are m servers. Therefore, we have $(i - m)p_i$ in Eqn 16).

$$n^{C_1} = \sum_{i=m+1}^{m+N} (i - m)p_i. \tag{16}$$

Therefore, n^{C_1} can be rewritten as follows by using Eqns. (9) and (16):

$$\begin{aligned} n^{C_1} &= \sum_{i=m+1}^{m+N} (i - m)p_0 \frac{\mu_{tm}^m \rho^i}{\prod_{j=1}^{m} \mu_{tj}} \\ &= p_0 \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} \sum_{i=m+1}^{m+N} (i - m)\rho^i. \end{aligned} \tag{17}$$

After simplification of Eqn. (17) by using geometric series, finally we will have the following expressions for n^{C_1};

$$n^{C_1} = \begin{cases} p_0 \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} \rho^{m+1} \left(\frac{1-(N+1)\rho^N + N\rho^{N+1}}{(1-\rho)^2} \right), & \rho \neq 1 \\ p_0 \frac{\mu_{tm}^m}{\prod_{j=1}^{m} \mu_{tj}} \left(\frac{N(N+1)}{2} \right), & \rho = 1 \end{cases} \tag{18}$$

Using Little's law, and Eqns. (14) and (18), average delay can be obtained as follows:

$$\delta^{C_1} = \frac{n^{C_1}}{\gamma^{C_1}}. \tag{19}$$

5.3 Performance Metrics of the System

In subsection 5.2, the performance metrics of C_1 are derived. Similarly, the performance metrics of C_2 can similarly be obtained for the system. In this subsection, the system performance metrics are derived by using the performance metrics of C_1 and C_2.

5.3.1 Average occupancy

We have taken sum of the average C_1 and C_2 occupancies in the system to find the overall average occupancy in the system. Overall average occupancy, n, can be computed as follows:

$$n = n^{C_1} + n^{C_2}. \tag{20}$$

5.3.2 Drop probability

After finding the drop rate of each class of requests in the system, we have computed the drop rate of a request, D, without considering type of request, as follows:

$$D = \frac{\lambda_1 D^{C_1} + \lambda_2 D^{C_2}}{\lambda_1 + \lambda_2}. \tag{21}$$

5.3.3 Throughput

By using throughput of C_1 and the number of C_2 requests in the system, the total throughput of the system, γ, can be obtained as follows:

$$\gamma = \gamma^{C_1} + \gamma^{C_2}. \tag{22}$$

5.3.4 Average delay

The average delay which is experienced by each request in the system can be obtained as follows;

$$\delta = \frac{\gamma^{C_1}\delta^{C_1} + \gamma^{C_2}\delta^{C_2}}{\gamma}. \tag{23}$$

6 Simulation of the System

Discrete event simulation has been carried out under the aforementioned scheduling policies in Section 4 by following $M/M_i/c/N$ [37] procedures. However, the simulation is implemented by using parameters in Table 2 with a dynamic approach. To make the process simple, we assume each VM service rate is 1.

Table 2 Parameters of cloud simulation

The number of servers	8
Service rates of servers	μ = 1, 2, ... 8
Initial Server Setups	μ = 1, 2, 3 for C_2 and μ = 4,... 8 for C_1
Arrival rates of C_1 and C_2	$\lambda_1 = \{i\}$, $\lambda_2 = \{2i\}$, where i = 1,2, ...,10
Used priority levels	$\Psi_1 = \{1.5, 5\}$, $\Psi_2 = \{1\}$
Queue Lengths of Q_1 and Q_2	30

The values of the selected parameters are given in Table 2. The values of the parameters are selected to observe behaviors of the system when the system is under light and heavy traffic flows of IoT requests. Arrival rates of C_1 and C_2 depend on i to keep track the relation of two arrival rates. We assume that C_1 represents the higher priority request; the priority of this class is higher because of premium membership of users or application types. In order to observe priority level effects on the system performance, two priority levels of C_1 are used. On the other hand, C_2 represents the second priory type requests because of standard users or low priority applications. Therefore, the arrival rate of C_1 is lower than C_2 requests. This approach well represents the real life scenario as it is seen in similar type behaviors in commercial cloud systems [40]. The values of the service rates in Table 2 represent the only service time of a request in the servers. The values of service rates are selected to presents heterogeneous server environment [38]. Creation and preparation times of VMs are ignored because IoT type requests are generally software type requests and do not experience delays due to creation or preparation of VMs. Buffer lengths are kept small [41], similar to real system to reduce respond time.

7 Results

In this section, the derived formulas in Section 5 are verified with an extensive simulation and the performances of the proposed DDSS and h-DDSS methods are evaluated by comparing with existing DSS method.

7.1 Validation of Analytical Expressions

In this subsection, the analytical and simulation results of h-DDSS are compared to evaluate the accuracy of the approximation of the performance metrics which are derived in Section 5 for FSF allocation.

7.1.1 Average occupancy and delay

Figs. 8 and 9 show the average C_1 and C_2 occupancies and delays of h-DDSS obtained through the simulation and analytical model. The simulation and analytical occupancy results are closely matched. Although the delay results of the simulation are higher than the analytical ones, they have followed a similar pattern. The reason is that the scheduling algorithm dynamically arranges the number of assigned servers for C_1 and C_2 and does not allow newly assigned servers to drop the requests in service. Such dynamic behavior cannot analytically be formulated. Therefore, the delay results of the simulation are generally a little higher than the analytical delay results. The occupancy and delay results of C_1 and C_2 are almost zero until $i \leq 6$ ($\lambda_1 \leq 6$ and $\lambda_2 \leq 12$). However, after $i > 6$, the delay and occupancy are sharply increasing until reaching to maximum occupancy value ($= 30$ because queue size is 30).

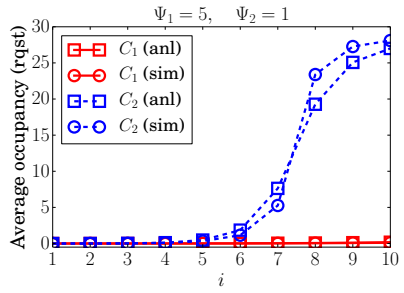


Fig. 8 Average C_1 and C_2 occupancies of h-DDSS (FSF allocation) obtained through the simulation and analytical model.

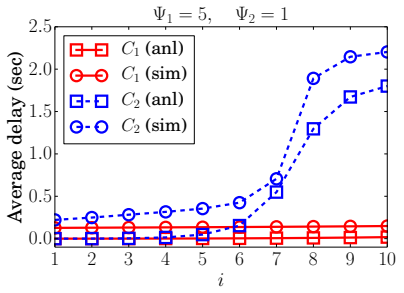


Fig. 9 Average C_1 and C_2 delays of h-DDSS (FSF allocation) obtained through the simulation and analytical model.

7.1.2 Drop rate and throughput

Figs. 10 and 11 show C_1 and C_2 drop rates and throughput of h-DDSS obtained through the simulation and analytical model. The analytical results of the throughput and drop rates also matched with the simulation ones. C_1 throughput is lower than

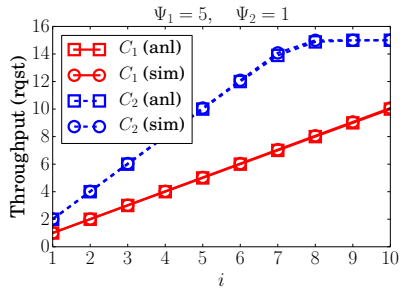


Fig. 10 C_1 and C_2 throughput of h-DDSS (FSF allocation) obtained through the simulation and analytical model.

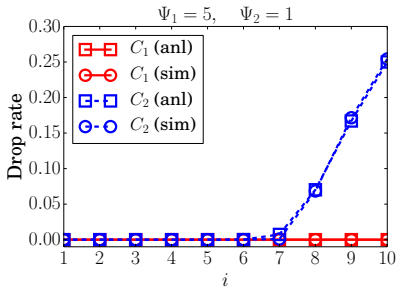


Fig. 11 C_1 and C_2 drop rates of h-DDSS (FSF allocation) obtained through the simulation and analytical model.

C_2 throughput because the arrival rate of C_1 (λ_1) is lower than the arrival rate of C_2 (λ_2). However, after $i = 7$ ($7 \leq \lambda_1 \leq 10$ and $14 \leq \lambda_2 \leq 20$), because of the priority levels of C_1 ($\Psi_1 = 5$) and C_2 ($\Psi_2 = 1$), C_2 throughput is stable, and C_1 throughput is continuously increasing. This means that the assigned servers to C_1 and C_2 are enough to serve the incoming C_1 traffic, but the assigned servers are not enough for C_2 traffic after $i = 7$.

C_1 and C_2 drop rates are close to zero while $i \leq 7$ ($\lambda_1 \leq 7$ and $\lambda_2 \leq 14$). On the other hand, C_2 drop rate is higher than C_1 drop rate after $i \geq 7$ ($7 \leq \lambda_1 \leq 10$ and

$14 \leq \lambda_2 \leq 20$) because the number of the assigned servers for C_2 is not enough to serve C_2 requests.

As shown in Figs. 8, 9, 10, and 11, the obtained analytical and simulation results closely match and follow a similar pattern, thereby validating our analytical approximations. Therefore, the obtained analytical expressions can be used to estimate the performance of a large set of servers in heterogeneous cloud systems and heavy traffic flows of IoT requests approximately.

7.2 Performances of Classes in DSS, DDSS and h-DDSS

In this subsection, the performance metrics, occupancy, throughput, and utilization, of DSS, DDSS, and upper (FSF) and lower (SSF) bounds of h-DDSS are presented. Note that we sometimes use h-DDSS or upper and lower bounds terms instead of FSF and SSF in the rest of the paper. Average delay is not given because average delay and occupancy follow similar patterns for all methods. Moreover, confidential intervals because of realization results from different simulation runs are not given. The reason is that the packet arrivals follow Poisson Distribution; thus, the differences between the obtained results from the different realizations are insignificant (it means the variation is insignificant). To avoid complex presentation of the figures, the confidential intervals are not presented in the figures.

DSS and DDSS results are obtained by averaging service rates of heterogeneous servers while keeping every other parameters similar because of homogeneous server assumption of DSS and DDSS. For example, if the total service rate of eight heterogeneous servers is equal to 40, the average service rate of eight homogeneous servers is five. Therefore, each server rate of homogeneous servers is five. Moreover, we also present the effects of priority levels of classes on the performance of the cloud systems by keeping the priority level of C_2 fixed at 1 ($\Psi_2 = 1$) while changing the priority level of C_1 as 5 and 1.5 (Ψ_1 as 5 and 1.5). Therefore, the impacts of the priority levels of the classes on the performance of DSS, DDSS and h-DDSS are obtained.

7.2.1 Average class occupancy

Figs. 12 and 13 show the average C_1 and C_2 occupancies of DSS, DDSS and h-DDSS (upper and lower bounds) when $\Psi_1 = 5$ and $\Psi_2 = 1$. From Figs. 12 and 13, the impacts of the arrival rates on C_1 and C_2 occupancies are observed as follows: C_1 and C_2 occupancies are almost zero until $i = 5$ ($\lambda_1 \leq 5$ and $\lambda_2 \leq 10$) because of light traffic flows. However, continuously rising arrival rates of C_1 and C_2 ($5 \leq \lambda_1 \leq 10$ and $14 \leq \lambda_2 \leq 20$) leads sharp increases in C_2 occupancies and slight changes in C_1 occupancies because the priority level of C_2 ($\Psi_2 = 1$) is lower than the priority level of C_1 ($\Psi_1 = 5$).

From Figs. 12 and 13, the effects of the arrival rates on the occupancies of DSS, DDSS and h-DDSS are also observed as follows: The occupancies of DSS, DDSS, FSF and SSF are equal and close to zero when $i \leq 5$ for both C_1 and C_2 because all scheduling algorithms well handle the light traffic flows. Therefore, the differences

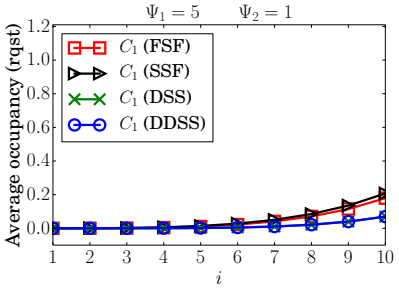


Fig. 12 Average C_1 occupancies of DSS, DDSS and h-DDSS ($\Psi_1 = 5$).

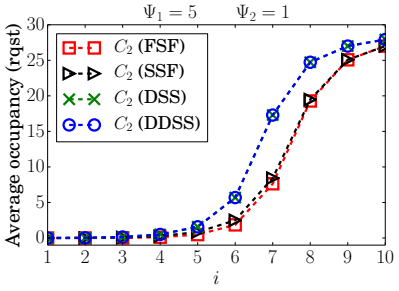


Fig. 13 Average C_2 occupancies of DSS, DDSS and h-DDSS ($\Psi_1 = 5$).

between scheduling algorithms are not significant. However, continuously rising arrival rates of C_1 and C_2 shows that the occupancies of DSS and DDSS (homogeneity) are equal, and the occupancies of upper (FSF) and lower (SSF) of h-DDSS (heterogeneity) are very close for both C_1 and C_2 (when $\Psi_1 = 5$ and $\Psi_2 = 1$). It is worth mentioning that after $i = 5$, C_1 occupancies of h-DDSS are up to 2 to 3 times higher than C_1 occupancies of DSS and DDSS, and it is opposite for C_2.

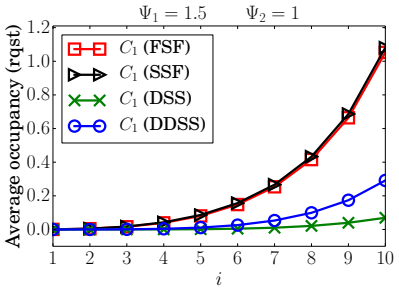


Fig. 14 Average C_1 occupancies of DSS, DDSS and h-DDSS ($\Psi_1 = 1.5$).

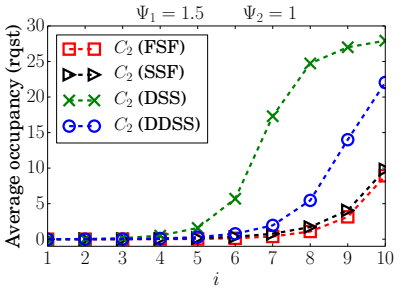


Fig. 15 Average C_2 occupancies of DSS, DDSS and h-DDSS ($\Psi_1 = 1.5$).

In Figs. 14 and 15, we change the priority level of C_1 (Ψ_1) from 5 to 1.5 while keeping other parameters same to observe the effects of priority levels on the occupancies. The occupancy of DSS is not changed by the priority levels because DSS does not take into account the priority levels while assigning servers. However, DDSS and h-DDSS dynamically arrange servers by using priority levels. Therefore, as observed from the occupancy figures, when the priority level of C_1 is decreased, C_1 occupancies of DDSS and h-DDSS are increasing but C_2 occupancies of DDSS and h-DDSS are reduced. It is important to note that the changes in the priority levels affect h-DDSS occupancy more than DDSS occupancy.

7.2.2 Throughput

Figs. 16 and 17 show C_1 and C_2 throughput of DSS, DDSS and h-DDSS when $\Psi_1 = 5$ and $\Psi_2 = 1$. C_1 throughput of all cases is same because arrivals of C_1 traffic ($\lambda_1 = i = 1, \dots 10$) are completely served. Moreover, C_2 throughput is also same

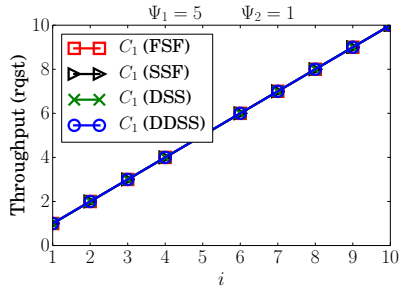


Fig. 16 C_1 throughput of DSS, DDSS and h-DDSS ($\Psi_1 = 5$).

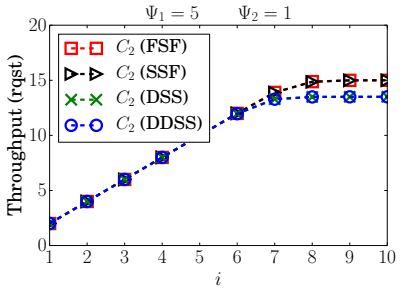


Fig. 17 C_2 throughput of DSS, DDSS and h-DDSS ($\Psi_1 = 5$).

until $i = 7$ ($\lambda_2 \leq 14$) for all cases since the assigned number of servers is enough to serve the C_2 traffic up to $i = 7$. However, after $i = 7$ ($14 \leq \lambda_2 \leq 20$), C_2 throughput reaches maximum possible throughput level and cannot go higher because of the service rate limitations of the assigned servers. As observed from Fig. 17, when the system is under heavy traffic flows, the upper and lower bounds of C_2 throughput of h-DDSS are slightly higher than C_2 throughput of DSS and DDSS because h-DDSS assigns more servers for C_2 traffic.

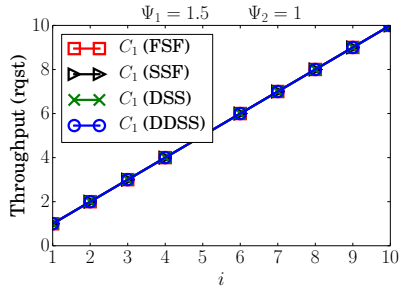


Fig. 18 C_1 throughput of DSS, DDSS and h-DDSS ($\Psi_1 = 1.5$).

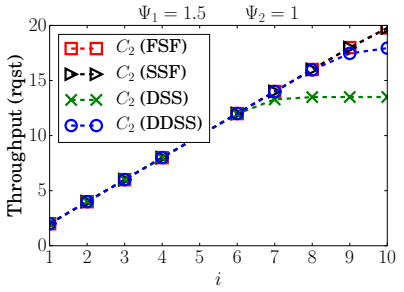


Fig. 19 C_2 throughput of DSS, DDSS and h-DDSS ($\Psi_1 = 1.5$).

In Figs. 18 and 19, we change the priority level of C_1 (Ψ_1) from 5 to 1.5 while keeping other parameters same to observe the priority levels effects on the through-

put. The throughput of DSS is not affected because of non-dynamic server assignment policy of DSS. However, when the priority level of C_1 is decreased, C_2 throughput of DDSS and h-DDSS are increasing after $i = 7$ because of dynamic server assignment policies of DDSS and h-DDSS. In addition, although C_2 throughput of h-DDSS is still higher than C_2 throughput of DDSS, C_2 throughput of DDSS is improved more than C_2 throughput of h-DDSS. It is significantly important to note that DDSS and h-DDSS notably increase C_2 throughput without decreasing C_1 throughput after changing the priority level of C_1. This shows that the performance in DDSS and h-DDSS can be optimized by selecting priority levels of classes.

7.2.3 Utilization

Utilization is a performance metric which reflects the efficiency of server usage. Here, utilization is computed as the ratio of the arrival rates of incoming class traffic to the total service rates (of all the dedicated servers) for the same class. We are interested in the impacts of the priority levels on utilizations of DSS, DDSS, and h-DDSS.

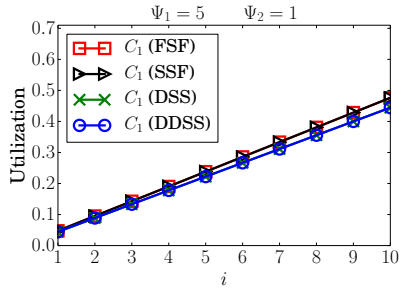


Fig. 20 C_1 utilization of DSS, DDSS and h-DDSS ($\Psi_1 = 5$).

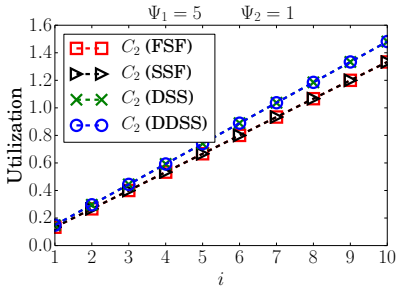


Fig. 21 C_2 utilization of DDSS and h-DDSS ($\Psi_1 = 5$).

Figs. 20 and 21 show C_1 and C_2 utilizations of DSS, DDSS and h-DDSS when $\Psi_1 = 5$ and $\Psi_2 = 1$. C_1 utilizations of DSS and DDSS are equal and slightly lower than C_1 utilization of h-DDSS (FSF and SSF). However, C_2 utilizations of DSS and DDSS are larger than C_2 utilization of h-DDSS. In addition, the gap between the utilizations of DSS and DDSS, and the utilization of h-DDSS are getting higher for both C_1 and C_2 utilizations while the arrival rates of C_1 and C_2 are increased.

In Figs. 22 and 23, we change the priority level of C_1 (Ψ_1) from 5 to 1.5 while keeping other parameters same. As a result of the priority level alteration, there are two important changes. First, utilizations of DSS and DDSS are not equal anymore, and C_1 utilization of DDSS is higher than C_1 utilization of DSS, and C_2 utilization of DDSS is lower than C_2 utilization of DSS. Second, the gap between utilization of DSS, DDSS and h-DDSS is becoming much larger. It is worth mentioning that reducing C_1 priority level increases C_1 utilization and decreases C_2 utilizations of DDSS and h-DDSS.

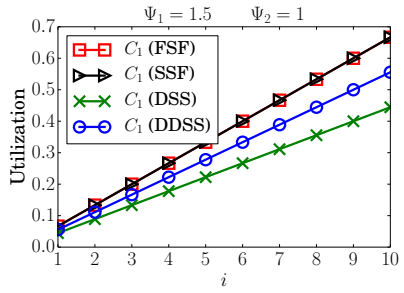


Fig. 22 C_1 utilization for DSS, DDSS and h-DDSS ($\Psi_1 = 1.5$).

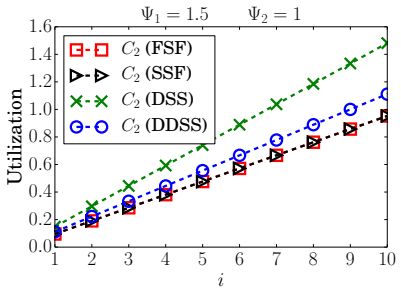


Fig. 23 C_2 utilization for DDSS and h-DDSS ($\Psi_1 = 1.5$).

7.3 Summary of Results

Based on the results, we make the following observations: (i) Priority levels do not significantly affect the performances of homogeneous and heterogeneous systems when the system is under light traffic flows. (ii) The throughput of the homogeneous server systems is affected more than the throughput of the heterogeneous server systems by priority levels when the system is under heavy traffic flows. (iii) Although the upper (FSF) and lower (SSF) bound performances of h-DDSS are same when the system is under the pressure of IoT requests, FSF is generally better than SSF. (iv) Heterogeneous systems can become more efficient than homogeneous systems by assigning proper priority levels to types of Internet of Things requests. (v) Importantly, DDSS and h-DDSS increase the performance of the system up to 40% compared to DSS.

8 Conclusion and Future Direction

In this paper, we have proposed a dynamic scheduling algorithm for cloud computing by considering homogeneous and heterogeneous servers with priorities of Internet of Things requests. Analytical expressions of the upper and lower bound performance metrics for heterogeneous Dynamic Dedicated Server Scheduling are derived and verified by an extensive simulation. The performances of homogeneous and heterogeneous cloud computing systems have been presented in terms of throughput, occupancy, and utilization. Results show that the proposed scheduling algorithm increases the performances of the homogeneous and heterogeneous systems up to 40%. The results obtained in this paper will assist cloud service providers build efficient cloud computing server scheduling algorithms which are adaptable to homogeneous and heterogeneous server systems for different types of Internet of Things requests.

As a future work, the effects of heterogeneity level of servers and Internet of Things requests on cloud computing performance will be investigated by using Gini Index.

Acknowledgements This research was supported in part by U.S. NSF grants NSF-1404981.

References

1. C. Wang, Z. Bi, and L. D. Xu, "IoT and cloud computing in automation of assembly modeling systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1426–1434, May 2014.
2. T. Shon, J. Cho, K. Han, and H. Choi, "Toward advanced mobile cloud computing for the internet of things: Current issues and future direction," *Mobile Networks and Applications*, vol. 19, no. 3, pp. 404–413, June 2014.
3. L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, Oct 2010.
4. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645 – 1660, Sep 2013.
5. J. Tan and S. Koo, "A survey of technologies in internet of things," in *IEEE International Conference on Distributed Computing in Sensor Systems*, Marina Del Rey, CA, May 26-28, 2014, pp. 269–274.
6. F. Richter. (2013, Sep) Smartphone sales break the billion barrier. Accessed: June 12, 2014. [Online]. Available: http://www.statista.com/chart/777/global-connected-device-shipments/
7. W. Kim, "Cloud computing: Today and tomorrow," *Journal of Object Technology*, vol. 8, pp. 65–72, Jan 2009.
8. L. Wang, G. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, Apr 2010.
9. T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Principles and Practice of Parallel Programming*, Chicago, IL, June 15-17, 2005, pp. 186–195.
10. A. V. Dastjerdi, , and R. Buyya, "Compatibility-aware cloud service composition under fuzzy preferences of users," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 1–34, Mar 2014.
11. J. Mars and L. Tang, "Whare-map: Heterogeneity in "homogeneous" warehouse-scale computers," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 619–630, June 2013.
12. J. Mars and L. Tang, "Whare-map: Heterogeneity in "Homogeneous" Warehouse-scale Computers," in *40th Annual International Symposium on Computer Architecture*, Tel-Aviv, Israel, June 23-27, 2013, pp. 619–630.
13. C. Delimitrou and C. Kozyrakis, "Qos-aware scheduling in heterogeneous datacenters with paragon," *ACM Trans. Comput. Syst.*, vol. 31, no. 4, Dec 2013.
14. Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 1–34, Mar 2014.
15. W. Ellens, M. Zivkovic, J. Akkerboom, R. Litjens, and H. van den Berg, "Performance of cloud computing centers with multiple priority classes," in *IEEE 5th International Conference on Cloud Computing (CLOUD)*, Honolulu, HI, June 24-29, 2012, pp. 245–252.
16. Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, "Resource provisioning for cloud computing," in *Conference of the Center for Advanced Studies on Collaborative Research (CASCON '09)*, Toronto, Canada, Nov 2-5, 2009, pp. 101–111.
17. H. S. Narman, M. S. Hossain, and M. Atiquzzaman, "DDSS:Dynamic dedicated servers scheduling for multi priority level classes in cloud servers," in *IEEE International Conference on Communications (ICC)*, Sydney, Australia, June 10-14, 2014, pp. 3082 – 3087.
18. H. S. Narman, M. S. Hossain, and M. Atiquzzaman, "h-DDSS:Heterogeneous dynamic dedicated servers scheduling in cloud computing," in *IEEE International Conference on Communications (ICC)*, Sydney, Australia, June 10-14, 2014, pp. 3475 – 3480.
19. V. Goswami, S. S. Patra, and G. B. Mund, "Performance analysis of cloud with queue-dependent virtual machines," in *1st International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, Mar 15-17, 2012, pp. 357–362.
20. H. peng Chen and S. chong Li, "A queueing-based model for performance management on cloud," in *6th International Conference on Advanced Information Management and Service (IMS)*, Seoul, Nov 30-Dec 2, 2010, pp. 83–88.

21. A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 931–945, June 2011.
22. H. Khazaei, J. Misic, and V. Misic, "Performance analysis of cloud computing centers using M/G/m/m+r queuing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 936–943, May 2012.
23. K. Xiong and H. G. Perros, "Service performance and analysis in cloud computing," in *IEEE Congress on Services*, Los Angeles, CA, July 6-10, 2009, pp. 693–700.
24. S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," *Telecommunications Policy*, vol. 34, pp. 115–131, Oct 2010.
25. T. A. Henzinger, A. V. Singh, V. Singh, T. Wies, and D. Zufferey, "Static scheduling in clouds," in *3rd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'11, Portland, OR, June 14-17, 2011.
26. T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Trans. Softw. Eng.*, vol. 14, no. 2, pp. 141–154, Feb 1988.
27. F. Xhafa and A. Abraham, "Computational models and heuristic methods for grid scheduling problems," *Future Gener. Comput. Syst.*, vol. 26, no. 4, pp. 608–621, Apr 2010.
28. Y. Lee, S. Leu, and R. Chang, "Improving job scheduling algorithms in a grid environment," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 991–998, Oct 2011.
29. S. Bansal, B. Kothari, and C. Hota, "Dynamic task-scheduling in grid computing using prioritized round robin algorithm," *International Journal of Computer Science Issues*, vol. 8, no. 2, pp. 472–477, Mar 2011.
30. S. Ghanbari and M. Othman, "A priority based job scheduling algorithm in cloud computing," *Procedia Engineering*, vol. 50, pp. 778–785, 2012.
31. L. Yang, C. Pan, E. Zhang, and H. Liu, "A new class of priority-based weighted fair scheduling algorithm," *Physics Procedia*, vol. 33, pp. 942–948, 2012.
32. S. N. M. Shah, M. N. B. Zakaria, A. K. B. Mahmood, A. J. Pal, and N. Haron, "Agent based priority heuristic for job scheduling on computational grids," *Procedia Computer Science*, vol. 9, pp. 479–488, 2012.
33. H. A. Abba, N. Zakaria, S. N. M. Shah, and A. Pal, "Design, development and performance analysis of deadline based priority heuristic for job scheduling on a grid," *Procedia Computer Science*, vol. 9, 2012.
34. H. Chtourou and M. Haouari, "A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling," *Computers & Industrial Engineering*, vol. 55, no. 1, pp. 183–194, Aug 2008.
35. B. Yang, F. Tan, Y.-S. Dai, and S. Guo, "Performance evaluation of cloud service considering fault recovery," in *Cloud Computing*, Beijing, China, Dec 1-4, 2009, pp. 571–576.
36. S. Patel and U. Bhoi, "Priority based job scheduling techniques in cloud computing a systematic review," *International Journal of Scientific & Technology Research*, vol. 2, no. 11, pp. 147–152, Oct 2013.
37. D. Gross and C. M. Harris, *Fundamentals of Queueing Theory (Wiley Series in Probability and Statistics)*. Wiley-Interscience, Feb 1998.
38. F. S. Q. Alves, H. C. Yehia, L. A. C. Pedrosa, F. R. B. Cruz, and L. Kerbache, "Upper bounds on performance measures of heterogeneous M/M/c queues," *Mathematical Problems in Engineering*, vol. 2011, May 2011.
39. H. S. Narman, M. S. Hossain, and M. Atiquzzaman, "Multi class traffic analysis of single and multi-band queuing system," in *IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, Dec 9-13, 2013, pp. 1422 – 1427.
40. J. Novet. (2015, Nov) Dropbox hits 150k paying business customers, with 50k joining this year. Accessed: May 20, 2016. [Online]. Available: http://venturebeat.com/2015/11/04/dropbox-hits-150k-paying-business-customers-with-50k-joining-this-year/
41. G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *Computer Communication Review*, vol. 34, pp. 281–292, Oct 2004.

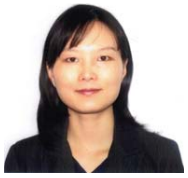
Husnu S. Narman received his B.S. in Mathematics from Abant Izzet Baysal University, Turkey, M.S. in Computer Science from University of Texas at San Antonio, and PhD in Computer Science from University of Oklahoma in the year 2006, 2011 and 2016, respectively. Currently he is working as a Postdoc in Holcombe Department of Electrical and Computer Engineering, Clemson University. His research interests are in queuing theory, network management, network topology, Internet of Things, LTE and cloud computing.

Md. Shohrab Hossain received his B.Sc. and M.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in the year 2003 and 2007, respectively. He obtained his Ph.D. degree from the School of Computer Science at the University of Oklahoma, Norman, OK, USA in December, 2012. He is currently serving as an Assistant Professor in the Department of Computer Science and Engineering at Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. His research interests include mobility of IPv6 networks, mobility models, security, scalability and survivability of mobile networks, Cloud Computing. He has several conference and journal papers published by IEEE and Springer.

Mohammed Atiquzzaman obtained his M.S. and Ph.D. in Electrical Engineering and Electronics from the University of Manchester (UK). He is currently holds the Edith Kinney Gaylord Presidential professorship in the School of Computer Science at the University of Oklahoma, and is a senior member of IEEE.

Dr. Atiquzzaman is the Editor-in-Chief of Journal of Networks and Computer Applications, founding Editor-in-Chief of Vehicular Communications and has served/serving on the editorial boards of IEEE Communications Magazine, International Journal on Wireless and Optical Communications, Real Time Imaging journal, Journal of Communication Systems, Communication Networks and Distributed Systems and Journal of Sensor Networks. He also guest edited 12 special issues in various journals. He has served as co-chair of IEEE High Performance Switching and Routing Symposium (2011 and 2003) and has served as symposium co-chairs for IEEE Globecom (2006, 2007, 2014) and IEEE ICC (2007, 2009, 2011, 2012) conferences. He co-chaired ChinaComm (2008), and SPIE Next-Generation Communication and Sensor Networks (2006) and the SPIE Quality of Service over Next Generation Data Networks conferences (2001, 2002, 2003, 2005). He was the panels co-chair of INFOCOM05, and is/has been in the program committee of numerous conferences such as INFOCOM, ICCCN, and Local Computer Networks.

He serves on the review panels of funding agencies such as the National Science Foundation and National Research Council (Canada) and Australian Research Council (Australia). In recognition of his contribution to NASA research, he received the NASA Group Achievement Award for outstanding work to further NASA Glenn Research Centers effort in the area of Advanced Communications/Air Trafc Managements Fiber Optic Signal Distribution for Aeronautical Communications project. He is the co-author of the book Performance of TCP/IP over ATM networks and has over 270 refereed publications which are accessible at www.cs.ou.edu/~atiq. His research interests are in communications switching, transport protocols, wireless and mobile networks, ad hoc networks, satellite networks, Quality of Service, and optical communications. His research has been funded by National Science Foundation (NSF), National Aeronautics and Space Administration (NASA), U.S. Air Force, Cisco, Honeywell, Oklahoma Department of Transportation, Oklahoma Highway Safety Office through grants totaling over $7M. His publications can be accessed at www.cs.ou.edu/~atiq.



Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the Program Co-Chair for a number of international conferences and member of the Program Committees of many leading conferences. She is a Microsoft Faculty Fellow of 2010 and a member of the IEEE and ACM. Her publications can be accessed at http://shenh.people.clemson.edu.