EXECUTION TIME AND POWER CONSUMPTION OPTIMIZATION in FOG COMPUTING ENVIRONMENT

Anwar Alghamdi [1†], Ahmed Alzahrani [2††], Vijey Thayananthan [3††],

Department of Computer Science, King Abdul-Aziz University, Jeddah, Saudi Arabia

Summary
The Internet of Things (IoT) paradigm is at the forefront of present and future research activities. The huge amount of sensing data from IoT devices needing to be processed is increasing dramatically in volume, variety, and velocity. In response, cloud computing was involved in handling the challenges of collecting, storing, and processing jobs. The fog computing technology is a model that is used to support cloud computing by implementing pre-processing jobs close to the end-user for realizing low latency, less power consumption in the cloud side, and high scalability. However, it may be that some resources in fog computing networks are not suitable for some kind of jobs, or the number of requests increases outside capacity. So, it is more efficient to decrease sending jobs to the cloud. Hence some other fog resources are idle, and it is better to be federated rather than forwarding them to the cloud server. Obviously, this issue affects the performance of the fog environment when dealing with big data applications or applications that are sensitive to time processing. This research aims to build a fog topology job scheduling (FTJS) to schedule the incoming jobs which are generated from the IoT devices and discover all available fog nodes with their capabilities. Also, the fog topology job placement algorithm is introduced to deploy jobs into appropriate resources in the network effectively. Finally, by comparing our result with the state-of-art first come first serve (FCFS) scheduling technique, the overall execution time is reduced significantly by approximately 20%, the energy consumption in the cloud side is reduced by 18%.

Key words:
Fog Computing, Job Scheduling, Resource Management, Execution Time, Power Consumption.

1. Introduction

A modern trend in technology and communications is the Internet of Things (IoT). The IoT can be defined briefly as anything that can be connected to the Internet and provide or produce data [1], including all online objects such as smart cameras, wearable sensors, environmental sensors, smart home appliances, cars, etc. Currently, the number of IoT devices in our world has been reaching approximately 75 billion things in 2025 [2]. The IoT technology improves and facilitates the quality of human life; hence a huge amount of data is generated through the IoT devices, which produces an unnecessary burden for data storage and analysis systems [3]. As a result, cloud computing is a critical source that can deal with this enormous data and applying some analysis on it. There are a lot of applications can be considered as sensitive applications for time responding such as smart traffic control applications, health monitoring applications, and surveillance camera system. Obviously, the enormous data produced by some of these applications can impose heavy network burdens. It is not efficient to offload all of this amount of data to the cloud and then return it [4]. Therefore, in 2012, Bonomi proposed a new term called fog computing [1]. Fog computing is a modern model which considered as an extension of clouds to provide services to network parties and offers a technique to solve the previous problem.

The distributed fog computing is placed between the IoT devices and cloud servers. Three layers are used in the Fog computing architecture: the bottom layer contains IoT devices such as surveillance cameras, health wearable devices, smart home appliances. The fog computing layer is the middle layer, which has resources like routers, computers, and gateways. The cloud layer consists of servers and data centers, as shown in Fig.1.

In general, fog computing and cloud servers are complementary to each other, and the significant goal of placing fog computing in an IoT environment is to support and increase cloud efficiency. Recently, Google has invented the federated learning (FL) approach, which mitigates the offloading to the cloud server. The main idea is to assign a specific dataset for each IoT device and an aggregation server at the network edge. Generally, the IoT device has its own model that trains the data locally instead of federating it to the centralized cloud. This approach has been adopted in many applications, such as in smart cities [6] and health care [7].

Both fog computing and cloud computing have similar features. However, fog computing is characterized by including geographical distribution, interaction in real-time, mobility support, heterogeneity, and interoperability [5]. Moreover, in fog computing, all nodes can execute the jobs instead of a single node when high performance is needed. Overall, fog computing is a suitable concept to increase the efficiency of IoT-cloud environments since it can make a reduction in latency time, network traffic, and energy consumption. However, this concept also has challenges because of the novelty. One of these challenges is mentioned to resource utilization and scheduling [1].

Manuscript received January 5, 2021
Manuscript revised January 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.1.18

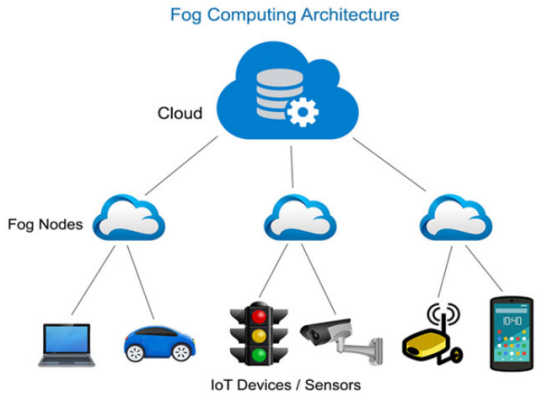

Fig. 1 The cloud-fog architecture

The concept of job scheduling in fog computing means placing a series of jobs to fog resources effectively. Placing too many jobs in fog resources can reduce the execution time, but the power consumption might be increased. However, forwarding the jobs into the cloud side can migrate the power consumption in fog computing, but the execution time would be increased. Therefore, an effective job scheduling and placement strategy are necessary considering the tradeoff between energy consumption and execution time. Also, optimize the usage of these resources Quality of Service (QoS) requirements. In this paper, we propose a tradeoff fog topology job scheduling (FTJS) strategy for scheduling the incoming jobs and place them in the suitable cloud-fog resources by utilizing all available resources in the whole network with their capabilities. The main contributions to this paper are following:

1- The fog topology job scheduling (FTJS) algorithm is designed to receive the incoming job and detect all available resources in the system.

2- The fog topology job placement (FTJP) algorithm is designed to place the ordered jobs in the suitable nodes.

3- The energy consumption and execution time have been reduced significantly.

2. Related Work

The scheduling concept definition is to determine an best solution for placing a set of jobs $J= \{j_1, j_2,...,j_n\}$ on a set of machines $M = \{m_1,m_2, ..., m_m\}$. One of the challenges in fog computing is to select suitable edge resources to place computation jobs from cloud and IoT devices. There is needed for efficient selector algorithms that can address this issue by considering the availability of edge resources with their capabilities [8]. In [9], the authors proposed a new method for managing mobile and edge devices. The devices are distributed in decentralized nodes. The IoT devices can be connected as peer-to-peer and decentralized, so this paper has solved the concepts related to IoT infrastructure. The problem of distributing tasks in fog computing has gained attention from researchers recently. The authors in [10] have analyzed the offloading policy between multiple fog nodes in a ring topology. In [11], a distributed policy for tasks assignment that can be executed efficiently in the network edge cloud has been proposed. The author has not considered the communication between the fog-to-cloud and IoT-to-cloud. The scalability in this model is limited since the cloud servers send their status continuously to the mobile subscribers, and it will be difficult with a larger amount of edge devices.

A collection of predefined constraints and objective functions [12] can be used to plan the work. Maximizing the use of available resources and minimizing the waiting period on a job is one of the objectives of work scheduling [13]. In [14], the authors divide the scheduling algorithms for cloud and edge computing into two groups: traditional algorithms and smart algorithms. For small scheduling problems, traditional algorithms are suitable, but the issue arises in large scheduling jobs. Therefore, they tried to enhance the solutions by selecting efficient algorithms such as meta-heuristic algorithms and heuristic for large complex problems.

The purpose of the research in [15] is to reduce network usage by presenting an optimization policy for data placement in the fog environment. This can be achieved by finding out the closest path between the fog device and the data source (IoT device). Minimizing the execution time and maximizing the throughput are achieved in the paper [16]. The algorithm distributes the workload on the fog resources environment. Also, a job scheduling technique is applied for Virtual Machines (VM) based on the service level agreement. In the paper [17], the authors demonstrate an architecture for mapping and migrating the service between the cloud and fog computing. The decision rule relies on three conditions: completion time, services sizes, and the capacity of fog resources.

In the first come first serve (FCFS) scheduling method, new jobs are placed at the end of the queue. From the beginning of the queue, the first job still runs first. The FCFS method for scheduling tasks is the basis of the round-robin method. For fixed times, resources are allocated to jobs. This strategy has the benefit of load balancing [18]. The authors in [19] propose an algorithm that reduces the cost of the delay and energy consumption through assigning resources and communications to global user equipments (UEs). In their algorithm, they study a multiuser offloading challenge with the indeterminate job requirement, even though the performance is enhanced significantly and cannot guarantee execution delay.

3. Fog Topology Job Scheduling

In this section, we propose a fog topology job scheduling (FTJS) algorithm, which can reduce the waiting times caused by the FCFS strategy. Most of the fog computing systems use the FCFS strategy, which executes one job at a time. Obviously, this strategy is not efficient when the system is dealing with a huge number of jobs. Moreover, the job priority is not considered in this strategy as well.

Suppose the system topology consists of 4 main areas, and each area has 10 fog resource nodes. So, we have 40 fog resources that can execute the job in a fog computing network. When any nodes in the system cannot accept any more jobs, it would be migrated to the cloud side. In the proposed approach, we add a distribution model between the incoming jobs and the system. The size of the model is L, which is the number of jobs to be executed in the system, as shown in Fig.2.

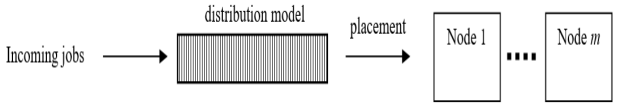

Fig. 2 Fog topology job environment

Once the scheduling process starts, all the jobs would be placed into the distribution model and allocate to the appropriate nodes in the fog system. Also, the devices in the system would be scanned in each periodical scheduling cycle. The purpose of the scanning technique is to detect all available resources and their capabilities in the system. After determining the free and suitable resources in the system, we acquire a set of waiting jobs in the distribution model order by the priority.

Algorithm 1 describes the periodical scheduling cycle. Firstly, the algorithm scans the system and discovers the set N of M free resources. Secondly, gathering the set J of L from the distribution model ordered by the priority. Thirdly, placing each job J into N until all N resources are full. Fourthly, if all the jobs L are placed, the current scheduling cycle s will be terminated and suspended until the next scheduling cycle. Finally, if the job is rejected and cannot be executed in the fog system, it will be migrated to the cloud side to be executed. This usually happens when the jobs require multi-core processors to be implemented, such as big data applications.

Algorithm 1 fog topology job scheduling (FTJS)

If scheduling cycle *s* is launched **then**
  scan the fog system and discover the set *N* of *M* free resources: $N = \{n_1, \ldots, n_M\}$
  gather the set *J* of *L* from distribution model: $J = \{j_1, \ldots, j_L\}$
  Job Placement (*J* , *N*)
→ **Algorithm 2**
  If all the jobs in L are executed **then**
    terminate the scheduling cycle *s*+1
  else if $j_i \in J$ is rejected **then**
    if multi_core(j_i) == *true* **then**
      migrate_to_cloud (j_i)
      terminate the scheduling cycle *s*+1
    else
      reserve space in distribution model

4. Fog Topology Job Placement

The topology of the fog computing network consists of three layers: cloud server layer, fog nodes layer, and mobile or sensor devices layer. The fog nodes layer usually includes more than areas in the same level, and each area has many fog device nodes. The red nodes cannot accept any more jobs. The blue nodes have some jobs to be executed and can accept more jobs. The green nodes are idle, and no jobs have been allocated to them, as shown in Fig3. So, we introduce another algorithm, which determines the placements for the set of jobs in the distribution model into the free nodes in the fog computing network, so the system utilization is exploited.

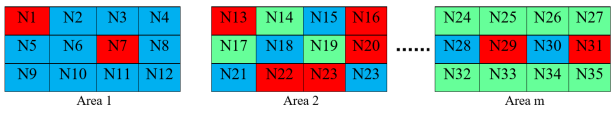

Fig. 3 Areas and nodes in fog system

Let J = { j1, j2, …, jL} be the set of L waiting jobs in the distribution model that are ordered by the priority. The waiting job jj in the distribution model has its own priority wi . The meaning of the priority is the required cores to execute the job. Let N = { n1, n2, …, nm } the sets of available nodes, which derives from the set acquired in Algorithm 1. Eventually, we want to determine a placement for the set of jobs J into the free nodes N, with the profit of maximizing the system utilization, as shown in equation (1).

$$Max: \sum_{i=1}^{L} \sum_{j=1}^{M} x_{ij}\, n_i \qquad (1)$$

$$S.T: \sum_{j=1}^{m} x_{ij} \leq 1, \forall i = 1,2,3, \ldots., L$$

$$\sum_{i=1}^{L} x_{ij}\, w_i \leq C_j \; \forall j = 1,2,3, \ldots., M$$

When the job j_i is placed in the fog node n_i, the value of $x_{ij} = 1$, in case the job j_i is not placed in the fog node n_i, the value of $x_{ij} = 0$. Each fog node n_i has the capacity of C_j. Obviously, the capacity will be reduced if more jobs are coming in that fog node.

The key goal of our solution is to reduce the job execution time and power consumption in the fog computing network by improving the job placement process. The job scheduling and placement strategies play a significant role in saving energy and speeding up the system performance through utilizing all available nodes considering their capabilities. For instance, in Fig.3, if we place more jobs in Area1, there would be a waste of time and energy because no fog nodes can accept any more jobs at this current time. Therefore, the Area2 is a better choice since there are three idle fog nodes.

Algorithm 2 describes the job placement process. The algorithm receives two inputs: the set of free and available nodes N and the set of jobs J waiting to be executed, which are coming from Algorithm 1. Firstly, the fog nodes are sorted and ranked increasingly depending on their remaining space for a new job. Secondly, the jobs are sorted depending on their priority. In this case, we assume the job is already ordered by assigned priority. Thirdly, the algorithm scans the whole system regularly and updates the set of fog nodes. Finally, the jobs are placed in the available fog nodes according to the priority.

Algorithm 2 Job Placement

Input: the set N of M nodes: $N = \{ n_1, n_2, \dots , n_m \}$
the set J of L waiting jobs in the distribution model: $J = \{ j_1, j_2, \dots , j_L \}$
sort and rank each n_i increasingly by free space for jobs
sort and add the job j_i to PR by priority
for *each job* $\in j_{pr}$ **DO**
scan the system to obtain updated set N of free fog nodes
if n_i has more space for j_{pr} **then**
return placing j_{pr} in n_i
else
continue

5. Simulation and Result

The simulation focuses on applying the proposed fog topology job scheduling algorithm in the fog computing environment to present and validate the effusiveness of our method. For simplicity and taking into account the principle, we consider the scenario with 4 main areas, and each area has 10 nodes. It means we have at maximum 44 available resources distributed in two layers of fog and IoT. Table 1 illustrates the characteristics of each device in the topology.

Table 1: Characteristics of the devices in the system

Characteristic	Cloud	Fog Resources		
		Proxy	Router	Camera
CPU	50000 (MIPS)	3000 MIPS	2500 (MIPS)	600 (MIPS)
RAM	1 T.B	6 G.B	4 G.B	1 G.B
Uplink latency	None	100	5	2
Uplink bandwidth	100	10000	10000	10000
Downlink bandwidth	100	10000	10000	10000
Parent	None	Cloud	Proxy	Router

To present the efficient performance of our method at varied workloads, we selected 4 groups of tasks. The total workloads for each group are 10, 20, 40, and 60, respectively. We conduct the simulation using the iFogSim tool on a computer equipped with Intel® i5 Core 2.40 GHz running windows 10 with 64-bit.

Since the FCFS method is the most popular in cloud and fog computing, and it is already implemented in the iFogSim simulator, we compare our method FTJS with the state-of-art FCFS scheduling technique.

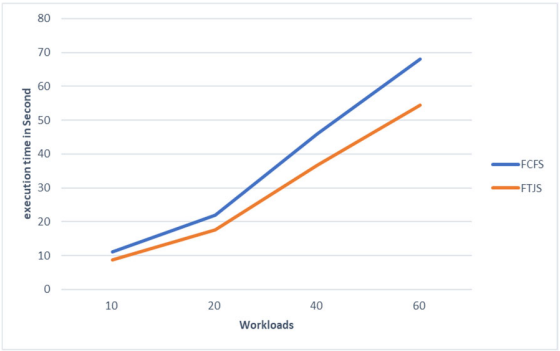

Fig. 3 Execution time comparison

Table 2 demonstrates the average execution time for our method FTJS and FCFS.

Table 2: Comparison of average execution time

Method	Group	Workloads	Execution time
FTJS	1	10	9
	2	20	18
	3	40	37
	4	60	54
FCFS	1	10	11
	2	20	22
	3	40	46
	4	60	68

The average execution time for each method as follow:

- FTJS: (9+18+37+54)/4 = 29.5
- FCFS: (11+22+46+68)/4 = 36.75

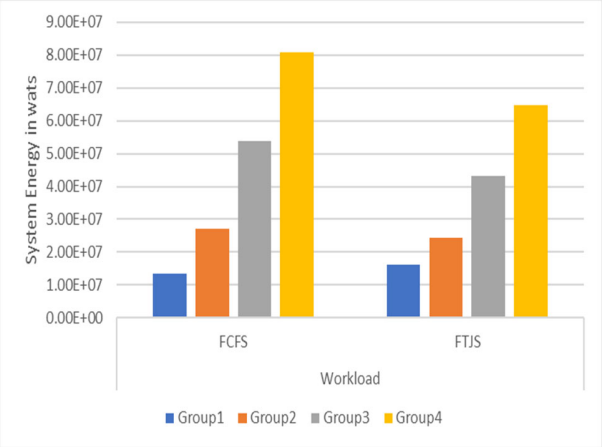

Fig. 5 Energy consumption in cloud server

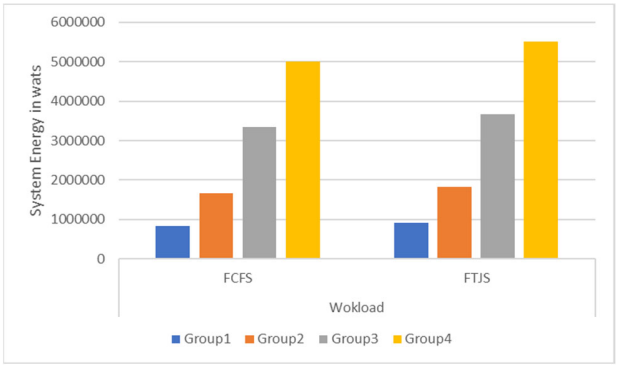

Fig. 6 Energy consumption in fog nodes

As shown in Fig.4, the execution time is reduced in our proposed FTJS compared with FCFS by 20%. The reason for this is to use all possible fog nodes in the topology and try to keep the job execution in the fog layer rather than in the cloud side. As mentioned before, the benefit of using fog layers is to migrate the delay in the IoT-cloud environment by implementing the job close to the IoT devices.

The energy consumption on the cloud side is shown in Fig.5. In our method FTJS, the power consumption is lower than in FCFS by 18%. This is due to the FCFS method of forwarding the jobs into the cloud side periodically when a node cannot accept that job. In our method, the distribution model can detect and monitor the status of all possible fog nodes, then place that job into them instead of forwarding the job into the cloud side. However, our method FJTS in fog resource network is higher than FCFS by 12% as shown in Fig.6. The reason for this is our method utilizes the whole system's capabilities. The idle devices are considered in our method, and the algorithm will not offload jobs into the cloud unless it cannot be implemented in any possible fog nodes.

5. Conclusion

In this paper, we introduce a tradeoff method that can realize magnificent execution time with power consumption. To solve the previous issue, we built two algorithms that can utilize all fog resources in the system and do the work close to the IoT devices. The experimental results demonstrate that there is a reduction in the execution time by 20% while the power consumption in cloud server by 18%. However, the power consumption in fog resources has increased by 12%. We anticipate enhancing the job scheduling in fog computing by implementing the jobs in parallel for future work. The fog nodes are heterogeneous, so it is possible to detect multi-core CPUs in fog computing.

References

[1] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli, "Fog computing and its role in the internet of things," Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16, 2012.

[2] Mckinsy Global Institute website "The Internet of Things: Mapping The Value Beyond The Hype," Accessed 5.1, 2021. [Online]. Available: https://www.mckinsey.com

[3] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," IEEE Access, vol. 5, pp. 8284–8300, 2017

[4] K. Kambatla, G. Kollias, V. Kumar, A. Grama, "Trends in big data analytics," Journal of Parallel and Distributed Computing, vol.74, no.7, pp. 2561-2573, 2014.

[5] F. A. Kraemer, A. E. Braten, N. Tamkittikhun and D. Palma, "Fog Computing in Healthcare–A Review and Discussion," in IEEE Access, vol. 5, pp. 9206-9222, 2017.
[6] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, "Federated learning in smart city sensing: Challenges and opportunities," Sensors, vol. 20, no. 21, p. 6230, 2020.
[7] J. Xu and F. Wang, "Federated learning for healthcare informatics," arXiv preprint arXiv:1911.06270, 2019.
[8] R. Kolcun, D. Boyle, and J. McCann, "Optimal processing node discovery algorithm for distributed computing in IoT," in The 5th International Conference on the Internet of Things pp.7279, 2015.
[9] R.-I. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, G. Mastorakis, "Drop computing: Ad-hoc dynamic collaborative computing," Future Gener. Comput. Syst., vol. 92, pp. 889-899, Mar. 2017.
[10] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," ACM Trans. Model. Perform. Eval. Comput. Syst., vol. 1, no. 4, p. 16, 2016.
[11] X. Guo, R. Singh, T. Zhao, Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," Proc. IEEE Int. Conf. Commun. (ICC), pp. 1-7, May 2016.

[12] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A hyper-heuristic scheduling algorithm for cloud," IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 236–250, 2014.
[13] N. Patil and D. Aeloor, "A review-different scheduling algorithms in cloud computing environment," 11th International Conference on, IEEE, pp. 182–185, 2017.
[14] N. Patil and D. Aeloor, "A review-different scheduling algorithms in cloud computing environment," 11th International Conference on, IEEE, 2017, pp. 182–185, 2017.
[15] I. Lera, C. Guerrero, and C. Juiz, "Comparing centrality indices for network usage optimization of data placement policies in fog devices," in Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC), vol. 1, no. 1, pp. 115–122, 2018.
[16] S. Agarwal, S. Yadav, and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," Int. J. Inf. Eng. Electron. Bus., vol. 8, no. 1, pp. 48–61, 2016.
[17] A. A. Alsaffar, H. P. Pham, C.-S. Hong, E.-N. Huh, M. Aazam, "An architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing," Mobile Inf. Syst., vol. 2016, Aug. 2016.
[18] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," in Advances in Computing, Communications and Informatics (ICACCI), International Conference on. IEEE, pp. 658–664, 2014.
[19] N. Eshraghi and B. Liang, "Joint offloading decision and resource allocation with uncertain task computing requirement," in IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, pp. 1414–1422, 2019.

Anwar Alghamdi is a lecturer in faculty of computer and information technology at Al-Baha University. He obtained the BSc degree in computer science from King Saud University, Riyadh, Saudi Arabia in 2008. He obtained the MSc degree in computer science form Gannon University, USA in 2015. Currently, he is pursuing a Ph.D. in the field of computer science at King Abdulaiziz University. His research interests are in the fields of fog computing, Internet of Things, and big data.

Ahmed Alzahrani received the B.S. degree in computer science from King Abdul Aziz University in 2000, and the M.S. degrees in information security from University of Glamorgan, Cardiff, UK in 2005. He received the Ph.D. degree in computer networks from University of Bradford, UK in 2009. He is currently an Associate Professor with the Computer Science Department, and the current Vice Dean of Deanship of Graduate Studies for academic affairs, King Abdul Aziz University. His current research interests include computer networs, networks security, quality of service routing, quantum computing, deep learning, big data, in-memory computing, and high performance computing.

Vijey Thayananthan: is a Professor at Computer Science Department in the King Abdulaziz University, Jeddah, KSA. He obtained his Ph.D. in Engineering (Digital communication engineering) from Department of Communication Systems, University of Lancaster, UK, in 1998. Since 2000, he had been working as a Research engineer and senior algorithm development engineer in Advantech Ltd, Southampton University Science Park, UK and Amfax Ltd, UK respectively. His research interests include wireless communication algorithm design and mobile communication analysis, security management of communication network and big data, computer security and wireless sensor network. He has been a full-time member of the Institution of Engineering Technology (IET), UK since 2005.