Resource Management Technique for IoT in Fog Computing Supported by Distributed SDN

Ahmed Jawad Kadhim
Computer Engineering Department
Ferdowsi University of Mashhad
Mashhad, Iran
Ahmed.kadhim@mail.um.ac.ir

Seyed Amin Hosseini Seno
Computer Engineering Department
Ferdowsi University of Mashhad
Mashhad, Iran
hosseini@um.ac.ir

Abstract

Internet of Things (IoT) refers to the interconnection of a very large number of heterogeneous-limited-resource-devices that senses and collects information about their environments. The traditional method to solve the resource scarcity problem in IoT is to leverage the required resources from cloud environment. Things continuously send the requests to the cloud through internet connection. But this is not an optimal solution due to the latency and bandwidth expensive, so the optimal solution is a fog computing. The idea behind the fog computing is moving the resources to the network edge to be close to the IoT devices. In this paper we propose efficient resource management technique based on software defined network (SDN) capabilities to enhancing the QoS of IoT by exploiting the collaboration between the fog and cloud computing. We propose architecture of clusters of fog devices controlled by distributed SDN controllers. In addition the proposed architecture contains central SDN controller connects to all distributed SDN controllers and all cloud servers. So that it contains global view of the network. This paper investigates many issues: task scheduling, mitigate the load, resource discovery and resource selection to reduce the response time and guarantee execution all the hard real time tasks within their deadlines and produces the best effort to execute the soft real time tasks to reduce the penalty.

Keywords:- internet of things; fog computing; software defined network; cloud computing; resource management

Introduction

Internet of things is a new paradigm of connecting large number of heterogeneous things that found around the people in different environments. Things may be sensors, actuators and mobile devices interconnect with each other and collect information about these environments in order to take true decision without any intervention from humans. This generates huge data that needs to storage, analysis, and process in good manner. Cloud computing considered as a solution to solve the IoT's limitations [2].

Cloud computing is large, flexible, scalable and reliable world of powerful physical resources that can be accessed in shared manner through the internet. Cloud computing uses these resources to provide applications, platforms, services and data storage to the user demands in virtual manner [5]. But there are some challenges that make the cloud is not an optimal choice for IoT due to the high and unacceptable latency between the IoT devices and cloud which leads to catastrophe in some hard real time tasks. So this was the motivation to find new concept to serve IoT efficiently which is the fog computing.

The idea behind fog computing is migration of the resources to the devices that exist at the network edge to be closer to things. Fog device can be any device has the ability of connection, processing and storage such as server, router… etc. that distributed in different locations. Fog computing is not substitute for the cloud computing but a complement to its work. Another good advantages of fog computing are: supporting the mobility, location awareness and supporting real time applications that require high quickly responses. It provides efficient, easy and flexible solution for IoT to enhancing QoS because it helps in decreasing the power consumption, transmission delay and increasing the throughput [11].

SDN is developed to simplify the architecture of traditional networks and make it programmable. It is a technology or architecture that separates the traditional network architecture into two planes (data plane and control plane). Data plane contains simple devices which only forward the data packets while the control plane contains the controller(s) that represents the brain and very important part of this architecture [3]. The control plane can be centralized or distributed [4]. There are many benefits from merging SDN with fog computing to improve the IoT network that are: first, SDN provides simple management for different heterogeneous fog devices. Second, SDN provides global view about the IoT network that can help in controlling the network resource and infrastructures [8]. Third, the SDN controller has all knowledge about the resources and tasks [10]. So that we will use the capabilities of distributed SDN controllers to produce efficient resource management technique in fog computing.

The main contributions of this paper are as follows: There are many researches focused on the collaboration between fog and cloud environments, so we first studied these works and diagnosed their challenges.

We propose architecture of clusters of fog devices that work under the control of SDN controllers to perform the resources provisioning purpose. Also we propose efficient resource management technique by exploiting the fog-cloud platform and SDN capabilities to investigate the task scheduling, mitigate the load, resource discovery and resource selection issues to improve the QoS of IoT.

The rest of the paper is organized as follows: in section II we discuss the related works. Section III introduces our system model and problem formulation. Section IV describes our proposed algorithm. At last section V concludes our work.

Related Works

There are many researchers studied the interplay between fog computing and cloud computing for resource provisioning purpose and optimizing QoS. In [1] proposed workload allocation technique between edge and cloud computing based on tradeoff between the delay and energy consumption. Also they proposed architecture of four layers: user interface, edge computing, dispatch, and cloud. They used interior-point and generalized benders decomposition methods to create tradeoff between delay and energy consumption in edge and cloud computing respectively, but in this paper there is high latency because the fog device forwards the request directly to the cloud computing if it fails to handle that request.

In [6] presented load balancing algorithm upon the dynamic graph partitioning in fog computing. They implemented hierarchical fog computing framework decomposed of four layers that are physical resource, resources layer of cloud atomization, service management and platform management layer. The system carried the cloud atomization technique on fog nodes to create virtual machines. The disadvantage of this paper is that the load balancing implements the protocols at the traditional hardware that is increase the delay and overhead as well as it is difficult for the network management.

In [7] presented heuristic scheduling algorithm to provide tradeoff between response time and monetary cost. They used architecture of three layers: IoT devices, fog, and cloud computing. Fog computing layer contains n fog devices managed by the broker. IoT devices send the tasks to the broker that schedules and forwards them to the best fog devices or cloud servers. The disadvantages of this paper are delay and overhead in the broker. Also sending all tasks to the broker instead of closest fog devices may lead to additional delay especially when the best fog device is the closest one.

In [9] proposed hierarchical architecture of n layers: mobile devices, multilayers of fog devices and cloud layer. Each one of fog computing multilayers consists of arbitrary number of servers and each server connects with all other servers. All edge devices exchange some of their information among each other. The mobile device sends the requests to the closest edge device. If that device fails to execute the request, it forwards that request to the higher layer. The authors proposed request placement algorithm depends on the branch and bound to convert the large complex problem to many small problems to select the optimal server to execute the request based on the capacity and transmission time. This technique suffers from high overhead due to the exchanging of the information among the fog devices at different layers.

In [12] used the standard architecture of three tiers: clients, fog, and cloud. They proposed load balancing algorithm based on replication technique. The client sends the request to the nearest fog device. If it fails to execute the request then it will broadcast that request to other fog devices. If there is a certain one can execute that request, will send the required data to the source fog device which will replicate it to be used in the future, else it will send the request to the cloud server. The broadcasting of the request is not optimal solution because more than one fog device may execute that request and send the response, so this technique leads to high overhead and increase the energy consumption.

In [13] used classical architecture of three layers: clients, fog computing and cloud computing layer. Each client sends its request to the closest fog device which executes the hard real time request based on the data availability and workload while the soft real time requests will wait until finishing all hard requests. If the fog device fails to execute the hard request, it will send that request to the cloud directly, but this leads to high delay in hard and soft requests and increases the lost soft requests in the congestion states.

In [14] proposed architecture of three layers: Embedded client, storage server, and computation server. The storage servers save the required data to process the tasks. The embedded clients process the delay sensitive tasks while other tasks are handles by the computation servers. There are three types of delay: I/O interrupt, computation and transfer delay. They used task placement technique to minimize I/O interrupt delay and scheduling algorithm to decrease the computation delay. The disadvantage of this paper is that the size of data may be very large that leads to high I/O interrupt delay.

It is necessary to address the challenges of the previous works, so we produce resource management technique for fog computing based on the SDN capabilities to optimize the response time.

System Model and Problem Formulation

In this paper we propose architecture of five layers named from bottom to top: IoT devices, fog computing, distributed SDN controllers, central SDN controller and cloud computing as shown in figure 1. Each group of IoT devices $D = \{d_1, d_2, \ldots, d_n\}$ connects to one fog device in local area network. Fog computing layer contains number of clusters $C = \{c_1, c_2, \ldots, c_n\}$, and each cluster c_j has arbitrary number of SDN-based-fog-devices. Let $F = \{F_{c1}, F_{c2}, \ldots, F_{cn}\}$ is the set of all fog devices in the network and $F_{cj} = \{f_{1cj}, f_{2cj}, \ldots, f_{ncj}\}$ is the set of fog devices in c_j. Let $S = \{s_1, s_2, \ldots, s_n\}$ is the set of SDN controllers in distributed SDN controllers layer. Each s_j contains complete view of c_j. In layer 4, C_c represents the central SDN controller (powerful SDN controller) that controls and connects to all SDN controllers in layer 3 and the cloud servers in layer 5. It is contain complete view of the network. In highest layer there are cloud servers $K = \{k_1, k_2, \ldots, k_n\}$.

Normally each SDN controller has database which contains information about each path in its domain such as average of lost packets, delay and number of hops to travel from one point to another. In addition to this database we propose that each SDN controller has multi agents that used to perform many functions such as scheduling, monitoring and decision making. The scheduling agent schedules the requests based on Earliest Deadline First scheduling algorithm. The monitoring agent sends periodically messages to fog devices and cloud servers to collect information about their current state such as the number of tasks in the queue and average service time. The scheduling agent sends the requests to the decision making agent that based on the information of the monitoring agent and the database of SDN controller selects the optimal solution.

Let $T = \{t_1, t_2, \ldots, t_n\}$ is the set of task requests that launched from IoT devices to the fog device f_{icj}. Each t_i in T has some parameters $\{ar_{ti}, d_{ti}, e_{ti}\}$ where ar_{ti} is the arrival time, d_{ti} is the deadline and e_{ti} is the maximum execution time of t_i. In table 1 there are other notations. For each t_i, f_{icj} needs to check the feasibility and available capacity.

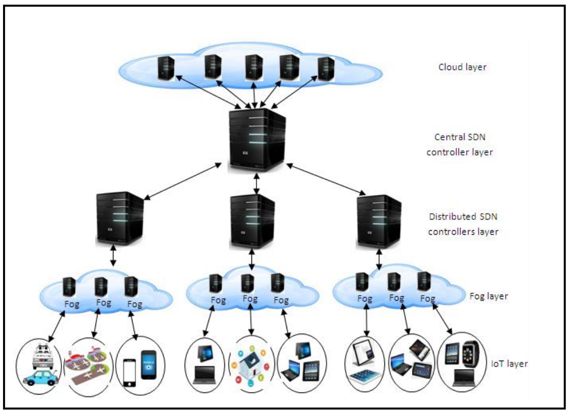

Fig 1: The proposed architecture.

Table 1: Notations

Symbol	Description
Ttti	The type of ti that may be hard real time task or soft real time task.
HTQ	Hard task queue
STQ	Soft task queue
A_{TFsj}	The address of optimal fog device in the same region (cluster) that selected by s_j to execute t_i
A_{TFc}	The address of optimal fog device in other region that selected by C_c to execute t_i
A_{TK}	The address of optimal cloud server that selected by C_c to execute t_i
OP	The optimal path to reach from f_{icj} to A_{TFsj}, A_{TFc} or A_{TK}
R_h	Hard request that tells s_j or C_c that f_{icj} has hard real time task needs to be executed in other fog device or cloud server.
R_s	Soft request that tells s_j or C_c that f_{icj} has soft real time task needs to be executed in other fog device or cloud server.
HCS_{sj}	Hard candidate set of fog devices in the same cluster. This set has been created by s_j to execute certain hard real time task.
SCS_{sj}	Soft candidate set of fog devices in the same cluster which have not any hard real time task. This set has been created by s_j to execute certain soft real time task.
HCS_c	Hard candidate set of fog devices in the other clusters. This set has been created by C_c to execute certain hard real time task.
SCS_c	Soft candidate set of fog devices in the other clusters which have not any hard task. This set has been created by C_c to execute soft real time task.
D^{comp}_{ificj}	It is the computation time that required to execute t_i in f_{icj}
D^{wait}_{ificj}	It is the total waiting time of t_i in f_{icj} until selecting the appropriate device to execute it.
$D^{comm}_{Rhficj\text{-}sj}$	It is the communication time required to send R_h from f_{icj} to s_j.
$D^{comm}_{ificj\text{-}TFsj}$	It is the communication time needed to send t_i from f_{icj} to TF_{sj}.
D^{comp}_{iTFsj}	It is the required time to execute t_i in TF_{sj}.
$D^{comm}_{TFsj\text{-}ficj}$	It is the communication time required to send response from TF_{sj} to f_{icj}.
D^{dis}_{sj}	It is the required time to discover the candidate fog devices in c_j and select the optimal one to execute t_i.
$D^{comm}_{sj\text{-}ficj}$	It is the required time to send A_{TFsj} and OP from s_j to f_{icj}.
$D^{comm}_{ificj\text{-}TFc}$	It is the needed time to send t_i from f_{icj} to TF_c.
D^{comp}_{iTFc}	It is the required time to execute t_i in TF_c.
$D^{comm}_{Rhsj\text{-}Cc}$	It is the communication time needed to send R_h from s_j to C_c.
$D^{comm}_{TFc\text{-}ficj}$	It is the communication time required to send response from TF_c to f_{icj}.

D^{dis}_{Cc}	It is the needed time to discover the candidate fog devices in other fog clusters and select the optimal one to execute t_i.
$D^{comm}_{tificj-TK}$	The required time to send t_i from f_{icj} to TK.
D^{compu}_{tiTK}	The required time to execute t_i in TK.
$D^{comm}_{TK-ficj}$	It is the communication time required to send response from TK to f_{icj}.

A. Feasibility Constraint

In some time intervals there are high intensity of tasks, so some tasks may not meet their deadlines, let X_{tificj} denotes whether t_i meet its deadline in f_{icj} or not

$$X_{tificj} = \begin{cases} 1, & \text{if } t_i \text{ can be executed in } f_{icj} \text{ and meet its deadline} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

To sovle this problem, all these tasks must be sent to the higher layers to find the appropriat fog device or cloud server to execute them.

B. Fog Device Capacity Constraint

Let max^{ficj} is the upper bound of fog device f_{icj} capacity and $Load^{ficj}$ is the workload allocated to fog device f_{icj} so

$$0 \leq Load^{ficj} \leq max^{ficj} \quad (2)$$

Let L_{ficj} illustrates whether the fog device is overloaded or not.

$$L_{ficj} = \begin{cases} 1, & \text{if } f_{icj} \text{ is overloaded} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

C. Cloud Server Capacity Constraint

In each cloud server there are number of virtual machines, each one has limited capacity. Let min^{kivi} and max^{kivi} is minimun and maximum capacity of virtual machine v_i in cloud server k_i, and L^{kivi} is the workload allocated to v_i so:

$$min^{kivi} \leq L^{kivi} \leq max^{kivi} \quad (4)$$

If f_{icj} fails to execute hard task t_i, it will send request R_h to s_j. This request tells s_j that f_{icj} is overloaded. s_j by using its multi-agents checks other fog devices in its domain and creates candidate set to find the optimal one (TF_{sj}) to execute t_i. Then s_j sends to f_{icj} the address of TF_{sj} and the optimal path to reach from f_{icj} to TF_{sj}. At this point f_{icj} sends t_i to TF_{sj}. Finally TF_{sj} execute t_i and sends response to f_{icj}. If s_j fails to find the appropriate fog device to execute t_i because all other fog devices in its domain are overloaded, then will forward R_h to C_c. C_c will select optimal device from other fog clusters TF_c or select the optimal cloud server TK from cloud computing layer if all fog devices are overloaded. Then C_c sends to f_{icj} the address of selected device and the optimal path to reach it. As well as sends updated flow tables to all intermediate points in this optimal path. f_{icj} sends t_i to it (TF_c or TK) for processing which later sends response to f_{icj}.

D. Task Compeltion Constraint

To enhance the QoS of real time system, all the hard real time tasks must complete their execution within their deadlines that leads to optimize the response time and throughput. For each task t_i there is probability to be executed by closest fog device (f_{icj}), other fog device in the same cluster (TF_{sj}), fog device in other cluster (TF_c), or in worst case by the cloud server (Tk), so

$$\sum_{ficj \in Fcj} Ptific + \sum_{TFsj \notin Fcj} PtiTFs + \sum_{TFc \in F} PtiTF + \sum_{TK \in K} PtiTk = 1, \forall t_i \in T \quad (5)$$

E. Fog Computation Delay

Fog device from the view point of queuing theroy considered as m/m/1 queuing model. Let A_{ficj} is the service rate and B_{ficj} is the traffic arrival rate of fog device f_{icj}. The fog computation delay in f_{icj} can be computed as

$$D_{ficj}^{compt} = \frac{1}{Aficj - Bficj} \quad (6)$$

F. Cloud Computation Delay

In each cloud server k_i there are m virtual machines, so it is consider as m/m/n queuing model. Let B_{ki} is the traffic arrival rate and A_{ki} is the service rate. The cloud computation delay in k_i can be computed as

$$D_{ki}^{compt} = \frac{C(m, Bki/Aki)}{mAki - Bki} + \frac{1}{Aki} \quad (7)$$

G. Communication Delay

In our model there are number of communication delay types. Some of them are little significantly like the communication delay between f_{icj} and s_j, delay between s_j and C_c while some other of them is very high such as the delay between f_{icj} and cloud server.

The communication delay types are as follows:

- **Fog Device to SDN Controller Delay**

The traffic rate transferred from f_{icj} to SDN controller s_j is w_{ficjsj} and the delay from f_{icj} to s_j is d_{ficjsi}. The communication delay between fog device and SDN controller is

$$D_{ficj-sj}^{comm} = w_{ficjsj} * d_{ficjsi} \quad (8)$$

- **SDN Controller to Central SDN Controller Delay**

When the traffic rate launched from SDN controller s_j to central SDN controller C_c is w_{sjCc} and the delay from s_j to C_c is d_{sjCc}, the communication delay between SDN controller and central SDN controller can be computed as

$$D_{sj-Cc}^{comm} = w_{sjCc} * d_{sjCc} \quad (9)$$

- **Central SDN Controller to SDN Controller Delay**

Let w_{Ccsj} denotes the traffic rate transferred from central SDN controller C_c to SDN controller s_j and

d_{Ccsj} denotes the delay from C_c to s_j. The communication delay between central SDN controller and SDN controller is

$$D^{comm}_{Cc-sj} = w_{Ccsj} * d_{Ccsj} \tag{10}$$

- **SDN Controller to Fog Device Delay**

When the traffic rate transferred from SDN controller s_j to the fog device f_{icj} is w_{sjficj} and the delay from s_j to f_{icj} is d_{sjficj}, the communication delay between s_j and f_{icj} is

$$D^{comm}_{sj-ficj} = w_{sjficj} * d_{sjficj} \tag{11}$$

- **Fog Device to Optimal Fog Device Delay**

If the traffic rate launched from fog device f_{icj} to the optimal fog TF is w_{ficjTF} and the delay from f_{icj} to TF is d_{ficjTF}, then the communication delay between fog device and optimal fog device is

$$D^{comm}_{ficj-TF} = w_{ficjTF} * d_{ficjTF} \tag{12}$$

- **Optimal Fog Device to Fog Device Delay**

Let w_{TFficj} represents the traffic rate launched from optimal fog TF to fog device f_{icj} and d_{TFficj} represents the delay from TF to f_{icj}. The communication delay between optimal fog device and fog device is

$$D^{comm}_{TF-ficj} = w_{TFficj} * d_{TFficj} \tag{13}$$

- **Fog Device to Optimal Cloud Server Delay**

Let w_{ficjTK} is the traffic rate transferred from f_{icj} to TK and d_{ficjTK} is the delay from f_{icj} to TK. The communication delay between fog device and optimal cloud server can be computed as

$$D^{comm}_{tificj-TK} = w_{ficjTK} * d_{ficjTK} \tag{14}$$

- **Optimal Cloud Server to Fog Device Delay**

Let w_{TKficj} is the traffic rate transferred from TK to f_{icj} and d_{TKficj} is the delay from TK to f_{icj}. The communication delay between optimal cloud server and fog device can be computed as

$$D^{comm}_{TK-ficj} = w_{TKficj} * d_{TKficj} \tag{15}$$

The hard real time tasks must be executed within their deadlines, so:

Case 1: if t_i can be executed in f_{icj} then

$$D^{compu}_{tificj} \leq d_{ti} \tag{16}$$

Case 2: if t_i can be executed in TF_{sj} then

$$D^{wait}_{tificj} + D^{comm}_{tificj-TFsj} + D^{compu}_{tiTFsj} + D^{comm}_{TFsj-ficj} \leq d_{ti} \tag{17}$$

Where

$$D^{wait}_{tificj} = D^{comm}_{Rhficj-sj} + D^{dis}_{sj} + D^{comm}_{sj-ficj} \tag{18}$$

Case 3: if t_i can be executed in TF_c then

$$D^{wait}_{tificj} + D^{comm}_{tificj-TFc} + D^{compu}_{tiTFc} + D^{comm}_{TFc-ficj} \leq d_{ti} \tag{19}$$

Where

$$D^{wait}_{ificj} = D^{comm}_{Rhficj-sj} + D^{comm}_{Rhsj-Cc} + D^{dis}_{Cc} + D^{comm}_{Cc-sj} + D^{comm}_{sj-ficj} \quad (20)$$

Case 4: if t_i can be executed in TK then

$$D^{wait}_{ificj} + D^{comm}_{ificj-TK} + D^{comp}_{iTK} + D^{comm}_{TK-ficj} \leq d_{ti} \quad (21)$$

Where

$$D^{wait}_{ificj} = D^{comm}_{Rhficj-sj} + D^{comm}_{Rhsj-Cc} + D^{dis}_{Cc} + D^{comm}_{Cc-sj} + D^{comm}_{sj-ficj} \quad (22)$$

The communication times required to transfer t_i from one location to another are different where

$$D^{comm}_{ificj-TFsj} < D^{comm}_{ificj-TFc} < D^{comm}_{ificj-TK} \quad (23)$$

$$D^{comm}_{TFsj-ficj} < D^{comm}_{TFc-ficj} < D^{comm}_{TK-ficj} \quad (24)$$

The Proposed Algorithm

In this section we explain the proposed algorithm to perform the resource management technique to provide the resource and assign each task to the appropriate fog device in order to enhance the QoS of IoT network. According to (23) and (24) the main goal of our algorithm is assigning more tasks to the fog devices that are close to the IoT devices and decrease transferring of tasks to far cloud servers to minimize response time and guarantee that all hard real time tasks execute within their deadlines. By using decision making agent, SDN controller s_j selects the optimal fog device TF_{sj} from its domain and central SDN controller C_c selects the appropriate fog device TF_c from other domains or TK from cloud computing layer based on the information of their database and information of monitoring agent.

Our Proposed Algorithm

d_i sends t_i to f_{icj}
if TT_{ti} = "hard" then
  if $L_{ficj} = 0$ & $X_{ficj} = 1$ then
    f_{icj} adds t_i to HTQ
    f_{icj} reschedules the hard tasks based on priority (deadline) by using EDF
  else
    f_{icj} sends R_h to s_j
    s_j reschedules the hard requests based on priority by using EDF
    s_j creates HCS_{sj} for R_h
    decision maker of s_j selects TF_{sj} from HCS_{sj}
    s_j sends A_{TFsj} and OP to f_{icj}
    f_{icj} sends t_i to TF_{sj}
    TF_{sj} reschedules the hard tasks based on priority (deadline) by using EDF
    TF_{sj} executes t_i and sends response to f_{icj}
    If $HCS_{sj} = Ø$ then
      s_j sends R_h to C_c
      C_c reschedules the hard requests based on priority by using EDF
      C_c creates HCS_c for R_h
      decision maker of C_c selects TF_c from HCS_c
      C_c sends A_{TFc} and OP to s_j
      s_j sends A_{TFc} and OP to f_{icj}
      f_{icj} sends t_i to TF_c
      TF_c reschedules the hard tasks based on priority (deadline) by using EDF
      TF_c executes t_i and sends response to f_{icj}
      If $HCS_c = Ø$ then
        decision maker of C_c selects TK
        C_c sends A_{TK} and OP to s_j
        s_j sends A_{TK} and OP to f_{icj}
        ficj sends ti to TK
        TK reschedules the hard tasks based on priority (deadline) by using EDF
        TK executes ti and sends response to ficj
Else (TT_{ti} = "soft")
  f_{icj} adds t_i to STQ
  f_{icj} reschedules the soft tasks based on priority (deadline) by using EDF
  If HTQ is empty then
    f_{icj} executes t_i
  Else
    f_{icj} sends R_s to s_j
    s_j reschedules the soft requests based on priority by using EDF
    s_j creates SCS_{sj} for R_s
    decision maker of s_j selects TF_{sj} from SCS_{sj}
    s_j sends A_{TFsj} and OP to f_{icj}
    f_{icj} sends t_i to TF_{sj}
    TF_{sj} reschedules the soft tasks based on priority (deadline) by using EDF

```
TF_sj executes t_i and sends response to f_icj
If SCS_sj = Ø then
    s_j sends R_s to C_c
    C_c reschedules the soft requests based on
    priority by using EDF
    C_c creates SCS_c for R_s
    decision maker of C_c selects TF_c from SCS_c
    C_c sends A_TFc and OP to s_j
    s_j sends A_TFc and OP to f_icj
    f_icj sends t_i to TF_c
    TF_c executes t_i and sends response to f_icj
    If SCS_c = Ø then
        decision maker of C_c selects TK
        C_c sends A_TK and OP to s_j
        s_j sends A_TK and OP to f_icj
        f_icj  sends t_i to TK
        TK reschedules the soft tasks based on
        priority (deadline) by using EDF
        TK executes t_i and sends response to f_icj
```

Conclusion

In this paper, we studied the existing resource management techniques and diagnosed the problems in each one. Then we proposed efficient technique to solve the load and lack of resources problems by using SDN capabilities to exploit the resources in fog computing landscape and cloud computing efficiently with take into account the transmission delay and workload. The main goal of our algorithm is assigning more tasks to the fog devices that are close to the IoT devices and decrease transferring of tasks to the far cloud servers in order to minimize response time and increase the throughput. Our proposed algorithm can provide good performance for the real time systems than cloud computing and existing strategies in the related works. In the future, we will add load balance technique in each SDN controller and take into account the latency and bandwidth consumption. Also we will use the simulation to build the proposed architecture and evaluate the proposed algorithm in terms of response time, the percentage of hard real time tasks that meet the deadline and the bandwidth cost.

References

[1] Deng, R., Lu, R., Lai, C., Luan, T.H. and Liang, H. (2016), “Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption”, IEEE internet of things journal, vol. 3, No. 6, pp. 1-11.

[2] Gubbia, J., Buyyab, R., Marusic, S. and Palaniswami, M. (2013), “Internet of things (iot): a vision, architectural elements, and future directions”, Elsevier, vol. 29, No. 7, pp. 1645-1660.

[3] Jammal, M., Singh, T., Shami, A., Asal, R. and Li, Y. (2014), “Software defined networking: state of the art and research challenges”, Elsevier, pp. 74-98.

[4] Kabo, H.I., Abu-mahfouz, A.M. and Hancke, G.P. (2017), “A survey on software-defined wireless sensor networks: challenges and design Requirements”, Browse Journal & Magazines-IEEE Access, vol. 5, pp. 1872-1899.

[5] Lakra, A.V. and Yadav, D.K. (2015), “Multi-objective tasks

scheduling algorithm for cloud computing throughput optimization", Elsevier, vol. 48, pp. 107-113.

[6] Ningning, S., Chao, G., Xingshuo, A., and Qiang, Z. (2016), "Fog computing dynamic load balancing mechanism based on graph repartitioning", China Communication, pp. 156-164.

[7] Pham, X. and Huh, E. (2016), "Towards task scheduling in a cloud-fog computing system", IEEE Japan, [proceedings of 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEICE].

[8] Qaisar, S. and Riaz, N. (2016), "Fog networking: an enabler for next generation internet of things", Springer International Publishing Switzerland, pp. 353-365.

[9] Tong, L., Li, Y. and Gao, W. (2016), "A Hierarchical edge cloud architecture for mobile computing", IEEE USA, [The 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM].

[10] Truong, N.B., Lee, G.M. and Doudane, Y.G. (2015), "Software defined networking-based vehicular adhoc network with fog computing", IEEE, , [IFIP/IEEE IM 2015 Workshop: 7th International Workshop on Management of the Future Internet (ManFI)], pp. 1202-1207.

[11] Vaquero, L.M. and Merino, L.R. (2014), "Finding your way in the fog: towards a comprehensive definition of fog computing", ACM SIGCOMM Computer Communication Review, vol. 44, No. 5, pp. 27-32.

[12] Verma, M., Bhardwaj, N. and Yadav, A.K. (2016), "Real time efficient scheduling algorithm for load balancing in fog computing environment", I.J. Information Technology and Computer Science, vol. 4, pp. 1-10.

[13] Verma, S., Yadav, A.K., Motwani, D., Raw, R.S. and Singh, H.K. (2016), "An efficient data replication and load balancing technique for fog computing environment", India, IEEE, [3rd International Conference on Computing for Sustainable Global Development (INDIACom)], pp. 5092-5099.

[14] Zeng, D., Gu, L., Guo, S., Cheng, Z. and Yu, S. (2016), "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system", IEEE Transactions on Computers, vol. 65, No. 12, pp. 1-11.

تقنية إدارة الموارد ل IOT في الحوسبة الضبابية بدعم من الموزعة SDN

احمد جواد كاظم
قسم هندسة الحاسبات
جامعة فردوسي
مشهد/ ايران
Ahmed.kadhim@mail.um.ac.ir

سعيد امين حوسيني سنو
قسم هندسة الحاسبات
جامعة فردوسي
مشهد / ايران
hosseini@um.ac.ir

الخلاصة:-

يشير إنترنت الأشياء (IoT) إلى الترابط بين عدد كبير جدا من الأجهزة غير المتجانسة وذات الموارد المحدودة والتي تستشعر وتجمع المعلومات عن بيئاتها. الطريقة التقليدية لحل مشكلة ندرة الموارد في انترنت الاشياء هو استخدام موارد بيئة الحوسبة السحابية. الأشياء ترسل الطلبات باستمرار إلى الحوسبة السحابية من خلال شبكة الانترنت. ولكن هذا ليس الحل الأمثل بسبب التأخير (Latency) وعرض النطاق الترددي المكلفة، لذا فإن الحل الأمثل هو الحوسبة الضبابية. الفكرة وراء الحوسبة الضبابية هي نقل الموارد إلى حافة الشبكة لتكون قريبة من أجهزة ال IoT. في هذه المقالة نقترح تقنية فعالة لإدارة الموارد بالاعتماد على قدرات الشبكات المعرفة بالبرمجيات (SDN) لتحسين جودة الخدمة للـ IoT من خلال استغلال التعاون بين الحوسبة الضبابية والحوسبة السحابية. نقترح معمارية من مجموعات من ألاجهزة الضبابية التي تسيطر عليها وحدات التحكم الـ SDN الموزعة. وبالإضافة إلى ذلك تحتوي المعمارية المقترحة على وحدة تحكم SDN مركزية حيث تتصل بجميع وحدات التحكم SDN الموزعة وبجميع الخوادم السحابية. بحيث يحتوي على رؤية شاملة عن الشبكة. هذا البحث يدرس العديد من القضايا: جدولة المهام، تخفيف الحمل، اكتشاف الموارد واختيار الموارد لتقليل زمن الاستجابة وضمان تنفيذ جميع مهام الوقت الحالي الصارمة في المواعيد النهائية، وتقديم أفضل جهد لتنفيذ مهام الوقت الحالي المرنة لتقليل الغرامة.