Load Balancing of Fog Computing Centers: Minimizing Response Time of High Priority Requests

Dilip Rathod, Girish Chowdhary

Abstract*: *Fog computing is one of the enabling computing technology which primarily aims to fulfill the requirements of the Internet of Things (IoT). IoT is fast-growing networking and computing sector. The scalability of users, devices, and application is crucial for the success of IoT systems. The load balancing is an approach to distribute the load among computing nodes so that the computing nodes are not overloaded. In this paper, we propose the priority-based request servicing at fog computing centers. We particularly address the situation when the fog node in fog computing center (FCC) receives more workload than their capacity to handle it. The increased workload is shifted to nearby fog nodes rather than to the remote cloud. The proposed approach is able to minimize the offloading the high priority request to other nodes by 11% which proves the novelty of our proposed.

Keywords* : *Internet of Things, Middleware, Scalability, Load balancing,

I. INTRODUCTION

The Internet of things (IoT) is one of the prominent technology in computer science in this decade [1]. The emergence of IoT has led to changes in many existing Internet protocols (e.g. Web sockets), the emergence of new protocols (e.g. CoAP), new computing paradigm (e.g. fog computing), many more changes in vertical and horizontal aspects of networking as well as computing field. The IoT has yet to bloom in its full potential. IoT also has stimulated a lot of research and development in the area of computer network. The horizon of computing has reached to every kind of objects we come across in our daily life such as home, office, farm, city, energy grid, forest, etc. making each of them smart (the first-class citizen of the Internet).
From software engineering perspective following are requirements of IoT based systems: interoperability, security, scalability, extensibility, data management. For the smooth growth of IoT based systems/application, the researchers/designers must address these issues. The middleware is the well-known answer to these requirements of Internet of Things [2]. Middlewares has been used in many fields of distributed computing, and in a variety of ways since they were introduced e.g. Java RMI, CORBA. Middlware in IoT context is layer of software sits between edge IoT network and end IoT Applications. In this paper, we extend our work on the scalability requirement of IoT Middleware [3]. We particularly address the situation when the fog node in fog computing center (FCC) receives more workload than their capacity to handle it. Based on the growth rate we are witnessing IoT [4]; this will be a common scenario. The increased workload is shifted to nearby fog nodes rather than to the remote cloud. The processing of requests near the edge of IoT network helps in avoiding the communication delay with remote cloud data center. Our contribution in this paper is to use priority queueing model for processing of incoming service request. The high priority (latency sensitive) requests are processed first followed by low priority (delay tolerant) requests. The results of implementation prove the novelty of our proposed approach which is able to minimize the offloading the high priority request to other nodes
The paper is organized as follows. The existing work on load balancing among fog nodes is briefly discussed in section 2. Section 3 describes the hierarchical architecture of IoT which includes sensors, fog computing nodes, and cloud data center as main components. Our contribution using priority queueing is detailed in section 4. The implementation details and results obtained are given section and section 6 concludes the paper.

II. RELATED WORK

The existing work on load balancing mainly considers shifting the computational burden to the cloud data center. The issue arises in when fog node is unable to handle the latency-sensitive request and need to offload it to cloud. This will cause to SLA violation. There are only a few works [5] [6] [7] [8] which addresses the issue of offloading latency-sensitive request to nearby fog center which helps in minimizing response introduced because of communication delay with the remote cloud. Authors in [5] uses random walk techniques for horizontal load balancing. Two types of workload is considered in [6] namely: lightweight and heavyweight. The response time is calculated based on it. Game theoretical approach to load balancing is applied in [7]. The load balancing among fog nodes is mainly used for minimizing energy consumption and mobility support in [8]. Our work differs

Manuscript published on 30 September 2019.
***Correspondence Author(s)**
Dilip Rathod*, Research Scholar, Dept. of Computer Science and Engineering, Research Center: Shri Guru Gobind Singhji Institute of Engineering and Technology, Under SRTM University, Nanded, Maharashtra, India. Email: rathod.dt@gmail.com
Girish Chowdhary, Professor and Director, School of Computational Science, Swami Ramanand Teerth Marathwada (SRTM) University, Nanded, Maharashtra, India. Email:girish.chowdhary@gmail.com

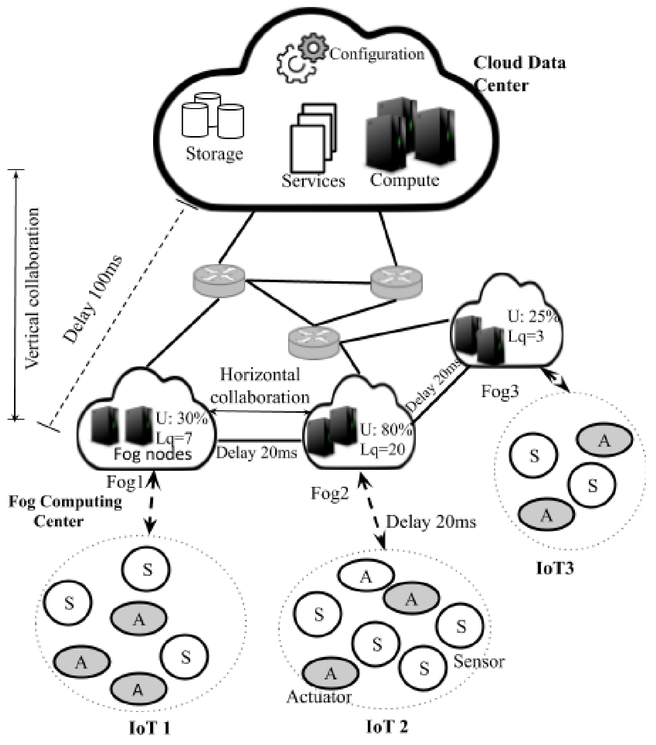

Fig 1: Fog computing based IoT architecture

with the existing work, we not only try to load balance the increased workload at certain fog node by offloading it to nearby fog nodes but also process every incoming service request based on its priority and load type. Our proposed sensitive to load of fog nodes as well as priority of IoT service request.

III. FOG COMPUTING MODEL

The architecture of cloud fog based IoT middleware is as shown in Fig. 1. There are three layers namely: IoT layer, fog computing layer and cloud layer. IoT device generates data/observations. IoT device includes sensors and actuators. These devices are resource-constrained in terms of computation, memory, networking capability and battery operated. The fog computing layer does intermediate processing and also fulfill the latency-sensitive requests. Fog computing centers consists of one or more fog nodes which have high computing capacity compared to IoT devices and are geographically located near the edge of the IoT network. The cloud layer process computationally intensive jobs do data analytics and storage. All the value added services the IoT user may demand are provisioned by cloud such as analyzing trends and pattern in data, predictive analytics, understanding data and improve the system operation. The machine learning is increasingly applied on IoT data. The machine learning jobs are computationally intensive and are carried out in cloud data center.

Our idea in this paper is when any of the fog nodes become overloaded due to increased arrival rate. The arrival rate is greater than capacity of fog node to handle it. The excess load can be shared with nearby fog nodes rather than offloading requests to the remote cloud. Due to geographical proximity, transmission delay between fog nodes is very less compared to cloud.

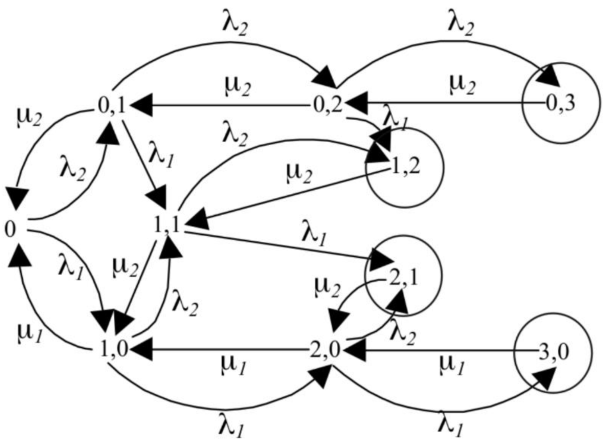

Fig. 2: State transition diagram priority queue

IV. LOAD BALACING OF FOG COMPUTING CENTERS

A. Priority Queue

Queueing theory is a widely used technique for modeling and analyzing the performance of distributed systems [8]. The fog nodes process the data generated by sensors. The arrival rate of data (service request) follows Poisson distribution. The service rate of fog as well as cloud nodes are exponentially distributed. When service request arrive fog node start processing it. If service arrive and the fog node is busy, the request is put into queue. The requests in the queue are processed in certain order based on demand of application. IoT application are diverse in nature. The processing the request on first come first serve (FCFS) order may lead to violation of latency demanded by application. We assign a priority to every request as specified in Service Level Agreement (SLA). In this paper, we pick the service request from queue based on its priority. Four types of requests are considered namely: high priority lightweight request (based on the size of payload), high priority heavyweight, low priority lightweight and low priority heavyweight requests. The state transition diagram for M/M/1/3 priority queue is as shown in Fig. 2. There is one server and queue capacity is a maximum of 3 requests. The balance equations for all the states are given below.

$$P_0(\lambda_1 + \lambda_2) = P_{0,1}\,\mu_2 + P_{1,0}\,\mu_1 \quad (1)$$
$$P_{0,1}(\lambda_1 + \lambda_2 + \mu_2) = P_{0,2}\,\mu_2 + P_0\,\lambda_2 \quad (2)$$
$$P_{1,0}(\lambda_1 + \lambda_2 + \mu_1) = P_{0,2}\,\mu_1 + P_{1,1}\,\mu_2 + P_0\,\lambda_1 \quad (3)$$
$$P_{1,1}(\lambda_1 + \lambda_2 + \mu_2) = P_{1,0}\,\lambda_2 + P_{1,2}\,\mu_2 + P_{0,1}\,\lambda_1 \quad (4)$$
$$P_{1,2}\mu_2 = P_{1,1}\,\lambda_2 + P_{0,2}\,\lambda_1 \quad (5)$$
$$P_{2,1}\mu_2 = P_{1,1}\,\lambda_1 + P_{2,0}\,\lambda_2 \quad (6)$$
$$P_{0,2}(\lambda_1 + \lambda_2 + \mu_2) = P_{0,1}\,\lambda_2 + P_{0,3}\,\mu_2 \quad (7)$$
$$P_{2,0}(\lambda_1 + \lambda_2 + \mu_1) = P_{1,0}\,\lambda_1 + P_{2,1}\,\mu_2 + P_{3,0}\,\mu_1 \quad (8)$$
$$P_{0,3}\mu_2 = P_{0,2}\,\lambda_2 \quad (9)$$
$$P_{3,0}\mu_1 = P_{2,0}\,\lambda_1 \quad (10)$$

The incoming service request will find the server is busy. The request has to wait in queue for service is given below.

$$P_B = P_{1,2} + P_{2,1} + P_{3,0} + P_{0,3} \quad (11)$$
$$P_{high\_priority} = P_{0,1} + P_{1,1} + P_{2,1} \quad (12)$$

$$P_{low\_priority} = P_{1,0} + P_{2,0} + P_{3,0} \quad (13)$$

$$P_0 = P_b + P_{m\_deadline} \quad (14)$$

P_o is probability of offloading request, $P_{m\_deadline}$ is probability of missing deadline.

For M/M/1/3 priority queue when the number of request waiting for service in the queue of either type or both type is 3. The queue is full. The requests arriving request any further will need offloading.

The end-to-end latency is the sum of communication latency, service time and queueing delay as given below.

$$T_{total} = T_s + T_q + T_{comm} \quad (15)$$

B. Load balancing

Processing as well as arranging the request waiting for service in the queue for servicing based on its priority give an edge over FCFS processing of the request. The high priority requests are processed first, and arranged in the front part of the queue. This leads to the amount of time high priority request has to wait in queue is minimized. Also, the probability of high request offloaded because the queue is full is reduced. The existing work on load balancing [5], [6], [7], and [8] does not consider the priority of request for processing it.

Our idea about load balancing is as shown in Fig. 3. Load balancing is the process of distributing load almost uniformly among all the nodes managed by load balancing unit. We use fog controller for load balancing. The load balancing is achieved among autonomously function fog nodes. The fog nodes may be heterogeneous in their in terms of computing capability, memory size, etc. The fog controller (FC) maintains the table of all fog nodes involved in load balancing along with their processing capacity, number of servers, queue length, and utilization. The fog nodes send their status at every interval called update interval, to fog controller. There are two fog controllers primary and secondary. The main task of load balancing is done by the primary fog controller (PFC). The secondary fog controller (SFC) acts as a backup and will be used if PFC fails. Use of fog controller helps in the number of message update message exchanged compared to a distributed approach.

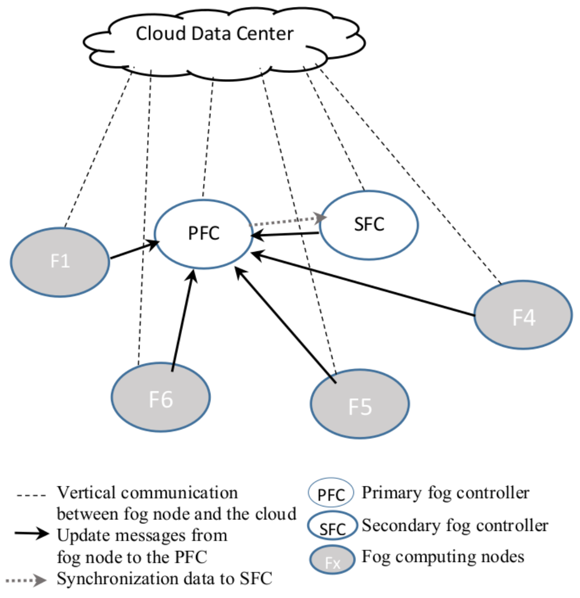
Cloud Data Center
PFC
SFC
F1
F4
F5
F6
Vertical communication between fog node and the cloud
Update messages from fog node to the PFC
Synchronization data to SFC
PFC Primary fog controller
SFC Secondary fog controller
Fx Fog computing nodes

Fig. 3: Collaborative Load balancing among fog nodes

Thus reliving individual fog nodes from the task of load balancing. Thus helps to reduce the computational complexity of the algorithm. The steps followed by individual fog node and fog controller are given in following subsection.

C. Algorithm

Part I: Fog Node

Step 1: Register with fog controller using their fog id, service arrival rate, processing capacity, number of servers, queue length and utilization.

Step 2: At every update, interval sends update message containing arrival rate queue length and utilization.

Step 3: If the utilization is above threshold or queue is full, set Offloading flag to TRUE.

Step 4: If offloading is required to get the list of lightly loaded fog nodes from PFC

Step 5: If offloading list is non-empty the excess arriving request will be forwarded to lightly loaded nodes otherwise to cloud.

Step 6: Set Offloading flag to false.

Part II: Fog Controller

Step 1: Create Fog_Table containing fields fog id, service arrival rate, processing capacity, number of servers, queue length and utilization.

Step 2: Maintain fog table using update message received from every fog node

Step 3: Synchronize the fog table with SFC.

Step 4: Upon receiving an offloading request from fog node return the list of node suitable for handling the request offloaded request.

Step 5: At every update interval, prepare a list of lightly loaded nodes sorted according to processing capacity, queue length, and utilization.

Step 6: Return the list of lightly loaded nodes

V. Evaluation and Results

We have implemented the algorithm for load balancing algorithm given section 4. The libraries compatible with Python 3.7 on Ubuntu 18.04 and Dell Inspirion 15 notebook with 8GB RAM and i5 processors. The details of the simulation parameter used are given in table 1. The number of fog nodes was varied from 3 to 11 each with different configuration representing the heterogeneity of IoT environment. The arrival rate (800-1500 req/sec), service rate (1500-2000 req/sec) was varied randomly in multiple of 100.The queue capacity was varied randomly between 50 -100 in multiple of 10. The configuration particularly includes the arrival rate of request and service time arrival rate and the number of server in fog center. The service priority 1-4 for four types of packets were considered. The packets were generated following Poisson distribution. The packet with payload less

Table 1: Simulation parameters

Parameter	Value
No. of fog nodes	3-11
No. of controller node	2
Arrival rate	8-15 req/ms
Service rate	12-20 req/ms
Queue size	20
Request size	Lightly packets, Heavy packets
Priority	1 – delay-sensitive light packets 2 – delay-sensitive heavy packets 3 – delay-tolerant light packet 4 – delay-tolerant heavy packet
Delay	IoT device to fog node - 2 ms Fog node to fog node - 20 ms Fog node to cloud - 100 ms

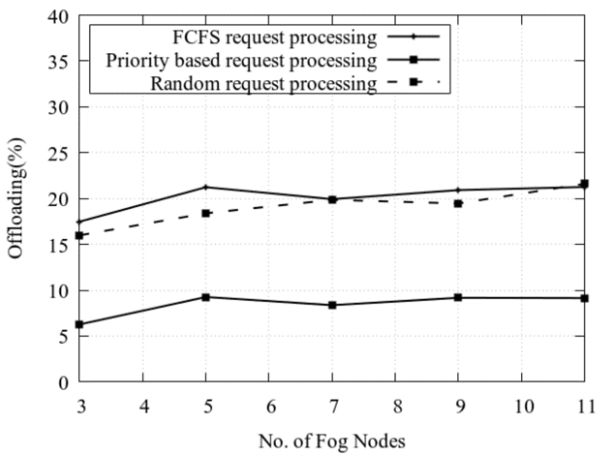

Fig. 4: Offloading of high priority request to neighboring fog nodes

than 1KB was lightweight otherwise classified as heavyweight packet. The result of the simulation is as given in Fig. 4. The experiment was run for 300 iterations and the average of 10 runs is plotted in Fig. 4 for three types of request processing considered: First Come First Serve (FCFC), priority-based and random processing. It is observed that in case of the priority-based processing archives 11% of reduction in offloading of high priority request which is a considerable improvement. Hence our proposed method of load balancing is capable of serving high priority request more effectively as well as reducing the computational complexity of the algorithm. Also, sensitive to load of fog nodes as well as priority of IoT service request.

VI. CONCLUSION

The IoT systems are continuously developed for diverse applications. Each application has its demand for regarding how the request should be fulfilled. The traditional QoS mechanism will not apply to IoT services. In this paper, we attempt to address the demand for latency-sensitive IoT service requests. The requests with higher priority were processed first compared to requests with lower priority. Based on results of simulation it is clear that our proposed approach can minimize the amount of high priority request offloaded to nearby fog nodes. This further helps in minimizing response time compared to FCFS and random processing of requests. Also, proposed algorithm is sensitive to load of fog nodes as well as priority of IoT service request. Our possible future work include modelling and processing the latency-sensitive service request in real-time.

REFERENCES

1. "Top trends in the gartner hype cycle for emerging technologies, 2017," Aug2017. [Online Available]: https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/, accessed on August 20, 2019.
2. D. Rathod, G. Chowdhary, "Survey of middlewares for Internet of things", Proc. IEEE Int. Conf. on Recent Trends in Advanced Computing: CPS, Chennai, India, Sep. 10-11, 2018, pp.129-135.
3. D. Rathod, G. Chowdhary, "Scalability of M/M/c queue based cloud-fog distributed Internets of Things middleware", Int. J. Advanced Networking and Applications, UGC approved, ISSN: 0975-0290, 2019, vol.11(1), pp.4162-4170.
4. D. Evans, "The Internet of Things: how the next evolution of the Internet is changing everything", white paper, Cisco, 2011.
5. C. Fricker, F. Guillemin, P. Robert, G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing". ACM Trans. Model. Perform. Eval. 2016. vol.1(4) pp. 16:1-18.
6. M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, Y. Jararweh, "Improving fog computing performance via Fog-2-Fog collaboration", Future Generation Computer Systems, 2019. (to be published)
7. H. Xiao, Z. Zhang, Z. Zhou, GWS—A Collaborative load-balancing algorithm for Internet of Things", Journal of Sensors, 2018, vol.18(8), 2479.
8. W. Zhang, Z. Zhang, H. Chao, "Cooperative fog computing for dealing with Big Data in the Internet of Vehicles: architecture and hierarchical resource management", IEEE Communications Magazine, December 2017.
9. M. Harchol-Balter, Performance modeling and design of computer systems:queueing theory in action, Cambridge University Press, 2013.

AUTHORS PROFILE

Dilip Rathod is Research Scholar at Department of Computer Science and Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, Maharashtra, India. He has completed ME(CSE) in 2009 from Government College of Engineering, Aurangabad, Maharashtra, India. He received Bachelor of Engineering in Information Technology degree from Dr. Babasaheb Marathwada University in 2004 Maharashtra, India. Since 2005, he is working at P. E. S. College of Engineering, Aurangabad. Currently designated as Assistant Professor in Senior Scale at Computer Science and Engineering department. His membership of professional bodies include Computer Society of India(CSI), Institution of Electronics and Telecommunication Engineers (IETE, Life Member), and Indian Society for Technical Engineers(ISTE, Life member). Published 12 papers in national and International conferences. His area of Interest include IoT, Computer Network, and Cloud computing.

Dr. Girish V. Chowdahry is Professor and Director at School of Computational Science, Swami Ramanand Teerth Marathwada (SRTM) University, Nanded, Maharashtra, India. He has completed is Ph.D. from Indian Institute of Technology, Madras, Tamil Nadu, India. He received his undergraduate degree from Shri Guru Gobind Singhji Institute of Engineering and Technology, Under SRTM University, Nanded. He worked as Research Dean, for Engineering and Technology Section at SRTM Nanded, Maharashtra, India. His area of research includes Internet of Things, Wireless Sensor network and Optical WDM networks, pattern recognition. He has published more than 20 research papers in International journals and conferences.