Cooperative Fog Communications using A Multi-Level Load Balancing

Nour Mostafa
College of Engineering and Technology
American University of the Middle East (AUM),
Egaila, Kuwait
Email: {Nour.Moustafa}@aum.edu.kw

***Abstract*— Fog Computing has been introduced as an emergence technology in Internet of Things (IoT). Fog Computing is introduced as a solution for many enterprises, which include computation, storage, and data exchange capabilities of edge, devises. Users, resources, and data are steadily increasing in number, making scalability and extensibility important issues. With the emergence of the Cloud/Fog it is expected that run time estimates will also be used for resource selection, workflow management, load balancing, and job monitoring. User preferences is the core elements of the fog/cloud service provider, as the idea is to learn, predict, optimize etc. Predictions are a very important step towards automatic resource management. This paper proposes a cooperative fogs system, which consider the user preferences e.g. delay, cost, and privacy and find the optimal choice that meet the user preferences. In addition to ensuring efficient utilization of resources to balance the load in fog computing.**

***Keywords*—cloud, fog, fog-to-cloud, fog-to-fog, IoT, e2e delay**

I. INTRODUCTION

Fog computing systems are growing in complexity. They are having to handle ever-greater numbers of users, resources and task requests. The number of tasks produced by users and used in fog systems are also increasing. Delays in tasks execution and longer task execution times are a consequence of systems having to search for best resource in response to individual user requests.

While the volume of requests that needs to be handled is increasing, ensuring efficient selection and access to such huge and widely distributed resources is a serious challenge to network and fog designers. In order to tackle the problem of scalability, we propose a cooperative resources selection service for fog systems that will meet a user's preferences in the shortest response time. The proposed model presents a predictive component to determine/predict which resource is the best to fulfill the user's task.

It can be argued the problems associated with how fog resources are selected are not well serviced by current approaches. Finding the best suitable fog/cloud resources for job execution is difficult due to the increased complexity of fog systems. In this paper, we provide an extension to the work introduced in [1], where a fog resource selection algorithm (FResS) is integrated within the IoT framework. The resource selector is used to select the best host/resource by taking care of all user and job requirements. To enable fog selection in advance to execute the task, it requires some form of assistance to avoid incurring overhead when executing the task on resource-limited fogs. The FResS approach was proposed to make a rum time prediction, which estimate the time and number of needed resources to complete a new task. The proposed solution provides an accurate prediction model that are usable at a user level, resulting in a decrease of the overall e2e latency of the system.

When a new task is generated by the user, the proposed system will select a preliminary fog to connect to it based on location. Prediction is one widely accepted solution in distributed environments. By storing the users' log file in a task manager database and sharing it with the cloud (i.e. for learning), for new request, the proposed system will try to find the best solution (predict) the best resource if the prediction fails, a system can operate using the preliminary fog, consequently availability and fault tolerance will be improved. At the same time, sharing the users' log files among multiple sites will led to increasing possibility to find the required resources close to the site where the request originated, thus improving the response time, server request latency, and improving extensibility of the overall system[2].

The rest of the paper is organized as follows: Section II provides a summary of the related work in the literature. Section III introduces the fog prediction and selection scheme. Section IV introduces the multi-level load balancing technique. Section V provides some of the data gathered from simulation results. Finally, Section VI concludes the paper and provides further research directions for future work.

II. RELATED WORK

With the emergence of the cloud it is expected that run time estimates will also be used for resource selection, workflow management, load balancing, and job monitoring. Predictions are a very important step towards automatic resource management.

Studies on fog computing are still premature, some work have proposed fog to cloud computing architecture. The proposed architecture in [4] splits the service into slices to enable parallel execution. The desired services will be distributed/matched among the available Fog 2 Cloud (F2C) resources to achieve low delay on service allocation. The author

did not take into consideration the resource capabilities (processing power, speed, etc.).

In [5] (F2C) architecture was proposed which introduced a new layers of cloud by splitting the cloud and fog architecture into layers. In this paper, a proposed management system was introduced to choose the best fog that match the service's requirements based on available resources. The proposed system assumes that the service can be divided into small sub-services then execute them on different fog layers. However, the proposed approach did not show or explain how the service will be split into sub-services, and what are the criteria that the management system is using to choose the best fog.

In [6] a graph partitioning theory has been proposed to construct a system model of fog computing that achieve load balancing among fogs by combining the graph theory and characteristics of fog computing. A dynamic load balancing was achieved by considering a mechanism of graph repartitioning. The author in [7] presented an optimal workload allocation in Fog-Cloud computing systems. The author investigated the power consumption-delay tradeoff issue in F2C scenarios by developing a systematic framework. The problem investigated in this paper focused on solutions for three approximate sub-problems , namely, 1) convex optimization techniques was used to solve the power consumption-delay tradeoff for fog 2) a mixed integer non-linear programming problem was used to solve the power consumption-delay tradeoff for cloud, and 3) using the Hungarian method in polynomial time to minimize communication delay [8].

Although there exists research in solving issues related to cooperative and sharing resources in fog computing is still premature. As mentioned earlier, our work considers a solution to resource selection in fog computing based on users' preference and previous results of successful resource selection runs.

III. Fog-Prediction and Selection Scheme

In our paper [3] we proposed a resource selection service based on a run-time predictions within fog environments. There are two parts in the proposed solution: i) the fog execution time prediction model, and ii) the fog resource selection model called Fog Resource Selection (FResS) in the fog layer to manage requests between IoT devices and fog resources. Figure 1 shows how the FResS Module incorporates within the F2C architecture which consists of three modules a Task Scheduler, Resource Selector and Task Manager. Each Module is performing a different activity, these activities reflect the data and/or control flow dependencies between fog components.

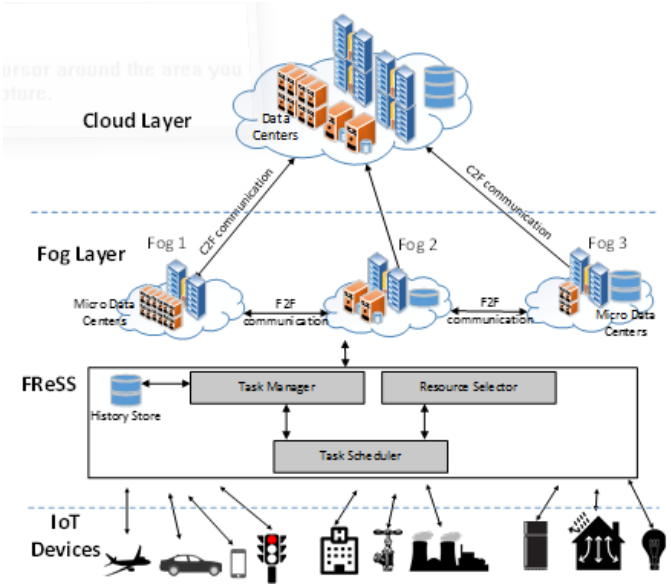

Fig.1 Incorporating the Fog Resource Selection (FResS) Modules within the F2C architecture.

A. Task Scheduler

The Task Scheduler perform the task of mapping the tasks between IoT devices and the Fog layer by mapping the task onto the best available resources to meet certain objectives functions. Objective functions are user defined constraints which can be in terms of time, cost, or some other quality of Services (QoS) metrics.

The Task Manager receives the task description from the Task Scheduler to find similar tasks in the execution history to obtain run times and the required resources for executing the task. Task execution history has been a very useful tool in predicting many aspects of resource performance, e.g. run time, queue waiting times, and data transfer delays. After finding a similar task, the next step is to provide a run-time prediction of the resource requirements and execution time for incoming tasks. The Resource Selector performs fog selection by making a run time prediction that select the fog(s) that are most suitable to complete the task in addition to the required resources.

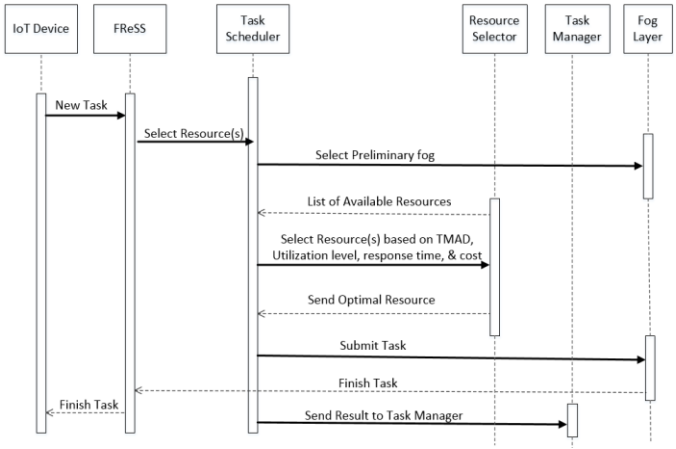

Fig.2 Sequence diagram showing interaction between different components of FReSS for new user/device.

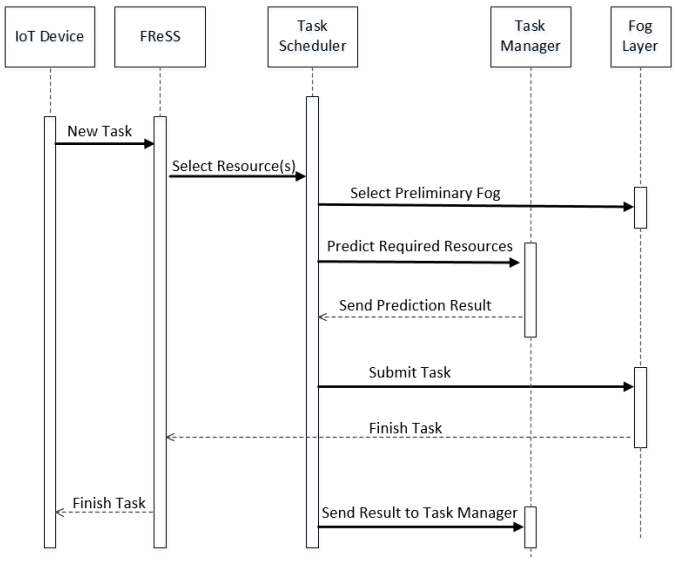

Fig.3 Sequence diagram showing interaction between different components of FReSS for existing user/device.

The Task Scheduler Module receives a list of selected resources along with a prediction of the execution time for each task from the Resource Selector depending on the configuration. After every execution, the result of the execution log are forwarded by the Task Scheduler to the Task Manager as an execution history and subsequently used to make predictions for future tasks. A sequence diagram shown in Figure 2 and 3 explains the interaction between the proposed modules by showcasing the sequence event during a typical task submission process.

The Resource Selector module is a multi-step process. The first step is to discover the fog resources then compare them against the minimum QoS criteria, e.g. availability, delay, memory, privacy, and cost. If there are no deadline constraints, then all the selected resources are entitled to execute the current task. The next step is to evaluates resources predicted execution time in terms of their ability to meet deadline. The Task Scheduler will utilize these predictions for optimum task placement. Additional constraints have been considered such as cost, workflow constraints, and user preferences to achieve optimum resource selections. The Resource Selector send a ranked list of resources along with predictions to the Task Scheduler which in turn evaluates these constraints. In the second Step, the Task Scheduler determine the required duration of advanced resource reservations based on the availability of predictions, which play a key role by providing the possible outcomes of a certain resource selection.

B. Task Manager Module

The Task Scheduler Module forwards the input task description to the Task Manager Module to generate run-time predictions, this Module compare the historical execution logs with the new submitted task to find similar tasks. This process is conducted using Artificial Neural Networks (ANNs) [9]. The predictions about the run times are made on all available resources Once similar tasks are found. The finished list is forwarded to the Resource Selector Module.

ANNs are suitable for training over hundreds or even thousands of passes of data sets [10][11]. Fog/cloud computing deals with such very large number of users, resources, and data which are steadily increasing in number, making scalability and extensibility important issues. ANNs generally have at least three layers models these layers are referred to as input, middle (or "hidden"), and output [10]. The starting point where the data enters the system is the input layer. The hidden layer is the intermediate processing unit at which the input data is passed to for processing, and then passes the new signal onto the output layer.

The key element of processing information in neural network is its interconnection weights. Weights express the relative strength of the input data or the various connection that transfer data from layer to layer. The network learns through repeated adjustment of weights, and how these weights are adjusted during the learning process is known as the learning algorithm.

The operation of a network consists of two components, the learning and recall phases. The learning phase involves modifying the weights at the input layer in order to get accurate results by passing input values to the input layer and measuring how close the network's predictions are to the training sets. The next figure shows a single unit network.

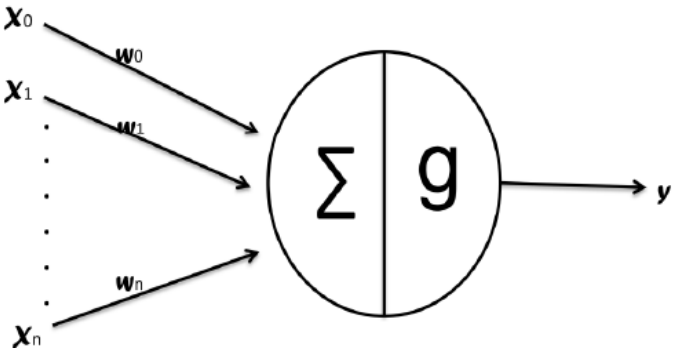

Fig.4 An Artificial Neuron Unit [12].

The weight adjustment is achieved by changing an amount proportional to the difference between the desired output and the actual output [12]. The general mathematic definition is as shown equation [13]:

$$y(x) = g(\sum_{i=0}^{n}(w_i x_i)) \quad (1)$$

x represents the input neuron with (n+1) input dendrites ($x0$ ….. xn) and the output is computed by the axon $y(x)$. The weights are initialised with (w 0….. wn) to determine how much the inputs should be weighted. g represents an activation function that uses the sum of the input to weight the strength of the output and how powerful it should be from the neuron.

The goal is to design and train the network using the training set to produce trainable weights where the output of the network matches the desired output [14].

The historical execution logs will be examined to find similar task by the Task Manager, once the execution time predictions are generated for all resources in the list. The Resource Selection Module will receive the complete list. Algorithm 1 forms the core of FResS, which is used to determine the required resources needed for a task to execute. There are five parameters used to identify the resources: delay, price, previous execution time, memory requested, and submission time. It is noted that many users of fog networks tend to do same tasks using the same data [10][15]. Therefore, an opportunity was created to use parameters from previous execution to predict future behavior of users in terms of required resources of the new task from execution logs.

The results of every execution are stored as an execution history log file and subsequently used by ANN to make predictions for future tasks. The accuracy of predictions will increases as a result of the increasing number of the historical executions [16][17]. Prediction is made using the proposed Task Manager module to determine the required and the result is returned to Task Scheduler.

The Task Manager generates estimates for the required resources of the task with sufficient accuracy. Resources Selector uses these predicted results to determine the optimal resources to execute the task. The proposed Modules have addressed the performance issues related to tasks submission of IoT devices that communicate directly (or randomly) with the remote fog/cloud for which incurs high delays and a network overload.

The prediction results are kept up to date by FResS by adding new prediction results and removing old ones, if a similar task in the history logs matching the new submitted task is found, the result will be sent to the Task Scheduler. If not, the preliminary fog will take place to execute the task.

The feedback from past history executions is not supported by current solutions used in fog computing which treats each task as a new task. On the other hand, the proposed model uses a prediction mechanism to inform the system about incoming tasks, and the database is updated thereby ensuring that an accurate, and up to date, view of resources performance is maintained. Consequently, a dynamic state information is provided by the proposed model database which help provide more accurate and efficient predictions.

C. Resource Selector Module

The Resource Selector Module is responsible for resource selection of the most suitable resources for the incoming task. After a new task arrives, it is forwarded to the Resource Selector Module by the Task Scheduler Module. The Resource Selector performs resource selection by making run time predictions. After resource selection is done, the Resource Selector returns a list of the selected resource or resources depending on the configuration, along with their run time predictions, to the Task Scheduler Module. After the task has finished its execution, the Task Scheduler Module forwards the execution log to the Task Manager for storage.

IV. A MULTI-LEVEL LOAD BALANCING TECHNIQUE IN FOG COMPUTING

Internet traffic is expected to grow 50 times by 2020 [19], it is very essential for the service provider to facilitate higher availability of quality services. In IoT and fog computing, the big data and number of services explosion are resulting in more load all across the network. The goal of balancing the load is to increase the response time and accomplish better utilization ratio and QoS, in addition to reducing the number of task rejections. The dynamic nature of the fog environment and the nature of the tasks submitted by users may cause under-utilization or over-utilization. To overcome this issue, a multi-level load balancing approach has been proposed in this section. The below algorithm will be used when user submit a task while considering user preferences. If the below metrics and parameters are satisfied optimally then it will improve the performance of the fog system.

Algorithm 1: Task Execution Resource Prediction
Data: User Preferences and Task log history
Input: New task submission description file
Result: Required resource(s) predictions for incoming task

```
Start
While Taskset <> empty
do
Process request for Task(s) ki;
    if (ki submitted by new user/device) then
        Resource Selector send available resources to Task Scheduler;
        Task Scheduler send user preferences to Task Manager;
        if (Task Manager found optimal resource(s))
            send back optimal resource(s) Task Scheduler;
            execute ki on the optimal resource.
    else if (ki submitted by existing user/device) then
            NN predict required resources For ki
            send back predicted resource(s) To execute ki;
            execute ki on the predicted resource.
    else
        Execute ki on the preliminary fog;

    update Task Manager log history.
    select next task;
    end;
end;
```

A. Metrics

- Cost: the efficient algorithm tries always to minimize the cost (processing, storage or transfer cost) by scheduling the incoming task to a resource based on the cost.
- Response time: is the time taken by the service provider between sending tha task by the user and receiving of response. Its least value is desirable.

- Delay: this work adopt the tenant maximum acceptable delay (TMAD) [20], this is an important metric in terms of QoS, which is used to sort the users priorities according to their important for the system based on the subscription plan each user pay for which can lead to a higher priority task or lower priority task.
- Scalability: The fog computing system should be able to support scalability, avoiding bottlenecks, this metric determine if the system is able to accomplish load balancing with a restricted number of resources.
- Fault Tolerant: the ability of the system to perform correctly in case of failure by having a backup.

B. *Steps of proposed algorithm*

Step 1: when the Task Scheduler receives a task from the user, a preliminary fog will be selected to connect to it based on users' location. Task Scheduler receives the predicted list of fogs that can serve the submitted task from Resource selector.

Step 2: If the list of the selected resources has more than one resource, then Task Scheduler is responsible for finding the optimal resource to fulfill users' request based on utilization level. In addition, the resource selection will take into consideration the delay, which must be less than the (TMAD) of the current user.

Step 3: Store the result in a Task Manager in a log file database. This database will be kept up to date to have the latest analysis of the network load balance. In addition to the generating different QoS results i.e. delay, cost, privacy etc that meet the users' preference. Each workflow will have a different effect on the network load balance. Then, the predicted results will be shared with the cloud for learning.

Step 4: For new task, the Task Scheduler will try to find best solution (predict) the required resources without going through step 2 and 3. This approach consider the users' preference while assigning task to resource(s) which is more suitable for fog computing environment. Such algorithm will constantly monitor the resources and task progress and take decision based on that.

The proposed algorithm is designed to overcome the disadvantages of static and random resource selection by combining them with the historical resource selection. The traditional resource selection also suffers from the problem of uneven load distribution. Consequently, the severity of this problem is less because of the presence of a feedback loop as shown in Figure 5.

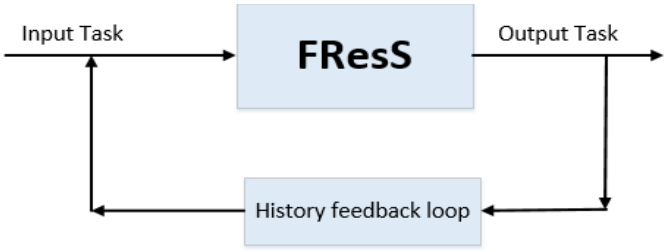

Fig.5 History as a feedback loop in the scheduling process.

V. Simulation Results

The evaluation of the FResS was carried out using GridSim [18], selected for its flexibility. A detailed comparison of the proposed solution with traditional cloud solution is carried out to highlight the differences and advancement. IoT devices have been simulated as geographically distributed sites. The storage sites in the fog were modeled as a set of nodes where a graph represent the network. A uniform bandwidth, computing power, memory and storage capacity have been configured to all nodes. The number of tasks, required files, and capacity of storage nodes were varied to simulate different scenarios. One cloud storage site, three fog storage sites, and 150 IoT devices were defined by the simulator. The total number of task requests is 1500. The connectivity of the bandwidth is up to 2000 MB/Sec.

To test our new approach against large number of tasks, we generated a prediction model using actual query data from GridSim and the resultant resource predictions from our ANN. A number of tasks were executed, first using the traditional model then the prediction FResS model. The time taken to execute tasks in both models was calculated.

This section explains the different times involved which can contribute to the total execution time of a task on the fog. The prediction model presented above will provide run time predictions, which provide estimates of the processing time. Tasks on the fog are executed in many stages, which start with the task submission and end with the results retrieval. These stages can be divided into four major components by ignoring lesser time delays and are given below.

1. Stage in: transfer of task(s) and executable to the destination node.
2. Waiting time: waiting in the batch queue.
3. Run time.
4. Stage out: transfer of completed task(s) to the requesting node.

Stage one, also called stage in, consists of transferring the data or code from the source node to the executing node. Once all the required data and code has been transferred to the destination node/nodes, the task is submitted to the local task manager, which in turns puts it in the waiting queue. Depending on the load on the executing node, the task will spend some time in the queue for its execution turn. This time is called queue wait time and is represented by stage two. At the end of the queue wait time, the task is assigned for execution which marks the start of the run time in stage three. Once the task completes its execution, results are returned to the originating node in stage four. Returning results to the originating nodes is called the stage out phase. The total turnaround time, which is also called total Time to Delivery (TDD), is given in Equation 1.

$$T_{TRT} = T_{SIT} + T_{QWT} + T_{RT} + T_{SOT} \qquad (2)$$

Where,

T_{TRT} = Total Run Time

T_{SIT} = Stage In Time

T_{QWT} = Queue Wait Time

T_{RT} = Run Time

T_{SOT} = Stage Out Time

Simulation runs were conducted to analyse the overall time performance by varying the number of tasks. Result outlined in figure 6 show that the adopted FResS model outperform the traditional model such that the tasks response time was decreased by 21% (as shown in table 1) for the overall time taken for task processing showing significant time savings.

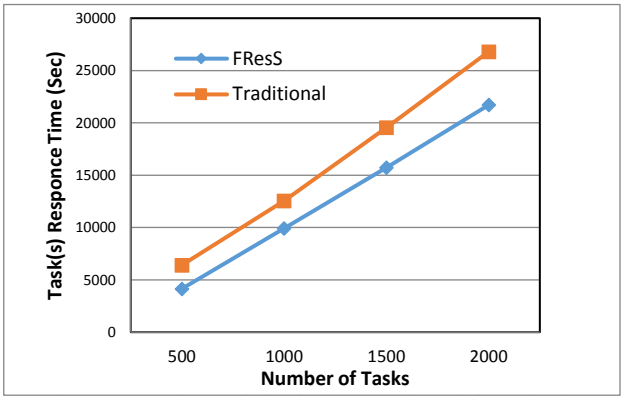

Fig.6 Comparative analysis of Response Time with no of tasks.

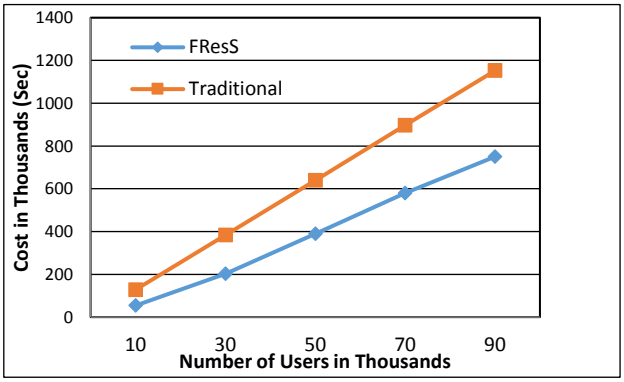

Fig.7 Comparative analysis of Cost with no of users.

Figure 7 shows that the cost increases with the increase number of users and the FResS outperform the traditional model showing a decrease in the total cost.

Table 1. Simulation results: Task Turnaround Time using FResS and Traditional model

No. of tasks	Task Turnaround Time using Traditional model (Sec)	Task Turnaround Time using FResS (Sec)	Difference (Sec)
500	6414	4150	2264
1000	12540	9920	2620
1500	19550	15740	3810
2000	26806	21710	5096
Total	65310	51520	13790
Average	16327	12880	21%

The network bandwidth is an important factor, which affects the task execution time, in the fog/cloud environment broader bandwidth can be provided between nodes within a region whereas bandwidth between nodes across region is relatively narrow. Since many nodes within a region try to connect to other nodes from other regions through a narrow bandwidth, this interregional link is highly congested with network traffic and it causes hierarchy of bandwidth. Therefore, it was important to carry out some experiments to show the tasks execution time for both the proposed model and traditional model with both narrow and broad bandwidth, a set of experiments were run to evaluate the performance of both models by varying the bandwidth between sites.

As shown in figure 8, when the experiments were run on a narrow bandwidth, the FResS outperforms the traditional model. However, the differences of task execution time decreased as broader bandwidth is set, though, the difference is still significant. Consequently, it can be concluded that FResS model can be effectively utilized and outperform the traditional model.

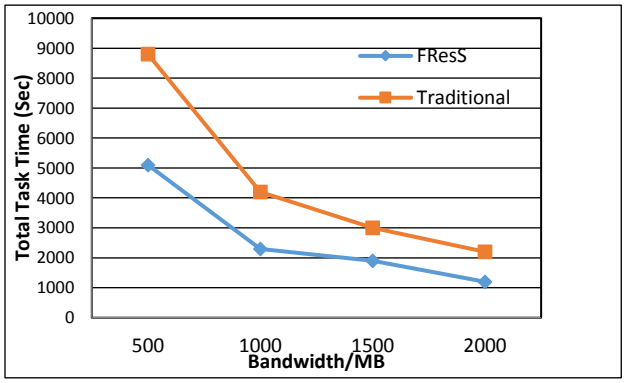

Fig.8 Total job time with varying bandwidth.

VI. Conclusion and Future Work

Cloud computing has emerged as a means of sharing computational resources and information services. Fog computing was introduced to overcome the shortcomings of the cloud computing scheme within the IoT environment by bringing the resources closer to the devices where the request originated, thus increasing the performance of the system, reducing bandwidth consumption, and improving scalability of the overall system. Fog systems are growing in complexity, they are having to handle ever-greater numbers of users, resources and tasks requests. Therefore, current approaches, however, often exhibit an increase in response time.

This paper proposed a cooperative fog communications using a multi-level load balancing in corporation with FResS. The proposed prediction model uses past history to inform the system about incoming tasks, and the database is updated constantly with new predictions generated by the ANN. The resulting prediction model is simple, has low overheads, and provides very high accuracy. The experiments showed that the proposed approach could distribute the load based on the user preferences. The resources utilization was increased by decreasing the cost, response time, and bandwidth usage. In the

future work more metrics will be added to improve the algorithm, in addition to improving the load balancing on the cloud layer by distributing the proposed model on the cloud layer.

REFERENCES

[1] W. Masri, I. Al Ridhawi, N. Moustafa, P. Pourghomi, "Minimizing Delay in IoT Systems Through Collaborative Fog-to-Fog (F2F) Communication," the 9th International Conference on Ubiquitous and Future Networks (ICUFN 2017).

[2] G. Sushant and R. Buyya, "Data Replication Strategies in Wide Area Distributed Systems", Enterprise Service Computing: From Concept to Deployment, Robin G. Qiu (ed), pp. 211-241, ISBN 1-599044181-2, Idea Group Inc.,Hershey, PA, USA, 2006.

[3] N. Mostafa, I. Al Ridhawi, M. Aloqaily, "Fog Resource Selection using Historical Executions" IEEE Third International Conference on Fog and Mobile Edge Computing (FMEC 2018).

[4] V.Barbosa, X.Masip-Bruin, E.Marín-Tordera, W.Ramírez, S.Sánchez, "Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios", in IEEE GLOBECOM 2016, Washington, USA, December 2016.

[5] X.Masip-Bruin, E.Marín-Tordera, G.Tashakor, A.Jukan, G.Ren: Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. IEEE Wireless Commun. 23(5): 120-128 (2016).

[6] S. Ningning, G. Chao, A. Xingshuo and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," in China Communications, vol. 13, no. 3, pp. 156-164, March 2016.

[7] R. Deng, R. Lu, C. Lai, T. H. Luan and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," in IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171-1181, Dec. 2016.

[8] H. W. Kuhn, "The Hungarian method for the assignment problem," NavalRes. Logist., vol. 2, nos. 1-2, pp. 83-97, 1955.

[9] S. Yashpal, S. Alok, "NEURAL NETWORKS IN DATA MINING", Journal of Theoretical and Applied Information Technology, 2005 - 2009 JATIT.

[10] C. Krieger, "Neural Networks in Data Mining", technician report, 1996.

[11] S. Duggal, R. Chhabra , "Learning Systems and Their Applications: Future of Strategic Expert System". Issues in Information Systems, Vol. III, 2002.

[12] G. jha, "ARTIFICIAL NEURAL NETWORKS," International journal of computer science and issues, Indian Research Institute, PUSA, New Delhi, (2005).

[13] S. Nissen, "Implementation of a fast artificial neural network library (FANN)," Technical report, Department of Computer Science University of Copenhagen (DIKU), (2003), http://fann.sf.net.

[14] D. Shanthi, G. Sahoo and N. Saravanan, "Designing an Artificial Neural Network Model for the Prediction of Thromboembolic Stroke," International Journals of Biometric and Bioinformatics (IJBB), Volume (3).

[15] S. Chen, R. Lu, J. Zhang, "An Efficient Fog-Assisted Unstable Sensor Detection Scheme with Privacy Preserved", CoRR abs/1711.10190 (2017).

[16] I. Rao and E. Huh, "A probabilistic and adaptive scheduling algorithm using system- generated predictions for inter-grid resource sharing," Journal of Supercomputer, 45, pp: 185-204 (2008).

[17] N. Moustafa, I. Al Ridhawi, and A. Hamza, "An Intelligent Dynamic Replica Selection Model within Grid Systems, " in Proc. 8th IEEE GCC conference on Towards Smart Sustainable Solutions, pp. 1-6, 1-4 February 2015.

[18] R. Buyya, and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing", John Wiley & Sons Ltd, 2002.

[19] Yong-Hee Jeon, "Impact of Big Data: Networking Considerations and Case Study", International Journal of Computer Science & Network Security; Dec2012, Vol. 12 Issue 12, p30, 2012.

[20] B. Bhuyan, H. Sarma, N. Sarma, A. Kar and R. Mall, "Quality of Service (QoS) Provisions in Wireless Sensor Networks and Related Challenges", Wireless Sensor Network, Vol. 2 No. 11, 2010, pp. 861-868. doi: 10.4236/wsn.2010.211104 Case Study", International Journal of Computer Science & Network Security; Dec2012, Vol. 12 Issue 12, p30, 2012.