International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 16, No.1 (Aug-24)

http://dx.doi.org/10.12785/ijcds/160145

Design of Node Level Load Balancing in Hierarchical Fog Structure

Jai Geetha[1], Jayalakshmi D S[1], Chandrika Prasad[1], Srinidhi N N[2*] and Naresh E[3*]

[1]*Department of Computer Science and Engineering, Ramaiah Institute of Technology, Bangalore, Karnataka, India*
[2]*Department of Computer Science and Engineering, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, India*
[3]*Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, India*

Received 30 Oct. 2023, Revised 19 Apr. 2024, Accepted 26 Apr. 2024, Published 1 Aug. 2024

Abstract: Fog computing refers to the operations performed in a distributed network of nodes on the edge of the network to provide faster output generation for emergent requests. The fog layer brings computation closer to the devices, thereby reducing latency in the network. However, in recent years, fog computing has been subjected to several problems, including load balancing. Load balancing ensures appropriate allocation and distribution of resources and workload in a fog network. That being said, it's important to note that the nature of nodes in a fog network can be heterogeneous. In such a situation, it's crucial to have a load balancing mechanism that routes requests to the appropriate node based on the type of the requests, the load on the nodes, and the total load on the system. One way to solve the issue would be applying the conventional load balancing algorithms, but traditional load balancing schemes don't apply here since they tend to work amongst homogeneous sets of nodes with similar resources. In the proposed research, three load balancing schemes, namely- Highest Capacity Mode, Earliest Predicted Response Time, Equal Capacity Mode have been proposed. Finally, an optimal approach, a hybrid algorithm, and taps on the benefits of the 3 types of load balancing schemes have been proposed with the best performance among all the schemes.

Keywords: Fog Computing, Load Balancing, Node-RED, Hierarchical Fog Architecture

1. INTRODUCTION

IoT networks are made up of loosely connected networks that are linked together by a heterogeneous network of nodes. In general, the goal of creating such environments is to collect and analyse data from IoT devices to mine and find trends, perform predictive analysis or optimization, and then make a better decision promptly. Data collected and aggregated from IoT devices are sent to the centralized cloud for storage and processing. But the processing of data in a centralized cloud does not scale to the requirements, especially in the case of applications such as health monitoring and emergency response that require less latency. And that is where fog computing comes into action [1]. Fog networking or fogging is a decentralized computing structure for edge devices to carry out complex calculations, storage, and communication locally and/or over the internet [2]. Here, resources like data and applications get placed in logical locations between the data source and the cloud. However, in the case of fog computing, the network nodes tend to be heterogeneous which raises the issue of balancing the load amongst the nodes in the network. To cater to this issue, load balancing schemes are applied that ensure a uniform and appropriate distribution of load in the network. Given the heterogeneous nature of the nodes in a fog network, it is crucial to have effective load balancing to decide the best fog node for handling a request. Traditional load balancing schemes don't apply here since they tend to work amongst homogeneous sets of nodes with similar resources. Another important aspect of the nodes in the fog system is the appropriate architectural organization of the nodes within a fog layer [3]. An effectively designed architecture can bring about better results and better resource allocation. This paper aims to introduce a hierarchical structure to the nodes in a fog environment. And secondly, provide a load balancer scheme where the decision is taken at the node level, rather than using a dedicated load balancer.

2. THEORITICAL BASIS

Aazam et al. [4], presented a six-layered Fog architecture. Physical and virtualization layers maintained and managed physical nodes and sensors. Network and underlying node activities were monitored by the next, Monitoring layer. After that, the temporary storage layer stores temporary data in the Fog resources. For private data, pri-

E-mail address: geetha@msrit.edu, jayalakshmids@msrit.edu, chandrika@msrit.edu, srinidhi.nn@manipal.edu, naresh.e@manipal.edu

vacy, encryption, and integrity-related services are provided by the security layer. Secured and pre-processed data is uploaded to the cloud by the topmost layer. Advantages seen were that most of the processing was done in the Fog environment and that allowed the cloud to deal with more complex services. However, the proposed architecture has the following shortcomings:

1) Achieves low latency but not to the required level.
2) No direct communication between consumers and the layers is allowed.

Arkian et al. [5], proposed a four-layer Fog architecture consisting of data generation, cloud computing, data consumer and fog computing layers. The requests are submitted to the three layers and responses are sent for the required services. IoT devices are present in data generation layer and communication to cloud layer is through the fog layer. Decision making at the dynamic level is supported by cloud computing layer where complex and log-term analysis of behavior takes place. Dynamic decision making includes pattern recognition, large scale event detection and relationship modeling. The advantages of this model include having vast monitoring abilities and services and common cloud content. The model also allows direct communication among the three-layered architecture of the model and its consumers. Lack of proper validation due to the open nature of architecture was a grave disadvantage. Shaik et. al [6], proposed an organization technique to group fog nodes into hierarchical, layered structures where each layer in the hierarchy is grouped according to the similarity in resources and behavior. The paper focuses on grouping nodes based on geographic locations, resource availability, and various other characteristics. It is noted that the fog nodes in the lower levels tend to have lower resources and computing strength, while also having a much lower latency. This includes mostly the gateway nodes as well. The following advantages are noted-

1) It is noted that in the cases of hierarchical structure, it paves the way for better resource allocation, load balancing, resource utilization, autonomy, and allows for a logical grouping of nodes by similar characteristics, thus allowing some form of homogeneity within each layer.
2) As one goes higher in the levels of the hierarchy, the available resources, availability, and reliability of the fog nodes tend to be better than at lower levels.

The disadvantages are

1) The main downside of this approach is the lack of knowledge of the complete system state.
2) Due to inter-node communication between nodes in the hierarchy, there tends to be a latency trade-off for inter-level communication.

Chandak et. Al. [7], explore the various load balancing methods used in Fog computing environments such as Software Defined Network (SDN) approach, Graph Partitioning, Hill Climbing, Secure and Sustainable Approach, Distributed Approach, Adaptive Load Balancing, Centralized Load Balancing, Dynamic Resource Allocation, and Tabu Search. The inferred result was that various load balancing schemes are surveyed with appropriate parameters identified, with issues and improvements discussed as well. The SDN approach lacks in terms of software complexity to dynamically allocate resources. It can be observed that graph Partitioning does a better job at the dynamic allocation of resources and uniform graph partitioning or a balanced graph partition problem can be shown to be NP-complete. In hill-climbing, if the threshold is left too high or low, it may take several iterations to acquire desired values for estimation. In DaeWon et. Al [8], the authors proposed a load balancing technique with graph coloring based on a genetic algorithm. The algorithm takes the input of G=(V+E) where V is the number of fog nodes and E the available edges. The algorithm's output is the number of colors used and the goal is to minimize this number, the decrement or increment of the numcolor variable and satisfaction of the condition that no two adjacent nodes bear the same color, determines whether the edge node will offload the incoming task into the cloud server or perform the action itself. Observing this method, the advantages are 1. The use of graph coloring helps to minimize the resources used as the colors represent nodes. 2. Genetic algorithms facilitate ease of implementing solutions both in parallel and distributed settings. However, 1. The algorithm employs graph coloring as well as a genetic algorithm that includes the calculation of a fitness function and evaluation of condition variables at each iteration, thus making the process costly. 2. The graph coloring starts with a random number, to begin with, and the algorithm is repeatedly executed until an accurate value for the number of colors used is obtained. This is a very computationally heavy procedure., are some noticeable disadvantages Alex X. Liu, et. Al. [9], introduce DRAM, which stands for Dynamic Resource Allocation Method, for load balancing in fog environments. It combines static resource allocation with dynamic service migration to achieve load balancing among fog nodes in the system. The load balancing mechanism is handled in four consecutive steps, namely, Fog service partition, spare space detection (for computing nodes), static resource allocation (for fog service subset), and then the global resource allocation strategy with the purpose of load balancing. It is found that the load-balance variance is drastically reduced, since each node has a certain degree of resource utilization, and the average resource utilization is balanced. DRAM enables the usage of fewer computing nodes, compared with other schemes, and can even get higher efficiency on a given number of computing nodes than Bidirectional Forwarding Detection. The authors suggest that further analysis of the negative effects of service migration on the fog nodes may be required, including traffic on the nodes, the cost-of-service migration and data transmission, and performance degradation. Furthermore,

it may be required to develop a method to balance the mentioned negative effects, such that the positive effects of service migration outweigh the negatives. Verma et. Al [10], offers the RTES algorithm, which balances the load by utilizing available bandwidth, completing real work ahead of schedule, and responding to clients less time than a standard cloud strategy. The technique is implemented on a simulator called CloudSim, and the results are compared to existing task scheduling algorithms like FCFS based on the turnaround time parameter. The proposed algorithm outperforms the current ones. It has a short execution time, responds quickly to client requests, and completes real jobs ahead of schedule while maintaining data consistency and proper resource and bandwidth use. The suggested algorithm is 90% efficient, however it might be even better if other QoS criteria like security were included. Rahman et al. [11], proposes minimization of bandwidth cost and efficient resource management in a cooperative three-layer fog-cloud computing environment as well as a novel MILP optimization formulation in the three-layered structure. The data collected by sensors from IoT devices is thought to require bandwidth and server resources in order to be processed. Each request is represented as a tuple (h, r), with h denoting bandwidth and r denoting resource demand. The requests made by the sensor nodes within a certain region are pooled to build the MILP model, and these requests also include two demands, named H and R denoting aggregated bandwidth demand and aggregated rescue demand respectively. The requests are then prioritized, and various situations are investigated to see how SDN may assure optimal network and server resource utilization in a given setting (Software-defined network approach). Following that, both homogeneous and heterogeneous server resource demands are considered, and performance variations are examined in terms of bandwidth cost, server and connection usage, and the number of servers deployed. The priority level can be changed by adjusting the weight parameters, such as bandwidth cost, link-level utilization, and server resource consumption. This allows the service provider to prioritize tasks based on network conditions. Due to temporal and spatial volatility in requirements, the suggested model allows for changing priority levels from time to time. As a result, this approach can assist fog service providers in appropriately allocating limited resources. This research could be improved further by combining server and network consolidation using virtualization approaches in a dynamic traffic environment to optimize resource allocation even more. Saleh et al. [12], presents a Load Balancing and Optimization Strategy (LBOS) based on Reinforcement Learning and a Genetic Algorithm. LBOS continuously monitors network traffic, gathers information about each server's load, processes incoming requests, and distributes them evenly among available servers using dynamic resource allocation algorithms. As a result, in real-time systems in fog computing, such as healthcare systems, LBOS is simple and efficient. The system's primary purpose, however, is to achieve low latency. The fog layer here is made up of two primary components: (1) a load balancer agent (LBA) and (2) a resource allocator (RA). The LBA is the software that determines which fog server (FS) can handle the incoming request. To attain a high LB for fog conditions, the RA module uses the RL algorithm. The importance of choosing a given FS is determined by the adaptive weight value (AW). To get the optimal AW value, a genetic algorithm (GA) is applied. The authors use MATLAB to conduct a comparison experiment between the Adaptive Weighted Round Robin (AWRR) algorithm and several alternative LB schemes in order to verify the performance of the LBOS algorithm. Experiments were conducted, and the findings revealed that the proposed method improved quality-of-service in the cloud/fog computing environment by lowering allocation costs and decreasing reaction time after using state-of-the-art algorithms to compare the suggested approach. The results of the experiments also showed that it can cut power usage and the number of migrations, the quantity of data saved within the load balancer may be limited due to the increasing amount of memory for the model during the reinforcement stage, and changes can be made to make the model more lightweight. Nanda et. Al [13], discusses new techniques for load balancing to discover lightly loaded data centers in the edge for allocating tasks and for the secure edge data center authentication. The approach discussed brings forward an authentication technique that is adaptive assisted with a central cloud data center. The process of device authentication originates from the cloud data center and is then approved by all other edge data centers approve authentication by passing on the key pairs and valid credentials. Moreover, the solution provides dynamic load allocation by considering the load of destination nodes of which the information is shared among nodes during the authentication phase. The authentication and load balancing schemes are combined to come up with secure load balancing. Authentication is performed through public and private keys with Trusted Machines and is propagated in a distributed manner from the authenticated nodes to the other nodes. The dynamic load balancing is done by Bread First Search (BFS), where if one EDC is overloaded, a control package is broadcasted by forwarding requests to all edge data centers with respect to their Ids. This model offers secure authentication and improved load balancing performance at the edge data center devices. This is done by accessing the target EDC load during the authentication procedure. Due to the added overhead of the authentication scheme, the load balancing scheme tends to sometimes suffer and slow down when under severe stress to propagate changes. Harshit et. Al [14], talks of growing dependency of industry on IoT, thus overwhelming the storage infrastructure in place and also the data analytical techniques applied by the large amounts of data generated. Scaling the rapidly increasing processing and storage requirements at the architectural levels is offered by services provided by the cloud layer. The issues arising with this security offered by the cloud can be seen in the health care and emergency response application that require a significantly low latency and relay of data to and from the cloud will prove to be catastrophic. In response

to overcoming these very persistent limitations, the Fog computing layer was introduced wherein the edge of the network has an extended cloud on top of it to reduce the congestion in the network and process latency. To evaluate performance and promote innovation in the currently employed resource management algorithms comprehensively, a simulator is employed. In the future, this technique can be used to facilitate time and priority sensitive applications in a multi tenanted environments. Parvedy et. Al [15], proposes iFogStor for an innovative data organization technique to be employed by the fog architectures. The goal of the proposed system is to gain advantage from the position and diverse behavior of fog devices and to observe significantly lower latency in the push and pull operations at the devices. All the fog nodes at the storage and operational sites are widely heterogenous when their capability of data storage and performance efficiency are taken into account. These fog nodes can be from a vast range of possibilities, such as base stations and set-up boxes that are equipment with limited resources. iFogStor exploits the complex nature of the fog architecture to adopt the data arrangement strategy taking into consideration the node properties. Specifically, the strategy takes into account the characteristics such as the nature of data stored in the node, its location and performance. The future scope of the proposed model explores the idea of introducing more sensitive nodes. These nodes can also have added advantage of archiving services in association with the improved data allocation design thus introducing a rather optimal solution. In conclusion, the literature survey brought to light the various aspects that need to be taken into consideration for the effective performance of a Fog Network. The architecture of the layers in view, appropriate techniques for load balancing, and efficient and optimal algorithms for resource allocation are vital to the cause. Taking into consideration this information, the project proposes a hierarchical structure to the nodes in a fog environment and a load balancer scheme where the decision is taken at the node level, rather than using a dedicated load balancer. The hierarchical structure safely ensures that the nodes are clustered together based on similarities in their bandwidth and latency characteristics. This ensures proper communication between all fog nodes and the optimal algorithm to effectively choose the correct level of fog layer to assign the task to. The increase in latency is in many cases negligible due to the low latency between interconnected fog nodes. The potential advantages of a hierarchical structure were emphasized, along with the potential downfall of the lack of global system state knowledge. This project utilizes the hierarchical structure with the advantages, along with a shard of the system state knowledge, to make routing decisions at the node layer.

3. METHOD

A. Design

Figure I, Shows a sample two-level hierarchical structure, with the lowest level (called the gateway level) consisting of gateway nodes to which the IoT devices connect, and the connection between each layer is routed through a router

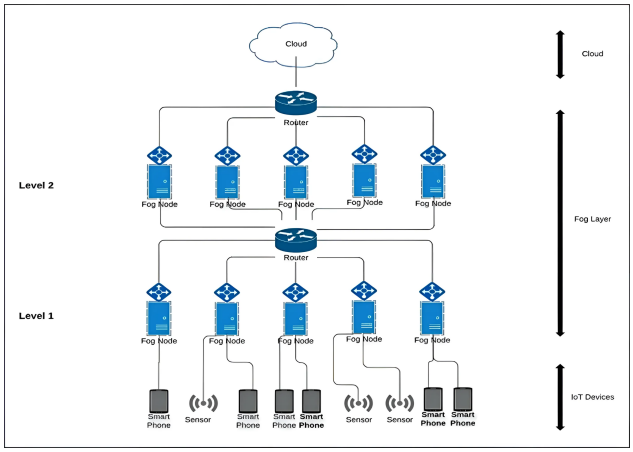

Figure 1. VHDL Synthesis Process

which forwards requests as per previous level decisions. Theoretically, this can be an N-level architecture with M number of fog nodes in them. After the final level of fog nodes is the cloud server. Requests are screened by these nodes to assess them based on the available capacity and bandwidth in the nodes. If the nodes fail to be able to process the incoming request, the request is passed on to the higher level of nodes in the hierarchy and so on until the cloud layer is reached.

B. Design Graphical User Interface

Figure 2, shows the GUI of the form for a fog node, where one can enter the fields for the various thresholds and allocations of a fog node. This can be done within the same GUI of the flow simulation to quickly change metrics before rerunning the simulation. There are set of nodes arranged according to the proposed fog layer architecture and the arrangement is used for simulating the processing of requests in real time. Based on the results of the simulation, graphs are generated to show the load balancing and resource allocation among nodes in the various layers. The arrangement has start and end nodes which are meant to pass signal to the IoT devices at the edge layer. Followed by that, the Fog layer is divided into hierarchical layers, layer 1 and layer 2. Based on the requirement of the request raised by the IoT devices, the load is balanced between these layers based on parameters like capacity and turnaround time. After the request is processed by one of the nodes, its performance is recorded in terms of the time taken to complete the request and the number of nodes jumped to assign the request. The data is then populated in a graph to analyze the efficiency of the model.

C. Proposed System

Figure 3, explains the following about the proposed system configuration:

1) The proposed system is a collection of custom fog nodes that can be used on node-red for fog simulation purposes. A flow-on node-red can be created

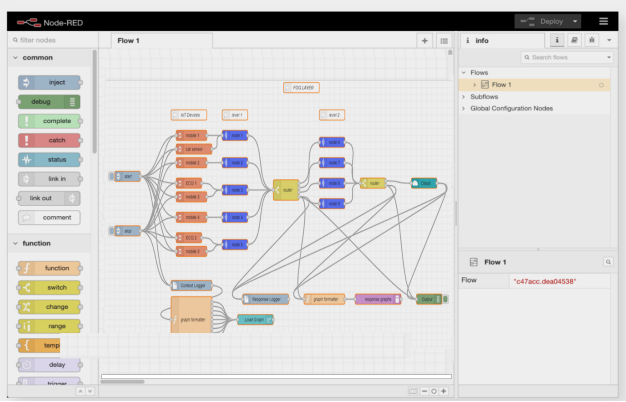

Figure 2. GUI of Fog Nodes in Simulation Environment

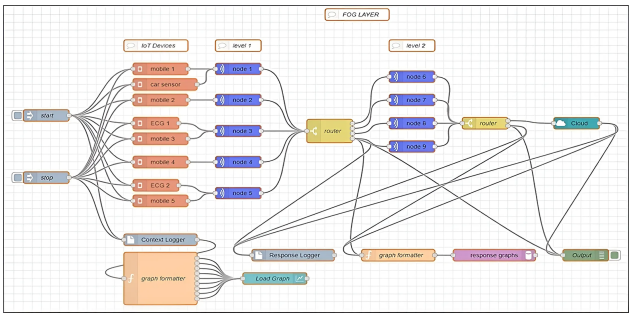

Figure 3. System Modeling for Hierarchical Fog Structure

to simulate a fog environment using the custom fog nodes. There are (currently) 3 types of nodes on the system available: fog nodes, devices, and the cloud.

2) Fog nodes act as the fog devices in the fog layer, which handles and redirects responses appropriately. A device is an IoT device like a mobile phone or an ECG that periodically sends requests to the gateway nodes. The cloud has seemingly infinite resources.
3) The fog layer is represented as a hierarchical structure, where the higher levels have somewhat higher resources but have a higher latency as well, to incentivize each node to complete the task itself.
4) A few QoS factors like latency, Instructions Per Cycle (IPS), and Capacity (represents RAM) are based on which the performance of load balancing is evaluated. A demo flow with a hierarchy of 2 levels for illustration purposes has been created for better visual understanding.

Requests are scheduled based on these factors and three load balancing schemes have been proposed based on them: Highest Capacity (naive), Earliest Predicted Time, and Equal Average Load. An optimal approach is taken which combines the capabilities of “Equal Average Load” mode and “Earliest Predicted Time” to redirect requests to effectively balance the load, thus improving resource utilization, as well as improving the overall response times of each node.

D. Algorithm

Figure 4, shows a flowchart depicting the various actions taken by the entities in the system when given a request by an IoT device. The full flow is explained as below in steps.
Step 1: First, the selected node receives the incoming request, which is the gateway node assigned to a particular device.
Step 2: The request is served by the current node if the current node has sufficient resources to handle the load. Else it redirects the request to the nodes one level higher than it in the hierarchy.
Step 3: The decision to which node the request is forwarded to is taken depending on the type of load balancing scheme.
Step 4: Steps 1-3 are repeated until the request is served by a node and it keeps getting redirected to a higher level till then.
Step 5: If the nodes at the highest level cannot handle the request, it is forwarded to the cloud which is assumed to have infinite resources compared to a fog node.
Step 6: At the end, the metrics and results of the path and request are logged in a file for further analysis.

Figure. 5, shows the pseudocode for the Optimal mode algorithm, where the decision is taken based on the product of the earliest predicted response time and the existing load percentage on higher level nodes. If no decision is made within the specified thresholds, the node with the highest capacity is chosen as a fallback. In the function signature, higherLevelNodes refer to the array of the fog nodes that are one level higher than the fog node making the decision, capacity and instructions refer to the required capacity and instructions of the incoming request.

E. Implementation of Modules

The proposed system primarily consists of following modules:

- Fog node module: This module is in charge of receiving requests and routing them to the relevant node. Each node in the module evaluates and accordingly directs incoming requests based on characteristics such as message payload within the node or the number of instructions. The nodes also store the state of a node higher in the hierarchy so that if they are unable to serve the request, they can determine which node should get it. In addition, the nodes have customizable capacities, a maximum load threshold, and a maximum duration to serve the request.

- IoT device module: Each device in this module can transmit requests of different capacities over a period that can be determined by the user.

- Data logging module: This module is primarily concerned with data formatting for analysis as well as logging response times to create real-time graphs.

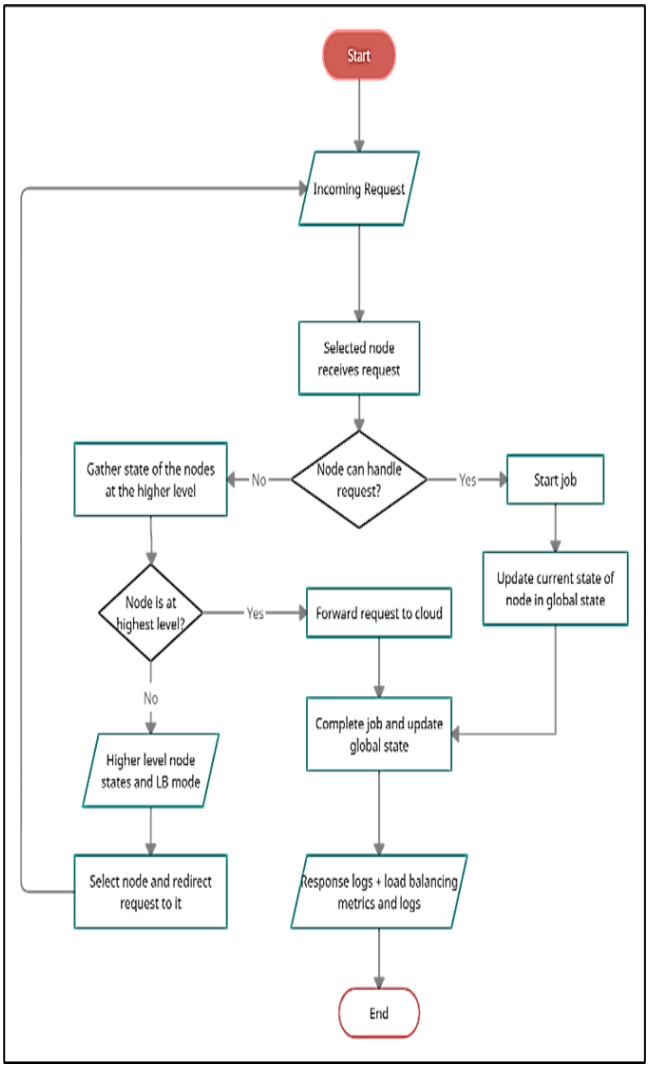

Figure 4. Flowchart Explaining the Execution flow

Algorithm 1 Optimal Load Balancing Scheme

```
function OptimalRedirect(higherLevelNodes,capacity,instructions)
    lowestWeightedAverage ← 999999999
    lowestPredictedTime ← 99999999
    lowestPercentage ← 100
    highestCapacity ← −1
    chosenNode ← null
    highestCapacityNode ← null

    for node in higherLevelNodes do
        if node.capacity > capacity then
            instructionTime ← (instructions/node.IPS) * 1000
            predictedTime ← node.latency + instructionTime
            weightedAverage ← predictedTime + node.loadPercentage

            if weightedAverage < lowestWeightedAverage then
                lowestPredictedTime ← predictedTime
                lowestPercentage ← node.loadPercentage
                chosenNode ← levelNode
            end if
        end if
    end for

    if !chosenNode then
        chosenNode ← highestCapacityNode
    end if

return chosenNode
end function
```

Figure 5. Algorithm for the Proposed Work

4. USE CASE

The proposed algorithim is aimed at choosing the best node with appropriate capacity to perform the task at hand and that has lowest possible latency. This algorithm would be at best performance in a hierarchical architecture as the complexity of switching nodes on the same layer of the network would be almost negligible. The intension behind this algorithm-architecture combo is to aceive maximum efficiency in as little time as possible. This combination will be best suitable for tasks where the payload requires certain specific capacity and configuration requirements to be fulfilled. The nodes on the fog layer would be chosen based on best suitability with respect to configuration match, available capacity and availability of nodes. The payload once processed can be forwarded to the cloud layer for transmission. An appropriate use case for the proposed algorithm would be in smart homes and smart cities with a wide range of transaction requests. The appliances in settings like these require different kinds of task performed with varied computational requirements. These requirements can be optimally fulfilled by the various nodes available in the proposed fog architecture. Due to grouping of nodes based on configuration and computational capacities, the algorithm is faster in assigning a node to the incoming pay load. As the required computational power is known to the system beforehand, the algorithm places the request at the appropriate fog level based on computational power offered by the nodes. Among the available nodes at the said layer, the algorithm choses the best suitable node with appropriate capcity to complete the task. This way, the system, a combination of hierarchical architecture and optimized algorithm, achieves maximum request fulfilment efficiently and provides the scope of batch processing of requests based on the varied range of devices being in action within a smart home setting.

5. COMPARING PROPOSED ALGORITHM TO A BENCHMARK SCHEDULING ALGORITHM

One of the very famous algorithms used in fog network node scheduling is based on swarm optimization, proposed by [16]. This algorithm is based on calculating the makespan given by the summation over n processors performing a task T in time given by the function ET taking the task and processor as parameters. The constraint of the algorithm being that the makespan is always lesser than the deadline. The execution cost (ECU) for each processor is taken into account along with the transmission cost per unit distance (TCU).

$$Makespan = max_{j=1}^{M} Time_{Pj} = max_{j=1}^{M} \Sigma_{i=1}^{n(p_j)} ET(T_i, P_j) \quad (1)$$

Eq 1. gives the equation for the scheduling algorithm used in [16]. It calculates the max time taken by a processor j among M processors by finding maximum summation over n processors for the execution time calculated for a task i and process j. The proposed algorithm is at power with the optimization achieved by [16]. The algorithm proposed in the paper reduces the transmission cost significantly due to the hierarchical architecture by virtue of which the transmission cost at the same level reduces to null. As the nodes are grouped based on the capacities and configurations, the different hierarchy layers within the fog layer itself contribute very less to the transmission cost due to target mapping performed by the algorithm to place the incoming request on the appropriate level which is then followed by a choice between the available nodes at that layer.

6. WORKING OF THE PROPOSED ALGORITHM WITHIN THE FOG NETWORK

The fog network comprises of the physical and virtualization layer, monitoring layer, pre-processing layer, temporary storage layer, security layer and the transport layer going from bottom to top. The physical layer consists of sensors and are incharge of collecting incoming data that needs to be processed. The next layer, the monitoring layer is responsible for monitoring nodes and resources within the network, as well as response and service monitoring. The pre-processing layer performs data analysis and filtering, and the temporary storage layer works on data deduplication, distribution and replication. The security layer perfoms encryption and decryption on the data which is transmitted to the cloud layer via the transmission layer.

Looking at the functions of the various fog layers, it is evident that the architecture and algorithm proposed by this paper are well suited for the monitoring layer where the nodes and resources monitoring is performed and appropriate nodes are chosen for the task at hand. The monitoring layer will perform more efficiently when facilitated with the algorithm proposed here and will also have scope for batch processing and optimal resource allocation at low latency will be achieved.

This architecture will also be suitable for the pre-processing layer where in nodes perform computation and data analysis. At this layer the choice of nodes for the job is a crucial task and can be completed with better efficiency when applied in association with the proposed algorithm.

7. RESULTS AND DISCUSSION

A. Performance Analysis

On running tests on the proposed algorithms, their results and performance parameters were plotted in graphs and compared. The following is a better insight into the results obtained.

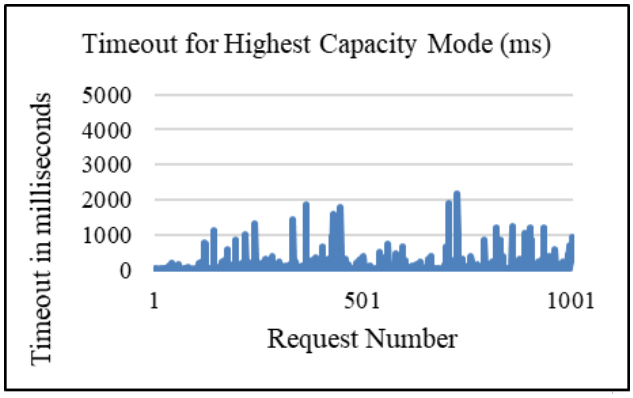

Figure 6. Average Load on the fog nodes (Highest Capacity Mode)

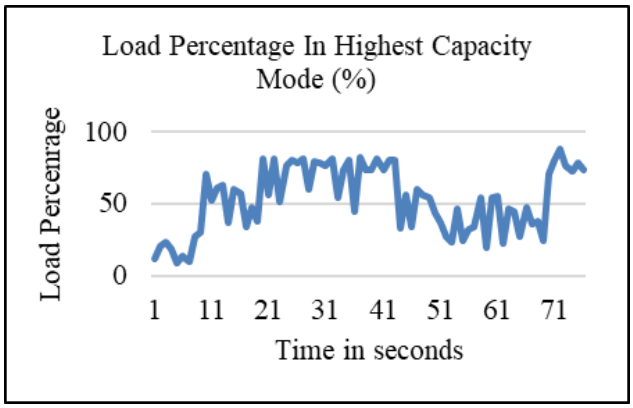

Figure 7. Response times in Highest Capacity Mode

1) Highest Capacity Mode

Figure 6, shows the average load on fog nodes as time progresses. The load is shown as percentage of capacity used up in the fog node. Fog nodes with no load on them at the time are not counted for calculations. The following graph shows the Highest Capacity Mode. In Figure 7, the graph shows the response times of the requests of a sample 1000 requests in milliseconds. The y-axis is the response time in milliseconds and x-axis is the request number, which goes from 1 to 1000. This shows the response times for the Highest Capacity Load Balancing Scheme.

- Load is very unevenly distributed with some nodes reaching almost 90-100 load percentage.
- Response time reaches up to 8000 milliseconds in some cases, and many requests have a response time of 2000 milliseconds.

2) Earliest Predicted Response Time Mode

Figure 8, shows the average load on fog nodes as time progresses. The load is shown as percentage of capacity used up in the fog node. Fog nodes with no load on them at the time are not counted for calculations. The following graph is for the Earliest Predicted Response Mode.

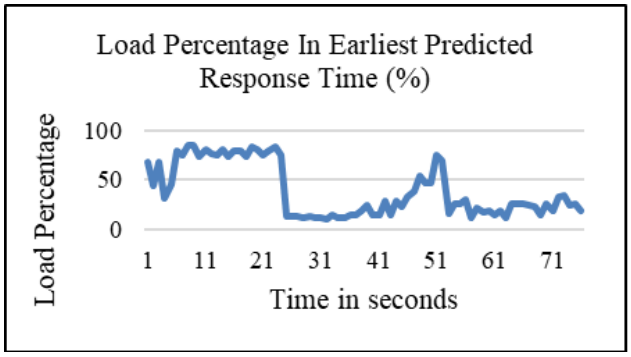

Figure 8. Average load on the fog nodes (Earliest Predicted Response Mode)

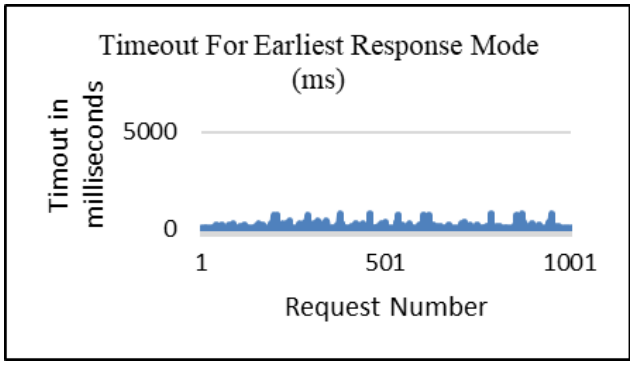

Figure 9. Response times in Earliest Predicted Response Mode

In Figure 9, the graph shows the response times of the requests of a sample 1000 requests in milliseconds. The y-axis is the response time in milliseconds and x-axis is the request number, which goes from 1 to 1000. This shows the response times for the Earliest Predicted Response Load Balancing Scheme. i. Load is much more evenly distributed than Highest Capacity Mode although a lot of requests get served by the cloud. ii. Response time is always less than 1000 milliseconds.

3) Equal Capacity Mode

Figure 10, shows the average load on fog nodes as time progresses. The load is shown as percentage of capacity used up in the fog node. Fog nodes with no load on them at the time are not counted for calculations. The following graph is for the Equal Capacity Mode.

In Figure 11, the graph shows the response times of the requests of a sample 1000 requests in milliseconds. The y-axis is the response time in milliseconds and x-axis is the request number, which goes from 1 to 1000. This shows the response times for the Equal Capacity Scheme.

- It may handle resource utilization more but, load balancing is still unstable.
- Response time is higher than 2000 milliseconds frequently and reaches over 6000 milliseconds sometimes.

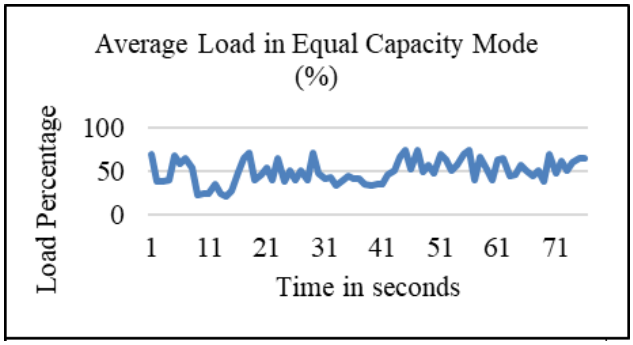

Figure 10. Average Load of the fog nodes (Equal Capacity Mode)

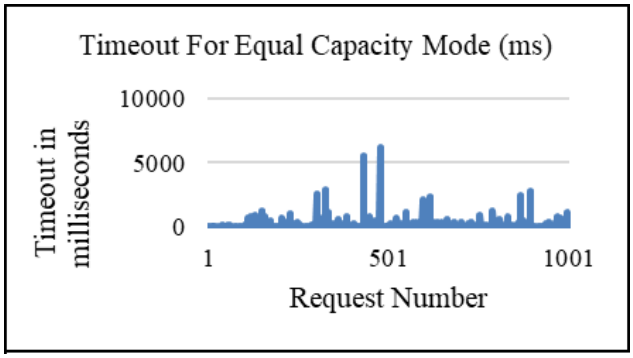

Figure 11. Response times in Equal Capacity Mode

4) Optimal Mode

Figure 12, shows the average load on fog nodes as time progresses. The load is shown as percentage of capacity used up in the fog node. Fog nodes with no load on them at the time are not counted for calculations. The following graph is for the Optimal Mode.

In Figure 13, the graph shows the response times of the requests of a sample 1000 requests in milliseconds. The y-axis is the response time in milliseconds and x-axis is the request number, which goes from 1 to 1000. This shows

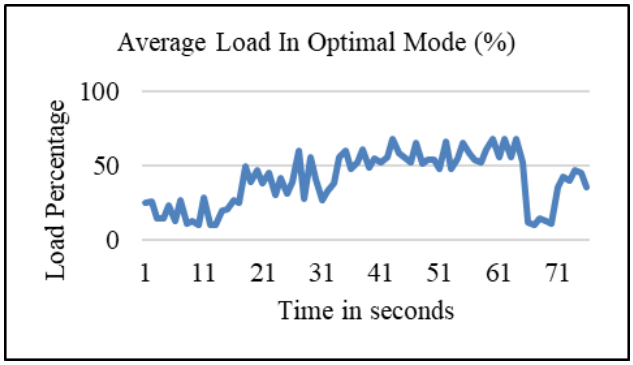

Figure 12. Average Load of the fog nodes (Optimal Mode)

TABLE I. Comparison between all four algorithms

Algorithm	Load Capacity	Turn Around Time
High-Capacity Mode	Uneven distribution of load, Nodes reaching 100 % capacity	Worst-case time taken is around 2000ms
Equal Capacity Mode	Resource utilization satisfactory but unbalanced distribution of load	Several edge cases over 2000ms for high resource tasks, worst case 6000ms
Earliest Prediction Mode	Load is evenly distributed but resource utilization is uneven among nodes	Time taken is always under 1000ms, thus generally optimal
Optimal Approach	Load is evenly distributed and resources have optimal utilization	Time is very satisfactory and always under 500ms

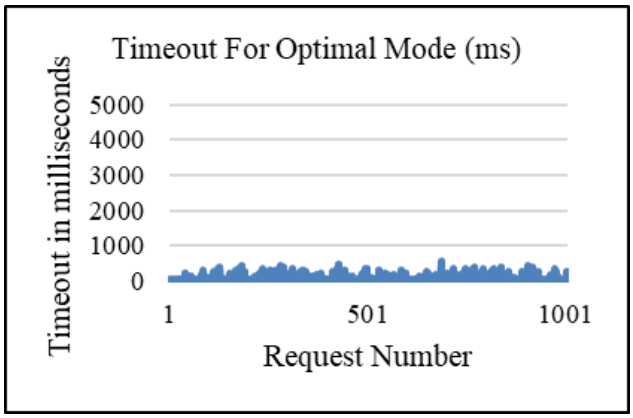

Figure 13. Response times in Optimal Mode

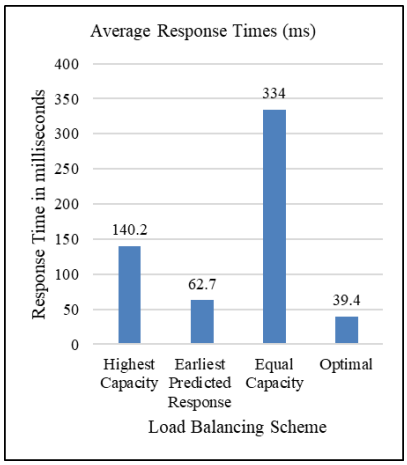

Figure 14. Average response times for the four modes (milliseconds)

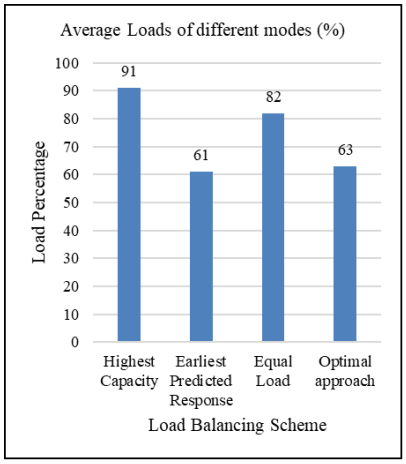

Figure 15. Average load of fog nodes for the four modes

the response times for the Optimal Mode.

- Able to effectively select nodes more evenly, with better resource utilization than "Earliest Response", but not as good as "Highest Capacity".
- It also shows a better general response time, where all requests are served in under 500 milliseconds.

B. Results Comparison

Figure. 14, Shows the average response times in milliseconds for requests coming in the four load balancing schemes and Figure. 15, shows the average load of a given fog node at a given time in the four load balancing schemes, in percentage of capacity used up when serving a request. A tally of all the results is tabulated in Table 1.

8. Conclusions and Future Work

A fog simulator was built from scratch in Node-Red which is extensible and scalable and can have N number of levels and [M1, M2, ... Mk] number of nodes at each level. The Hierarchical structure proposed at the Fog Layer adds an advantage to the computing efficiency of the network. 3 different kinds of load balancing algorithms were built and compared: Highest Capacity, Earliest Predicted Response Time, and Equal Average Load and Optimal Approach algorithm which combines the three. Optimal approach that taps in on the benefits of the other 3 modes of load balancing schemes was found to outperform them with average response time less than 40 milliseconds and average load percentage of only 63 %. There is a need for improvement in the working of Fog Computing with a focus on various QoS factors. It is crucial to have effective load balancing in a Fog Computing environment to decide the appropriate node for handling a request. The load balancing scheme introduced in this project primarily aims to cater to the issue of latency in a typical cloud computing setting. That means

the project aims to create an architecture that can ensure the appropriate distribution of load in a network in such a way that minimizes the time required in receiving and processing the requests. This idea can further be extended to incorporate other factors such as QoS to improve the performance of the fog network. In addition to that, this idea can be improved further by making it more scalable to solve real-world problems where the number of nodes in a network is quite large.

References

[1] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga, "Resource management techniques for cloud/fog and edge computing: An evaluation framework and classification," *Sensors*, vol. 21, no. 5, p. 1832, 2021.

[2] J. Lim and D. Lee, "A load balancing algorithm for mobile devices in edge cloud computing environments," *Electronics*, vol. 9, no. 4, p. 686, 2020.

[3] S. K. Mani and I. Meenakshisundaram, "Improving quality-of-service in fog computing through efficient resource allocation," *Computational Intelligence*, vol. 36, no. 4, pp. 1527–1547, 2020.

[4] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for iot," in *2015 IEEE 29th international conference on advanced information networking and applications*. IEEE, 2015, pp. 687–694.

[5] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications," *Journal of Network and Computer Applications*, vol. 82, pp. 152–165, 2017.

[6] S. Shaik and S. Baskiyar, "Hierarchical and autonomous fog architecture," in *Workshop Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–8.

[7] A. Chandak and N. K. Ray, "A review of load balancing in fog computing," in *2019 International Conference on Information Technology (ICIT)*. IEEE, 2019, pp. 460–465.

[8] J. Lim and D. Lee, "A load balancing algorithm for mobile devices in edge cloud computing environments," *Electronics*, vol. 9, no. 4, p. 686, 2020.

[9] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun, A. X. Liu *et al.*, "Dynamic resource allocation for load balancing in fog environment," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[10] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *Int. J. Inf. Technol. Comput. Sci*, vol. 8, no. 4, pp. 1–10, 2016.

[11] M. M. S. Maswood, M. R. Rahman, A. G. Alharbi, and D. Medhi, "A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment," *IEEE Access*, vol. 8, pp. 113 737–113 750, 2020.

[12] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (lbos) using reinforcement learning in fog computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 4951–4966, 2020.

[13] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.

[14] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[15] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "ifogstor: an iot data placement strategy for fog infrastructure," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2017, pp. 97–104.

[16] W.-C. Yeh, C.-M. Lai, and K.-C. Tseng, "Fog computing task scheduling optimization based on multi-objective simplified swarm optimization," in *Journal of Physics: Conference Series*, vol. 1411, no. 1. IOP Publishing, 2019, p. 012007.

Dr. Geetha J is working as an Associate Professor in Computer Science and Engineering Department of Ramaiah Institute of Technology, Bangalore. Her areas of interest include cloud computing, big data, semantic web, graph theory and web design. She has 17 years of teaching experience and has 21 publications, the most recent one being "Implementation and Performance Comparison of Partitioning Techniques in Apache Spark". Dr. Geetha is also a member of IEEE and ISTE.

Dr. D S Jayalakshmi is working as an Associate Professor in the Computer Science and Engineering Department of Ramaiah Institute of Technology, Bangalore. Her areas of interest include cloud computing, big data, and computer graphics. She has 27 years of teaching experience and has 30 publications, the most recent one being "Simulation of MapReduce Across Geographically Distributed Data Centers Using CloudSim". Dr. Jayalakshmi is also a life member at ISTE.

Chandrika Prasad working as an Assistant Professor in the department of Computer Science and Engineering. She has 16 years of experience in teaching and 7 years of experience in research. Her area of interest is Cloud Computing, High performance computing, Text processing in Indian regional languages, Natural language processing. She has published around 20 articles in international conferences and journals.

Dr. Srinidhi N N working as an Assistant Professor in the Department of Computer Science Engineering at Manipal Institute of Technology Bengaluru, MAHE Manipal. He has published more than 40+ research articles in International Journals including Elsevier, Inderscience, Springer, Taylor Francis and International Conferences. He is involved in research and teaching for 8 years and has 3 years of Industrial experience. He served as reviewer for various reputed journals including Springer, IEEE, Elsevier and delivered expert talk on WSN, IoT and Robotics in various colleges including IIT, NIT, DIAT and other premier institutions. His current research lies in the areas of Sensor Networks, Cloud Computing, Fog Computing, Edge Computing and IoT. He served as Guest editor, editorial member, session chair, TPC member, etc for various journals and conferences. He is an international reviewer for research projects from Sultan Qaboos University, Oman and reviewed 4 different projects. He has worked as a Project Fellow for a SERB-DST sponsored research project in the area of IoT worth 40 lakhs.

Dr. Naresh E holding Ph.D in Computer Science and Engineering and Master's degree in software engineering. Currently, he is working as an Assistant Professor (Selection Grade) in the Department of Information Technology at Manipal Institute of Technology Bengaluru. He has published more than 60 articles in reputed journals, conferences, and book chapters. His research interests include Software cost and effort estimation, Empirical software quality engineering, Software process improvement, Artificial Intelligence, Data Analytics, and Internet of Things. He served as editorial member, session chair, TPC member, etc for various journals and conferences. He is a senior member for ACM, International Association of Engineers, Indian Society for Technical Education, ICSES and Internet Society –Global Member.