Journal of Engineering Science and Technology
Vol. 17, No. 6 (2022) 4487 - 4504
© School of Engineering, Taylor's University

IMPLEMENTING LOAD-BALANCED CONCURRENT SERVICE LAYER FOR IMPROVING THE RESPONSE TIME OF AN IOT NETWORK

J. K. R. SASTRY[1,*], K. V. SOWMYA[2]

[1]Department of Electronics and Computer Science, KLEF University
Vaddeswaram, Guntur District, Andhra Pradesh, India
[2]Department of Electronics and Communication Engineering, KLEF University
Vaddeswaram, Guntur District, Andhra Pradesh, India
*Corresponding Author: drsastry@kluniversity.in

Abstract

Most IoT networks suffer from poor response as designing such a network is done without considering the expected response time. Performance of IoT networks suffers due to several issues that include heterogeneity among the devices, varying speeds for communication, lack of alternative paths for communication, frequent failure of the devices, etc. The issues relating to the performance vary from layer to layer. The performance of an IoT network suffers due to the lack of a proper working system to handle service requests from either side of the users and the devices. In this paper, a load balancing system, a system of architecting the service management, and the decision relating to optimum RESTful servers have been addressed that enhance the performance of an IoT network by more than 69.5% compared to the performance enhancement achieved through other methods.

Keywords: Load balancing, Networking topology in the services layer, Performance enhancement of IoT network, RESTful service management.

1. Introduction

The Internet of Things (IoT) describes the network of physical objects - "things"—that are embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet [1]. D. All things connected to the Internet can talk to each other, enabling a much smarter system where the decision-making process becomes easy. IoT is being applied in almost all sectors, including industries, retail, medicine, etc., paving the way to a fully digitalized world.

IoT networks are multi-layer networks consisting of different layers such as the device layer, controller layer, restful services layer, gateway layer, and cloud layer. Although an IoT application looks simple, it is only with all these layers' involvement that makes an IoT application is complete.

The controller layer keeps the status of the devices, buffers the data, and initiates any control required through actuating. The device layer consists of devices viz., sensors responsible for gathering the data related to physical surroundings into which they are deployed. The data collected from these devices is communicated to the Controller layer through a base station, where this layer processes the data received from the sensors. The device layer consists of devices viz., sensors responsible for gathering the data related to physical surroundings into which they are deployed. The data collected from these devices is communicated to the Controller layer through a base station, where this layer processes the data received from the sensors. The device layer consists of devices viz., sensors responsible for gathering the data related to physical surroundings into which they are deployed. The data collected from these devices is communicated to the Controller layer through a base station, where this layer processes the data received from the sensors.

REST stands for Representational State transfer Architecture for implementing lightweight WEB services, generally called RESTful services. The WEB services are implemented through API that can be called whoever a service is required. An API is treated as a resource, and service is requested by referring to a resource. The APIs are scalable in the sense that an API can be called any number of times as different instances of the same service

The Restful Services layer provides different end-user services for moving the data to the cloud where the data is stored or retrieved. The design of the service layer becomes the key as most of the processing is undertaken in the layer. The services are implemented as WEB services using API, referred to as Restful API. A user can initiate the request to an API by invoking a request for an API by transmitting a message in XML format to the server that hosts the services as RESTful API. Django server is generally used for hosting the web services as RESTful API. The services hosted in this layer must be lightweight as the processing has a heavy bearing on the response time within which the transactions initiated by the user or devices must be answered. Most of the time, processing the transactions initiated by the user is consumed in the services layer. Traffic handling capacity is the key to bearing on the scheduled and processed services. The design and implementation of the layer thus become quite critical.

The gateway layer acts as a medium for transporting the data from the services layer to the cloud layer. Finally, the cloud layer stores the received from the services layer. The flow of data in an IoT network is always bi-directional, wherein the

bottom to top approach the data from the devices is sent to the cloud. In contrast, in a top-to-bottom approach, a user controls the devices from a remote location where the data flows from the user via the cloud layer to the device layer.

A technology that can connect things and communicate is the Internet of things (IoT). Devices in an IoT network can sense the physical parameters and send this information to the higher layers through many intermediate layers. Various applications can effectively communicate using web services available on the Internet. All the communications in a web service are encoded using XML Web services can also be called information exchange systems that enable one application to exchange data with another using the Internet. An overview of web services is given, and a discussion about the two types of web services viz. SOAP-based web services and Restful web services are carried out. The Restful services layer is important for handling the requests sent by the remote clients [2].

An IoT network can be more efficient if it can provide more services to its clients. In [3-5], the importance of Restful web services, i.e., the web services that follow Representational state transfer (REST) architecture in which there is a stateless connection between the server and client highlighted. Restful web services are gaining much importance nowadays since they are lightweight, simple, and stateless. IoT is a technology sandwiched with other technologies to enhance its importance and deliver promising solutions to all problems.

The data collected from devices in an IoT network is stored in the cloud. Due to the tremendous increase in the number of devices connected to an IoT network, data storage in the cloud is also increasing, a challenge. Data gathering and processing technologies should be so that they pre-process the data before being transmitted to the cloud to eliminate any unwanted data. Edge computing is one such technology that has emerged and proved to be an effective solution for the increased storage capacity at the cloud layer. Porter et al. [6] discussed an effective data gathering and processing system based on Restful web services, and the MERN stack is used to build an effective web service for the IoT system.

An IoT network's performance depends on several factors like the protocol used, heterogeneity among the layers, transmission speeds, latency, availability of various alternate paths for communication, etc. An IoT network's performance depends on the performance of individual layers, which can be computed as a summation of performances at all the layers mentioned above.

Each layer in an IoT network needs to be connected to a different topology to derive maximum performance from the whole network. Establishing communication between these layers having different topologies is a real challenge. In addition to the topology, other issues must be addressed at each layer, which affects the topology issue. Different other issues must be addressed at each layer, which affects the issue of other issues that must be addressed at each layer, affecting the response time.

Power minimization and extending the longevity of devices is the key issue relating to the device layer. In the Controller layer communicating with a base station, internal control processing and the flow of requests to the services layer become the key issue. In the Restful server layer, the server should provide the required services, and the rate of service should match the speed at which the data is communicated. The gateway layer establishes high-speed communication into

the cloud while dealing with different communication protocols to pump the data into the gateway.

The important metrics used in accessing the performance of an IoT network are response time and throughput, which are dependent on various other factors such as the bandwidth available, packet loss ratio, protocol conversions that happen at different layers, speed of data communication, and latency of the devices, etc.

The main issue that needs to be addressed in the service layer is handling the services request traffic emanating from the devices and the users on the cloud side. One critical issue that must be addressed is to match the speed of communication matching to the speed of processing the request. Many users might request the same simultaneously, requiring heavy service request instance management. Load balancing of the traffic when several servers are used for service management is another critical issue. The decision to use an optimum number of servers is another critical issue that must be considered. When several servers are used, establishing a connection with the controllers through a load balancer by choosing a proper networking topology is one of the most important key issues that must be addressed. The relationship between all these issues with the overall performance of an IoT network must be assessed, which is the prime focus of this paper.

The novelty of the work

Enhancing the response time of the IoT network through load balancing at services layer into which several redundant, concurrent, and similar servers are added. The balancing system is fully integrated with performance enhancements done in the controller and the device layer.

In the rest of the paper, section 2 covers related work which identifies the GAP regarding performance optimization within the services layer. Section 3 describes a prototype network and its related performance computations considering the transmission of three data packets commencing from a device until it reaches the cloud. Section 4 presents a revised IoT network, focusing on the device and controller layers. Performance computations and the efficiencies that accrue when optimum Wi-Fi data packet size and speed, Optimum CDMA data packet size and speeds have been elaborated in this section. Section 5 deals with the limitations of using a single server at the services layer. Section 6 presents an architecture that shows how RESTful services can be used to implement the service layer of an IoT network. In section 7, a load balancing system is explained, and it has been shown how the performance of the IoT is improved. Architecture is presented in this section, which shows that the load balancing at the services layer is achieved. An algorithm that implements the load balancing is presented in this section. The results relating to the performance of the load-balanced IoT networks have also been discussed in this section

2. Related Work

Several issues have been discussed in the literature connected to the IoT network's performance in one way or the other. A review of the contributions is placed in this section.

Sowmya and Sastry presented [7] a detailed review of the basic issues that must be addressed to enhance the performance of the IoT networks. The same authors [8] also presented a solution for enhancing IoT networks' performance by implementing a clustering algorithm and a multi-stage network within the device level. Next, a method that enhances the performance in the controller layer through high-speed

communication and using effective and sufficient data buffers has been presented [9, 10]. A parallel architecture for controller coupled with clustered architecture for device layer is presented, which proved a very high improvement in the performance of the IoT network. Sasi Bhanu et al. [11] presented a high-speed computing at the services layer, though the solution did not look much into service layer orientation. Sastry et al. [12] proposed a coupler-based implementation to enhance the performance at the gateway level.

Numbers of authors have presented algorithms relating to selecting a device to act as a cluster Head to effect communication of the sensed data and receive the data for affecting the actuating function. The issue of power minimisation of the sensors has been discussed in [13-21].

Some contributions focus on how the sensed data is stored in the cloud [22]. Some issues related to securing the data for transmitting from source to destination have been presented by Kwon and Park [23].

Determination of an optimum number of clusters is another important issue addressed. Some of the contributions in this regard include Puschmann et al. [24].

Various methods of selecting cluster head were proposed in [25-28]. Researchers of [29-37] discussed several issues in handling heterogeneity among devices in the IoT networks, ways of clustering devices at the device layer, etc.

High-performance computing requirements have been felt necessary for building IoT networks, especially to build the services layer, as the customers require a fast response of the order of a few seconds. Implementation of HPC within IoT, however, is expensive. IoT being lightweight, high performance in the services layer must be achieved without adding to the cost.

HPC implementation within cloud computing systems to implement all the desired computing requirements has been discussed [38]. The connection between IoT and HPC has been discussed in [39]. New paradigms and devices that can be used with HPC implemented within IoT layers have been presented. He also has presented how IOT can be supported with HPC.

The performance of the IoT networks generally suffers due to the need to deal with more unwanted data than the actual data transmitted [40]. Maintenance of redundancy within the IoT networks to make the network fault tolerance also leads to poor performance. They have also suggested including another layer within the IoT network, called the Local IoT controller layer, to improve performance.

They have shown how education can be imparted by implementing HPC as a service within a Cloud computing system. Boobala et al. [41] have presented Different ways of optimizing the performance of cloud computing systems when HPC is implemented as a part of the cloud has been discussed in [41]. HPC as a service can be implemented for a user [42]. The process of optimizing performance when HPC is offered as a service is demonstrated by implementing a different kind of task of scheduling.

Implementing microservices-based middleware that implements an intelligent API layer has been discussed [43]. The API also implements various components that include external service assembler, service auditor, service monitor, and service router component to coordinate service publishing, subscription, decoupling, and service combination within the architecture.

Several performance improvement techniques have been presented in the literature [44-46] that include performance monitoring of lathes through IoT performance monitoring UPS. Batteries through IoT, Implementation of asymmetric processing on multi-core processors to implement IoT applications on GNU/Linux framework, testing of message scheduling middleware algorithm with SOA for message traffic control in IoT environment, and Development of hybrid execution service-oriented architecture (HESOA) to reduce response time for IoT application

The literature mostly concentrated on performance optimization within Cluster Layer and Layer to a certain extent. The issues related to performance enhancement in the services layer have been dealt with the extent of implementing high-performance computing, not much focusing on the issue of service requests related traffic, and dependence on the performance at the service layer on the methods implemented in other layers the IoT network,

3. Prototype IoT Network for Experimentation

The IoT network has been built considering all the typical and comprehensive IoT network layers, including the device, controller, services, gateway, and computing layers. The Devices in the device layer are connected as a cluster. A typical IoT network developed for carrying out the experimentation is shown in Fig. 1.

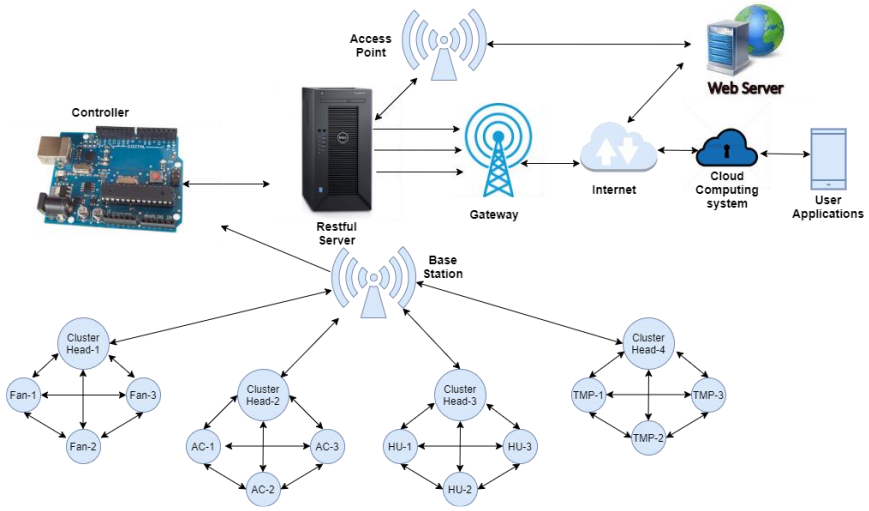

Fig. 1. Experimental IoT network.

Four clusters have been used to sense two inputs (temperature and Humidity) and actuate two outputs (Fans and Air conditioners). Each cluster is provided with a cluster head that communicates with the base station on behalf of all the devices in the cluster to transmit or receive the data. The communication between the cluster heads and the base stations and in between the base stations and the controller is implemented through CDMA. Certain optimizations are implemented within the cluster to determine optimum speeds in our earlier works. Controllers are connected to the Restful service's server through a USB interface. Restful services are primarily designed to implement different types of service requests. The Restful services server is connected to a gateway to connect the local network to the cloud through the Internet.

Computing the response time is the real challenge before improving the IoT network's performance. Time consumed by every device and within every layer for sensing, processing, protocol conversion, receiving, and transmitting must be computed and added to find the overall response time. A process is added into every layer to log data in terms of receiving the data, unpacking the data, packing the data, and transmitting the data. Table 1 shows the time components that must be considered in each layer. Summation of time taken in each of the layers will show the total time taken by an IoT network to complete the transactions initiated from a low-end device until the stage of storing the data in the cloud or the time taken to receive the message from the end-user until the device where the request of the user is processed.

Table 1. Time components that affect the performance of an IoT network.

Serial	Time Component Symbol	Time component Description
1	**TRL_i**	Time taken to receive the data in the i^{th} layer.
2	**$TUPL_i$**	Time taken to unpacking and packing data in the i^{th} layer
3	**TTL_i**	Time taken to transmit the data in the i^{th} layer.
4	**$TTLL_i$**	Time Taken to Log the Data
5	**$TSUM_i$**	Total time taken to process the data. $\sum_{i=1}^{N} TRLi + TUPLi + TTLi + TTLLi$

The prototype model's performance computations have been carried out considering the 4 clusters of sensing and actuating devices, each headed by a cluster head that receives the data using the Wi-Fi protocol, fixed at 11Mbps, the data size of 13 Bytes, and the CDMA communication speed fixed at 110Mbps. The ethernet speed is fixed at 110Mbps to transmit the data to the cloud through the gateway. The time calculations for transmission of the data are shown in Table 2. It can be seen from this table that it has taken 1152 microseconds to complete the transmission of data to a cloud.

Table 2 Time computations of prototype IoT network.

		Reception						Conversion Time					Transmission						
Transmitting Device Code	**Device Description**	**Total Data in Bytes received**	**Protocol used for receiving the -data**	**Incoming data Packet Size**	**Incoming speed used for receiving the data Mbps**	**Total time spent for reception in Microseconds**	**Number of Paths available for reception**	**Time Spent for receiving per path**	**Protocol used for conversion**	**Protocol Conversion time**	**Latency Time**	**Total Data in Bytes to be Transmitted**	**Protocol used for transmission**	**Outgoing data Packet Size**	**Transmission Speed in Mbps**	**Time Taken to Transmit**	**Number of paths available for transmission**	**Micro Secs of time spent for transmission per path**	**Total time Spent within the device for receiving and Transmission**
N0010	Fan-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0011	Fan-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0012	Fan-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
CLUS1	Cluster Head-1	13	Wi-Fi,	48	11	34.91	1	34.91	CDMA	0.001	0.0000	13	CDMA	68	110	4.95	1	4.95	39.856
N0020	AC-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0021	AC-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0022	AC-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
CLUS2	Cluster Head-2	13	Wi-Fi,	48	11	34.91	1	34.91	CDMA	0.001	0.0000	13	CDMA	68	110	4.95	1	4.95	39.856
N0030	HU-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0031	HU-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0032	HU-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
CLUS3	Cluster Head-3	13	Wi-Fi,	48	11	34.91	1	34.91	CDMA	0.001	0.0000	13	CDMA	68	110	4.95	1	4.95	39.856
N0040	TEMP-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0041	TEMP-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
N0042	TEMP-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	1	34.91	34.913
CLUS4	Cluster Head-4	13	Wi-Fi,	48	11	34.91	1	34.91	CDMA	0.001	0.000	13	CDMA	68	110	4.95	1	4.95	39.856
BASI	Base Station-1	13	CDMA	68	170	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	110	4.95	1	4.95	8.145
N005A	Controller-1	156	CDMA	211	170	9.929	1	9.93	USB	0.001	0.003	156	USB	215	120	14.33	1	14.33	24.267
N006	Restful server	156	USB	215	480	3.583	1	3.583	NA	0.000	0.000	156	Wifi	191	11	138.91	1	138.91	142.492
N007	Gateway	156	Wifi,	211	11	153.455	1	153.455	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	167.384
N008	Access point	156	Wifi	211	11	153.455	1	153.455	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	167.384
N009	WEB server	156	Ethernet	211	100	16.88	1	16.88	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	30.809
N00A	Cloud	156	Ethernet	174	100	13.92	1	13.92	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	27.849
N00B	User Device	20	thernet	38	100	3.04	1	3.04	Ethernet	0.004	0.005	20	Ethernet	38	100	3.04	1	3.04	6.089
Total response Time							24	497.11									24	655.60	1152.798

4.4. Enhancing the Performance at the Device layer and Controller Layer - Revised IoT Network

The prototype IoT network shown in Fig. 2 is the modified network considering the device Layer and the Controller Layer through a multi-stage network in the device layer and providing the parallel computing considering both the base stations and Microcontrollers. Parallel paths are designed into the networks s to communicate between the devices and the controllers faster. Four Base stations and four controllers are added to the network. Three cluster heads are implemented within each of the clusters. The devices are connected to the cluster heads through a multi-stage network. This arrangement provided several communication paths between the devices and cluster heads. Three cluster heads are connected to two base stations to provide dual paths for communication. Again, parallel communication is provided between a base station and a set of Microcontrollers to implement parallelism at the controller level. All the controllers are connected to a services server through a USB interface. The revised network provides several paths for communication leading to high fault tolerance and availability of any paths for effecting communication simultaneously. Services server is connected to the gateway to be the onward connection to the Internet.

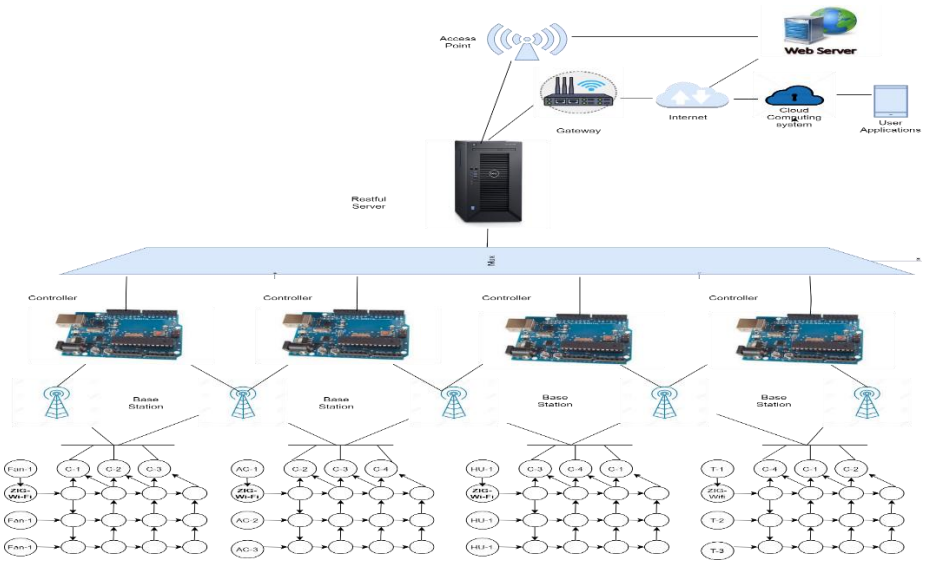

Fig. 2. Prototype IoT network with modifications effected at device and controller layer.

The detailed performance computations of an IoT network with changes made into the device layer and the controller layer are shown in Table 3. From Table 3, one can see that choice of Wi-Fi speed at 11Mbps, Wi-Fi data size 48 bytes per packet can be transmitted using the Cellular speed fixed at 170 Mbps, and data size fixed at 64 bytes/packet using zero latency time of data transmission at Cluster Head. However, 0.001 Microseconds of time will be sent for repacking and packaging into cellular data packets. It can be seen from Table 3 it took only 792 Microseconds of time to transmit 174 Bytes of payload as against 1152 Microseconds of time taken to transmit the size of payload that effecting to timesaving of 57%.

Table 3. Performance computations - Revised IoT (With Changes DEV +CNT) network.

Transmitting Device Code	Device Description	Reception							Conversion Time			Transmission						
		Total Data in Bytes received	Protocol used for receiving the -data	Incoming data Packet Size	Incoming speed used for receiving the data Mbps	Total time spent for reception in Microseconds	Number of Paths available for reception	Time Spent for receiving per path	Protocol used for conversion	Protocol Conversion time	Latency Time	Total Data in Bytes to be Transmitted	Protocol used for transmission	Outgoing data Packet Size	Transmission Speed in Mbps	Time Taken to Transmit	Number of paths available for transmission	Micro Secs of time spent for transmission per path	Total time Spent within the device for receiving and Transmission
N0010	Fan-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0011	Fan-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0012	Fan-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS1	Cluster Head-1	13	Wi-Fi,	48	11	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
N0020	AC-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0021	AC-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0022	AC-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS2	Cluster Head-2	13	Wi-Fi,	48	11	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
N0030	HU-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0031	HU-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0032	HU-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS3	Cluster Head-3	13	Wi-Fi,	48	11	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
N0040	TEMP-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0041	TEMP-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0042	TEMP-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS4	Cluster Head-4	13	Wi-Fi,	48	11	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
BASI	Base Station-1	13	CDMA	68	170	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	1	3.20	6.400
BAS2	Base Station-2	13	CDMA	68	170	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	2	1.60	4.800
BAS3	Base Station-3	13	CDMA	68	170	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	2	1.60	4.800
BAS4	Base Station-4	13	CDMA	68	170	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	2	1.60	4.800
BAS5	Base Station-5	13	CDMA	68	170	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	1	3.20	6.400
N005A	Controller-1	13	CDMA	68	170	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	6.404
N005B	Controller-2	13	CDMA	68	170	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	17.459
N005C	Controller-3	13	CDMA	68	170	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	6.404
N005B	Controller-4	13	CDMA	68	170	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	6.404
MUSB	Multi-USB Port	156	USB	215	480	3.583	4	0.896	NA	0.000	0.000	156	USB	72	480	1.20	1	1.20	2.096
N006	Restful server	156	USB	215	480	3.583	1	3.583	NA	0.000	0.000	156	Wifi	191	11	138.91	2	69.45	73.038
N007	Gateway	156	Wifi,	191	11	138.909	1	#####	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	152.838
N008	Access point	156	Wifi	191	11	138.909	1	#####	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	152.838
N009	WEB server	156	Ethernet	174	100	13.92	1	13.92	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	27.849
N00A	Cloud	156	Ethernet	174	100	13.92	1	13.92	Ethernet	0.004	0.005	156	Ethernet	174	100	13.92	1	13.92	27.849
N00B	User Device	20	Ethernet	38	100	3.04	1	3.04	Ethernet	0.004	0.005	20	Ethernet	38	100	3.04	1	3.04	6.089
Total response Time							43	405.41									52	375.63	792.192

5. Limitation of a Single Server in an IoT network

A layer that deals with user requests, the services layer, works like middleware in an IoT network. The services layer is usually developed using a server interfaced with a gateway on one side and the Microcontroller on the other side. In Fig. 2, it can be seen that the service's servers get connected to the Microcontroller through a USB interface-driven with a speed of 120 Mbps which means 15×10^6 Bytes driven per second. On average, if the data packet size of 15 Bytes is considered, and when the transaction is converted into XML standard, the request size will be nearly 100 bytes leading to a request load of 10,000 requests per second. A typical server designed with the 2.5 GHz speed of the processor can process 2500 requests per second, implying 40 Microseconds of processing one request. If all the 10,000 requests are to be processed by a single processor, it will take 10,000×40 microseconds of time which is a huge time not acceptable to any user. With a single server used in the IoT network, there will be a considerable delay in response time, which is unacceptable to the end-user.

6. Implementing RESTful Services Server within an IoT Network

The services server is implemented as a RESTful (Representational state transfer) Server (Application) to communicate using an API for availing different services. The REST full server implements a different API. The user can request to run the API through the initiation of the REST full server implements different APIs. The user can request to run the API by initiating an XML message transmitted through either HTTP or TCP/IP protocol. The architecture of a restful services server is shown in Fig. 3.

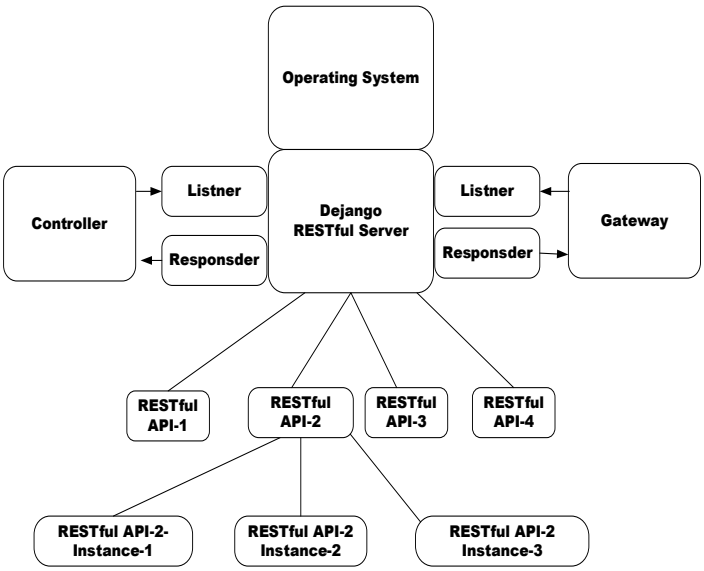

Fig. 3. Restful services server architecture.

A Listener program keeps waiting to receive a service request called through a gateway or controller in an XML message. The Listener receives the message, parses the API required, and the RESTful server will initiate the API in a separate thread by passing the requested parameters. The API will process the request, gets the output, converts the same to an XML message, and transmit the same to the user who initiated the request. When several users request the same service, several service instances are instantiated.

The application developer can develop different APIs and register the same with the Django server by providing the specification of the function in terms of the function name, arguments list and the return value, and the format in which the output must be returned. The DData is stored or retrieved using API functions through one of the database engines. Django REST Framework is used to implement the RESTful API. For the prototype application to be implemented, 12 different services (Status of the Fan1, Status of the Fan2, Status of the Fan3, Status of AC1, Status of AC2 Status of AC3, State of Humidity1, State of Humidity2, State of Humidity3, State of Temp1, State of Temp2, State of Temp3) are considered for implementing typical IoT network. Users can query any of the statuses, and the status is provided as a text string (ON/OFF) to the user. The number of instances to be handled depends on the traffic scheduled to each server.

A single restful service server is generally not sufficient. Therefore, many servers must be added to the system to cater to the load and respond within an acceptable time. A single restful service server is generally not sufficient. Therefore, many servers must be added to the system to cater to the load and respond within an acceptable time. A single restful service server is generally not sufficient. Therefore, many servers must be added to the system to cater to the load and respond within an acceptable time single restful service server is generally not sufficient. Therefore, many servers must be added to the system to cater to the load and respond within an acceptable time. Two thousand four hundred requests, when evenly distributed over

12 different types of requests, 200 instances for each of the API (request) have to be processed, which means the server should be able to cater to 2400 threads' massive processing requirement. Two thousand four hundred requests, when evenly distributed over 12 different types of requests, 200 instances for each of the API (request) have to be processed, which means the server should be able to cater to 2400 threads' massive processing requirement. Two thousand four hundred requests, when evenly distributed over 12 different types of requests, 200 instances for each of the API (request) have to be processed, which means the server should be able to cater to 2400 threads' massive processing requirement. Two thousand four hundred requests, when evenly distributed over 12 different types of requests, 200 instances for each of the API (request) have to be processed, which means the server should be able to cater to 2400 threads' massive processing requirements.

7. Improving the Design of the IoT Network at the Service Layer for Enhancement of the Response Time

The response time can be improved by adding more servers into the services layer. In that case, many issues are to be addressed. The load on the servers needs to be managed such that all the servers must be equally loaded to get an acceptable response time. Figure 4 shows the redesign of the IoT network with multiple servers to address the service request traffic.

The network explained earlier has been improved further by intruding a load balancer to which the four controllers are connected through a USB interface, and four restful servers are connected to the output side of the Balancer. The restful servers are identical, and each server can provide any service designed into the IoT network. All the restful servers are connected to the Internet through a multi-channel gateway.

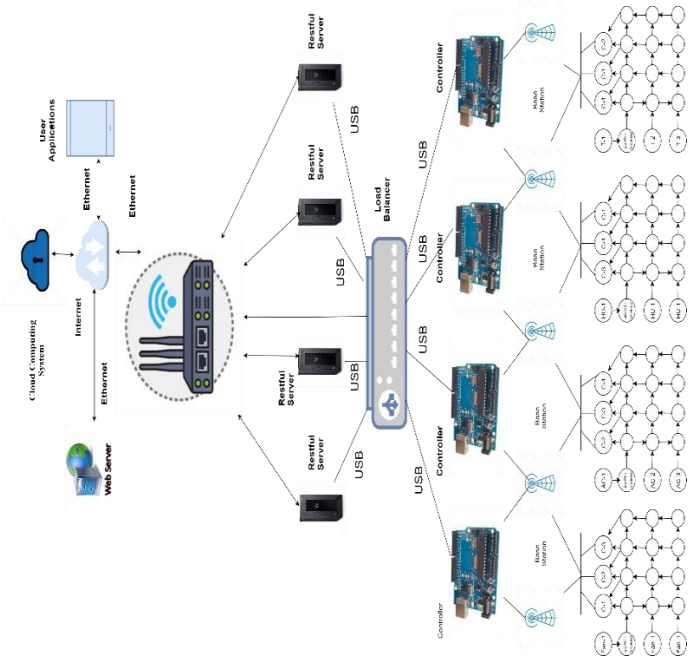

Fig. 4. Revised IoT network - more servers in service layer with service requests distributed through a load balancer.

7.1. Load balancing the service requests

A load balancer is a hardware-software solution mainly meant to maintain even request load on each server. The software architecture of the load balancer is shown in Fig. 5.

Four listeners keep communicating with its peer controller side Transmitter within the Load Balancer. The listeners receive a request, assign a priority based on the type of request, write a priority based on the type of request, write a priority based on the type of request, write a priority based on the request, and write it into a priority queue. Priorities are assigned to the requests as per their urgency of servicing. The requests within the queue are ordered according to the queue. A dispatcher program is an intelligent program that keeps track of the load on each physical server's physical servers by maintaining a memory-resident table, as shown in Table 4. A loading strategy is to select the less loaded server until the server reaches the peak load. The loading of the requests must also be done in a round-robin fashion and skipping the server that has reached the maximum load.

Architecture reveals how different software components are related and interact to accomplish overall requirements. There is one program that runs in each controller to transmit the data. Four individual threads of the Load balancer keep listening on the controller port and receive the data as and when transmitted by the controller. The listers write the data transmission requests into a queue by attaching priority based on the data centricity. The dispatcher program keeps account of the load on each server and schedules the requests to the server, which is less loaded. The server writes the output to the gateway channel to which the server is connected.

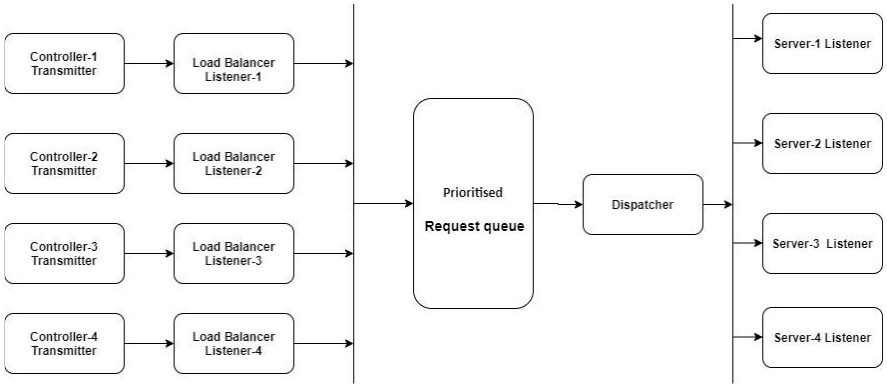

Fig. 5. Load balancer architecture.

Table 4. Memory resident table for keeping status of the load.

Server-ID	Number of requests under process	Maximum Capacity of the server (Number of requests	Load on the server in percentage
1	2000	2500	80
2	1900	2500	76
3	1800	2500	72
4	1700	2500	68
Total	7400	10000	74

The number of servers required can be decided based on the Maximum Load expected. From Table 5, three servers are sufficient to handle 7400 requests, with a maximum of 7500 requests. There is a need to estimate the number of concurrent

requests that the services layer of the IoT network must cater to. If the load varies heavily, then one must have the strategy to decide on the number of maximum servers required. There should also be a strategy to manage the servers such that the required number of servers only be connected that are sufficient to manage the load at any point in time

Algorithm (Load balancer Listener process)

{
QUEUE [i] // is the request queue where i stands for the request number.
PRIORITY [j] // is the priority to be set for the i^{th} service as per Table 5
Receive the request packet, which is in the XML message format.
Parse the message, the Kind of reservation requested.
Attach priority to the request and write into the queue with incremented for every request.
}

Table 5. Priority set for the services.

Service Number	Service	Priority
1.	AC1ONOFF	950
2.	AC2ONOFF	900
3.	AC3ONOFF	850
4.	FAN1ONOFF	800
5.	FAN2ONOFF	750
6.	FAN3ONOFF	700
7.	SET-HUMIDITY1	650
8.	SET-HUMIDITY2	600
9.	SET-HUMIDITY3	550
10.	SET-TEMP1	500
11.	SET-TEMP2	450
12.	SET-TEMP3	400

Algorithm (Despatcher process)

LOAD [i] // Load on each of the servers where i stands for server
THRESHOLD[j] // Threshold load on each of the server j
QUEUE [k] where k stands for a request placed in the QUEUE
CURRQST // Current Request

```
While (True)
{
    Read CURRQST
    i= 0
        LOOP
      {
        i = i+1
        if (i > 12) then i=0
        If LOAD[i]  <  THRESHOLD[i] THEN
        {
        LOAD[i] = LOAD[i] +1
        TRANSMIT, the request to SERVER[i]
        If SERVICECOMPLETED = TRUE then LOAD[i] = LOAD[i]-1
        }
    }
}
```

7.2. Discussion and results

The performance computation considering the revised IoT network considering that the IoT network is developed using four servers is shown in Table 6 which is placed after references section. Comparative analysis of response time computation considering typical scenarios is shown in Table 7. The response time is the least when the data size is 468 bytes considering the changes made in the device layer + Controller Layer and considering device layer + Controller Layer + services Layer. With the Multi-servers introduced in the services layer, the response time is decreased from 3.63 Microseconds/byte to 1.14 Microseconds per byte, an Improvement in response time by 69.5%.

Table 6. Performance computations - Revised IoT (With Changes DEV +CNT+Services) network.

		Reception							Conversion Time			Transmission						
Transmitting Device Code	Device Description	Total Data in Bytes received	Protocol used for receiving the data	Incoming data Packet Size	Incoming speed used for receiving the data Mbps	Total time spent for reception in Microseconds	Number of Paths available for reception	Time Spent for receiving per path	Protocol used for conversion	Protocol Conversion time	Latency Time	Total Data in Bytes to be Transmitted	Protocol used for transmission	Outgoing data Packet Size	Transmission Speed in Mbps	Time Taken to Transmit	Number of paths available for transmission	Micro Secs of time spent for transmission per path	Total time Spent within the device for receiving and Transmission
N0010	Fan-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0011	Fan-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0012	Fan-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS1	Cluster Head-1	13	Wi-Fi,	48	11	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
N0020	AC-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0021	AC-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0022	AC-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS2	Cluster Head-2	13	Wi-Fi,	48	11	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
N0030	HU-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0031	HU-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0032	HU-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS3	Cluster Head-3	13	Wi-Fi,	48	11	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
N0040	TEMP-1	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0041	TEMP-2	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
N0042	TEMP-3	13	NA	13	NA	0.001	1	0.001	Wi-Fi	0.001	0.002	13	Wifi	48	11	34.91	2	17.45	17.459
CLUS4	**Cluster Head-4**	**13**	**Wi-Fi,**	**48**	**11**	34.91	2	17.45	CDMA	0.001	0.0000	13	CDMA	68	170	3.20	2	1.60	19.056
BASI	**Base Station-1**	**13**	**CDMA**	**68**	**170**	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	1	3.20	6.400
BAS2	**Base Station-2**	**13**	**CDMA**	**68**	**170**	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	2	1.60	4.800
BAS3	**Base Station-3**	**13**	**CDMA**	**68**	**170**	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	2	1.60	4.800
BAS4	**Base Station-4**	**13**	**CDMA**	**68**	**170**	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	2	1.60	4.800
BAS5	**Base Station-5**	**13**	**CDMA**	**68**	**170**	3.200	1	3.20	NA	0.000	0.000	13	CDMA	68	170	3.20	1	3.20	6.400
N005A	**Controller-1**	**13**	**CDMA**	**68**	**170**	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	6.404
N005B	**Controller-2**	**13**	**CDMA**	**68**	**170**	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	17.459
N005C	**Controller-3**	**13**	**CDMA**	**68**	**170**	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	6.404
N005B	**Controller-4**	**13**	**CDMA**	**68**	**170**	3.200	2	1.60	USB	0.001	0.003	13	USB	72	120	4.80	1	4.80	6.404
MUSB	**Load Balancer**	**52**	**USB**	**288**	**480**	4.800	4	1.200	NA	0.000	0.000	52	USB	111	480	1.85	1	1.85	3.050
N006A	**Server-1**	**13**	**USB**	**72**	**480**	1.200	1	1.200	NA	0.002	0.002	13	Wifi	48	11	34.91	2	17.45	18.659
N006B	**Server-2**	**13**	**USB**	**72**	**480**	1.200	1	1.200	NA	0.002	0.002	13	Wifi	48	11	34.91	2	17.45	18.659
N006C	**Server-3**	**13**	**USB**	**72**	**480**	1.200	1	1.200	NA	0.002	0.002	13	Wifi	48	11	34.91	2	17.45	18.659
N006D	**Server-4**	**13**	**USB**	**72**	**480**	1.200	1	1.200	NA	0.002	0.002	13	Wifi	48	11	34.91	2	17.45	18.659
N007	**Gateway**	**52**	**Wifi,**	**192**	**11**	139.636	1	139.63	Ethernet	0.004	0.005	52	Ethernet	70	100	5.60	1	5.60	145.245
N009	**WEB server**	**52**	**Ethernet**	**70**	**100**	5.6	1	5.6	Ethernet	0.004	0.005	52	Ethernet	70	100	5.60	1	5.60	11.209
N00A	**Cloud**	**52**	**Ethernet**	**70**	**100**	5.6	1	5.6	Ethernet	0.004	0.005	52	Ethernet	124	100	9.92	1	9.92	15.529
N00B	**User Device**	**20**	**Ethernet**	**38**	**100**	3.04	1	3.04	Ethernet	0.004	0.005	20	Ethernet	38	100	3.04	1	3.04	6.089
Total response Time							45	252.11									57	342.08	605.352

Table 7. Comparisons of performance considering the prototype model and the revised model with change in data size.

Type Model	Total data transmitted in bytes	Response Time in Microseconds	Time takes per byte Transferred in Microseconds
Prototype model	156	1152.80	7.39
Device + Controller Modified Model	156	792.19	5.08
	312	1244.48	3.99
	468	1696.77	3.63
	1560	4862.78	3.12
Device + Controller + Services Later Modified Model	156	605.35	3.88
	312	437.90	1.40
	468	533.71	1.14
	1560	1204.41	0.77

8. Conclusions and Future Work

Performance of the IoT network is the key, especially when such networks are built for the aerospace and automobile sectors. In this paper, performance improvement considering the Device layer + Controller Layer + Services layer has been presented. The performance improvements have been achieved considering each layer or set of layers or all the layers together. 69.5% performance improvement has been achieved by using four servers, each capable of handling a Load of 2500 Requests per second. And the load is distributed through an Intelligent Load Balancer. The servers could be estimated to meet the Highest traffic and manage the servers to be hooking on the network based on the actual load dynamically. It has been found a data load of 468 bytes per request will yield optimum response time.

Future work

Further research should be carried out to find an optimum number of servers that must be used for varying degree workloads in terms of the number of service requests that must be handled, and the cost incurred for implementing such a mechanism.

Further research could be done by considering different performance enhancement methods implemented at the gateway level integrated with the methods implemented in the services levels.

Further investigations are also to be carried out to find the possibility of using an ethernet interface between the Controllers and the Load balance and the connectivity between the servers and the gateway.

References

1. Oracle. (2022). What is the internet of things (IoT)? IoT topics. December 7, 2022, https://www.oracle.com/in/internet-of-things/what-is-iot/
2. Khachane, J.L.; and Desai, L.R. (2015). Survey paper on web services in IoT. *International Journal of Science and Research* (*IJSR*), 4(12), 635-637.
3. Zhang, L.; Yu, S.; Ding, X.; and Wang, X. (2014). Research on IoT RESTful web service asynchronous composition based on BPEL. *Proceedings of the* 2014 *Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, Hangzhou, China, 62-65.
4. Pan, L.; Xu, M.; Xi, L.; and Hao, Y. (2016). Research of livestock farming IoT system based on RESTful web services. *Proceedings of the* 2016 5*th International Conference on Computer Science and Network Technology* (*ICCSNT*). Changchun, China, 113-116.
5. Laine, M. (2011). RESTful Web services for the internet of things. internet hosted PDF, December 7, 2022, http://www.sensinode.com/EN/products/nano service.html.
6. Porter, P.; Yang, S.; and Xi, X. (2019). The Design and implementation of a RESTful IoT service using the MERN stack. *Proceedings of the* 2019 *IEEE* 16*th International Conference on Mobile Ad Hoc and Sensor Systems Workshops* (*MASSW*), Monterey, CA, USA, 140-145.
7. Sowmya, K.V.; and Sastry, J.K.R. (2018). Performance evaluation of IoT systems - basic issues. *International Journal of Engineering & Technology*, 7(2.7), 131-137.

8. Sowmya, K.V.; and Sastry, J.K.R. (2021). Performance optimization within the device layer of IoT networks. *Journal of Engineering Science and Technology* (*JESTEC*), 16(6), 5087-5109.
9. Sowmya, K.V.; and Sastry, J.K.R. (2022, in press). Improving the performance of heterogeneous IoT networks through multi-stage and parallel computing systems. *SCIENTIA-IRANICA*, 1-37.
10. Bhupathi, CH.; Sastry, J.K.R. (in press). Hybrid models for computing fault tolerance of IoT networks *TELKOMNIKA* (*Telecommunication Computing Electronics and Control*).
11. Sasi Bhanu, J.; Sastry, J.K.R.; Venkata, S.K.P.; Venkata, S.B.; and Sowmya, K.V. (2019). Enhancing performance of IoT networks through high-performance computing. *International Journal of Advanced Trends in Computer Science*, 8(3), 432-442.
12. Sastry, J.K.R.; Ramya, G.S.; Niharika, V.M.; and Sowmya, K.V. (2019). Performance optimization of IoT networks within gateway layer. *Recent Advances in Computer Science and Communications*, 13(6), 1338-1346.
13. Ananthi, J.V.; Chinnalagi, V.; Murugeswarai, R.; Priyadarsini, T.; and Rajalakshmi, K. (2017). An effective performance of smart sensor network using IoT. *International Journal of Advance Research, Ideas and Innovations in Technology*, 3(2), 638-645.
14. Ashwini, M.; and Rakesh, N. (2017). Enhancement and performance analysis of LEACH algorithm in IOT. *Proceedings of the* 2017 International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 1-5.
15. Behera, T.M.; Samal, U.C.; and Mohapatra, S.K. (2018). Energy-efficient modified LEACH protocol for IoT application. *IET Wireless. Sensor Systems*, 8(5), 223-228.
16. Bhandari, S.; Sharma, S.K.; and Wang, X. (2017). Cloud-assisted device clustering for lifetime prolongation in wireless IoT networks. *Proceedings of the* 2017 *IEEE* 30*th Canadian Conference on Electrical and Computer Engineering* (*CCECE*), Windsor, ON, Canada, 1-4.
17. Choi, D.-K.; Jung, J.-H.; and Koh, S.-J. (2018). Enhanced cluster-based CoAP in Internet-of-Things networks. *Proceedings of the* 2018 *International Conference on Information Networking* (*ICOIN*), Chiang Mai, Thailand, 652-656.
18. Choi, D.-K.; Jung, J.-H.; Kang, H.-W.; and Koh, S.-J. (2017). Cluster-based CoAP for message queueing in Internet-of-Things networks. *Proceedings of the* 2017 19*th International Conference on Advanced Communication Technology* (*ICACT*), Pyeong Chang, Korea (South), 584-588.
19. Geetha, V.; Kallapur, P.V.; and Tellajeera, S. (2012). Clustering in wireless sensor networks: Performance comparison of LEACH & LEACH-C protocols using NS2. *Procedia Technology*, 4, 163-170.
20. Khedira, S.E.L.; Nasri, N.; Wei, A.; and Kachori, A. (2014). A new approach for clustering in wireless sensors networks based on LEACH. *Procedia Computer Science*, 32, 1180-1185.
21. Liu, Y.; and Zhou, Y. (2018). Development of distributed cache strategy for the analytic cluster in an Internet of Things system. *Proceedings of the* 2018 *IEEE* 15*th International Conference on Networking, Sensing and Control* (*ICNSC*), Zhuhai, China, 1-6.

22. Tao, X.; and Ji, C. (2015). Clustering massive, small data for IoT. Proceedings of the 2014 2*nd International Conference on Systems and Informatics* (*ICSAI* 2014). Shanghai, China, 974-978.
23. Kwon, M.; and Park, H. (2016). The cluster formation strategies for approximate decoding in IoT networks. *Proceedings of the* 2016 *International Conference on Information Networking* (*ICOIN*), Kota Kinabalu, 366-368.
24. Puschmann, D.; Barnaghi, P.; and Tafazolli, R. (2017). Adaptive clustering for dynamic IOT data streams. *Journal of Internet of Things*, 4(1), 64-74.
25. Behera, T.M; Mohapatra, S.K.; Samal, U.C.; Khan, M.S.; Daneshmand, M.; and Gandomi, A.H. (2019). Residual energy-based cluster-head selection in WSNs for IoT application. *Journal of Internet Things*, 6(3), 5132-5139.
26. Ju, Q.; and Zhang, Y. (2017). Adaptive clustering for the internet of battery-less things. *Proceedings of the* 2017 *IEEE Wireless Communications and Networking Conference* (*WCNC*), San Francisco, CA, USA, 1-6.
27. Puschmann, D.; Barnaghi, P.; and Tafazolli, R. (2017). Adaptive clustering for dynamic IoT data streams. *IEEE Internet of Things Journal*, 4(1), 64-74.
28. Zhao, S.; Yu, L.; Cheng, B.; and Chen, J. (2017). IoT service clustering for dynamic service matchmaking. *Sensors*, 17(8), 1727.
29. Sastry, J.K.R; and Bhupathi (2020). Enhancing fault tolerance of IoT networks within device layer. *International Journal of Emerging Trends in Engineering Research*, 8(2), 491-509.
30. Sowmya, K.V.; Chandu, A.; Nagasai, A.; Sri Harsha Preetham, C.H.; and Supreeth, K. (2020). Smart home system using clustering based on internet of things. *Journal of Computational and Theoretical Nanoscience*, 17(5), 2369-2374.
31. Geethika, R.A.; Upendra, Y.; Sastry, J.K.R.; and Bhupathi (2020). An approach to compute fault tolerance of an IoT network having clustered devices using crossbar networks. *International Journal of Emerging Trends in Engineering Research*, 8(4), 987-1004.
32. Sastry, J.K.R.; Tejasvi, N.S.T.; and Aparna, J. (2017). Dynamic scheduling of message flow within a distributed embedded system connected through an RS485 network. *ARPN Journal of Engineering and Applied Sciences*, 12(9), 2809-2817.
33. Pavithra, T.; and Sastry, J.K.R. (2018). Strategies to handle heterogeneity prevalent within an I.O.T.-based network. *International Journal of Engineering & Technology*, 7(2.7), 77-83.
34. Rajasekhar, J.; and Sastry, J.K.R. (2020). Building composite embedded systems based networks through hybridization and bridging I^2C and CAN. *Journal of Engineering Science and Technology* (*JESTEC*), 15(2), 858-881.
35. Krishna, M.S.R.; Sastry, J.K.R.; and Bhanu, J.S. (2017). Building fault tolerance within wireless sensor networks: A butterfly model. *Research Journal of Applied Sciences*, 12(2), 139-147.
36. Priya, B.V.; and Sastry, J.K.R. (2018). A comparative analysis of the methods used to build information / content-centric networks over software-defined networks. *International Journal of Engineering & Technology*, 7(2.7), 746-753.
37. Rajasekhar, J.; and Sastry, J.K.R. (2020). On developing high-speed heterogeneous and composite embedded system networks through multi-

master interface. *International Journal of Advanced Computer Science and Applications*, 11(12), 320-333.

38. AbdelBaky, M.; Parashar, M.; Kim, H.; Jordan, K.E.; Sachdeva, V.; Sexton, J.; Jamjoom, H.; Shae, Z.-Y.; Pencheva, G.; Tavakoli, R.; and Wheeler, M.F. (2012). Enabling high computing performance as a service. *Computer*, 45(10), 72-80.
39. El Baz, D. (2014). IoT and the need for high-performance computing. *Proceedings of the* 2014 *International Conference on Identification, Information and Knowledge in the Internet of Things*. Beijing, China, 1-6.
40. Satpute; S.; and Deora, B.S. (2015). Improving the performance of Internet of Things by using local IoT controller unit. *Proceedings of the* 2015 *International Conference on Green Computing and Internet of Things* (*ICGCIoT*), Greater Noida, India, 1328-1330.
41. Muralitharan, D.B.; Reebha, S.A.B.; and Saravanan, D. (2017). Optimization of performance and scheduling of HPC applications in the cloud using cloudsim and scheduling approach. *Proceedings of the* 2017 *International Conference on IoT and Application* (*ICIOT*). Nagapattinam, India, 1-6.
42. Rajaei, H.; and Jamalian, S. (2016). HPC as a service in education. *Proceedings of the ASEE's* 123*rd Annual conference and expositions*. New Orleans, L.A.
43. Onoriode, U.; and Gerald, K. (2018). IoT architectural framework: connection and integration framework for IoT systems, electronic *First workshop on Architectures, Languages and Paradigms for IoT EPTCS* 264, doi:10.4204/EPTCS.264.1, Lancaster, UK, 1-17.
44. Murty, A.S.R.; Teja, K.; and Naveen, S. (2018). Lathe performance monitoring using IoT. *International Journal of Mechanical Engineering and Technology* (*IJMET*), 9(4), 494-501.
45. Vijaya, M.K.; Prabu, A.V.; Prathyusha, S.M.; and Varakumari, S. (2018). Performance monitoring of UPS battery using IoT. *International Journal of Engineering & Technology*, 7(2.7), 352-355.
46. Poonam, J.S.; Pooja, S.; Sripath, R.K.; Abhilash, K.; Arvind, B.V. (2018). Implementation of asymmetric processing on multi-core processors to implement IoT applications on GNU/Linux framework. *International Journal of Engineering & Technology*, 7(2.7), 710-713.