IMPROVEMENT IN QUALITY OF SERVICE IN IOT ENABLED APPLICATIONS

A Thesis Submitted
in Fulfillment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

In

WIRELESS COMMUNICATION & SENSOR NETWORK

By

SHIVANGI VERMA
(Roll no. 1180454005)

Under The Supervision Of
MR. ASHUTOSH RASTOGI
Assistant Professor
BBD UNIVERSITY

To the
SCHOOL OF ENGINEERING
BABU BANARASI DAS UNIVERSITY
LUCKNOW
MAY, 2020

CERTIFICATE

It is certified that the work contained in this thesis entitled **"ENHANCEMENT OF QUALITY OF SERVICE IN IOT ENABLED APPLICATIONS",** by **SHIVANGI VERMA** for the award of **Master of Technology** from Babu Banarasi Das University has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Supervisor	Head of Department
Mr. Ashutosh Rastogi	Dr. Nitin Jain
(Assistant Professor)	(Associate Professor)
BBD University	BBD University

Date:

ABSTRACT

The scope of internet is expanding beyond computing and computer devices being connected. It is envisaged to provide advanced level of services to society, businesses, etc. Internet of Things connects everyday 'things' embedded with sensors and electronic software to the internet enabling them to collect and exchange data. IOT will lead to unification of technologies viz. cloud computing, low power embedded systems, machine learning, big data and networking; and an era of ubiquity which means connectivity at anyplace and anytime. An enormous amount of data would be generated by billions of users which has to be handled and processed which requires a much wider and more complex network than the present day internet. The heterogeneous networks, gigantic number of links and information exchange within edge nodes in IOT makes the system very complex and hence creates hurdles for satisfaction of the dynamic QoS requirements. In such enormous smart devices connectivity it is of utter important to maintain better quality of service (QoS).

This thesis proposes weighted fair queueing based packet scheduling along with dynamic bandwidth allocation based on average queue length to improve the quality of service in IoT enabled applications. This predictive model employs AEWMA (Adaptive Exponentially Moving Average) method for the calculation of average queue length which follows a nonlinear method to find the average queue length so that accidental network bursts can also be considered. The scheduler composes of two different steps: firstly, the weighted fair queueing policy will take the previous slot average queue length into consideration for the calculation of current slot average queue length. Then, the bandwidth will be allocated by multiplying the average queue length (which is calculated based on the current queue size and the previous average queue size) with a coefficient. The proposed approach is tested on network model where we have data ranging from few kilobits to megabits based on which the packet loss, delay and jitter performance of the proposed scheme is measured.

ACKNOWLEDGEMENT

It is proud privilege to express a profound sense of gratitude and whole hearted thanks to my respected guide **Mr. Ashutosh Rastogi (Assistant Professor),** Department of Electronics & Communication Engineering, Babu Banarasi Das University, Lucknow, for his expert guidance, invaluable suggestions and encouragement throughout the thesis work. Finally, my heartfelt appreciation goes out to my parents and brother for their encouragement, prayers and good wishes, which helped me to write this thesis and obtain my Maters degree.

I owe a great deal of appreciation to **Dr. Nitin Jain (HOD),** Department of Electronics & Communication Engineering, Babu Banarasi Das University, Lucknow, for his keen interest and valuable guidance during the course of thid work. Their continuous encouragement inspired me throughout the work.

Shivangi Verma
Roll.no: 1180454005
BBD University

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IOT: Internet of Things

QOS: Quality of Service

VoIP: Voice over Internet Protocol

AC: Admission Control

RSVP: Resource Reservation Protocol

FIFO: First-in-First-out

FQ: Fair Queuing

PQ: Priority Queuing

WRR: Weighted Round Robin

WFQ: Weighted Fair Queuing

EWMA: Exponentially Weighted Moving Average

AEWMA: Adaptive Exponentially Weighted Moving Average

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The scope of internet is expanding beyond computing and computer devices being connected. It is envisaged to provide advanced level of services to society, businesses, etc. The term 'Internet of Things' was first coined by Kevin Ashton, a British entrepreneur, in 1999. It can be viewed as amalgamation of sensors and connected devices with persons, processes and technology to empower data manipulation, remote monitoring and trend evaluation of such devices. The early growth of connected devices was sluggish and only improved with the expansion of internet along with the influence of corporate and academic bodies like IBM, Siemens, Cisco, MIT, Cambridge, etc. Olivetti Research's "active badge" developed between 1989 and 1992 was said to be one of the first connected devices by a few, the purpose of which was to track a person's location with the help of infrared signals. It marked the shift from a physical world towards a digital world. John Romkey and Simon Hacket in 1990 created a connected toaster, the first internet 'device' which operated over an IP network using Simple Network Management Protocol (SNMP). Siemens M2M "One" technology empowered machines to communicate over wireless networks. Market players and researchers began to search for efficient methods to monetize the new technologies in wireless connectivity, and especially in M2M. A transformation in the way of business was witnessed in early 2000. With easy access to internet and smarter phones, rather than searching for a store, customers were willing to opt for online solutions. Companies like Uber and Amazon capitalized by providing doorstep services and structuring business around online customer interaction. The next step was a shift from the digital to a connected world. Market players started aiming

towards being connected and it took not long before connected homes, connected healthcare, connected lifestyle, connected car, connected devices became abuzz. The Internet of Things progressively began its accession by enabling the devices to communicate with each other. Starting from a cellular phone which can only connect to the internet, to houses that can automatically switch on the air conditioner for its owner depending on the time at which she/he may arrive keeping in view the roadways traffic; devices can also make intelligent decisions based on the data available to them marking a shift from the connected to an intelligent world. Smart watches can inform the medical practitioners about the medical condition of their patients which will in turn help them to provide timely treatment. Cars can directly request a software upgradation from the manufacturers without requiring the owner's intervention. The substantial amount of data generated allows connected devices and systems to meet and adjust to their needs in a self-sustained manner and offer end users a desirable quality of service (QoS). These services range from applications making our day-to-day life easier and comfortable to delivering benefits of emission reduction, energy saving, security improvement, healthcare services, etc.

The Internet of Things connects everyday 'things' embedded with sensors and electronic software to the internet enabling them to collect and exchange data[1]. It has grown into one of the vital technologies the present times. Since we can connect day to day objects ---cars, household appliances, gadgets —to the internet, smooth communication is possible between processes, people and things. Technological evolution, enhanced wireless connectivity, better speed and access to internet, improved computational capabilities and network infrastructure laid the path for connected devices and IOT. By means of low-cost computing, the cloud, analytics, big data and mobile technologies, physical things can gather and exchange data with marginal human intervention. In today's connected world, digital systems can store, observe, and regulate interaction between connected things. The physical world encounters the digital world and cooperate with each other. IOT will lead to unification of technologies viz. cloud computing, low power embedded systems, machine learning, big data and networking and create an era of ubiquity which means connectivity at anyplace and anytime. An enormous amount of data would be generated

by billions of users which has to be handled and processed which requires a much wider and more complex network than the present day internet. Cisco estimated that by 2020, 50 billion things will be connected to internet. We can understand the basic working of an IOT system by its four fundamental components [2]:

1. Sensors/Connected Devices: In an IoT environment various machines, sensors, actuators or other connected gear intended devices perform given action of collecting data of various degrees of complexity from its surroundings such as any temperature and pressure readings or images captured by a CC TV. A device can have multiple sensors bundled together to perform more functions than simply sensing data viz. communicating with one another, carrying out operations, transmitting and receiving data to/from the server/cloud. For example- smartphones have multiple sensors such as camera, GPS, accelerometer, etc. Fig. 1.1 shows the basic concept of an IoT system.

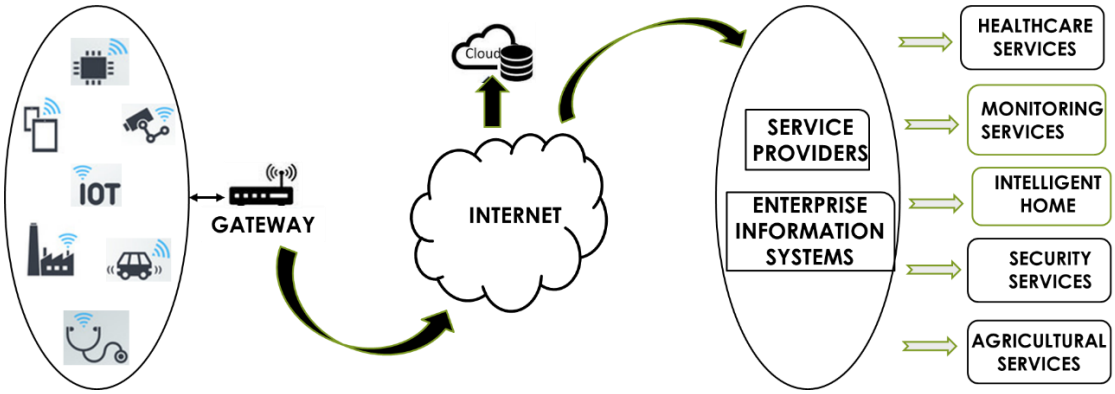

Fig. 1.1. Basic Concept of an IoT System

2. Connectivity: Based on network, this is the means through which the devices exchange data with one another and communicate the server/cloud. The data collected by sensors/devices is then sent to cloud by various ways: cellular networks, satellite, Wi-Fi, Bluetooth, low-power wide area networks etc. or directly to the internet via Ethernet. Whichever option we choose, it will have some specifications and tradeoffs between bandwidth, power consumption, reliability, network agility, security and range. Therefore, choosing an appropriate connectivity option is important for any particular scenario. An

Internet of Things (IoT) gateway is a physical device or software program that serves as the connection point between the cloud and controllers, sensors and intelligent devices. All data moving to the cloud, or vice versa, goes through the gateway, which can be either a dedicated hardware appliance or software program. An IoT gateway may also be referred to as an intelligent gateway or a control tier.

3. Data Processing: After the data reaches the cloud, processing is performed on the acquired data. The cloud is a large inter-connected network of servers that performs services for people and businesses. Henceforth, an action may be performed without user intervention or with the help of user interface in any input is required.

4. User Interface: After that, the information is sent to the user to provide an insight into the workings of the whole system and offering ways to interact with it. The main task of the application layer is to visualize the information received from the software and present it in a user-friendly way by methods such as notifying through e-mails or texts, triggering alarm on phones or the user can check in on the IoT systems via an interface. There are also cases where some actions perform automatically. By establishing and implementing some predefined rules, the entire IOT system can adjust the settings automatically and no human has to be physically present.

A real life example showing the working of an IoT system is the end-user requesting the temperature readings of his house from any remote location. Suppose an air conditioner is installed in the house. The temperature sensor of the AC will be connected to the internet via a gateway with the help of a cloud infrastructure which maintains comprehensive records of every connected device such as their id, status, the count of device access, last time of device access, etc. The connection of the device to the cloud is then implemented. The end-user can communicate with the device (AC) via the cloud through a mobile application. The cloud infrastructure will receive the request along with authentication and device information to guarantee cybersecurity. After authentication, the cloud sends the request through gateways to appropriate sensor network. The temperature sensor then responds to the request by observing the current room temperature and reverting it to the cloud which then identifies the end user and forwards the required information to the mobile application.

IoT transforms ‘things’ from being traditional to smart and helps us to achieve true potential of technology. The features of IoT are:

1. **Connec**t- It connects various things to IoT platform. This includes device virtualization and end point management. Device virtualization means standardized integration of devices with the IOT enterprise platform present on cloud/server. It is needed to ensure that a certain level of standard is present on the device so that it can connect to the IOT platform. The devices generate an enormous amount of data which helps us to understand and improve the system. Hence, high speed, reliable, secure and bidirectional communication between the cloud and devices is required. End point management means identification of devices from which data/command is coming and how it has to be processed, manage device end point identity, metadata and lifecycle states for all the devices.

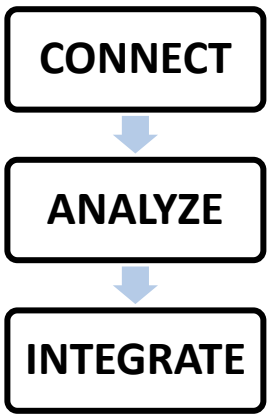

Fig. 1.2. Features of IoT

2. **Analyze**- Analyze the data collected and use it to build business intelligence. This includes stream processing, data enrichment and event storage. Stream processing is real time analysis of incoming data stream with event aggregation, filtering and correlation. Data enrichment means identification and processing of raw data streams with contextual information and generate composite streams which can be taken forward for future understanding. Event store is to query and visualize massive amount of data with integrated cloud service support and enable big data analytics.

3. **Integrate**- Integrate various models to improve user experience. This includes enterprise connectivity to dynamically dispatch critical IOT data and events to applications and process flow, integration with REST API for efficient and easier communication between enterprise, platform and 'things' & command and control integration to send data to devices from enterprise, mobile applications, voice based recognition; independent of device connectivity.

1.2 MOTIVATION

According to a forecast by the IoT connections outlook ("IoT connections outlook, Ericsson Mobility Report November 2019") there would be near about 24.9 billons Internet of Things (IoT) connections compared to 8.1 billon population by the year 2025 [3].

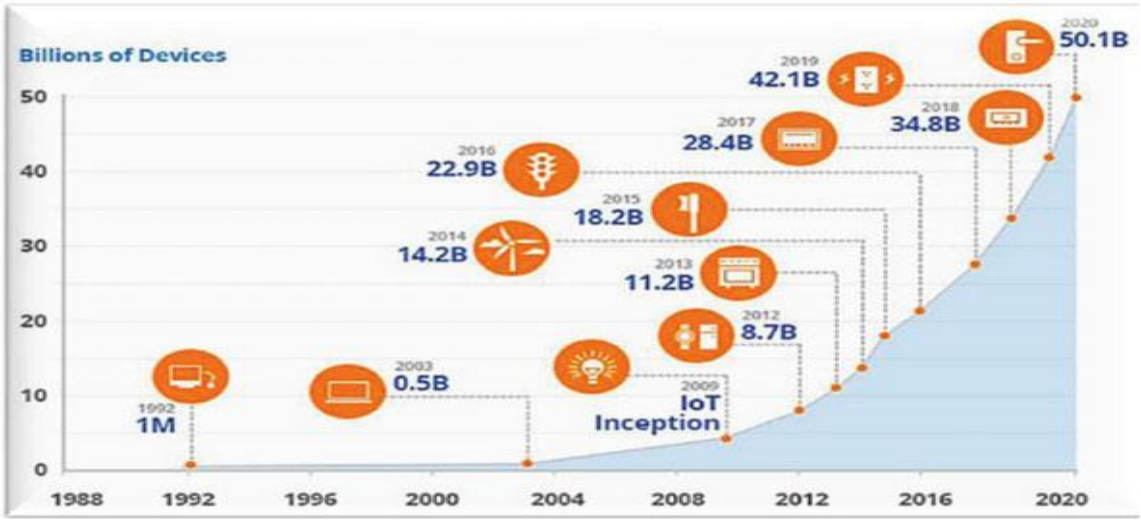

Fig. 1.3 Growth of IoT

The growing number of innovative and exciting applications and services are continuously been developed with the help to technologies like radio frequency identification, wireless local area network, wireless sensor network, mobile networks and internet which help in improving the efficacy and reducing the cost of processes[4]. The heterogeneous networks, gigantic number of links and information exchange within edge nodes in IOT makes the system very complex and hence creates hurdles for satisfaction of the dynamic QoS requirements. IoT offers the means, tools and environment for service creation but

suffers from common difficulties like component failure and limited energy. It is always desirable to provide service to application without any compromise in service quality. Generally two kinds of applications run in an IOT scenario, one, which needs throughput but is delay-tolerant and second, is delay sensitive and requires different levels of bandwidth for QOS requirement. QOS manages system capabilities and resources to offer efficient and desirable service to end-users. It enables customers to get a clear visibility regarding the usability and performance of services provided by the service provides. QoS metrics like jitter, bandwidth, throughput and efficiency, network connection time, monetary cost, availability, security and privacy, interoperability, service level agreement, monitoring, reliability, long term stability, response range, sensitivity, precision, power consumption, mobility support, scalability, pricing, etc. [5-6] will help the end-users to recognize the suitable IOT service for their applications and provide optimization of QOS. Different researches have covered various dimensions of IOT but QoS in IOT have still received slight consideration.

1.3 MAIN CONTRIBUTION AND OUTLINE OF THESIS

In the real world, the IoT network consists of various types of devices connected in a network like sensors with critical information, CCTV surveillance with huge amount of data, smart wearable gadgets. Nowadays the number of physical devices (IoT devices) are rapidly growing like wearable smart devices, smart video surveillance, security devices, medical applications, etc. which results into various streams of data which are stored on to the different cloud based servers so that the data can be used for various analytics and can be retained for a long period of time. In such enormous smart devices connectivity it is of utter important to maintain better quality of service (QoS). The internet based network model approach like DiffServe, IntServe cannot be used for addressing a network model for IoT devices because these models cannot address the real QOS related problems faced in the actual IoT. However, many other approach uses Markov chain for optimizing the QoS related issues in the communication of delay sensitive network, but due to uncertain type of devices in network it is not efficient in IoT. Apart from the network

models the scheduler also plays a vital role in terms of QoS. Many schedulers were proposed in the earlier approaches where often WRR was considered the most compared to others due to its simplicity. WRR is a round robin based scheduler for the network model, whereas we will use weighted fair queueing (WFQ) which is also a generalized process sharing policy (GPS) and extension to fair queueing policy. The methodology for achieving better QoS for each services cannot be same hence message based QoS alert approach categorized the services into 2 different type's i.e. critical and non-critical services. Few of the approach uses predictive models for predicting the average queue length by using the exponentially weighted moving average (EWMA) method and then bandwidth can be allocated efficiently by using the average queue length for high and medium priority data and thus the efficiency is obtained for both types of packets. But due to use of EWMA a steep increment was observed in the average queue size which resulted in allocation of higher portion of bandwidth to the high priority and thus more or less the medium, low priority packets get the same type of bandwidth allocation which resulted in the previous approaches. We have different approaches for improving the energy efficiency, network topology and performance related issues. In our approach the services are categorized into high, medium, low priority and also makes it possible to address the emergency services. This thesis proposes weighted fair queueing based packet scheduling along with dynamic bandwidth allocation based on average queue length to improve the quality of service of applications. The scheduler comprises of two different steps. First, the weighted fair queueing mechanism will take the previous slots' average queue size into consideration for the calculation of current slots' average queue length. The bandwidth will be allocated by multiplying the average queue length (which is calculated based on the current queue size and the previous average queue size) with a coefficient. Medium priority packets will be allocated bandwidth by calculating average queue length with simple average formula. The portion of bandwidth left out after allocation for high priority and medium priority is allocated to the low priority.

The present thesis is organized in five chapters as discussed below:

Chapter 1: This chapter includes general introduction, overview of IoT concepts, motivation and objective of the work. Finally, this thesis chapter explains the organization of the thesis.

Chapter 2: This chapter explains literature review of various journals and conference publications which includes the overview of IoT, its fundamental components, features and applications. Quality of service and its parameters, basic architecture of quality of service and existing QoS models and various queue scheduling disciplines are summarized.

Chapter 3: This chapter discusses the adaptive exponentially weighted moving average, which is used to calculate the average queue lengths of high, medium and low priority queues on the basis of which weights are assigned to different queues to schedule the incoming data packets**.** Thus, we investigate the QOS performance of the joint packet scheduling and dynamic bandwidth allocation algorithm in IOT applications.

Chapter 4: This chapter includes synthesis and simulated results of thesis work.

Chapter 5: This chapter includes the conclusion of the work and its limitations.

1.4 METHODOLOGY

The prediction of real world behavior can be achieved either by simulating the system or by use of theoretical analysis. Theoretical analysis approach seems to be cheaper and faster solution, whereas simulation approach can be used when the mathematical model of the system is very complex. However, simulation approach is an expensive and time consuming solution. In this thesis simulation approaches is used to analyze the proposed model. For simulation approach, a process based discrete event simulation library in PYTHON known as SimPy is used to analyze the packet loss, average end-to-end delay and average end-to-end jitter performance of incoming data streams.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

IOT is an extension of the hitherto heterogeneous networks and is emerging as a new technological paradigm composed of capabilities such as sensing, actuation, computation, communication, storage and networking to make real time data available to the end users. The 'things' connected to the internet interact among themselves paving the way towards a digital future but the gigantic connected 'things' and heterogeneous networks poses as a hurdle in the way of achievement different QOS requirements of various services. Quality of service is defined as ability of the network to offer better services to specific network traffic over different technologies, viz. frame relay, Ethernet, ATM, SONET or 802.11 networks, and IP-routed networks which might use some or all of these underlying technologies. The networks provide passage to the swarm of data and applications such as high-quality video, delay-sensitive data, real-time voice, etc. Although, the high bandwidth demand applications test the network capabilities and resources, they also supplement, add significance, and improve the business process. The major goal is to offer security along with dedicated bandwidth, controlled latency and delay variations (required by real time interactive traffic) & better loss characteristics. The services using IoT are evolving enormously every day. These services span in every walk of our lives like entertainment, eHealth, defense, security, smart cities, etc. Since the reach of connectivity and services is expanding, it affects our lives and safety. Also, the chances of system failure becomes greater than before. Hence, we need to develop good quality IoT systems to prevent loss of fortune and data. Therefore, quality of service is an important and valuable aspect of an IOT system. As the Internet users are expanding continuously, system performance must increase with them to satisfy desired standards.

Therefore, it is important to select a proper QoS architecture and subsequently develop an algorithm to implement it. Quality of Service (QoS) in IoT is an important factor that requires research and development for QoS execution, management, stabilization and optimizations.

2.2 PARAMETERS OF QoS IN INTERNET OF THINGS

The prevailing IoT architectures are planned and designed from the WSN perspective. Fig. 2.1 shows the different IoT architectures from which we can observe that IoT is a collective system, the basic building blocks of which are things, communication and computation. An understanding of these three integrated pillars is necessary to understand the functionality of QoS in IoT.

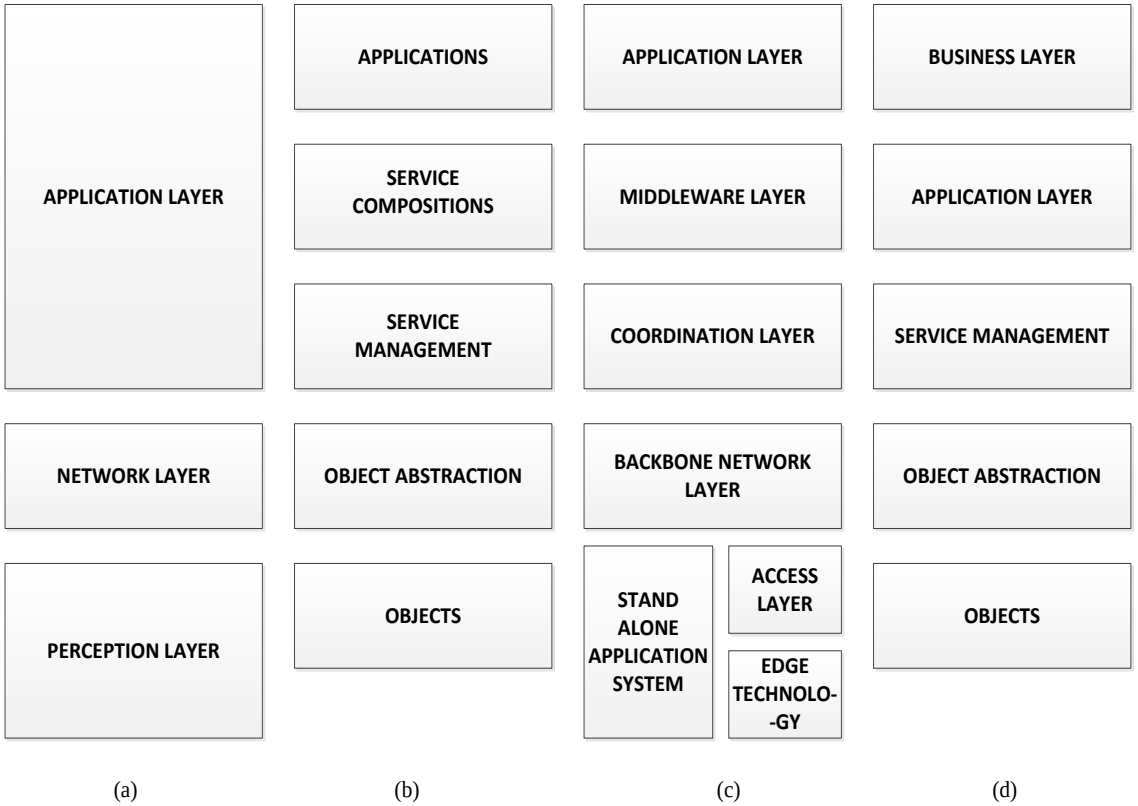

Fig. 2.1.(a) Three layer IoT architecture (b) SOA based IoT architecture (c) Middleware based IoT architecture (d) Five layer IoT architecture

Things are devices embedded with sensors and possessing the ability to connect with the internet. Communication involves unique elements, protocols and technologies that facilitate interaction. Computation involves analysis and replenishment of data which can be done in cloud, fog nodes and edge devices.

QoS enables users to get a lucid visibility of the performance of services offered by the service providers. QoS metrics or parameters will enable end users to specify better IoT services for their desired applications and also stabilize and optimize service quality. The parameters that are needed to be taken into consideration for the establishment of high end services that have specific set of quality requirements, can be organized under things, communication and computation. These three pillars are essential in the deployment, development and wider acceptance of IOT. The chief identified QoS metrics associated to these three components of IoT are:

a) QoS of Communication: A communication network is the pillar of an organization. These networks carry a huge amount of data from various applications, consisting of high -definition real time video, audio, delay-sensitive or delay-tolerant data, etc. Such bandwidth-intensive tasks, although put pressure on network capabilities and available resources, but at the same time they complement and add value to the business process. The performance requirements of network grows along with its users. Also, many new online services have high bandwidth demand and network performance requirements. Network performance is a matter of apprehension for both, internet service providers and users. Some of the identified QoS metrics in communication network are given below:

1. **Jitter**: It is the unwanted variance of time delay amid a stream of data packets in the network. Packets belonging to the same flow, rather than remaining constant, may possibly arrive at the destination with a rate somewhat different from what they were released from the source. i.e the time duration between the arrival of packets varies. These packets are queued, processed, de-queued, etc. independent of each other and can suffer congestion and configuration error and hence arrive out of order and suffer varying end-to-end delays. Therefore, they might arrive out of sequence, and their end-to-end delays might vary. The service provider must arrange for adequate bandwidth and plausible latency for warranting quality network connections to help

mitigate the problem of jitter. For applications involving audio or video packets, correct sequence of packets and data rate are required at the destination with which they were transmitted from the source. The de-jitter buffer can be used for this purpose. It is called so, as it reimburses the jitter caused by the network. If the jitter is within a certain limit, the de-jitter buffer holds, sorts and releases the packets to the application on the basis of Real-Time Transport Protocol time stamps.

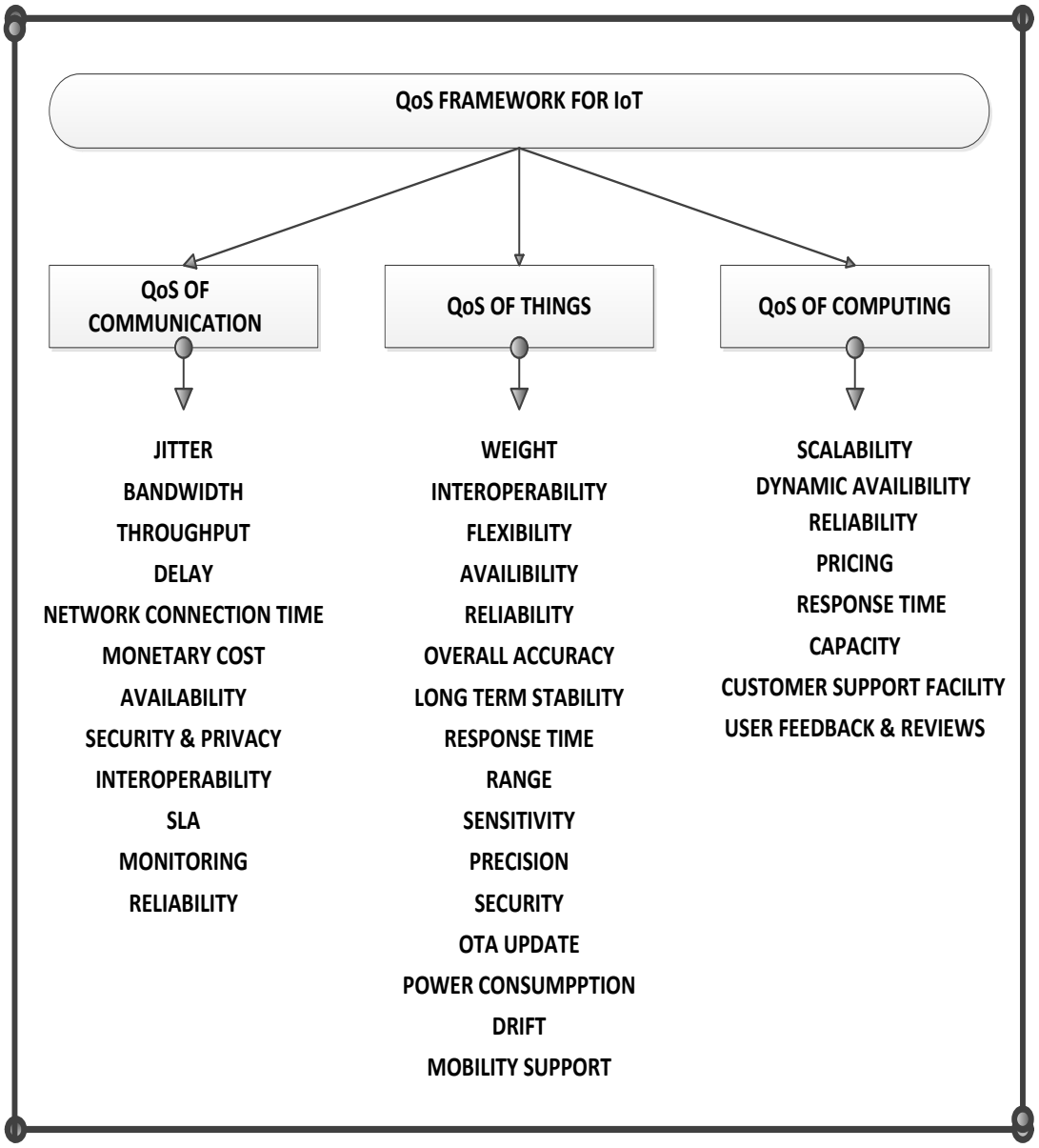

Fig 2.2. QoS Framework for IoT

2. **Bandwidth:** Bandwidth is the data volume which is allowed to traverse through a network in a defined time-period, but the tangible network speed might be much more lower (also called throughput). Bandwidth is measured in Mbps. Networks with high bandwidth availability are generally preferred for real time applications.
3. **Throughput and Efficiency:** It gives the measure of a number data packets transmitted or received above the network. It is the actual available bandwidth that the network has and is inversely proportional to network latency. Throughput is measured in bits per second (bps).
4. **End-to-end delay:** Delay refers to the amount of time required for the information to move from source to destination. It is a combination of different types of delays which influence the packets of specific flows or applications. The delay types that constitute the end-to-end delay are as below:

   i) **Processing Delay**: It can be defined as the time taken by devices such as Layer three switch or a router to complete all the obligatory tasks to transfer the data packet from ingress to egress .i.e. input interface to output interface. The CPU type, utilization efficiency, architecture of router, mode of switching, device configuration features influence the processing delay. e.g. distributed-CEF switched data packets on a VIP card (Versatile Interface Processor) do not cause CPU interrupts.

   ii) **Queuing Delay**: It is the defined as the time duration devoted by a data packet in the buffer of a router interface. Queuing and scheduling discipline, bandwidth, no. of packets already waiting in the buffer, utilization efficiency of server/router, etc. influence queuing delay.

   iii) **Serialization Delay:** Serialization delay is the time taken to transmit all the bits of a particular frame from the physical medium for transmission over the physical layer.

   iv) **Propagation Delay:** The time taken by all the bits of a particular frame to reach their destination crossing the physical link is known as propagation delay. Propagation delay also depends on number of factors one of which is the medium of transmission.
5. **Packet Loss:** Packet loss occurs when the data packets traversing the communication network are not able to reach their destination. Packet loss may occur when the buffer capacity of the intermediate/edge routers is not enough to hold the incoming packets

and thus end up dropping them or when a packet crosses the threshold time at which it should have reached a particular check point thus rendering the further transmission of that packet useless. A router can also discard certain packets based on the priority setting. Also, interface overrun and interface reset can also cause the packet to be flushed and dropped.

6. **Network Connection Time:** Network Connection Time is the time that the server takes to respond to any end-user/device request. Delay to respond to a request above a certain threshold can lead to a connection timeout.
7. **Monetary Cost:** Bandwidth can determine the network quality level. With increase in bandwidth network, price also increases proportionally. There is a tradeoff as the users generally prefer high bandwidth and lower costs. There are three kinds of pricing strategies defined in literature: (i) flat-rate scheme, (ii) capacity-based scheme and (iii) usage-sensitive pricing scheme.
8. **Availability:** When required for functioning, availability gives the amount of time the network is operational and active. Usually, it is considered in "scales of nine". VPN tunnel and internet connection affect the availability of network in communication. [7].
9. **Security and Privacy:** With increasing cybercrimes, network security is an important factor to prevent our system from various attacks. Network security and privacy protects the information that flows through a communication channel from unwanted interception and attacks and maintains the integrity of data. Many encryption techniques can be employed to secure the network. Certain Privacy Enabling technologies [8] are: (i) Transport Layer Security (TLS), (ii) DNS Security Extensions (DNSSEC), (iii) Onion Routing (iv) Virtual Private Networks (VPN), (v) Private Information Retrieval (PIR), etc.
10. **Interoperability:** Heterogeneous networks are very common in the vast stretch of internet. Different types of technologies work on different types of system software and possess different architectures. The ability of such diverse technologies and networks to communicate with each other and perform the operations is called as interoperability.

11. **Service Level Agreement:** Service level agreement defines the negotiating terms of services between users and providers. OcularIP [9] is a tool that can be used for measuring SLA. OcularIP can analyze inconsistencies of a network and is compatible with the mercurial network demands, hence can be used as a prospective tool to define service providers' potential by the users.
12. **Monitoring:** Monitoring of connected devices, routers, switches, network traffic, software applications, etc. is an important task to meet the service requirements. Monitoring can be performed at hardware level or at software level. Network elements use commands like netstat, ping, snmpwalk, lynx, etc. to monitor the network. They can change device configuration and scheduled tasks to the peripherals with the help of the information gathered by the command execution.
13. **Reliability:** Reliability can be verbalized in terms of the association amid the origin and destination nodes. A medium can be reliable only if it assures delivery of data packets to end user without any loss of information or security issues and provides acknowledgement. Group Communication System such as Appia network, Quick Silver scalable multicast, etc. are frameworks which offer strong reliability.

b) QoS of Things:

Internet of Things comprises gargantum devices embedded with sensors, furnished with multitude of technologies and connected to a smart network. Some of the QoS paremeters identified for 'things' that have a potential to supplement a new perspective to IoT applications are as follows:

1. **Weight:** Low weight and cost sensors are always preferred to those with heavy load. It is a physical parameter measured in grams.
2. **Interoperability:** Interoperability in 'things' is the capability of sensors to interchange information and resources with each other regardless of their configuration or architecture. Certain metrics like Potential Measure, Performance Measure and Compatibility Measure [10] are in place although no global standard is defined for measuring it.

3. **Flexibility:** The capability of devices to undertake hardware and software changes in their configuration and waveform is the flexibility of sensors. Change of waveform by a device operating in a particular band can be done during down peak or less usage time. Therefore, a device of higher flexibility that works in one radio band can attend a multitude of applications with distinct range, data rate and latency requirements.
4. **Availability:** It can be understood as accessibility to energy and resource to a wide range such that sensors can access data which enhances the sensing efficiency of the devices/sensors and offers faster recovery in case of software or hardware failure thus providing better operational service.
5. **Reliability:** Reliability can be understood as self-recovery and configuration in a dynamic environment. Errors are inevitable in any system hence for long-term operation, reliability must be taken into consideration during architectural planning and system deployment.
6. **Overall Accuracy:** Sensors accuracy is the maximum uncertainty amid the actual measured value and standard value fixed at output parameters. Accuracy is degree of correctness concerning proper sensing of the system. E.g. in pressure sensor system, 100kPa is the given value of atmospheric pressure at sea level, that is accurate: if measured as 105kPa, it is incorrect. Sensors having minimum difference between estimated and actual value are considered efficient. It is measured in terms of error.
7. **Long-Term Stability:** Steadiness in the output given by sensors over a period of time is stability. Stability may deteriorate due to ageing of components, noise, etc. subject to nature of sensors.
8. **Response Time:** The time it takes for a sensor to change to its output with respect to the input parameters is called response time. Low response time is always desirable for any application.
9. **Range:** Sensors operate on a predefined range of units, exceeding which might damage the sensor or provide inaccurate range is exceeded, the sensors may get damaged or may produce inaccurate results. Therefore, provision of nominal range adjustment are appreciated.

10. **Sensitivity:** Sensitivity requirements of different applications is different. Sensitivity can be described as the amount of change in output with the input parameters. [11] shows a characteristic curve in which the slope (dy/dx) of curve y=f(x), where x and y are input and output respectively gives the measure of sensitivity of sensors. Sensors having higher and constant sensitivity are preferred to the ones with lower sensitivity.
11. **Precision:** Precision is the capability of a sensor to identify and measure any deviation in the obtained output when same data flow or signal is observed recurrently under equivalent conditions. High precision is achieved when there is high correlation between output data, irrespective of the proximity of output with actual result
12. **Security:** Google offers the public a service called as Google Cloud IoT Core with a special characteristic of full stack security to millions of heterogeneous devices dispersed across the network [12]. It offers secure and reliable data management and connection from a vast set of resource. Secure communication requires security credentials such as device registration, authentication and authorization
13. **OTA Update:** OTA update means over the air update. It helps individuals having the hardware based on Android Things to update their devices. On-site system update is not practically feasible. Therefore, devices are updated by pushing the updated across the network.
14. **Power Consumption:** The devices are usually battery operated and are charged with the help of electricity. New methods such as energy harvesting (using solar energy) is also an appealing option to users nowadays.
15. **Drift:** Undesired physical changes in sensors due to external parameters such as wind, dust, vapor, moisture, humidity, chemicals, etc is drift which can cause changes in temperature, frequency or dimensions of sensors.
16. **Mobility Support:** [13] classified mobility of sensor nodes into two classes' .i.e. strong mobility and weak mobility. Nodes undergoing failure or running out of service due to any reason come under the category of weak mobility nodes while that are not affected by physical factors like wind, water, etc. fall under the category of strong mobility nodes.

c) QoS of Computing

Cloud, edge or fog computing plays an important role in Internet Of Things. It helps in scaling the gargantum of data generate by multitude of applications and increases the efficacy of the overall system. It reduces the tremendous stress faced by the network. Hence, it is important to consider the QoS metrics of computing in IoT. A few such identified metrics are:

1. **Scalability:** Scalability in terms of computing is to achieve highest throughput in lowest response time. IOT is one of the major source of big data. High scalability computational power will be required to fit in more users, one way of doing which is to add resources with certain overhead to the system. Scaling can be done either vertically or horizontally. Usually vertical scaling is preferred.
2. **Dynamic Availability:** Dynamic availability lets us know if under normal operating terms, the system is accessible for use or not. In terms of computing availability can be describes as the software and hardware ability of being at hand and operational.
3. **Reliability:** High reliability in computing means longer time duration during which the applications run adequately and satisfactorily on the cloud. As the number of connected devices keeps on increasing, the computing system works less efficiently. Hence, it is important to study the behavior of the resource abundant cloud. Computing service providers use MTBF (mean time between failures) to measure reliability.
4. **Pricing:** Selection of a computing service also depends on the pricing factor. Cost can be understood as a function of network, storage and computing.
5. **Response Time:** The duration between submission of a request and getting back a response is the response time. It is related to the infrastructure and the application submitting the request.
6. **Capacity:** Capacity measures the highest amount of computational resources offered by the service provider which can be accepted, processed and analyzed by the computational software. Four metrics: CPU, storage, network capacity, and memory capacity together give the measure of capacity.
7. **Security and Privacy:** The connected devices, cloud nodes, edge nodes, etc. in IOT environment are susceptible to various attacks such as erroneous data addition,

unauthorized data access, denial of service, etc. Security also incorporates integrity and confidentiality to be guaranteed at computing node. Trustworthiness of computing can be ascertained by security measures like physical security support, crypto algorithms and key management, data support, network security support, etc.[14].

8. **Customer Support Facility: For any event of system fault and discrepancies vendors provide** customer support facility. Type, cost and response time involved in offering the facility to the user is includes in this. Generally, computing services which have a reliable support system are preferred.
9. **User Feedback & Reviews:** The users experience, evaluation, assessment and criticism plays a significant role in choosing a computing service. New users generally look into the reviews of existing users before selecting the service provider. User experience gives an understanding of stability, availability, reliability, transparency and cost of services. Higher ratings means better customer experience.

2.3 BASIC ARCHITECTURE OF QOS

The basic QoS architecture consists of the following components [15]:

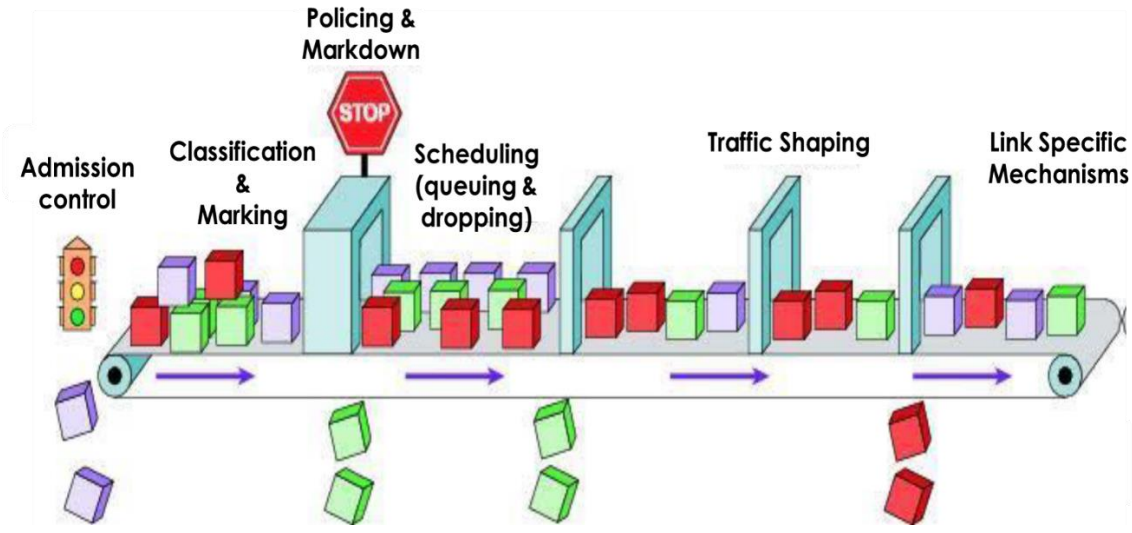

Fig.2.3 Basic Architecture of QoS [16]

1. **Admission Control**: Based on the service level agreement (SLA), we can know about the QoS requirements of any flow. Admission control tries to estimate that if a new flow enters a network, whether the required level of QoS can be satisfied for all the previous flows in addition to the new flow. Admission control replies to application demands for resources. The request is rejected if the resources cannot be spared for new request without harming the existing applications.

2. **Classification & Marking**: The traffic that belongs to a particular application and has made reservations of resource should be categorized and recognized by the routers in the passage to ensure that they can deliver appropriate services to those data packets. Based on the below mentioned application classes the incoming data packets are classified and marked-

i) Constant bit rate services (e.g.VoIP)

ii) Real time variable bit rate services (e.g. videoconferencing)

iii) Non-real time variable bit rate services (e.g.video streaming like IPTV)

iv) Available bit rate/best effort (e.g. file transfer)

3. **Policing & Markdown:** It is imperative to measure and observe that applications do not surpass resource utilization threshold outside their set profiles. Parameters such as rate and burst are used for measuring the applications behavior. Suitable action is taken based on the fact if the application obeys or exceeds its decided resource utilization. It sees whether a certain kind of flow or packet is significantly violating the QoS agreement. If so, the violating packet is dropped. The leaky bucket algorithm and token bucket algorithm have been described in literature for traffic policing.

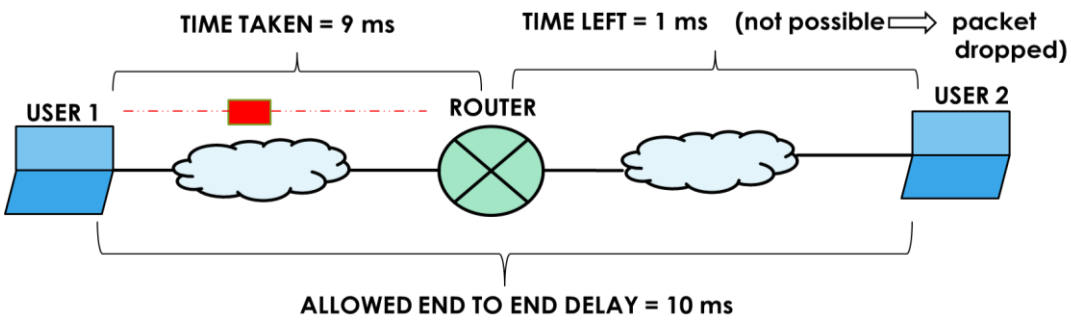

Fig. 2.4. Policing & Markdown

4. **Queuing &Scheduling**: It is necessary for network elements to have the ability to hold data packets when processing and forwarding others packets. Various queuing techniques store and forward data packets in different ways. The method in which queued data packets are chosen for transmission over the communication link is called scheduling. Based on the distinct requirement of the different traffic classes, traffic is prioritized. Some conventional scheduling disciplines are:

a) First-in-First Out (FIFO) queuing

b) Priority Queuing (PQ)

c) Fair Queuing (FQ)

d) Weighted Fair Queuing (WFQ)

e) Weighted Round Robin (WRR) or Class Based Queuing (CBQ)

f) Deficit Weighted Round Robin (DWRR)

5. **Traffic Shaping**: Traffic shaping is a technique of bandwidth management used in computer communication networks which delays a few or all the packets/datagrams to bring them in acquiescence with the sought after traffic profile. It is used to smoothen the jitter in the network.

Fig. 2.5. Reducing Jitter by Shaping

6. **Link Specific Mechanisms**: The jitter or delay thresholds of a VoIP packets can be surpassed because of serialization delay in low speed WAN. Techniques such as *Multilink PPP Link Fragmentation and Interleaving* and *Frame Relay Fragmentation (FRF 12)* can help to moderate serialization delay on such slow links. Compressed Real-Time protocol (cRTP) is a compression technique to abate bandwidth necessities and are extremely beneficial on slow links. Tx-ring is the closing interface at first-in-first-out queue which holds the frames that are to be straightaway transmitted across the physical interface. The

Tx-ring guarantees that a frame is permanently present at the interface when it is set to transmit the packets. This ensures that the link is utilized to 100% capacity [2].

2.4 MODELS OF QoS

Three models are generally described in literature for providing QoS services in the network[17]:

1. Best Effort Model
2. Integrated Services (IntServ).
3. Differentiated Services (DiffServ).

These three models are different from each other in terms of the way they aid applications to transfer data how the data delivery is handled by the network within predefined service level.

2.4.1 Best effort model

The simplest of the three models is BE model. By default, this model is used for Internet and it doesn't offer any quality mechanism at all due to which no complexity is linked to the best effort QoS model. BE does not incorporates resource reservation or other techniques for special treatment of certain applications by the network. Therefore, BE model is not well suited for real-time (RT) IoT traffic demands. This model must not be preferred when network resources are limited and unable to fulfill the desirable QoS requirements of the applications in terms of the metrics such as jitter, delay, packet loss, etc. In the network where applications compete for resources, the user experience could be of poor quality in case of absence of any mechanism to manage the unfairness.

The main advantages of this model are[5]:

1. Scalability- There is no scalability limit here. The bandwidth that is associated to the router interfaces specify throughput efficiency. The Internet can be called as a best-effort network.
2. Ease- No special configuration is needed in best effort model which makes it an easier and quick model to implement.

The downsides of the best-effort model are as below:

1. Lack of service guarantee- No guarantees of packet delivery, delay, or bandwidth availability is provided.
2. Lack of service differentiation- There is no differentiation of packets belonging to different applications based on different levels of significance.

2.4.2 Integrated services

This model was developed in mid-1990s and can be seen as the earliest thoughtful attempt to deliver end-to-end QoS that was necessary for real-time applications. IntServ uses explicit signaling, management and reservation of resources by the applications that require and need it. The IntServ model (Integrated Services) is also called hard QoS model because Hard-QoS warranties metrics such as delay, bandwidth and packet loss, thus offering a predictable level of service. IntServ uses a signaling protocol called Resource Reservation Protocol (RSVP). An application with predefined bandwidth necessity waits for RSVP to run from source to destination along the path, on a per hop basis and request for bandwidth reservation for any particular application flow. If the RSVP succeeds in its attempt to reserve bandwidth starting from source to destination, the application can start running. The entire time when the application is active, the routers must reserve the bandwidth for the application along its path. If the attempt to reserve the bandwidth by RSVP per hop from source tom destination fails, the application can't operate. IntServ model is a flow based model, i.e., source and destination IP addresses and ports. Here, applications demand explicit resource reservation in every flow from the network. All the flows that traverse the network are kept in check by network devices to see if new packets are part of an existing flow and whether the network has enough resources to provide service the packets. Reservation of resources per flow in the network, provides the applications with resource guarantees and predictable performance. On the basis of resources requests and available resources the IntServ model implements deterministic Admission Control (AC).There is a requirement of IntServ capable routers for implementation of this model in the network. RSVP allows a host to institute a connection above the connectionless IP Internet .i.e.

1. Applications appeal for some service level from the network beforehand transmitting data.

2. The network accepts or discards the per flow reservation based on resources that are available.

3. Once the request is furnished, the network expects the application traffic to maintain the requested profile.

This model has limited scalability because network nodes have heavy resource consumption due to the associated per flow processing. The network nodes have to preserve the reservation state of every flow crossing the node. The scalability problem is aggravated as RSVP is soft state protocol involving constant signaling load. In this model, new flows are acknowledged till the requested resources can be furnished, beyond which new application flows are rejected. The QoS request is made by RSVP for each flow. This request comprises identification of requestor, also called authorization object and required traffic policy (policy object) .RSVP provides information of flow parameters such as IP addresses and port numbers so that the routers lying between source and destination can recognize each flow. The advantages of the IntServ model are summarized below:

1. Clear end-to-end admission control.

2. Admission control per-request policy.

3. Signaling of port numbers.

4. Good resolution for handling flows in small networks.

5. Intserv permits hosts to bid per-flow, computable resources, along the data paths and to attain feedback concerning admissibility of such requests.

Some drawbacks of using IntServ are:

1. The overhead caused by continuous signaling of each active flow might become significantly large with increasing number of flows leading to heavy resource consumption, increased per flow processing, increased per flow state.

2. As each and every flow is traced and retained, IntServ is not very scalable option for implementation of large networks.

3. It is complex and difficult to implement.

2.4.3 Differentiated services

The development of Differentiated Services (DiffServ) which is the latest of the three models is targeted towards overcoming the limitations of the previous models. DiffServ is highly scalable, although it is not take QoS guarantee. RFCs 2474 and 2475 holds the discussion and description of DiffServ by Internet Engineering Task Force (IETF). DiffServ is also called as the "Soft QoS" model. In IntServ either the application request is denied or accepted for with guaranteed resources through signaling and admission control whereas signaling is not used in pure DiffServ, rather its basis is per-hop behavior (PHB). In PHB every hop in the network should be preprogrammed to offer a particular level of service for different classes of traffic. Thus, PHB do not need signaling till the traffic is marked to be recognized as one of the predefined traffic classes. DiffServ model is much more scalable as signaling and overhead status monitoring for every flow is not obligatory. Every node is deals with a limited range of traffic classes which implies that even if a large number of flows are active, they will be classified as one of the specified classes, and every flow will obtain the service level suitable for its class. The types of classes and service level received by then is decided upon by the business requirements.

In DiffServ model the traffic is initially classified and then marked. The type of service that a marked packet receives while traversing the network depends on the marking. DiffServ protects the network by techniques such as policing and admission control from oversubscription. E.g. voice traffic in a typical DiffServ is generally assigned a high priority with reserved bandwidth on every node. To forbid excessive voice calls to be active concurrently, call admission control can be deployed. All the voice packets belonging to the admitted calls belong one class. Three main points regarding the DiffServ model are:

1. The network traffic in DiffServ is classified.
2. QoS policies that are enforced are aimed at differential treatment of the predefined traffic classes.
3. Traffic classes and policies are specified on the basis of business requirements, hence, we select the service level for different traffic classes.

The advantages of DiffServ model are:

1. The DiffServ model is highly scalable.
2. No resource reservation needed at end hosts.
3. Simple operation, maintenance and configuration.
4. Supports intricate classification of traffic and conditioning on the edge.
5. DiffServ model can aggregate numerous application flows in limited number of TCs.
6. Overhead associated with signaling is reduced to maintain the policy on a per flow basis.
7. Diffserv network elements can process traffic much more effortlessly than Intserv devices.
8. It is a distributed service model. Far greater flexibility and efficiency is allowed in the routing process because of the distributed resource allocation among the routers of DiffServ domain.

The disadvantages of DiffServ QoS model are:

1. Coordination amid domains in end-to-end QoS service.
2. The guaranteed end-to-end QoS service may be affected by SPs QoS customization.

Table 2.1. Comparision of the three QoS models

QoS Service	Best Effort	DiffServ	IntServ
Isolation	No	On per aggregation basis	On per flow basis
Guarantee	No	On per aggregation basis (Traffic Class)	On per flow basis
Service	End-to-end service	Per domain service	End-to-end service
Suitable for Real Time traffic	Not suitable	Suitable (LLQ)	Suitable (resource reservation)
Scalability	High scalability	Scalable (the edge routers maintain per	It is not scalable (every router

		aggregate state and the core routers maintain per class state)	maintains a per flow state)
Complexity	Low complexity	Medium complexity	High complexity
Applicability	Default for internet	Can be used in networks of any size.	Suitable for small networks.
Admission Control	No applicable	Admission control is statistic based on traffic classes.	Admission control is deterministic based on flows.
Resource Reservation	Resource reservation is not available.	Resource reservation is as per Traffic Class on each node in the domain.	Resource reservation is as per flow on every node along the path from source to destination.

2.5 QUEUING AND SCHEDULING CONCEPTS

Queuing and scheduling when applied at an interface permits us split the traffic into manifold queues. After that the scheduler decides upon as to what kind of treatment should the traffic inside every queue receive. Differentiated behavior can be applied by the scheduler to various service classes. Buffers and bandwidth are significant parameters related to queuing and scheduling mechanisms. Buffer or length of queue is the amount of available memory for storing packets. Nevertheless, it is not necessary to queue up the entire packet; every so often a notification is been stored that represents the contents of the packets. The buffer value can be explained in two ways; either as the time duration for which packets are received at the queuing interface or in terms of physical size .i.e. the number of packets or notifications that can exist in the queue at one time. It can be defined as the quota of the memory that is available and expressed as an absolute number or milliseconds of traffic. The bandwidth parameter comes into picture in the scheduling part of the equation**.** The overall bandwidth is accessible to queuing and scheduling

mechanism. Scheduling governs how much bandwidth is assigned to each queue. The type of queuing and scheduling discipline used concludes the way in which the resources are allotted. The necessity for the incidence of queuing and scheduling is characteristically controlled by the occurrence of congestion. If the resources are adequate and no competition for resources is present, queuing is not needed. One of the ways to generate congestion is by placing more traffic than that can be supported by outgoing line speed at the interface. Congestion can also be artificially generated, by using a shaping rate at the interface which imposes a lower speed limit than the largest interface line speed. The excess traffic is throttled into the memory, which is apportioned through the actual queues. The scheduler provides service to the packets in the queues and is in charge of the rate with which packets are transmitted from each queue. The packet enters at the tail of the queue and remains there until it reaches the head of the queue, after which it leaves the queue.

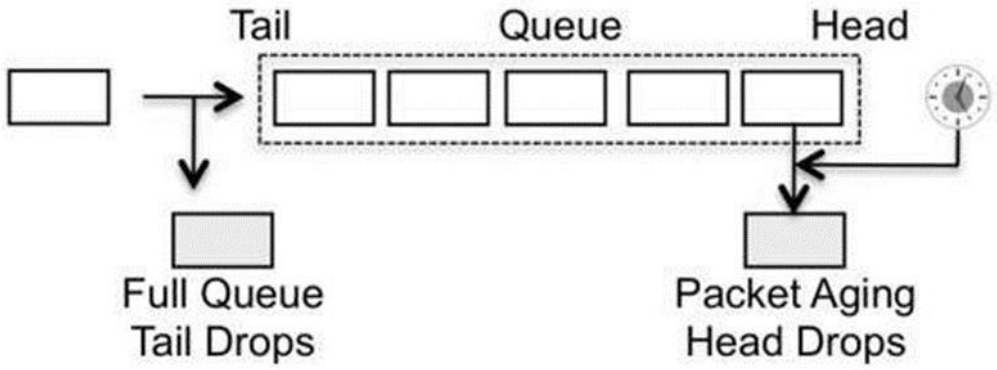

Fig. 2.6. Tail And Head Aging Drops In A Queue

In a queue, packets can either be dropped from the head or the tail of the queue, or from both at the once. Generally, packets in a queue are dropped from its tail. When the fill rate of the buffer is greater than removal rate, the buffer is completely filled up. As a result more packets can not enter the buffer, and the newly arrived packets have to be dropped. But packets can also be dropped from the head of the queue. Packets that are at the head of queue are those which have went from tail to head, and therefore have waited in the queue for the maximum duration of time. In the event of very high congestion and resource starvation, no scheduling slots are allocated to the queue. To evade stalling the queue, and

causing an extremely long delay to the packets inside the queue, a maximum time duration is prescribed on all packets as for how long are they allowed to stay in the queue, while waiting for a scheduling slot. This is termed as packet aging .i.e. packets are aged out of the queue as there is no longer any use of delivering them. Packet aging and tail drops are not mutually exclusive. Packets can be dropped from both the head and tail of the queue because of packet aging if the dropping rate at the head cannot keep pace with the rate of packets entering at the tail.

2.5.1 QUEUE SCHEDULING DISCIPLINES

Some chief scheduling disciplines are described below:

1. FIFO – First in- First out

FIFO is a commonly used acronym for First-In-First-Out, and is perhaps the most elementary queue scheduling discipline. The principle of FIFO queuing is equal treatment of all the packets, by placing all of them in the same queue. Only one queue is present which is served by a single scheduler. This mechanism indicates that the rate at which packets are removed is directly related to the interface speed or the shaping rate, in case a shaper applied. Packets receive service on a first come first serve basis as no overtaking takes place within the queue. For many router vendors, FIFO is perhaps the default setting, having one queue for transit traffic and another one for control plane packets that are locally generated viz. routing packets. Many hardware implementations require minimum one buffer for every interface for being able to build a packet before transmission. The purpose of the queue is to act as a placeholder to be used when the Layer 1 and 2 headers are appended to the packet. Offering a buffer of two to four packets size aids the system to utilize the interface proficiently and at line rate. FIFO is foreseeable and its algorithm is simple to be implemented in a router or a switch. Yet, the FIFO algorithm is possibly not a real "queuing and scheduling" solution as it does not offer any kind of differentiated services, because different classes of traffic share the same queue. In case of congestion, FIFO queuing initially introduces a delay when the queue starts filling up, and as the queue is completely occupied, all newly arriving packets will be discarded. Here, the manner of

drop behavior can be different. One way is to drop all the packets in the buffer to escape queue collapse beneath the problem of stalled sessions. The motivation behind this behavior is to evade a condition like TCP silly syndrome, where the receiving window becomes smaller and smaller since many packets in the queue memory have to wait to be processed. As a result, all sessions have to suffer a degradation in service**,** and the data packets are caught up in the queue in the same order in which they arrived. In spite of the probable availability of big buffer memory, the queues cannot be made use of accurately for the reason that only a small amount of senders' congestion window can be utilized. A different drop alternative is the use of more intelligent ways such as tail drop and packet aging.

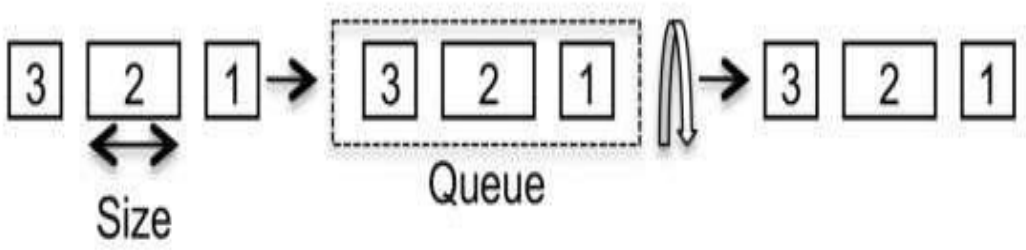

Fig. 2.7 FIFO scheduling

The FIFO is a simple algorithm to implement. It delivers satisfactory performance if the buffer saturation is low. Actually, FIFO can be comprehended as a burst-absorption execution than a true queuing mechanism. If the applications that require service are not delay-sensitive and are TCP-based, their removal from the buffer is well suited since the order of packets is preserved at the expense of introducing delay. In case of moderate congestion, due to RTT, TCP slows down, however retransmissions are limited to a minimum. However, it has some limitations. It does nor provides differentiated services. In the event of extreme congestion and queue usage, all services are likewise bad. Since the queue depth usage diverges, delay and jitter are not controlled. Hence, FIFO is not a good option for real-time applications. Example, a voice flow comprising many packets may be jammed after a 1500-byte TCP transfer having a large congestion window. Greedy flows are able to take up maximum of the buffer depth and therefore bursty flows consume entire available buffer memory. For the reason that TCP attempts to increase the size of transmission window, controlling the session is difficult with a single queue.

2. Fair Queuing

The main shortcoming of FIFO is that flows comprising of many data packets can consume large amount of the bandwidth for lesser bandwidth-intensive applications, since flows or packet streams are not separated in FIFO queuing. Fair queuing (FQ) is also called fairness algorithm. This scheduling scheme addresses the fundamental limitation of FIFO queuing. FQ segregates packet streams into multiple queues, presenting a fair scheduling mechanism for the data flows to access the link. Along these lines, FQ splits traffic and flows, and protects the applications that take up lesser amount of bandwidth from being starved by applications that take up more bandwidth. Scheduling is to a great extent statistical multiplexing between the queues, in which queues buffer packets belonging to different flows. Fair Queuing algorithm was defined in 1985 by Nagle. It considers the packet size to guarantee that each data stream obtains a fair opportunity to send an equal quantity of data. Every queue is allotted the same weight .i.e. same amount of bandwidth is apportioned to different queues. This arrangement evades the difficulties of handling large and small data packets in the queue, and hence the rate of packet removal from the queue does not depends on the number of packets but on the number of bits. Therefore, a queue having larger packets will have access to the same amount of bandwidth as a queue having smaller packets, since in each scheduling slot the queue is given service according to number of bits and not in terms of number of packets. Ironically, the main concern with the FQ is the point that it is undeniably fair, because it annuls the sought after QOS goal of delivering an uneven dissemination of resources across various service classes. Data streams that need lower delays can suffer if there is an obligation to visit many queues in a fair manner. A hash function splits flows from one another for mapping packets to a specific session in routers for implementing FQ scheduling. The hash function makes use of a rich set of variables viz. port addresses of source and destination, protocols or higher layer information above UDP, TCP, port numbers, etc. to compute a session or data flow. After the classification is completed, dynamic memory allocation and creation of logical queues which are present only if a data flow is active is an extremely resource-intensive task that is difficult to implement when a major portion of packet forwarding occurs in a hardware based architecture which has a

very low level of interaction with processing resources. Consequently, classification and dynamic queue creation for every active hashed data flow is basically not scalable. The fairness algorithm is a general approach to take care of oversubscribed backplanes on switches and routers. A general hardware architecture is having a one to two-stage fabric switch which connects line modules. In this setup, traffic that is incoming from several ports is forwarded towards a solo destination port over the fabric. At this point, fairness can warrant that the inbound ports on line modules will have equivalent access to the fabric so as to deliver them to ports on outbound module, as shown in figure 7.8. But, the fairness algorithm does not considers the element that the backplane is unaware of singular flows that is the foundation of the idea for using FQ to craft bandwidth fairness. So what is truly implemented is a slight variant of original FQ algorithm.

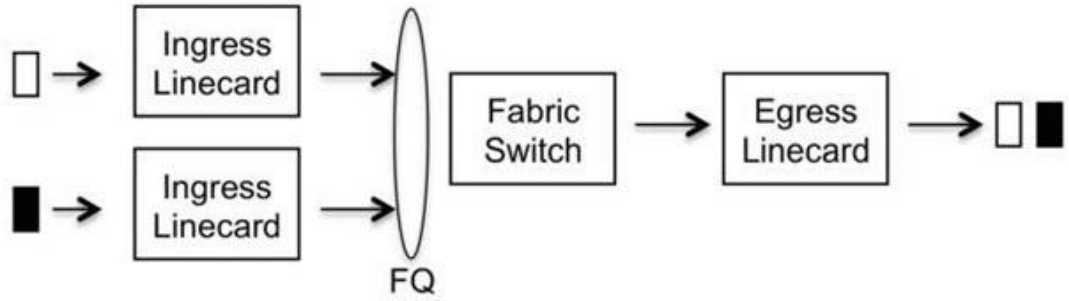

Fig. 2.8 Fairness algorithm

The FQ algorithm indicates that data rate might vary for small intervals, for the reason that the data packets are transmitted in sequence to realize fairness in each scheduling slot. This conduct can probably reduce the throughput, particularly in comparison to algorithms which emphasize more on higher throughput. On the positive note, nevertheless, FQ is actually effective in circumventing starvation of data flows under hefty loads. The FQ algorithm segregates every stream into its individual logical queue. Therefore, in theory, other queues are not affected by a greedy flow. Still, FQ suffer from its own set of limitations. It is very complicated to implement, and there is no example of any vendor implementation that exists on high-speed routers and routers that were created to handle enormous number of sessions and hefty amounts of traffic. FQ can be considered more of a kind of theoretic concept than a practical paradigm. It is a resource-intensive scheme

since a large number of states and hashes have to be computed and for the reason that memory should be allocated and modified based on variations in the session state. Since, every session hash is considered as a queue entity, delay and jitter can yet be issues. If numerous sessions are required to be scheduled, then depending on the number of active sessions that are currently transmitting packets, the multiplexing arrangement that is used to decide the session slot interval may fluctuate.

3. Priority Queuing

For delay sensitive applications and those applications that cannot handle packet loss, priority queuing (PQ) scheduling scheme offers a simple technique of backing up differentiated service classes. After a classification methodology classifies the data packets and places those packets into various queues of different priorities, the PQ scheme manages the packet scheduling using a priority-based model. Packets are arranged for transmission from the queue head of a specified queue only when all higher priority queues are empty. So the distinct levels of priority given to queues leads to the sought after unfairness concerning the way queues are serviced. Nevertheless, within a particular priority queue, same as other queuing and scheduling disciplines, data packets are scheduled in a first-in-first-out manner. As shown in Fig. 2.9, queue Q0 is having the highest priority, therefore, the scheduler will serve only queue Q0 till packets are present in it. Queue Q1 is having a low priority, therefore, packets are forwarded from this queue if queue Q0 is empty.

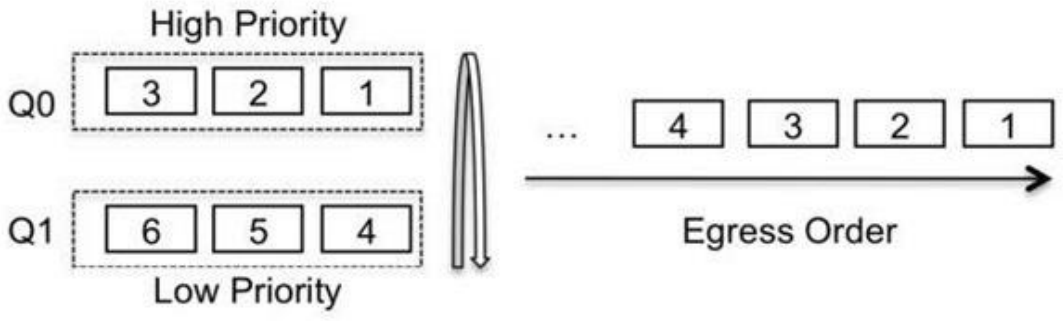

Fig. 2.9 PQ scheduling

PQ aids traffic with requirements such as lower delays and no packet loss. With the help of priority queuing, these kind of applications can be scheduled selectively and their

respective services can be distinguished from the greater part of the best-effort streams. The requirements of this discipline for queuing and scheduling are marginal and are not complex in comparison to other, more intricate queuing disciplines which also provide differentiated service. The PQ algorithm arrange for a simple way of backing up differentiated service classes, where various queues are serviced in different ways in accordance to the priority assigned. The computational requirements for buffering and scheduling are little and not very intricate, even in case of implementing PQ with high-speed links on network equipment. Real-time traffic with delay intolerant and loss sensitive data can be efficiently secured from supplementary greedy flows viz. TCP sessions, and high priority traffic can be warranted low delay and jitter. On the other hand, PQ suffers from certain limitations. If the bulk of high priority traffic turn out to be excessive, lower- priority traffic possibly will suffer due to complete resource starvation. If the queuing rate stays constant but the removal rate decreases, the data packets start dropping from the head or tail of the queue, or from both. As the buffer memory allotted to lower priority queues overflow, the drop rate of traffic placed in these queues increases. This results in increased packets drop and latency. As regards of TCP sessions, the TCP traffic might get stale if it is required to be retransmitted. Introduction of such a traditional queuing and scheduling scheme can disturb the network performance such that the service offered to the entire network drops due to starvation of the large streams of traffic and applications. Example, SIP traffic can be prioritized using PQ in a Session Initiation Protocol (SIP) session. Yet, if queries cannot be resolved by DNS servers and DHCP offers are not able to reach users and SIP servers gains acquired by preferring SIP traffic are limited since users would not be accessible to create SIP sessions. Furthermore, mischievous high priority streams can lead to a surge in delay and jitter in supplementary high priority flows which share the same queue. Also, a real time service consist of numerous different applications and different packet types, and all of them must be given comparable service levels so that the real-time service operates accurately. The consequence of absolute resource allocation and scheduling to the prioritized traffic in PQ can end up causing a network malfunction since low-priority traffic gets stalled. Offering specific kind of traffic higher priority penalizes lower-priority traffic. Proper provisioning

and rate controlling of high priority traffic must be done prevent low priority data from service breakdowns. Strict mode or rate controlled mode can be used to implement priority queuing. Example, in case the traffic rates go beyond a predefined threshold level which can be an explicit rate or percentage of link bandwidth, policing of the rate of priority queue can be done in order to drop or reclassify the traffic. This method prevents starvation of low priority data and can also regulate the delay introduced in high-priority traffic queues by instituting a boundary on the maximum quantity of traffic which can be queued.

4. Weighted Round Robin

Weighted round robin (WRR) scheduling addresses the inadequacies of priority queuing and fair queuing. The basic idea of WRR is to manage the scheduling of classes which demand different bandwidth. WRR achieves this by permitting numerous packets to be forwarded from the queue every time that particular queue gets a scheduling turn. It also addresses the problem with PQ where one queue can famish queues which have lower priority. This is done in WRR by permitting minimum one packet to be forwarded from every queue comprising packets in every scheduling slot. The number of packets that are serviced in every slot is defined by the weight of each queue which is generally a percentage of the total interface bandwidth, hence, representing the service difference within the queues and the class of traffic allocated to those queues. WRR is illustrated in Fig. 2.10.

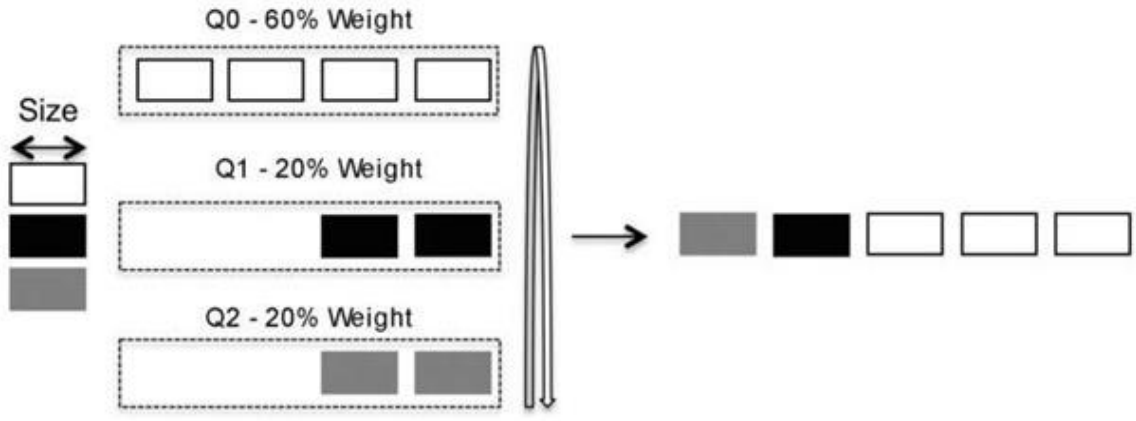

Fig. 2.10 WRR scheduling

It shows that three packets are forwarded from queue Q0 and one data packet is removed from both queue Q1 and queue Q2. The number of data packets forwarded reflects the weight of the queue. It can be seen that Q0 has three times higher weight in comparison to Q1 and Q2 therefore, it forwards three times more data packets in each scheduling slot. WRR does not have the true knowledge of the actual sizes of different packets in the queues that are to be scheduled. A general optimization for average packet size is done for queuing and scheduling. Still, the packet sizes are only based on estimation and do not have any true meaning concerning the actual traffic mix in every queue. This behavior of weighted round robin mechanism is both a blessing and a bane because WRR does not have complex resources which mandate state computation as in the case of weighted fair queuing that must convert bits to bandwidth scheduling, hence simple to implement. WRR is suitable for management of numerous flows and sessions, thus helping this scheme to become a core solution to deal with voluminous traffic and congestion. This scheduling scheme is easy to implement where every queue is assigned a weight which is a representation of the interface bandwidth for implementing differentiated services. It prevents starving of one queue at the expense of others for the reason that it is likely to offer service priority by assigning different weights to various queues. The downside of WRR scheduling is its unawareness regarding bandwidth due to the reason that it does not manages variable-sized packet. A 1500-byte packet is considered same as a 64-byte packet for attaining a scheduling slot. In actual fact, the weights assigned to packets' counts solely in every round of scheduling. When the mix of traffic is to a certain degree equivalent to classes and queues, this scheduling discipline is, over time, reasonably fair in its scheduling. On the other hand, for short term usages, the differences might be great. And in the case where the traffic classes have larger discrepancies in traffic mix and types, and therefore higher differences in packet sizes, this type of queuing can become somewhat unfair, preferring classes dominated by bigger packets. Services that impose strict demands on factors such as jitter and delay can be greatly affected by the order of scheduling of other queues. Example, real-time traffic that have relatively smaller packets will be at a loss when compared to TCP based traffic having larger packets as shown in Fig. 2.11. Here, both the queues have equal weights. But, since the packets in Q2 are half

the size of data packets in Q1, in effect Q2's bandwidth rate is half that of Q1.

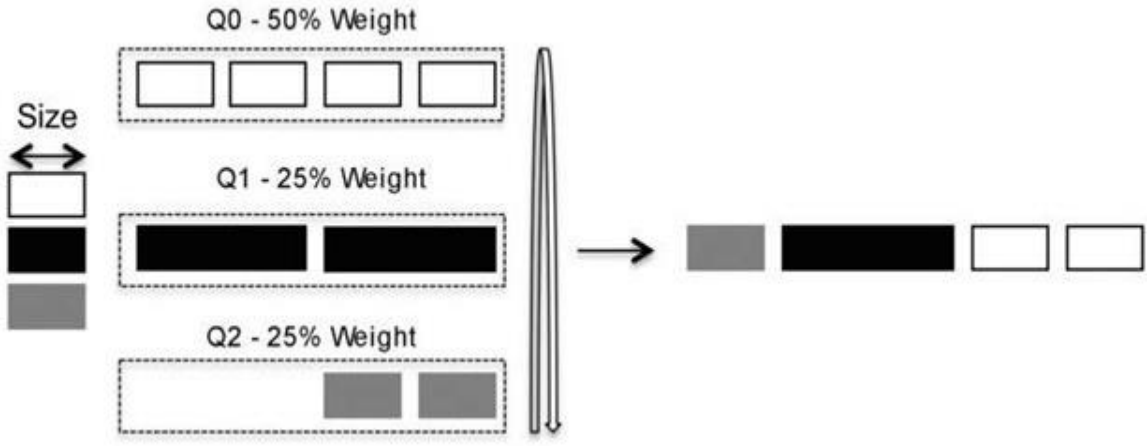

Fig. 2.11 Comparison of large and small packets in WRR

5. Weighted Fair Queuing

Weighted fair queuing (WFQ) is also called "bit-by-bit round robin", as it a queuing and scheduling method where the queue are provided service based on bits and not based on packets. In 1989, Demers, Shenke, Keshav, and Zhang developed WFQ. It emulates the concept of GPS (Generic Processor Sharing) of virtual processing time. Here, every queue is assigned a weight proportional to the shaping rate or the interface rate. WFQ is conscious of packet sizes and therefore can provision for variable-sized packets. The profit is that sessions having big packets are not given more scheduling time in comparison to sessions having smaller packets, for the reason that effectively the emphasis of WFQ is not packets but bits. So data flows having smaller packet sizes suffer no unfairness. At the head of queues, computation is performed on the bits of every packet based on which every queue is scheduled. For the reason that the traffic computation is performed on the basis of bits stream and not depending on packets, and as the router takes and transfers are indeed packets, there is an increase in complexity. The scheduling principle of WFQ is illustrated in Fig. 2.12. On the face of it, it may give the impression that WRR is analogous to WFQ. But the two of them are different as WFQ services bits whereas WRR handles packets in every scheduling slot. The figure shows that, the total bytes scheduled in every slot is equal for all the queues and is a reflection of the weight value as all queues have

equal weight. Queue Q0 forwards three packets having a total of X bytes, queue Q1 forwards one packet of Y bytes, and queue Q2 forwards two packets with a total value of Z bytes. The weight factor is actually an allowance for the amount of resources that can be allocated and used. The implementations done on the basis of WFQ algorithm afford service differentiation amongst classes and their respective traffic aggregates, instead of simply differentiating among individual flows. A weight assigned to every class segregates scheduling and bandwidth ratio for every class of traffic. Also, WFQ can manage variable lengths packet as it is bits aware. The downside of WFQ is that due to bit computation it is resource-intensive. The actual idea behind WFQ also takes up many resources as the flows are not grouped into classes having limited queues. Rather, every flow or stream acquires its own buffer quota or queue, in the same manner as FQ. Due to such high resource requirements and the intricate computations required to maintain the state for every stream and its packets, WFQ to a larger extent is implemented on CPU-based platforms where queuing is normally based on bus-based architectures.

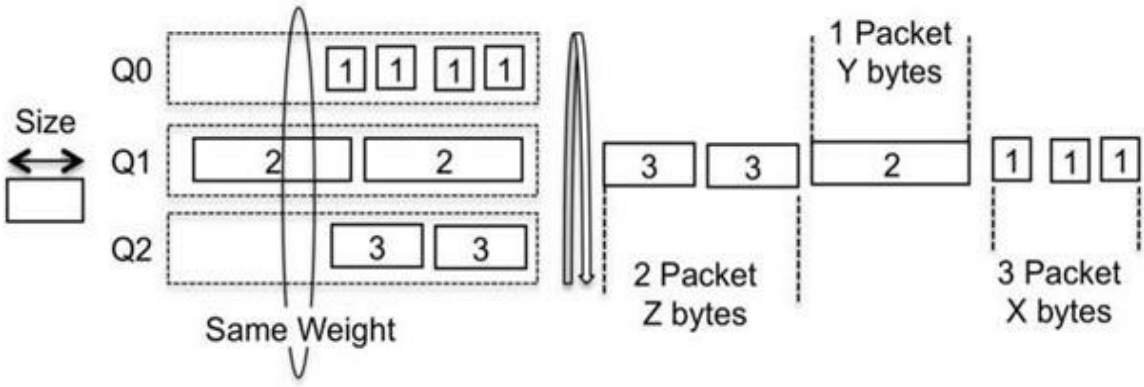

Fig. 2.12 WFQ scheduling

Table 2.2 shows the specific QoS requirements of different types of applications in terms of packet loss, delay, delay variation or jitter and bandwidth.

Table 2.2. Application specific QoS requirements

TYPE OF SERVICE	LOSS	DELAY (ONE WAY)	JITTER	BANDWIDTH
Voice	<=1%	<=150 ms	<=30 ms	21 kbps-320 kbps

Interactive Video	<=1%	<=150 ms	<=30 ms	On demand
Streaming Video	<=5%	<=buffer time	On buffer time	On demand
Data	-	-	-	Best effort

IP networks were designed for non-real time applications and were not highly sensitive to delay or jitter but VoIP applications require timely packet delivery with low delay, jitter and packet loss values [18]. Authors in [18] simulated a simple network with two queues: low priority and high priority queue and presented packet delay distributions of traffic at different network loads. DiffServ is proposed as a scalable method for providing quality of service over IP networks where packet scheduling is an important mechanism to achieve QoS. It helps in prioritizing data and reducing delay for sensitive applications. The scheduling scheme proposed in [19] adaptively adjusts the weights of scheduler according to the dynamics of average queue size and can be applied to weighted round robin and weighted fair queuing to achieve low loss rate, low queuing delay and delay jitter. Authors in [20] proposed a joint packet scheduling and channel allocation scheme for multiuser multichannel wireless networks in order to meet tradeoff between fairness and throughput using a Finite-State Markov Chain to define each channel. In [21], a cross-layer analytical framework, for radio link level delay statistics evaluation in a wireless network using adaptive modulation and coding, weighted round robin scheduling and automatic repeat-request based error control is analyzed. A comparison between various QSD algorithms: PQ (priority queuing), LLQ (low latency queuing), WRR (weighted round robin), WRR/SB (weighted round robin with save and borrow) and WRRPQ (weighted round robin with priority queuing) under different conditions from the point of view of delay, throughput and packet loss is done in [22-23] to analyze the possibility of their usage in modern converged communication networks. In [24], an analytical model

is proposed to abstract the equivalent service provided by router to each flow, which enables to calculate the per-flow worst case delay bound over the whole network. A weight allocation algorithm is developed to automatically assign proper weights to individual flows to satisfy their delay constraints and the *just-in-time* transmission policy is followed to avoid unnecessarily fast transmission. A Markov chain based model has been presented in [25] to predict dynamic bandwidth allocation in DiffServ networks. At a time-slot, the proposed Markov chain is used to predict the bandwidth requirement at the next time-slot, and resource is the allocated accordingly. The proposed scheme can effectively reduce the operation overhead in bandwidth allocation and further reduce the connection blocking probability. In [26-27], the authors propose a cost-effective analytical model for a finite capacity queueing system, for devices/things in IoT that have heterogeneous features with limited buffer capacity, storage and processing power. A pre-emptive resume service priority and push-out buffer management scheme is proposed to analyze the mean queue length and the blocking probability of high and low priority traffic for system with various capacities. In [28], network layer routing algorithm are also taken into consideration in message scheduling, aiming to provide a more optimal solution by applying certain degree of cross-layer methodology. Here sensor nodes are divided in IoT subgroups. Each subgroup has a broker delivering for all nodes and maintaining two queues for high priority and best effort messages respectively. QoS awareness is introduced in IoT subgroups by assigning traffic priorities and scheduling them with proposed algorithm making them energy efficient as well. A three-layer QoS scheduling model for service-oriented IoT is proposed in [29]. At application layer, the QoS schedule scheme explores optimal QoS-aware services composition by using the knowledge of each component service. At network layer, the model aims at dealing with scheduling of heterogeneous networks environment; at sensing layer, it deals with the information acquisition and resource allocation scheduling for different services. The proposed QoS-aware scheduling for service-oriented IoT architecture is able to optimize the scheduling performance of IoT network and minimize the resource costs. In [30], a priority based channel access and data aggregation scheme is proposed to reduce channel access and queuing delays in clustered industrial IoT networks. A prioritized channel access mechanism is developed

by assigning different MAC layer attributes to the packets coming from high and low priority nodes, based on application specific information provided by cloud center and then a preemptive M/G/1 queuing model is employed by using separate low and high priority queues before sending aggregated data to cloud. Authors in [31-32] present a dynamic Markov Chain based scheduling scheme. By estimating previous bandwidth allocated to prioritized traffic and the amount of decrease/increase in probability of average queue length of prioritized packets in consecutive cycles, the scheduler dynamically adjusts the percentage of link bandwidth allocated to it in each cycle. This helps in providing guaranteed services to real time traffic in IoT applications and simultaneously provide better throughput for non-real time traffic as well.

CHAPTER 3
PROBLEM FORMULATION AND SOLUTION METHODOLOGY

3.1 INTRODUCTION

According to a forecast by the IoT connections outlook there would be near about 24.9 billons Internet of Things (IoT) connections compared to 8.1 billon population by the year 2025. Nowadays the number of physical devices (IoT devices) are rapidly growing like wearable smart devices, smart video surveillance, security devices, medical applications, which results into various streams of data which are stored on to the different cloud based servers so that the data can be used for various analytics and can be retained for a long period of time. In such enormous smart devices connectivity it is of utter importance to maintain better quality of service (QoS). Many different novel technique [18,25,34] are implemented to improve the quality of service, majorly many of which propose to have separate queues for the different priority based data. Many of these techniques tried to improve the performance of high priority services by degrading the performance of low priority services in terms of bandwidth allocation. Few of the approach [34] uses predictive models for predicting the average queue length by using the exponentially weighted moving average (EWMA) method and then bandwidth can be allocated efficiently by using the average queue length for high and medium priority data and thus the efficiency is obtained for both types of the packets. But due to use of EWMA a steep increment was observed in the average queue size which resulted in allocation of higher portion of bandwidth to the high priority and thus more or less, the medium and low priority packets get the same type of bandwidth allocation which resulted in the previous approaches. EWMA uses a linear smoothing function due to which it smooths all the inputs including an accidental network bursts [35]. We have different approaches for improving the energy efficiency, network topology and performance related issues.

In real world the IoT network consists of various types of devices connected in a network like sensors with critical information, CCTV surveillance with huge amount of data, smart wearable gadgets. The internet based network model approach [36] like DiffServe, IntServe cannot be used for addressing a network model for IoT devices because these models cannot address the real QOS related problems faced in the actual IoT. However many other approach uses Markov chain [25] for optimizing the QoS related issues in the communication of delay sensitive network but due to uncertain type of devices in the network it is not efficient in IoT. Apart from the network models the scheduler also plays a vital role in terms of QoS. Many schedulers proposed [20,37] in the earlier approaches often took WRR into consideration compared to others due to its simplicity. WRR [37] is a round robin based scheduler for the network model. WFQ is a generalized process sharing policy (GPS) and extension to fair queueing policy. WFQ is considered to be better than the WRR which was used by the previous approach. The methodology for achieving better QoS for each services cannot be same, hence message based alert QoS approach [28] categorized the services into two different types i.e. critical and non-critical services. But many a times in a human centric IoT network few services are broadcasted as emergency which might be a non-critical service. Yet another [38] approach is used to acknowledge such services.

3.2 PROBLEM STATEMENT

Enhancement of Quality of Service in Internet of Things enabled applications.

3.3 QoS ISSUES IN CONVERGED IoT NETWORKS

A converged network provisions diverse applications, for instance, data, vedio and voice, at the same time over a shared infrastructure. Accommodating such diverse applications having different sensitivities and demands is a perplexing task. The end-to-end delay tolerable for (VoIP) packets is 150-200 ms. Moreover, the jitter in VoIP packets should be limited with the purpose of preventing the queues at the receiving end from being exhausted, resulting in breaks amid an audio flow. On the other hand, such rigorous delay

requirement are not present in data applications like downloading a file from FTP site, and jitter is not a big concern for such kinds of applications either. When many active VoIP, multimedia and data applications occur at once, mechanisms must be introduced such that a reasonable amount of less critical applications can remain active with acceptable quality as well, while the critically important application function properly. Countless data applications are based on TCP. The source retransmits a TCP segment after the timeout duration in case it is dropped. Hence, such applications show some tolerance towards packet drops. But, voice and vedio show minimal tolerance towards data loss. Therefore, network must introduce mechanisms to provide priority to such packets at times of congestion, to prevent packet drop. Network outages negatively influence all the applications and cause them to be disabled. Still, built in redundancy is introduced in well-designed networks, so that in an event of a sub-network failure the packets can be rerouted via a different path while the failed part is mended. The composite time taken to identify the failure, compute replacement paths, and rerouting the data packets need to be small enough such that the applications do not suffer and the users are not annoyed. Once more, data applications typically do not have as high network recovery demands as in case of multimedia applications. Fast recovery and network redundancy are a must in network outage. Network outage is undesirable, and mechanisms should be introduced to avoid it. On the basis of the preceding discussion, we will focus on four key concerns and challenges faced by IoT enabled applications.

1. **Available bandwidth -** Several concurrent data, voice and video applications contend over a limited amount of bandwidth. Absence of adequate bandwidth results in poor application performance in terms of delay, packet loss or delay variation. The users working on real-time applications perceive this immediately. The problem of bandwidth availability can be tackled in many ways such as increasing the link bandwidth, classifying and marking the traffic, deploying appropriate queuing mechanisms, etc. Increasing the bandwidth of the link is indisputably beneficial, but it cannot at all times be done quickly, and involves cost implications. Individuals/enterprises who merely increase the bandwidth when needed find this solution not to be very effective in case of heavy traffic bursts. Still, increasing the bandwidth of the link may be the initial action in some

scenarios, but not the final. Collective traffic classification and marking and congestion management, is a good method to offer satisfactory bandwidth for IoT applications.

2. **End-to-end delay -** Various actions and elements add to the total time taken by packets to travel from source to its destination such as compression, decompression, packetization, serialization, queuing, propagation and processing delay together contribute to the composite delay in packet transmission. Four main types of delay which results in end-to-end delay are:

i. Processing delay

ii. Queuing delay

iii. Serialization delay

iv. Propagation delay

Serialization and propagation delay are contributed by packet size, link bandwidth and average queue length. Delay can be reduced by increasing the link bandwidth. But, improvement in link capacity is time taking and costly, making this approach impracticable at times. Also, packets sensitive to delay can be prioritized and forwarded first using packet classification and marking and deploying queues. In certain cases the priority of packets might change as they enter from one domain to another therefore reprioritizing the packets can also help in reducing delay. Example, a critical marking packet leaving an enterprise network can be reprioritized as a best effort packet as it enters a provider network.

3. Delay variation (jitter) – On the basisi of the amount of simultaneous activity and traffic along with the network condition, packets belonging to the same flow can undergo varying delay as they traverse the network. The packets of any flow are processed, queued and dequeued independent of one another**.** As a result, they may reach out of sequence, and also their end-to-end delays can vary. For multimedia packets, it is important for the packets to reach the destination in the correct sequence and at the same rate with which they were released from the source which can be done with the help of a de-jitter buffer.

4. Packet loss - If the enormous amount of traffic depletes the capacity of the link or interface, packets may be dropped. Rapid bursts and failures are typically liable for these situations. Packet loss arises when network elements like a router exhausts its buffer space

to accept incoming packets and results in dropping them. Router can also drop few packets to make space available for higher priority packets. At times, interface reset can also cause packet flushing and dropping. Interface overrun can also result in packet drop. TCP retransmits the dropped packets; in the meantime, it decreases the send window size and slows down in case of high congestion and hefty network traffic volume. If a packet that is dropped belongs to UDP-based file transfer, the complete file may have to be retransmitted. This produces additional traffic in the network. Application which do not use TCP, and consequently are extra drop-sensitive, are known as fragile flows. In the course of a voice call, packet loss causes audio breakup. Video conferences can experience jerky images and out of sync audio in the event of packet drop or prolonged delays.

When traffic and congestion in the network are high, packet drops, prolonged delays and jitter is experienced by application users. Only with the help of an appropriate QoS configuration we can avoid such complications or at any rate bound them to low-priority packets.

3.4. SOLUTION METHODOLOGY

Future IoT networks are anticipated to sustain immense amount of heterogeneous devices/sensors in varied applications extending from eHealthcare to industrial control systems. Convergence of data, voice and video streaming into packet oriented network call for the support of QoS. The method of selection of packets waiting in the network nodes to be forwarded is a major factor in providing quality of service in packet based networks. Therefore, the QoS assurance offered by the network is prominently effected by the nature of scheduling mechanism used. The joint scheduling scheme used consist of following main features:

1. Different kind of traffic from various applications is separated on the basis of traffic characteristics such as bit rate of the traffic, rate of packet loss and acceptable delay into diverse classes and prioritized into high priority, medium priority and low priority data on grounds these characteristics.

2. We consider three queues having high, medium and low priority in order to store respective type of packets as defined above and then schedule them in a round robin fashion.
3. The high priority services cannot endue higher queuing delays and jitter and are delay sensitive, therefore, a small buffer memory for high priority queue is set aside. On the other hand, buffer memory for medium and low priority queue is kept large to minimize the packet loss.
4. Average queue length for current slot is calculated by using Adaptive Exponentially Weighted Moving Average (AEWMA).
5. The scheduler comprises of two different steps. First, the weighted fair queueing mechanism will take the previous slots' average queue size into consideration for the calculation of current slots' average queue length. The bandwidth will be allocated by multiplying the average queue length (which is calculated based on the current queue size and the previous average queue size) with a coefficient. The portion of bandwidth left out after allocation for high priority and medium priority is allocated to the low priority. This helps in weight optimization for the current time slot on the basis of increase or decrease in average queue length to apportion the bandwidth needed for high and medium priority services.
6. For keeping delay and jitter within tolerable limits for high priority data (\10 ms), the minimum threshold and maximum thresholds are selected that will act as a marker to allot the proper weights.

The proposed predictive average queue length based dynamic scheduling algorithm is as given below:

Initialization

While (until packets arrive):

Segregate the arriving packets into high, medium, low priority services;

Store the packets into different queues as per the priority;

Calculate the average queue length of high priority using AEWMA (Eqn1);

Calculate the average queue length of medium priority;

Assign the bandwidth weight based on average queue length for high priority using Eqn2;

Assign the bandwidth weight based on average queue length for medium priority using Eqn3;

Assign the bandwidth weight based on average queue length for low priority using Eqn 4;

Schedule the packets using the WFQ scheduler based on the dynamic bandwidth weights;

End.

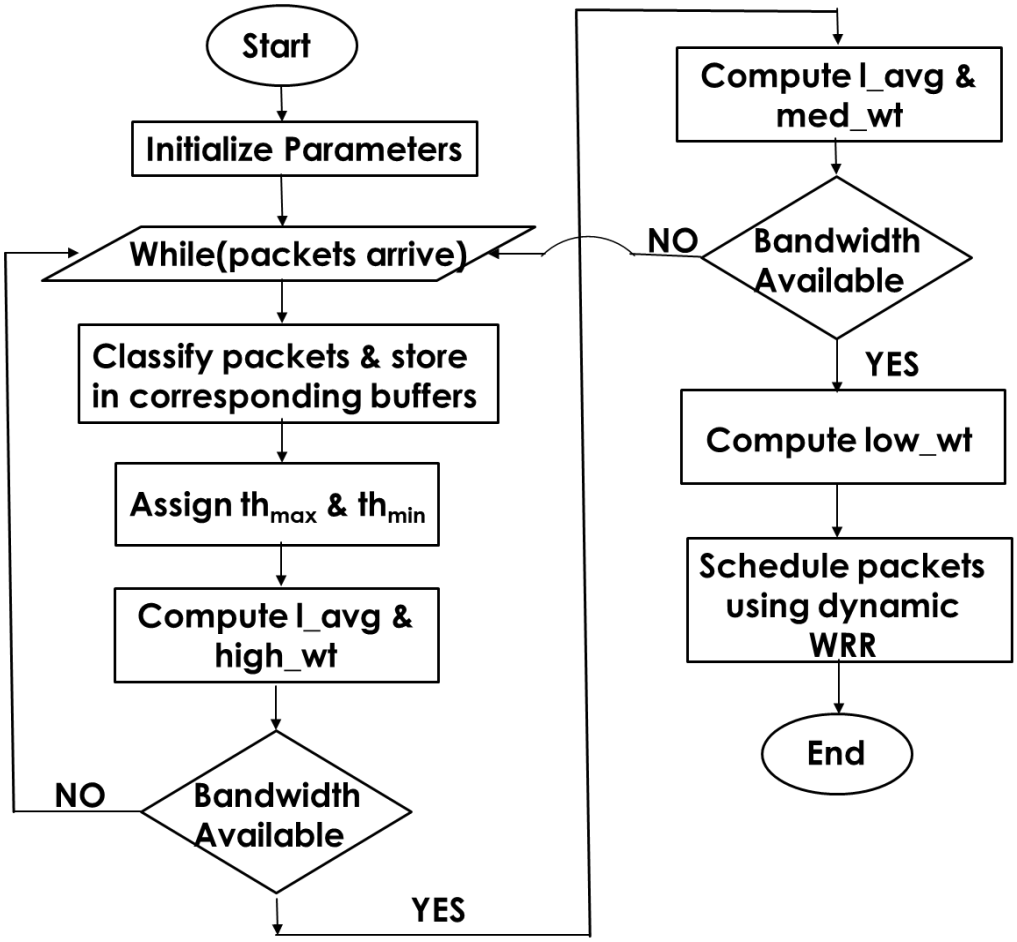

Fig 3.1. Solution Methodology

In initialization we will create three different queues respectively for high, medium, low priority services. The queue size of high priority data is only 10 as these services will offer lower delay, packet loss whereas the medium and low priority queue size will be 100. Initally the bandwidth weight for high priority is 0.3, medium priority is 0.3 and low priority is 0.4. All the bandwidth sum is equal to 1 which represents the current bandwidth

of the IoT network. When a new packet arrives in the network it is segregated into high, medium, low priority services based onto different properties which includes bit rate, tolerable delay, packet loss rate etc. then it is stored in different queues initially.

3.4.1 The adaptive exponentially moving average (AEWMA) algorithm

The AEWMA (Adaptive Exponentially Moving Average) algorithm was proposed in 2003 by Capizzi and Masarotto [41]. It is a type of improved EWMA algorithm with an adaptive smoothing function and is described below [35]:

$$S_0 = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad (n \in N^+), \tag{3.1}$$

$$S_i = \big(1 - w(e_i)\big)\, S_{i-1} + w(e_i)\, X_i \tag{3.2}$$

Here, Si is i^{th} AEWMA statistical value, X_i is i^{th} sample values and (e_i) adaptive smoothing function. Equation (3.3) can be derived from equation (3.1) and (3.2).

$$S_i = S_{i-1} + w(e_i)\,(X_i - S_{i-1}) \tag{3.3}$$

Now, we put $e_i = X_i - S_{i-1}$. Then, the score function (e_i), is:

$$\emptyset(e_i) = w(e_i)(X_i - S_{i-1}) \tag{3.4}$$

On combining equations (3.1), (3.3) and (3.4) we get-

$$S_i = S_{i-1} + \emptyset\,(e_i) \tag{3.5}$$

We can see that if place, $\emptyset(e_i) = \lambda_{EWMA}\, e_i$, we will obtain the classic EWMA algorithm. The typical EWMA algorithm is a special case of adaptive exponentially weighted moving average algorithm. The AEWMA scheme has the characteristics and benefits of the classical EWMA algorithm. Now the AEWMA algorithm, that makes use of the score function in place of fixed initial parameters, is seemingly much more adjustable to a wider range in comparison to the EWMA algorithm.

3.4.2 The predictive average queue length model

The predictive average queue length model employs AEWMA method for the calculation of average queue length of high priority services which considers the previous average queue length. Then by using equation (3.6) average queue length is calculated.

$$\text{avg\_len} = \text{avg\_len} + \varphi\,(e) \tag{3.6}$$

Here, avg_len represents the average queue length and φ (e) is a nonlinear smoothing weight function which is called as score function. The φ (e) score function is as defined below:

$$\varphi(e) = \begin{cases} e\{1-(1-\lambda_{ewma})[1-(e/k^2]^2]\} & (e < k) \\ e & (e > k) \end{cases} \quad (3.7)$$

λ_{ewma} is a smoothing factor which is set to 0.01, k is the threshold value which is set to 5.

$$e = len_{inst} - len_{avg} \quad (3.8)$$

Here, e is the difference between the instantaneous queue length and average queue length. EWMA follows a linear method for calculating the average length but AEWMA follows a nonlinear method to find the average queue length so that accidental network bursts can also be considered. Hence, we can say that the AEWMA is better than EWMA for finding the average queue length.

3.4.3 Dynamic bandwidth allocation

For calculation of the dynamic bandwidth weight the predefined method of EWMA approach [34] is used which states that if the average queue length is lower than the threshold queue size then the bandwidth weight is allocated as 0.3. If the average queue length increases above the minimum threshold then the weight also increases along with it up to 0.7 (max weight). For calculating the weight we use the equation (3.9).

$$high\_wt = \begin{cases} high\_init & ; l\_avg \in \{0, 0.833\} \\ (C1 * high\_len\_diff) + high\_pre\_wt & ; l\_avg \in \{0.833, 3.667\} \\ high\_upper & ; l\_avg \in \{3.667, 10\} \end{cases} \quad (3.9)$$

The value of high_init is 0.3 if the average queue length is below the minimum threshold queue length. If the queue size is between the maximum threshold queue length and 10 then the bandwidth weight goes up to 0.7 which is 70% of the available band width. high_len_diff is the difference between the current queue length and previous queue length. high_prev_weigths the previous bandwidth weight allocated for the high priority service. The scaling factor C1 which is a constant which is calculated by the below given

equation (3.10).

$$C1 = \frac{0.3}{high\_queue\_len_{max} - high\_queue\_len_{min}} \quad (3.10)$$

The value of high queue length max is 3.667 which represents allowable delay and high queue length min is 0.833 which represents a desirable delay. The average queue length for the medium priority service is calculated by simple average formula and the dynamic bandwidth weight is allocated by using the equation (3.11).

$$\text{med\_wt} = \begin{cases} \text{med\_init} & ; \text{l\_avg} \in \{0, 60\} \\ (1 - \text{high\_wt}) & ; \text{l\_avg} \in \{61, 100\} \end{cases} \quad (3.11)$$

The initial bandwidth weight for the medium priority queue is set to 0.3 until it reaches the threshold queue length of 60, otherwise the weight is total bandwidth weight subtracted with the high priority queue weight. The low priority queue weight is allocated with what is left after the allocation of weights for high priority queue and medium priority queue as shown in equation (3.12).

$$\text{low}_{wt} = 1 - (\text{high\_wt} + \text{med\_wt}) \quad (3.12)$$

3.4.4 Performance assessment

The performance assessment is done based on to the end to end delay, delay jitter which is most crucial for the high priority service and the total packet loss is required for all the services. We will calculate one way end to end delay, average delay, average delay jitter, average packet loss for all the sources.

$$ete_{delay} = R_n(i) - T_n(i) \quad (3.13)$$

Where $R_n(i)$ is the time at which packet 'i' is received by nth source and $T_n(i)$ is the time at which packet 'i' is sent by the nth source and ete_{delay} is the one way end to end delay. For calculating the average end to end delay for each type of services we will consider all the packets sent p and all the sources S. the equation as given below.

$$avg_{ete_{delay}} = \left(\sum_{n=1}^{s}\left(\sum_{i=1}^{p} ete_{delay_{n(i)}}\right)/p\right)/S \quad (3.14)$$

The delay jitter is calculated by the absolute difference of one way end to end delay of two consecutive packets. The one way end to end delay is calculated by equation (3.13).

$$delay_{jitter} = ete_{delay_{n(i)}} - ete_{delay_{n(i-1)}} \quad (3.15)$$

The packet loss ratio is obtained by the below equation.

$$packet_{loss} = \frac{\left(total_{packet_{transmitted}} - total\_packet_{recived}\right)}{total_{packet_{transmitted}}} \quad (3.16)$$

Here, $total_{packet_{transmitted}}$ is the total number of data packets that are transmitted from any source and $total\_packet_{recived}$ is the total number of data packets that reached their respective destinations.

CHAPTER 4

SIMULATION AND RESULT ANALYSIS

4.1 SYSTEM ARCHITECTURE

A sample interconnected network model as shown in Fig 4.1 is used for analyzing the predictive approach. The sample network model consist of various devices including CCTV camera which can generate data of larger size, smart phones, proximity sensors, BP monitoring devices, landline phones which can offer a larger delay . The devices takes about 10ms of inter arrival time. The bottle neck condition is of 2 Mbps. The performance analysis is applied on the sample interconnected IoT model.

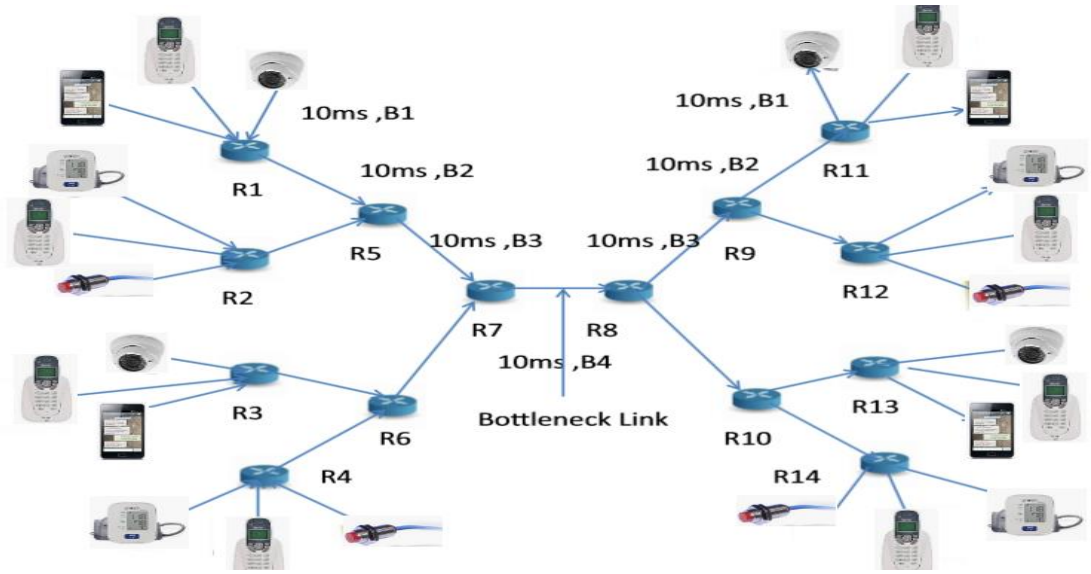

Fig 4.1 IoT interconnected network model

The incoming data packets are segregated on the basis of their traffic characteristics such as acceptable delay, bit rate, packet loss rate, etc. into high priority traffic, medium priority traffic and low priority traffic. After segregation, the data packets are placed in their

respective high, medium and low priority queues. As the packets keep on entering the queue in real time, the average queue length is calculated for queues using AEWMA algorithm and weights are allotted dynamically based on the average queue length. After that, the packets are scheduled using the weighted fair queuing scheduling algorithm as shown in Fig. 4.2.

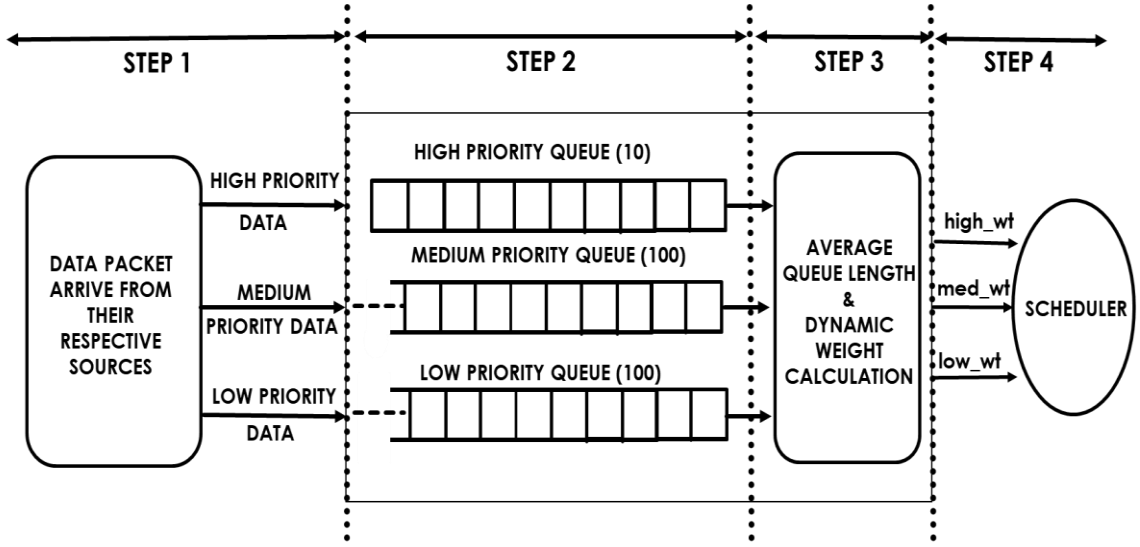

Fig. 4.2 Process Flow

4.2 SIMULATION TOOL

The simulation results are obtained using Python and SimPy, as Python is widely used in the domain of IoT, Data Analytics, and other platform.

4.3 SIMULATION RESULTS

The traffic characteristics are implemented on the interconnected network model shown in the fig 4.1. The high priority services are generated from the proximity sensors, digital BP monitoring devices, the medium services are generated from the mobile phones, low priority services includes the data from landline phone i.e. peer to peer connectivity, emails, chats. The network traffic type used for all the 12 sources is Poisson distribution. The packets are differentiated based on different characteristics like bit rate, tolerable

delay. The High priority packets are stored in a queue of length 10, medium priority packets are stored in a different queue of length 100 and the low priority packets are stored in a queue of length 100, then the packets are scheduled using the AEWMA based scheduling approach. The data size for high priority services goes up to 100 bytes as the data are majorly generated by the sensors. The data size for medium priority services ranges between 20 to 571 bytes which are generated by mobiles hence a wide range of data size is resulted, whereas the low priority services data size goes up to 100 bytes generated by the landline phones, peer to peer connectivity.

Table 4.1. Traffic characteristics used for the simulation

Parameters	Bottleneck with bandwidth of 2Mbps		
Service type	High Priority	Medium priority	Low priority
Traffic type	Poisson's distribution	Poisson's distribution	Poisson's distribution
Packet size	Up to 100 bytes	20-571 bytes	Up to 100 bytes
Number of flow at a time	12 flows at a time		
Node topology	Random Node Topology		
Data rate	2 Mbps data rate		
Number of nodes	38		
Traffic direction	Left to right		
Number of packets sent	16,022	439,906	438,909

Firstly, we have compared the EWMA and AEWMA for calculating the average queue length and dynamic weights.

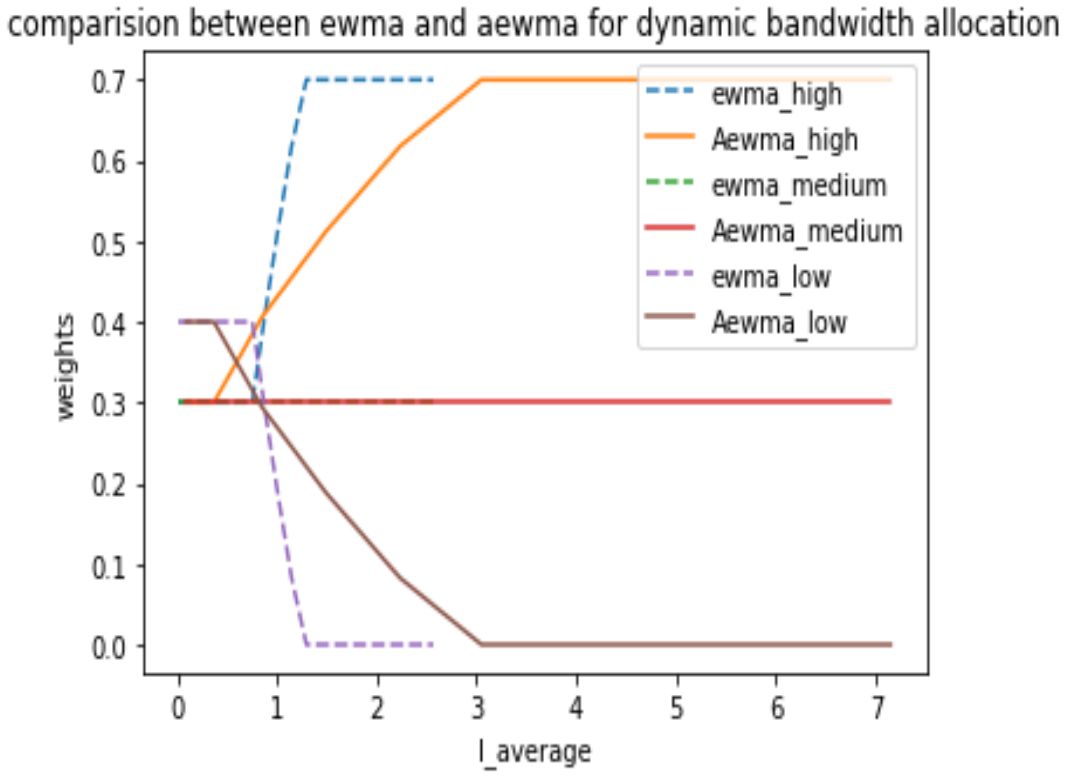

Fig 4.3 Comparison between the EWMA and AEWMA for dynamic band width allocation.

We can clearly see that the Fig 4.3 shows that there is an abrupt increase in the bandwidth weight for high priority services and the abrupt decreases for the low priority services due to the linear smoothing function used by the EWMA (exponentially weighted moving average), which would also smooth the accidental anomaly like network burst. Whereas the AEWMA (adaptive exponentially weighted moving average) which uses non-linear smoothing function, shows a gradual increase for the high priority service and gradual decrease for the low priority service. Hence we can clearly see that the AEWMA is an improved version of the EWMA. The pseudo-code for comparison of EWMA and AEWMA algorithm for dynamic weight calculation is given below.

For EWMA -

Initialization

While (Packets Arrive)

{

Classify packets and store in corresponding buffers;

Calculate l_avg and assign weights.

l_avg = (1-0.01)*l_avg+0.01*i

if(l_avg<0.833) : high_wt = 0.3

if(l_avg>0.833 and l_avg<3.667) : high_wt = 0.105857+high_wt

if (high_wt>0.7) : high_wt = 0.7

else : high_wt = 0.7

if i<60:medium_wt = 0.3

else : medium_wt = 1-(high_wt)

low_wt = 1-(high_wt+medium_wt)

}

For Adaptive EWMA:

Initialization

While (Packets Arrive)

{

Classify packets and store in corresponding Buffers;

Calculate l_avg and assign weights.

If less than threshold : l_avg = l_avg + e*(1-(1-0.01)*((1-(e/5)**2)**2))

else: l_avg = e

if (l_avg<0.833): high_wt=0.3

if (l_avg>0.833 and l_avg<3.667): high_wt=0.105857+high_wt

if (high_wt>0.7) : high_wt=0.7

else : high_wt=0.7

if i <60 : medium_wt=0.3

else : medium_wt=1-(high_wt)

low_wt=1-(high_wt+medium_wt)

}

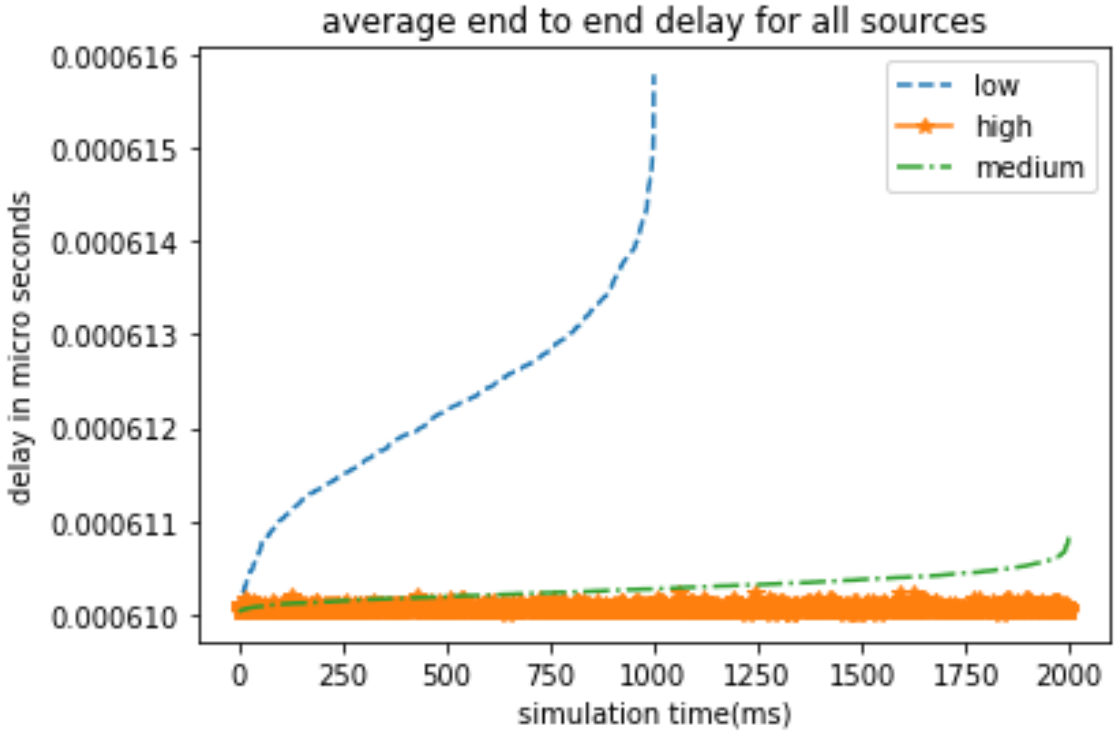

Fig 4.4 .Average end to end delay for all the sources

The results of average delay is shown in the Fig. 4.4 which shows the average delay of high priority services are 61ms, the delay of medium priority services are up to 61.1ms and the delay of low priority services goes up to 61.5 ms. The average delay includes the delay overhead by the link and the path. Fig 4.4 shows the average end to end delay for high, medium, low priority services from all the 4 sources from which we can see that high priority data undergoes the smallest amount of delay while the low priority packets suffer largest delay. The delay faced by medium priority data is more than that of high priority packets but lesser than low priority packets. Overall, the delay for all types of traffic is within tolerable limits.

Fig. 4.5 shows the average packet loss for high, medium, low priority services from all the 4 sources. The average packet loss ratio of high priority services from four sources is less than 1 percent, the total packets sent were 16022 out of which 103 were dropped. The average packet loss ratio for medium priority services from 4 sources is 1.8 percent, the

total packets sent were 439906 out of which 8039 were dropped. The average packet loss ratio of low priority services from four sources is 16.4 percent, the total packets sent were 438909 out of which 72329 were dropped

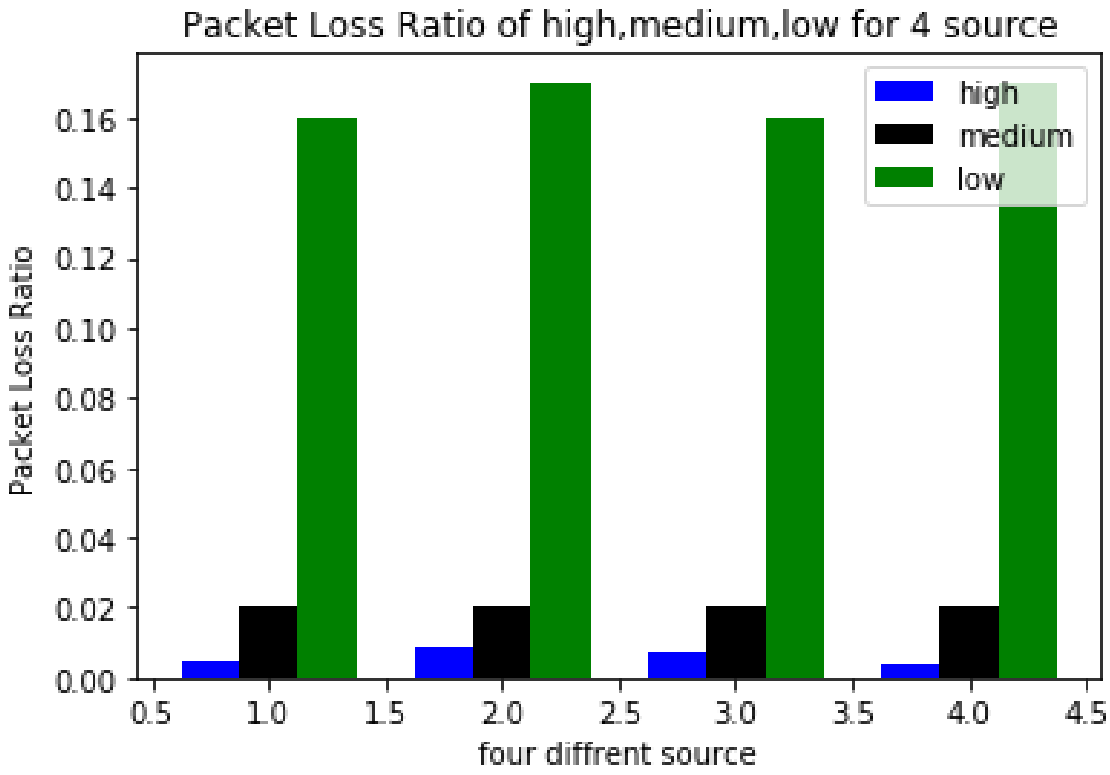

Fig 4.5 .Average packet loss ratio for all the sources

The average loss ratio for low priority service is higher as the max band width is allocated for high priority service and medium priority service as the queue length increases. Data rate used for high priority service is 84 Kbps below which no packet loss was detected. Data rate used for medium priority service is 0.8 Mbps below which no packet loss was detected. Similarly the Data rate used for low priority service is 1.116 mbps below which no packet loss was detected. The Data rate were detected by gradually increasing and identifying the packet loss. We can see from the figure that high priority data undergoes the smallest amount of loss while the low priority packets suffer highest losses. The loss faced by medium priority data is more than that of high priority packets but lesser than low priority packets. Overall, the loss for all types of traffic is within acceptable limits.

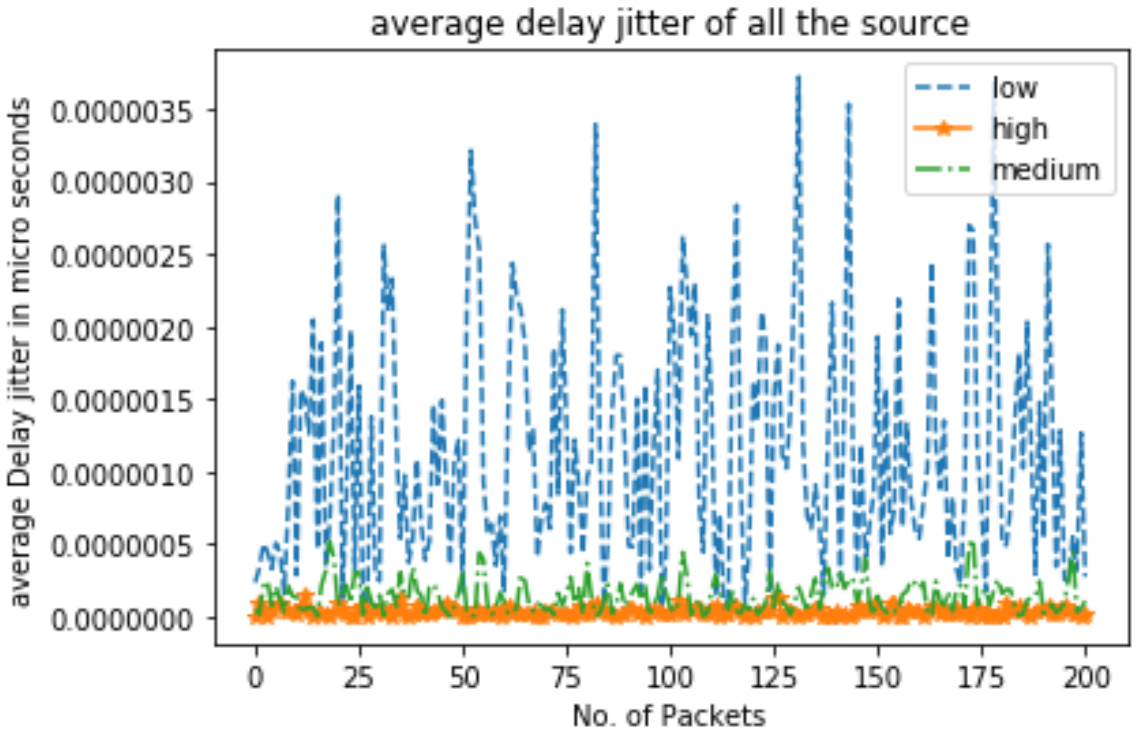

Fig 4.6 Average delay jitter for all the sources

The average delay jitter for high priority service is less than 0.5ms, for average delay jitter medium priority service is up to 0.5ms, and for the low priority services the average delay jitter is up to 2.5ms. The average delay jitter for low priority services is higher compared to high priority and medium priority services because the high priority services get complete bandwidth of low priority data when the average queue length is above the threshold. So, the bandwidth left for low priority is zero. Hence, the low priority packets gets more end to end delay and delay jitter. Fig 4.6 shows the average delay jitter for high, medium, low priority services from all the four sources.

CHAPTER 5

CONCLUSION

The Internet of Things is a new paradigm connecting every ‘thing’ and every one to the each another and eventually turning the physical world to a digital one. A simple click on a users’ smartphone can connect her/him to any remote location several kilometers apart and provide with necessary information or action as demanded by the user. With a tremendous growth in the available technologies and high speed internet, connectivity at anytime and anywhere has become possible. According to International Telecommunications Union, Internet of Things is ‘a global infrastructure for the information society enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies’. This paradigm will lead to innovations which will construct new kind of interactions between things and humans, and assist the realization of applications and services for enhancing the quality of life and exploitation of resources. The number of connected objects are increasing tremendously. Such enormous number of connected devices will generate massive volumes of data which has to be analyzed and stored. According to an IBM study, 2.5 quintillion bytes data is created every day. The storage and processing of this huge amount of data is not a trivial task.

The expectations of the users from the services offered by Internet of Things is not very much different from traditional computer based internet services .i.e. delivery of the service with guaranteed QoS levels. An example is that of ER (emergency response) IoT applications. Here, data should be received immediately from the deployed sensors and analyzed accurately to provide a timely response to probable damages caused by events such as natural disasters or medical emergencies. Such applications are extremely time sensitive and a delay in collection, transfer, assimilation and analysis of data can have catastrophic consequences. In IoT application QoS guarantee is challenging and difficult

to attain with the present state-of-the-art solutions offered in the background of IoT programming and resource management models (e.g. Amazon IoT, Google Cloud Dataflow, IBM Quark). An imperative challenge is that IoT eco-systems characteristically comprises of several layers like sensing layer, gateway layer, network layer, and cloud layer encompassing numerous heterogeneous hardware and software resources along with data types from different human and digital sensors. Providing QoS guarantees to customers demands the technical capability to warrant that their QoS needs will be managed across all the layers of the IoT application eco-system.

Many techniques for providing QoS guarantee have been described in literature and previous research works. But the traditional methods are not suitable for an IoT environment. This thesis proposes a dynamic bandwidth packet scheduling scheme to improve the quality of service of IoT applications in terms of bandwidth, end-to-end delay, packet loss and jitter. The data from different sensors and objects is divided into three classes .i.e. high priority data, medium priority data and low priority data on the basis of characteristics like bit rate, packet loss rate and tolerable delay. There are three different queues for different class of data. i.e. high priority queue, medium priority queue and low priority queue. Different type of data enters its respective queue and is scheduled according to weighted fair queuing mechanism. The weights for scheduling are calculated dynamically on the basis of average queue length of every type queue and the average queue length is calculated by a method known as AEWMA. In this way data packets are scheduled by dynamically assigning them bandwidth in accordance to the needs of different types of data. This method makes use of weighted fair queuing technique for packet scheduling. This can be seen as both, an advantage or limitation of this scheme. The weighted fair queuing mechanism does its computation based on the number of bits and not on the basis of number of packets. Therefore, the state computations become very complex. This problem can be removed by weighted round robin method where scheduling is done on the basis of number of packets and hence is a simple technique. But WRR does not have true knowledge of packet size and therefore flows having larger packets are preferred to those with smaller packets leading to their starvation. But the

WFQ mechanism has knowledge of the actual packet size and therefore prevents the starvation of data streams having smaller packet size.

The AEWMA based dynamic bandwidth allocation approach results shows a better (QoS) performance in terms of end to end delay, delay jitter, and packet loss ratio. The AEWMA approach is better than the EWMA approach which was previously majorly used in research. AEWMA approach is used for various data sizes ranging from 100 bytes to 571 bytes with different kind of devices in the network.it is clearly seen that the approach is good for high, medium priority services and the loss ratio, delay is lesser and within the limit . The low priority services also shows a better result compared to other previously available approaches.

REFERENCES

[1] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., 2015. "Intetnet of Things:A Survey on Enabling Technologies, Protocols and Applications". IEEE communication surveys & tutorials.Vol.17-1553-877X.

[2] J. Morgan, J., 2016. "A Simple Explanation of 'The Internet Of Things". Available online at: https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-canunderstand/#679a507e1d09.

[3] IoT connections outlook, Ericsson Mobility Report November 2019.

[4] Banafa, A., 2016. "The Internet of Everything (IoE)," 2016. Available Online at: https://www.bbvaopenmind.com/en/the-internet-ofeverything-ioe/.

[5] Bandyopadhyay, D., and Sen, J., 2011. "Internet of things: Applications and challenges in technology and standardization," *Wirel. Pers. Commun.*, vol. 58, no. 1, pp. 49–69, 2011

[6] Townsend, J., 2016. "The Function of Quality Assurance (QA) with the Internet of Things," 2016. Available online at: https://www.ctl.io/blog/post/qa-with-theiot/.

[7] Gridelli, B. S., 2014. "How to calculate network availability?", 2014. Available online at: https://netbeez.net/blog/how-to-calculate-networkavailability/.

[8] Weber, R. H., 2010. "Internet of Things - New security and privacy challenges," *Comput. Law Secur. Rev.*, vol.26, no. 1, pp. 23–30, 2010.

[9] "OcularIP". Available online at: https://www.localbackhaul.com/ocularip/.

[10] Ducq, Y. and Chen, D., 2008. "How to measure interoperability: Concept and approach," *IEEE Int. Technol. Manag. Conf.*, pp. 1–8, 2008.

[11] Kalantar-zadeh, K., "Sensors," Springer. ISBN 978-1-4614-5052-8. pp. 11–29, 2013.

[12] "CLOUD IOT CORE". Available online at: https://cloud.google.com/iot-core/.

[13] Raja, A., and Su, X., 2009. "Mobility handling in MAC for wireless ad hoc networks," *Wirel. Commun. Mob. Comput.*, vol. 9, no. 3, pp. 303–311.

[14] Habib, S.M., Ries, S., and Mühlhäuser, M., 2010. "Cloud computing landscape and research challenges regarding trust and reputation". *Ubiquitous Intell. Comput. 7th Int. Conf. Auton. Trust. Comput. (UIC/ATC), 2010 7th Int. Conf. on. IEEE*, pp. 410–415, 2010.

[15] Chakraborty, S. (2018). NPTEL Courses: Computer networks and internet protocol, NPTEL: National Program On Technology Enhanced Learning. Retrieved at July 2018.

[16] Szigeti, T. (2013). End-to-End QoS Network Design: Quality of service for rich media and cloud networks. Cisco press books, 2nd edition. CCIE No. 9794.

[17] Kurose, J.F. and Ross, K.W., (2007). "Computer Networking: A Top-Down Approach Featuring the Internet". ISBN 81-7758-878-8.

[18] Klepec, B., & Kos, A. (2001). Performance of VoIP applications in a simple differentiated services network architecture. IEEE international conference Eurocon 2001 (Vol. 1, pp. 214–217), Bratislava,Slovakia.

[19] Wang, H., Shen, C., & Shin, K. G. (2004). Adaptive weighted packet scheduling for premium service. IEEE communications, ICC (Vol. 6, pp. 1846–1850).

[20] Liu, L., & Xu, Z. (2005). Joint packet scheduling and channel allocation for wireless communications. In IEEE conference on signals, systems and computers (pp. 489–493).

[21] Le, L. B., Hossain, E., & Alfa, A. S. (2006). Service differentiation in multirate wireless networks with weighted round robin scheduling and ARQ based error control. IEEE Transactions on Communications,54(2), 208–215

[22]Balogh, T., & Medvecký, M. (2010) "Comparison of Priority Queuing Based Scheduling Alogrithm." Electrorevue ISSN 1213-1539.

[23]Balogh, T., Luknarova, D., Medvecký, M., (2010) 'Performance of Round Robin-based Queue Scheduling Algorithms' 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service. 978-0-7695-4070-2/10 $26.00 © 2010 IEEE

[24]Qian, Y., Lu, Z and Dou, Q., (2010) 'QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin' 978-1-4244-8935-0/10/$26.00 ©2010 IEEE

[25]Geng, X., Luo, A., Sun, Z., & Cheng, Yu. (2012). Markov chains based dynamic bandwidth allocation in diffserv network. IEEE Communication Letters, 16(10), 1711–1714.

[26] Awan, I., & Younas, M., (2013). "Towards QoS in internet of things for delay sensitive information". MobiWIS 2013 workshops, CCIS 183 (pp. 86–94), Springer International Publishing Switzerland.

[27] Awan, I., Younas, M., & Naveed, W.,(2014). "Modelling QoS in IoT applications". IEEE 17th international conference on network-based information systems (NBiS)(2014) (pp. 99–105).

[28] Abdullah, S., & Yang, K. (2013). A QoS aware message scheduling algorithm in internet of things environment. In IEEE online conference on green communications (pp. 175–180).

[29] Li, L., Li, S., & Zhao, S. (2014). QoS-aware scheduling of services-oriented internet of things. IEEE Transactions on Industrial Informatics, 10(2), 1497–1505.

[30] Bhandari, S., Sharma, S.K., Wang, X., (2017) 'Latency minimization in wireless IoT using prioritized channel access and data aggregation.' 978-1-5090-5019-2/17/$31.00 2017 IEEE.

[31] Sharma, R., & Navin, K., (2015) QoS-alert Markov Chain based Scheduling Scheme in Internet of Things. 978-1-4673-9526-7/15/$31.00 ©2015 IEEE.

[32] Sharma, R., Kumar, N., & Srinivas, T. (2014). Performance of new dynamic benefit-weighted scheduling scheme in diffserv networks. In Proceedings of IEEE international conference on advances in computing, communications and informatics, (ICACCI) (pp. 2578–2583), Noida.

[33] Sharma, R., & Navin, K., Srinivas, T.,(2018) Markov Chain Based Priority Queueing Model for Packet Scheduling and Bandwidth Allocation. © ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2018. doi.org/10.1007/978-3-319-73423-1_9.

[34] Sharma, R., & Navin, K., Srinivas, T., Gowda, N.B., (2018) 'Packet Scheduling Scheme To Guarantee QoS in Internet of Things.' Springer Science + Business Media, LLC, part of Springer Nature 2018.https://doi.org/10.1007/s11277-017-5218-8.

[35] Tang, D., Kai, C., Chen, X., Liu, H., and Li, X., 2014. "Adaptive EWMA Method Based on Abnormal Network Traffic for LDoS Attacks", Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2014, Article ID 496376, 11 pages http://dx.doi.org/10.1155/2014/496376.

[36] Sugeng, W., Istiyanto, J. E., Mustofa, K., & Ashari, A. (2015). The impact of QoS changes towards network performance. International Journal of Computer Networks and Communication Security., 3(2), 48–53.

[37] Cao, Y., & Li, V. (2002). "Scheduling algorithms in broad-band wireless networks". Proceedings of the IEEE, 89(1), 76–87.

[38] Monares, A., Ochoa, S. F., Santos, R., Orozco, J., & Meseguer, R. (2014). Modeling IoT-based solutions using human-centric wireless sensor networks. Sensors, 1424–8220, 15687–15713.

[39] L. Mohan, M. G. Bijesh, and J. K. John, "Survey of low rate denial of service (LDoS) attack on RED and its counter strategies," in Proceedings of the IEEE International Conference on Computational Intelligence & Computing Research (ICCIC '12),pp.1–7,Coimbatore,India,2012.

[40] Goel, S., & Imielinski, T. (2001). Prediction based monitoring in sensor networks: Taking lessons from MPEG. ACM SIGCOMM Computer Communication Special Issue on Wireless Extensions to the Internet, 31(5), 82–98

[41] G. Capizzi and G. Masarotto, "An adaptive exponentially weighted moving average control chart," 2003. Technometrics, American Statistican Association, vol.45, no.3,pp.199–207,2003.