Project Report

On

QUALITY OF SERVICE MANAGER

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning)

by

Ms.YASHASWINY SRIPADA-22WH1A6607

Ms.R.ISHWARYA – 22WH1A6609

Ms.S.AISHWARYA– 22WH1A6644

Ms.N.VAISHNAVI– 22WH1A6645

Under the esteemed guidance of

Ms. P Anusha

Assistant Professor, CSE(AI&ML)

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

2024-25

ABSTRACT

A Quality of Service (QoS) Manager in computer networks is a system designed to ensure efficient and reliable data transmission by prioritizing network traffic based on predefined policies. It allocates resources such as bandwidth and reduces latency, jitter, and packet loss for critical applications like video streaming, VoIP, and online gaming. The QoS Manager analyzes traffic patterns, enforces service-level agreements (SLAs), and adapts dynamically to changing network conditions. By optimizing performance and ensuring fairness, it enhances user experience while maintaining network stability. This technology is crucial in modern networks, especially with the increasing demand for high-quality, real-time digital services.

PROBLEM STATEMENT

As computer networks handle increasing volumes of traffic and diverse application demands, ensuring consistent performance becomes a significant challenge. Applications like video streaming, real-time gaming, and VoIP require specific levels of bandwidth, low latency, and minimal jitter to function effectively. However, the absence of an efficient Quality of Service (QoS) management system leads to several problems:

- **Traffic Congestion**: High network traffic results in delays, packet loss, and degraded user experiences.
- **Resource Mismanagement**: Inefficient allocation of bandwidth and other resources causes critical applications to underperform.
- **Inconsistent Service Levels:** Lack of prioritization among applications disrupts the performance of latency-sensitive and mission-critical services.
- **Inability to Meet SLAs:** Failure to monitor and enforce service-level agreements impacts customer satisfaction and trust.

This project addresses these issues by designing a QoS Manager for computer networks. It will focus on traffic prioritization, resource optimization, and performance monitoring to ensure that applications receive the required service levels while maintaining overall network efficiency. The solution aims to improve network reliability, enhance user experiences, and meet the growing demands of modern networked applications..

Functional Requirements

1. **Traffic Classification:**
    - The system should classify network traffic based on application type, priority, and QoS policies.
2. **Traffic Prioritization:**
    - The system must prioritize critical traffic, such as VoIP and video streaming, over non-critical traffic like file downloads.
3. **Resource Allocation:**
    - Dynamically allocate bandwidth and other network resources to meet application-specific QoS needs.
4. **Congestion Management:**
    - Implement traffic shaping and queue management techniques to prevent network congestion.
5. **Policy Enforcement:**
    - Apply and enforce QoS policies as defined by the administrator or network service provider.
6. **Performance Monitoring:**
    - Continuously monitor metrics like latency, jitter, packet loss, and bandwidth utilization.
7. **Alerting and Reporting**:
    - Generate real-time alerts for performance degradation and periodic reports on QoS performance.

Non-Functional Requirements:

1. **Reliability:**
   - The QoS Manager must operate with minimal downtime and ensure consistent service delivery.
2. **Performance:**
   - The system should introduce negligible latency while processing traffic to ensure smooth operation.
3. **Scalability:**
   - Capable of handling increasing traffic loads and scaling across distributed network environments.
4. **Usability:**
   - Provide a user-friendly interface for administrators to configure policies and monitor performance.
5. **Interoperability:**
   - The system must integrate seamlessly with existing network infrastructure and protocols, such as MPLS, DiffServ, and IntServ.
6. **Security:**
   - Ensure secure data handling and prevent unauthorized access to QoS settings and policies.
7. **Adaptability:**
   - The system should adapt to dynamic network conditions, automatically adjusting resource allocation and policies.

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Function to execute a system command and capture its output
void execute_command(const char *command) {
    FILE *fp;
    char buffer[128];

    // Open the command for reading
    if ((fp = popen(command, "r")) == NULL) {
        perror("Error executing command");
        exit(EXIT_FAILURE);
    }

    // Read and print the command output line by line
    printf("Command Output:\n");
    while (fgets(buffer, sizeof(buffer), fp) != NULL) {
        printf("%s", buffer);
    }

    // Close the file pointer
```

```
    pclose(fp);
}

int main() {
    const char *host = "8.8.8.8"; // Target host (Google DNS)
    int ping_count = 4;         // Number of ping attempts
    char command[128];

    printf("=== Simple Quality Network Service ===\n");

    // Construct the ping command
    sprintf(command, "ping -c %d %s", ping_count, host);

    // Execute the ping command and display the results
    execute_command(command);

    printf("\n=== Measurement Complete ===\n");

    return 0;
}
```

Compilation and Execution

Compile :

gcc simple_qns.c -o simple_qns

Run the executable code:

./simple_qns

Output:

```
=== Simple Quality Network Service ===

Command Output:

PING 8.8.8.8 (8.8.8.8): 56 data bytes

64 bytes from 8.8.8.8: icmp_seq=0 ttl=118 time=20.5 ms

64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=20.4 ms

64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=20.3 ms

64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=20.6 ms


--- 8.8.8.8 ping statistics ---

4 packets transmitted, 4 packets received, 0% packet loss

round-trip min/avg/max/stddev = 20.3/20.5/20.6/0.1 ms


=== Measurement Complete ===
```