

OPERATINGSYSTEM

**Project Name Network Protocol Analyzer**

**Team Members :**

|  |  |
| --- | --- |
| 22wh1a6603 | D. Megana Laxmi |
| 22wh1a6613 | I.Sivani |
| 22wh1a6623 | S.Deepika Preharshini |
| 22wh1a6633 | P.Preethi |
| 22wh1a6643 | N.Prasanna |
| 22wh1a6653 | G.Sanjana |
| 23wh5a6663 | N.Manasvi |

**Problem Statement :**

The problem statement for virtual memory management is to expand the network protocol analyzer project by incorporating support for analyzing multiple network protocols, such as TCP/IP, UDP, and HTTP. Discuss the importance of protocol analyzers in diagnosing network issues and ensuring the smooth flow of data in diverse networking environments..

**ABSTRACT :**

**Introduction:** In contemporary networking landscapes, the ability to comprehend and rectify network issues is indispensable for ensuring the seamless transmission of data and safeguarding the reliability and security of network communications.

**Project Objective:** This project endeavors to augment a network protocol analyzer to encompass support for multiple protocols, including TCP/IP, UDP, and HTTP.

**Importance of protocol Analyzer:** Protocol analyzers serve as pivotal tools in scrutinizing network traffic, pinpointing irregularities, and troubleshooting connectivity challenges.

**Enhanced utility:** By embracing support for diverse protocols, the analyzer amplifies its efficacy across a spectrum of networking scenarios, empowering network administrators and security practitioners to delve deeper into network dynamics and efficiently mitigate performance bottlenecks, security vulnerabilities, and other network-related impediments.

**Conclusion:** This paper underscores the criticality of protocol analyzers in diagnosing network malfunctions and underscores the imperative of accommodating multiple protocols to cater to the evolving requisites of contemporary networking environments.

Top of Form

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <pcap.h>

void packet\_handler(unsigned char userData, const struct pcap\_pkthdr pkthdr, const unsigned char\* packet) {

printf("Packet captured\n");

}

int main(int argc, char \*argv[]) {

char \*dev, errbuf[PCAP\_ERRBUF\_SIZE];

pcap\_t \*handle;

// Get a device to sniff on

dev = pcap\_lookupdev(errbuf);

if (dev == NULL) {

fprintf(stderr, "Couldn't find default device: %s\n", errbuf);

return 1;

}

// Open the session in promiscuous mode

handle = pcap\_open\_live(dev, BUFSIZ, 1, 1000, errbuf);

if (handle == NULL) {

fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);

return 1;

}

// Start capturing packets

pcap\_loop(handle, 0, packet\_handler, NULL);

// Close the session

pcap\_close(handle);

return 0;

}

**OUTPUT :**

Packet captured

Packet captured

Packet captured

Packet captured

...