CS23532-COMPUTER NETWORKS-LAB MANUAL

Practical 13

AIM: - Implement your own ping program

Algorithm:

```
import os
import sys
import socket
import struct
import time
import select

# Calculate checksum for ICMP packet
def checksum(source_string):
    sum = 0
    countTo = (len(source_string) // 2) * 2
    count = 0

    while count < countTo:
        thisVal = source_string[count + 1] * 256 + source_string[count]
        sum = sum + thisVal
        sum = sum & 0xffffffff
        count = count + 2

    if countTo < len(source_string):
        sum = sum + source_string[len(source_string) - 1]
        sum = sum & 0xffffffff

    sum = (sum >> 16) + (sum & 0xffff)
    sum = sum + (sum >> 16)
    answer = ~sum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
    return answer

# Create and send ICMP echo request
def send_one_ping(sock, dest_addr, ID):
    header = struct.pack('bbHHh', 8, 0, 0, ID, 1)
    data = struct.pack('d', time.time())
    chksum = checksum(header + data)

    header = struct.pack('bbHHh', 8, 0, socket.htons(chksum), ID, 1)
    packet = header + data
    sock.sendto(packet, (dest_addr, 1))

# Receive the ICMP echo reply
def receive_one_ping(sock, ID, timeout):
    time_left = timeout
    while True:
        start_time = time.time()
        readable = select.select([sock], [], [], time_left)
```

Practical 13

```
        time_spent = (time.time() - start_time)
        if readable[0] == []:  # Timeout
            return None

        time_received = time.time()
        rec_packet, addr = sock.recvfrom(1024)
        icmp_header = rec_packet[20:28]
        type, code, checksum, packet_id, sequence = struct.unpack('bbHHh', icmp_header)
        if packet_id == ID:
            bytes_in_double = struct.calcsize('d')
            time_sent = struct.unpack('d', rec_packet[28:28 + bytes_in_double])[0]
            return time_received - time_sent

        time_left = time_left - time_spent
        if time_left <= 0:
            return None

# Perform ping
def do_one_ping(dest_addr, timeout=1):
    icmp = socket.getprotobyname('icmp')
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
    except PermissionError:
        print("⚠️ Run this program as Administrator or Root.")
        sys.exit()

    my_id = os.getpid() & 0xFFFF
    send_one_ping(sock, dest_addr, my_id)
    delay = receive_one_ping(sock, my_id, timeout)
    sock.close()
    return delay

# Main Ping loop
def ping(host, count=4):
    dest = socket.gethostbyname(host)
    print(f"\nPinging {host} [{dest}] with custom ICMP packets:\n")

    for i in range(count):
        delay = do_one_ping(dest)
        if delay is None:
            print("Request timed out.")
        else:
            print(f"Reply from {dest}: time={(delay * 1000):.2f} ms")
        time.sleep(1)

    print("\nPing test completed.\n")
```

Practical 13

```
# Run program
if __name__ == '__main__':
    target = input("Enter host to ping: ")
    ping(target)
```

Output:

```
Enter host to ping: google.com

Pinging google.com [142.250.183.206] with custom ICMP packets:

Reply from 142.250.183.206: time=18.74 ms
Reply from 142.250.183.206: time=17.92 ms
Reply from 142.250.183.206: time=19.05 ms
Reply from 142.250.183.206: time=17.83 ms

Ping test completed.
```

Result

A **custom Ping program** was successfully implemented using **Python sockets**. It sends ICMP packets and measures round-trip time, replicating the basic functionality of the system `ping` command.