

Assignment 7: Application of Raw Sockets

Ping Application

Suyash Damle (15CS10057)
Arunansh Kaushik (15CS30004)

Objective:

The objective of this assignment is to use Raw Sockets to create a Ping Application.

To compile and run Ping:

```
g++ ping.cpp -o ping.out
```

```
sudo ./ping.out <address to ping>
```

Functions and their utility:

- **checksum()**: The function simply calculates the checksum value for the ICMP header and returns the value as short int.
- **display()**: The function displays the entire packet received – the IP and the ICMP headers – both.
Note : The global variable **show_packet** could be set to false to turn off this functionality
- **sig_handler()**: Gets activated at the arrival of SIGINT (- ^C) command- displays the statistics of the Round Trip Time (RTT) – average, min and max; and then exits the program
- **listener()**: does the simple job of receiving the packet, checking whether it is in response to the right ping and displays the basic info – the RTT value, and the sequence number.
- **ping_and_receive()**: The most important part of the whole application. Does the following jobs:
 - Creates and attaches the IP header: uses **memcpy()** call to add the IP header fields to the char array that forms the top part of the packet structure.
 - Creates and attaches the ICMP header: uses **memcpy()** as before to add the fields one-by-one to the char array that forms the part of the ICMP header in the packet
 - Adds Time stamp : Maintains 2 copies of the timestamp – one in the data part of the ICMP packet and another in an independent array **time_sent[]**. This part has only been done as an experiment to check whether the message part gets returned as-it-is. In all trial runs, it did get returned as-is.
 - Sends the packet to the designated address destination address over a *connectionless* socket.

Observations / Other specifics of Implementation :

- The application has to run on connectionless socket – hence binding and another connection – establishment activities are never used.
- The **setsockopt()** call has been used to set HDRINCL option to indicate to the kernel that **no IP header is to be added by the kernel**. The header is manually created and added to the packet.
- Source address and IP checksum have been left blank to indicate that the kernel may add these values in the packet by itself.
- A **sleep()** has been included after sending each packet to emulate the real linux ping command.
- The ICMP packets generated and those received were captured by **Wireshark** to check them. The RTT values generated by the application and those recorded by Wireshark were found to be in close agreement. A difference of a few microseconds, does exist and could be attributed to the fact that some instructions before

recording the time may have taken some time and that the process – running on multiprocessing environment may not immediately respond.

- The pings may not work with sites out of the AS as the proxy server only relays the HTTP and HTTPS messages - and hence, pinging was only done to *iitkgp.ac.in* which responds to the pings – as the server is within the AS.

Screenshot:

```
x - ■ suyash@suyash-ubuntu: /media/suyash/New Volume/IIT KGP/Networks/networks_lab
suyash@suyash-ubuntu: /media/suyash/New Volume/IIT KGP/Networks/networks_lab$ g++ ping.cpp
suyash@suyash-ubuntu: /media/suyash/New Volume/IIT KGP/Networks/networks_lab$ sudo ./a.out iitkgp.ac.in

64 Bytes of data received from 10.3.100.102      seq no: 1      RTT= 2.113 ms
-----
0: 45 00 00 40 df f5 00 00 3e 01 38 cd 0a 03 64 66
16: 0a 91 eb 00 00 00 0c 20 13 55 00 01 e4 d4 08 00
32: 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
48: 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 00
IPv4: hdr-size=20 pkt-size=64 protocol=1 TTL=62 src=10.3.100.102 dst=10.145.235.0
-----
ICMP: type[0/0] checksum[3104] id[4949] seq[1]
64 Bytes of data received from 10.3.100.102      seq no: 2      RTT= 1.863 ms
-----
0: 45 00 00 40 df f6 00 00 3e 01 38 cc 0a 03 64 66
16: 0a 91 eb 00 00 00 bd 15 13 55 00 02 33 de 08 00
32: 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
48: 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 00
IPv4: hdr-size=20 pkt-size=64 protocol=1 TTL=62 src=10.3.100.102 dst=10.145.235.0
-----
ICMP: type[0/0] checksum[48405] id[4949] seq[2]
64 Bytes of data received from 10.3.100.102      seq no: 3      RTT= 4.287 ms
-----
0: 45 00 00 40 df f7 00 00 3e 01 38 cb 0a 03 64 66
16: 0a 91 eb 00 00 00 73 0c 13 55 00 03 7d e6 08 00
32: 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
48: 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 00
IPv4: hdr-size=20 pkt-size=64 protocol=1 TTL=62 src=10.3.100.102 dst=10.145.235.0
-----
ICMP: type[0/0] checksum[29452] id[4949] seq[3]
64 Bytes of data received from 10.3.100.102      seq no: 4      RTT= 16.739 ms
-----
0: 45 00 00 40 df f8 00 00 3e 01 38 ca 0a 03 64 66
16: 0a 91 eb 00 00 00 98 f9 13 55 00 04 57 f8 08 00
32: 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
48: 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 00
IPv4: hdr-size=20 pkt-size=64 protocol=1 TTL=62 src=10.3.100.102 dst=10.145.235.0
-----
ICMP: type[0/0] checksum[39161] id[4949] seq[4]
^C
----- STATS -----
Average RTT: 6.2505 ms
Max RTT: 16.739 ms
Min RTT: 1.863 ms
suyash@suyash-ubuntu: /media/suyash/New Volume/IIT KGP/Networks/networks_lab$
```