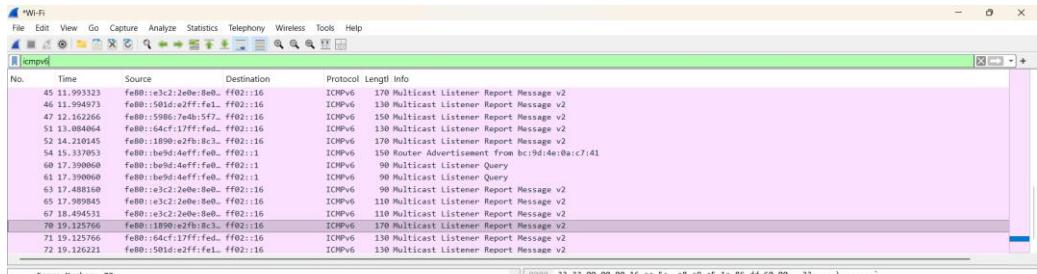


```

Frame Number: 70
Frame Length: 170 bytes (1360 bits)
Capture Length: 170 bytes (1360 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip6:hopopts:icmpv6]
Character encoding: ASCII (0)
[Coloring Rule Name: ICMPII]
[Coloring Rule String: icmp || icmpv6]
Ethernet II, Src: ea5c:e8:a9:a5:1e (ea5c:e8:a9:a5:1e), Dst: IPv6mcast_16 (33:33:00:00:00:16)
> Destination: IPv6mcast_16 (33:33:00:00:00:16)
> Source: ea5c:e8:a9:a5:1e (ea5c:e8:a9:a5:1e)
Type: IPv6 (86dd)
[Stream index: 9]
> Internet Protocol Version 6, Src: fe80::1890:e2fb:8c30:770, Dst: ff02::16
> Internet Control Message Protocol v6: Protocol
    Code: 0
    Checksum: 0x0bf8 [correct]
    [Checksum Status: Good]
    Reserved: 0000

```

Packets: 72 · Displayed: 34 (47.2%) · Dropped: 0 (0.0%) · Profile: Default

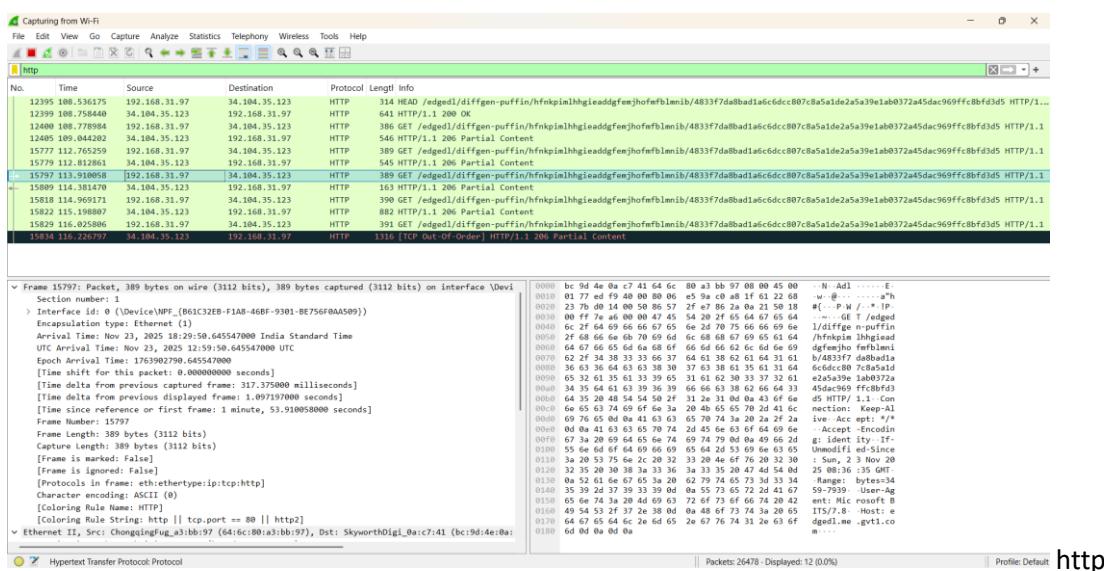
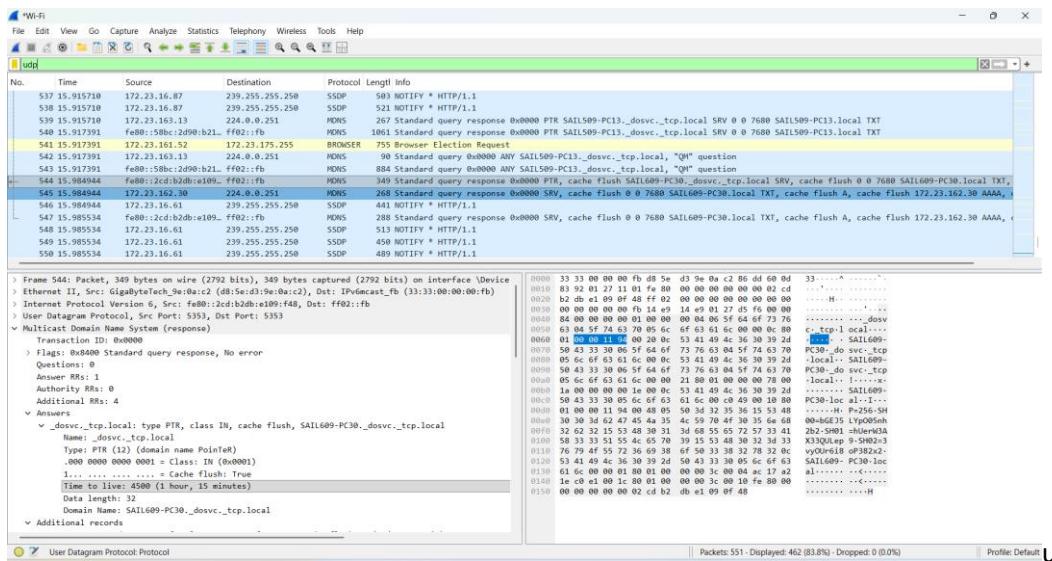
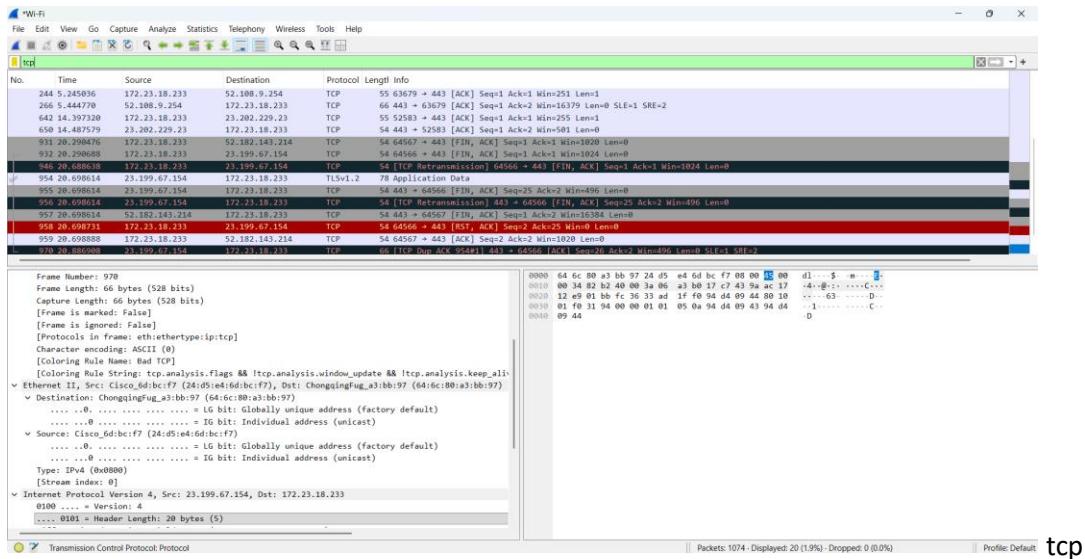


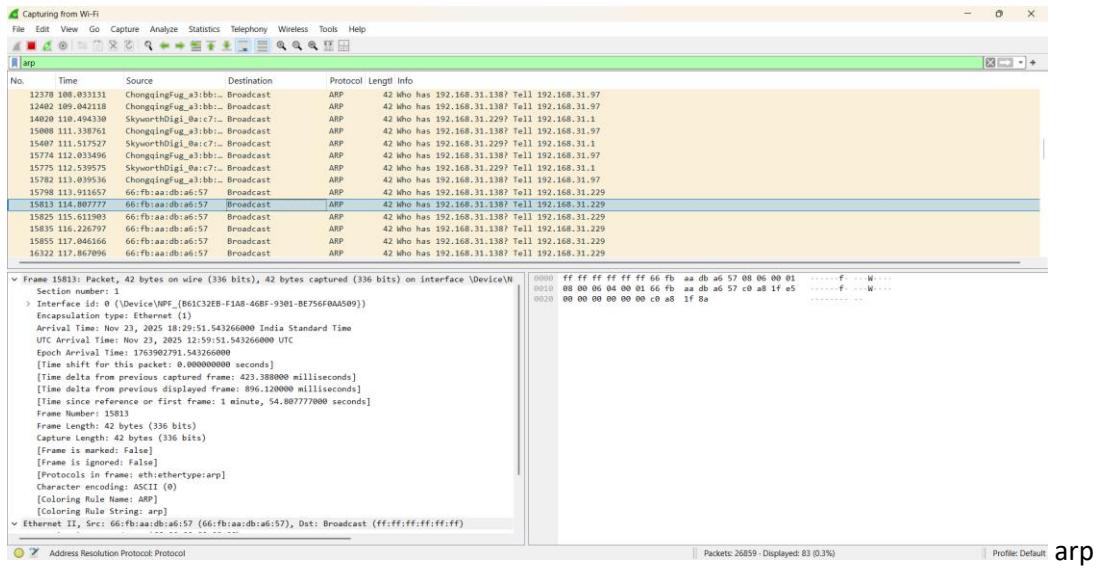
```

Frame Number: 70
Frame Length: 170 bytes (1360 bits)
Capture Length: 170 bytes (1360 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip6:hopopts:icmpv6]
Character encoding: ASCII (0)
[Coloring Rule Name: ICMPII]
[Coloring Rule String: icmp || icmpv6]
Ethernet II, Src: ea5c:e8:a9:a5:1e (ea5c:e8:a9:a5:1e), Dst: IPv6mcast_16 (33:33:00:00:00:16)
> Destination: IPv6mcast_16 (33:33:00:00:00:16)
> Source: ea5c:e8:a9:a5:1e (ea5c:e8:a9:a5:1e)
Type: IPv6 (86dd)
[Stream index: 9]
> Internet Protocol Version 6, Src: fe80::1890:e2fb:8c30:770, Dst: ff02::16
> Internet Control Message Protocol v6: Protocol
    Code: 0
    Checksum: 0x0bf8 [correct]
    [Checksum Status: Good]
    Reserved: 0000

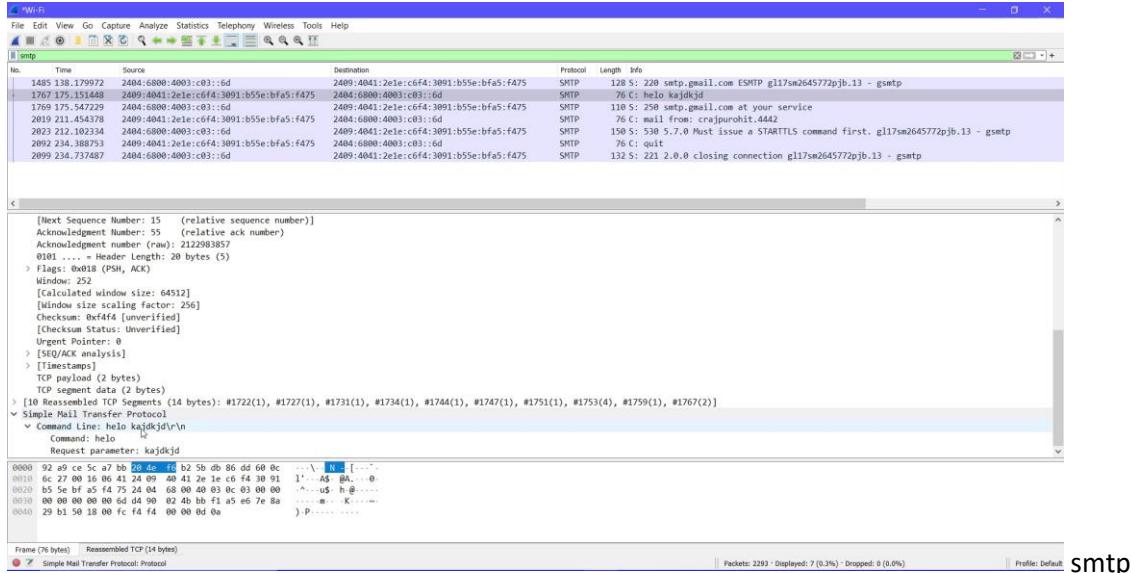
```

Packets: 72 · Displayed: 34 (47.2%) · Dropped: 0 (0.0%) · Profile: Default





## icmp



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar containing icons for various file types: Python (py), C/C++ (c/cpp), Java (java), JavaScript (js), and TypeScript (ts). The main area displays a C program named 'main.c'. The code implements bit stuffing and de-stuffing. It prompts the user for the number of bits (n), the raw data bits, and the length of the stuffed data. It then performs bit stuffing on the raw data and prints the result. Finally, it performs bit de-stuffing on the stuffed data and prints the original data. The output window shows the execution results.

```
main.c

72     printf("Enter number of bits: ");
73     scanf("%d", &n);
74
75     printf("Enter the bits (0/1):\n");
76     for (i = 0; i < n; i++) {
77         scanf("%d", &data[i]);
78     }
79
80     bitStuffing(data, n);
81
82     printf("Enter stuffed data length: ");
83     int stuffed_len;
84     scanf("%d", &stuffed_len);
85     int stuffed_data[100];
86     printf("Enter the stuffed bits (0/1):\n");
87     for (i = 0; i < stuffed_len; i++) {
88         scanf("%d", &stuffed_data[i]);
89     }
90
91     bitDeStuffing(stuffed_data, stuffed_len);
92
93     return 0;
94 }
95 }
```

Output

```
- Enter number of bits: 8
Enter the bits (0/1):
0 1 1 1 1 1 1 0
Data after Bit Stuffing: 011111010
Enter stuffed data length: 9
Enter the stuffed bits (0/1):
0 1 1 1 1 1 0 1 0
Data after Bit De-stuffing: 0111110
*** Code Execution Successful ***
```