CS 39006: Computer Networks Laboratory

Raw Sockets in POSIX Network programming

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Abhijnan Chakraborty
abhijnan@cse.iitkgp.ac.in

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

What Are Raw Sockets?

- A raw socket allows direct sending and receiving of IP packets without transport-layer encapsulation (e.g., TCP or UDP).

- Used when:
  - You want to implement custom protocols.
  - You need fine-grained control over packet structure.
  - You're working on network monitoring or security tools.

Key Characteristics

- Bypass the OS transport layer
- Requires root privileges
- Can be used to build or inspect packets manually
- Access to IP headers and payloads

Raw Socket Use Cases

- Writing custom protocols (e.g., your own discovery or telemetry protocol)
- Packet sniffing
- Penetration testing tools
- Network testing and simulation

POSIX Raw Socket API

```
int sock = socket(AF_INET, SOCK_RAW, protocol);
```

- `AF_INET`: IPv4
- `SOCK_RAW`: Raw socket type
- `protocol`: e.g., `IPPROTO_TCP`, `IPPROTO_RAW`, or a custom protocol number

POSIX Raw Socket API

```
int sock = socket(AF_INET, SOCK_RAW, protocol);
```

- `AF_INET`: IPv4
- `SOCK_RAW`: Raw socket type
- `protocol`: e.g., `IPPROTO_TCP`, `IPPROTO_RAW`, or a custom protocol number

- Optionally, you can pass the option to set the IP header manually

```
int opt = 1;
setsockopt(sock, IPPROTO_IP, IP_HDRINCL, &opt, sizeof(opt));
```

Simple Example (Send IP Packet)

```
int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

char packet[1024];
struct iphdr *ip = (struct iphdr *)packet;
strcpy(packet + sizeof(struct iphdr), "Hello!");

ip->version = 4;
ip->ihl = 5;
ip->ttl = 64;
ip->protocol = 253;  // Custom protocol
ip->saddr = inet_addr("192.168.1.10");
ip->daddr = inet_addr("192.168.1.20");
```

Simple Example (Send IP Packet)

```
struct sockaddr_in dst;
dst.sin_family = AF_INET;
dst.sin_addr.s_addr = ip->daddr;

sendto(sock, packet, sizeof(struct iphdr) + strlen("Hello!"), 0,
       (struct sockaddr *)&dst, sizeof(dst));
```

Receiving with Raw Sockets

```
int sock = socket(AF_INET, SOCK_RAW, 253);  // Custom protocol

char buffer[2048];
struct sockaddr_in src;
socklen_t len = sizeof(src);

int bytes = recvfrom(sock, buffer, sizeof(buffer), 0,
                     (struct sockaddr *)&src, &len);

struct iphdr *ip = (struct iphdr *)buffer;
char *payload = buffer + ip->ihl * 4;

printf("Received: %s\n", payload);
```

Receiving with Raw Sockets

```
int sock = socket(AF_INET, SOCK_RAW, 253);  // Custom protocol

char buffer[2048];
struct sockaddr_in src;
socklen_t len = sizeof(src);

int bytes = recvfrom(sock, buffer, sizeof(buffer), 0,
                     (struct sockaddr *)&src, &len);

struct iphdr *ip = (struct iphdr *)buffer;
char *payload = buffer + ip->ihl * 4;

printf("Received: %s\n", payload);
```

Why multiply by 4?

Some Useful Tips

- Understand the IP header structure (RFC 791)
- Always validate buffer sizes
- Use virtual machines or isolated networks for testing