

Laboratorio di Reti e Sistemi Distribuiti

8: Socket Raw di Livello 3

Roberto Marino, PhD¹

`roberto.marino@unime.it`

¹Dipartimento di Matematica, Informatica, Fisica e Scienze della Terra
Future Computing Research Laboratory
Università di Messina

Last Update: 13th March 2025

Definizione

I socket raw in C consentono di operare **a un livello più basso della pila di rete**, permettendo la creazione e manipolazione diretta dei pacchetti, inclusi gli header di protocollo (IP, TCP, UDP, ICMP, ecc.). Sono utilizzati per sviluppare strumenti di rete avanzati (sniffer, scanner, tool di sicurezza) o protocolli personalizzati e per fare **Packet Crafting**.

Creazione di un socket raw

Per creare un socket raw in C, si utilizza la funzione `socket()` con parametri specifici:

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

- `AF_INET/AF_INET6`: Lavora a livello di rete (IP) per IPv4/IPv6.
- `AF_PACKET` (Linux) o `AF_LINK` (BSD): Accesso al livello di collegamento (Ethernet).
- Tipo: `SOCK_RAW` indica un socket raw.
- Protocollo: Specifica il protocollo da gestire (es. `IPPROTO_TCP`, `IPPROTO_RAW`). Con `IPPROTO_RAW`, il kernel non aggiunge automaticamente l'header IP.

Analisi di ICMP_raw.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <sys/socket.h>
6 #include <netinet/in.h>
7 #include <netinet/ip.h>
8 #include <netinet/ip_icmp.h>
9 #include <arpa/inet.h>
10 #include <sys/time.h>
```

- Input/output (stdio.h).
- Gestione della memoria e funzioni di sistema (stdlib.h, unistd.h).
- Socket e protocolli di rete (sys/socket.h, netinet/*).
- Conversione di indirizzi IP (arpa/inet.h).
- Gestione del timeout (sys/time.h).

Analisi di ICMP_raw.c

```
1 unsigned short compute_checksum(void *addr, int len) {
2     int sum = 0;
3     unsigned short *ptr = addr;
4     while (len > 1) {
5         sum += *ptr++;
6         len -= 2;
7     }
8     if (len == 1)
9         sum += *(unsigned char *)ptr;
10    sum = (sum >> 16) + (sum & 0xFFFF);
11    return (unsigned short)(~sum);
12 }
```

- Somma tutti i word (16 bit) del pacchetto.
- Aggiunge eventuali byte rimanenti (se la lunghezza è dispari).
- Esegue il folding del risultato a 16 bit.
- Restituisce il complemento a uno (necessario per il checksum ICMP).

Analisi di ICMP_raw.c

```
1 void build_icmp_packet(struct icmphdr *icmp, int sequence) {
2     icmp->type = ICMP_ECHO; // Tipo: Echo Request
3     icmp->code = 0; // Codice: 0 (standard per Echo Request)
4     icmp->un.echo.id = htons(getpid()); // ID basato sul PID del
    processo
5     icmp->un.echo.sequence = htons(sequence); // Numero di sequenza
6     icmp->checksum = 0; // Azzera il checksum prima del calcolo
7     // Payload di esempio (riempito con 'A')
8     char *data = (char *)(icmp + 1);
9     memset(data, 'A', PACKET_SIZE - sizeof(*icmp));
10    icmp->checksum = compute_checksum(icmp, PACKET_SIZE); // Calcolo
    finale
11 }
```

- type = ICMP_ECHO (8): Indica una richiesta di ping.
- id: Usa il PID del processo per identificare la richiesta.
- sequence: Numero incrementale per tracciare i pacchetti.
- checksum: Deve essere calcolato dopo aver riempito tutti i campi.

Analisi di ICMP_raw.c

```
1 struct timeval tv;  
2 tv.tv_sec = TIMEOUT_SEC;  
3 tv.tv_usec = 0;  
4 setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO, &tv, sizeof(tv));
```

- SO_RCVTIMEO imposta un timeout per recvfrom(), evitando che il programma si blocchi all'infinito.

Analisi di ICMP_raw.c

```
1 struct sockaddr_in dest;  
2 memset(&dest, 0, sizeof(dest));  
3 dest.sin_family = AF_INET;  
4 dest.sin_addr.s_addr = inet_addr(argv[1]); // Converte l'IP in formato  
    binario
```

- Input/output (stdio.h).
- Gestione della memoria e funzioni di sistema (stdlib.h, unistd.h).
- Socket e protocolli di rete (sys/socket.h, netinet/*).
- Conversione di indirizzi IP (arpa/inet.h).
- Gestione del timeout (sys/time.h).

Struttura pacchetto ICMP

```
1 struct icmphdr {
2     uint8_t  type;          // Tipo del messaggio (es: 8 = Echo Request, 0
                             // = Echo Reply)
3     uint8_t  code;          // Codice (specifica ulteriormente il tipo,
                             // spesso 0)
4     uint16_t checksum;      // Checksum
5     union {
6         // Parte specifica per Echo Request/Reply:
7         struct {
8             uint16_t id;      // Identificatore (es: PID)
9             uint16_t sequence; // Numero di sequenza
10        } echo;
11        // Altri campi per altri tipi ICMP (es: Timestamp, errori)
12    } un;
13 };
```

Guardate dentro `/usr/include/netinet/ !!`

Little-Endian



