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



Socket Raw

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:

```
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.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <arpa/inet.h>
#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).



```
unsigned short compute_checksum(void *addr, int len) {
      int sum = 0;
      unsigned short *ptr = addr;
      while (len > 1) {
          sum += *ptr++;
          len -= 2:
      if (len == 1)
          sum += *(unsigned char *)ptr;
      sum = (sum >> 16) + (sum & OxFFFF);
10
      return (unsigned short)(~sum);
11
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).

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

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

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

```
struct sockaddr_in dest;
memset(&dest, 0, sizeof(dest));
dest.sin_family = AF_INET;
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).

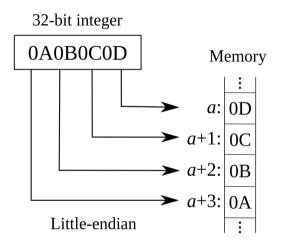
Struttura pacchetto ICMP

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

Guardate dentro /usr/include/netinet/!!



Little-Endian



Big-Endian

