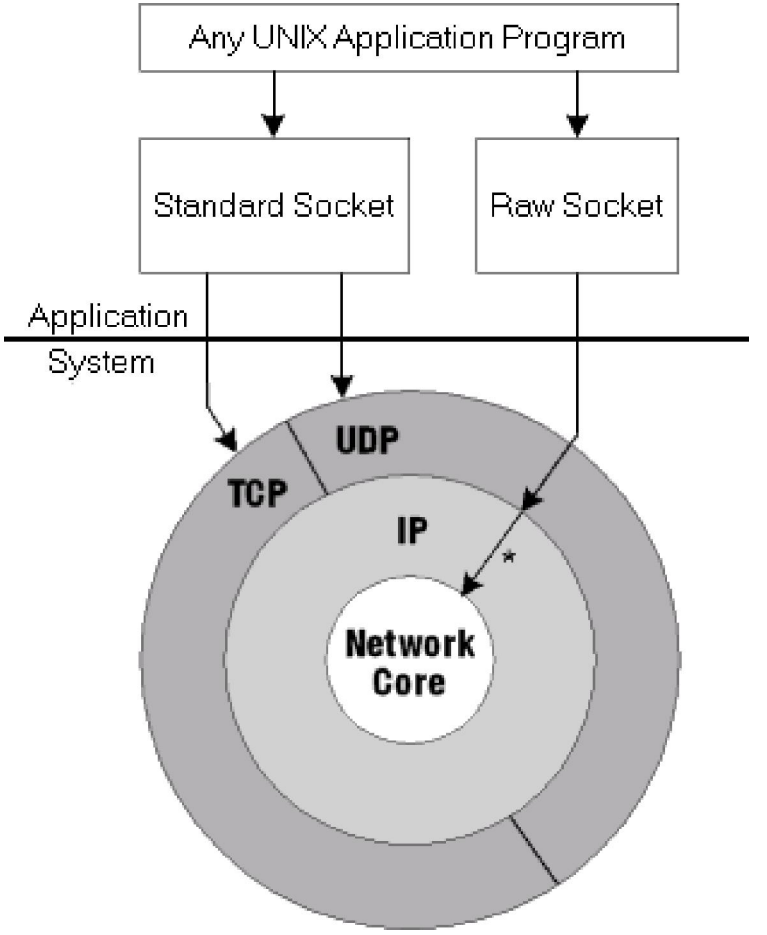
Raw sockets
- Socket thô

Có thể …

- Chúng ta muốn sử dụng chức năng của các giao thức ICMP, IGMP
  - Nhưng chúng ta không thể xử lý các gói tin ICMPv4, IGMPv4 và ICMPv6 với TCP/UDP sockets
- Chúng ta muốn viết chương trình cho bộ định tuyến
  - Nhưng chúng ta xử lý gói tin OSPF như thế nào?
- Chúng ta muốn tấn công vào máy tính nào đó với các gói tin giả mạo
  - Chúng ta tạo các gói tin đó như thế nào?

Câu trả lời là

- “Sử dụng socket thô”



Chúng ta có thể làm gì với socket thô?

- Cho phép tiến trình nhận và gửi các gói tin **ICMPv4, IGMPv4, và ICMPv6**
  - E.g. chương trình ping, traceroute
- Cho phép tiến trình nhận và gửi các gói tin IPv4 **không được xử lý bởi hệ điều hành**
  - Hầu hết các hệ điều hành chỉ xử lý các gói tin chứa trường giao thức là 1 (ICMP), 2 (IGMP), 6 (TCP), and 17 (UDP)
  - Giao thức định tuyến OSPF có trường giao thức là 89
- Cho phép một tiến trình tự xây dựng **IPv4 header**
  - sử dụng tùy biến IP_HDRINCL

Hạn chế của socket thô

- Không có cơ chế đảm bảo tính tin cậy truyền tin
- Không có số hiệu cổng
- Truyền tin không theo chuẩn
- Không có gửi gói tin ICMP tự động
- Không có TCP hoặc UDP thô
- Để tạo socket thô, cần có quyền root hoặc (hoặc administrator)

Tạo socket thô

- Chỉ có superuser mới có thể tạo socket thô

  #include <netinet/in.h>

  sockfd = socket(AF_INET, SOCK_RAW, *protocol*);

  - gán SOCK_RAW cho tham số thứ hai khi khởi tạo socket
  - tham số *protocol* là một trong các hằng số được định nghĩa bằng IPPROTO_xxx
    - E.g. IPPROTO_ICMP, IPPROTO_RAW

Thiết lập tùy biến IP_HDRINCL

- Để tự tạo **IPv4 header**
- const int on = 1;

  if (setsockopt(sockfd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0)

  error

Gửi dữ liệu với socket thô

- Gọi hàm *sendto* hoặc *sendmsg,* và thiết lập địa chỉ IP đích
  - Nếu tùy biến IP_HDRINCL không được thiết lập, địa chỉ đầu tiên của dữ liệu gửi đi tương ứng với byte đầu tiên ngay sau IP header
  - Nếu tùy biến IP_HDRINCL được thiết lập, địa chỉ đầu tiên của dữ liệu gửi đi tương ứng với byte đầu tiên của IP header
- Hệ điều hành sẽ phân mảnh gói tin nếu kích thước gói tin vượt quá MTU của giao diện mạng

Nhận dữ liệu với Socket thô

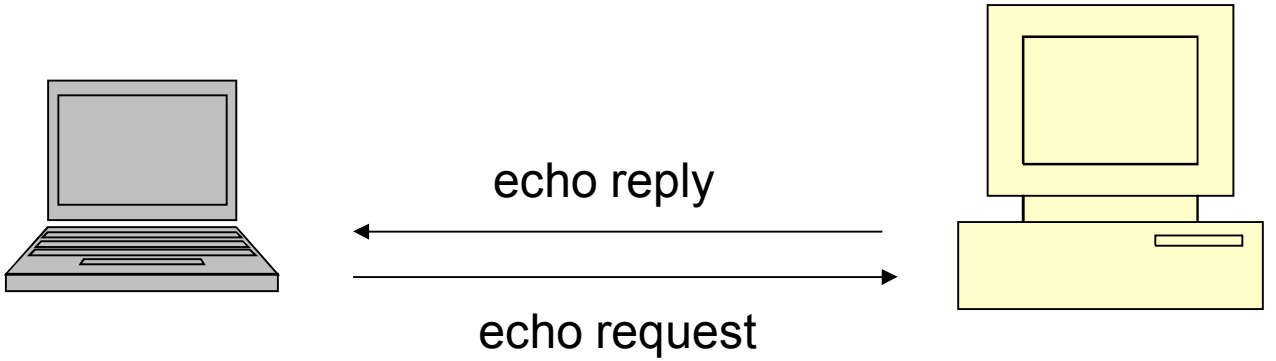
- **Thường dùng với hàm `recvfrom()`**
- Các gói tin UDP và TCP không bao giờ được gửi vào Socket thô
- Hầu hết các gói tin ICMP/IGMP được đưa vào Socket thô sau khi hệ điều hành xử lý xong gói tin ICMP
- Các gói tin IP với trường giao thức không được xử lý bởi hệ điều hành có thể được đưa vào Socket thô
- Nếu gói tin bị phân mảnh thì gói tin sẽ được đưa vào Socket thô chỉ khi các mạng được tập hợp và ghép mảnh đủ

Nhận dữ liệu với Socket thô (2)

- **Điều kiện để một socket thô nhận một packet**
  - **Nếu tham số `protocol` được thiết lập khi khởi tạo socket, chỉ có các gói tin có cùng trường giao thức đó được đưa vào socket.**
  - **Nếu hàm `bind()` được gọi trên một socket thô, chỉ có các gói tin có đích đến là địa chỉ IP được gán mới được đưa vào socket.**
  - **Nếu hàm `connect()` được sử dụng, chỉ có các gói tin được gửi từ địa chỉ đã chỉ định được đưa vào socket.**

Chương trình ping

- Hoạt động của chương trình ping rất đơn giản
  - Một thông báo ICMP echo request chứa nhãn thời gian được gửi tới địa chỉ IP của một node và node đó trả lời bằng một thông báo ICMP echo reply
    - RTT = thời gian nhận được thông báo ICMP echo reply - nhãn thời gian

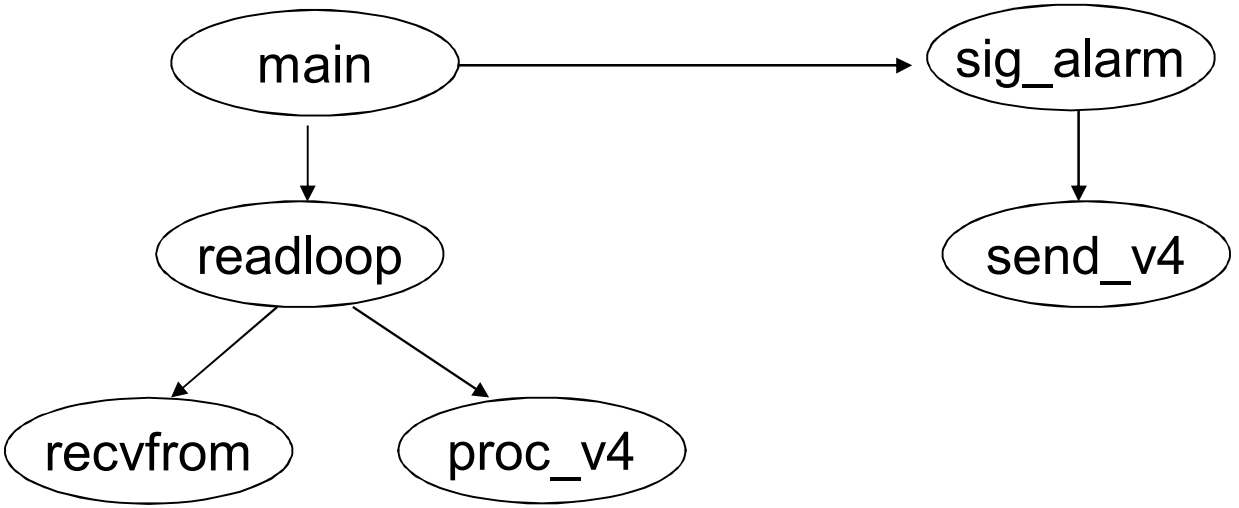


Định dạng của thông báo ICMP echo request và echo reply

Type	Code	CHECKSUM
Identifier		Sequence number
DATA (optional)		

- *type* = 8 và *code* = 0 với echo request
- *type* = 0 và *code* = 0 với echo reply
- Trường *Identifier* và *Sequence Number* được dùng phía client để so khớp thông báo reply với thông báo request đã gửi tương ứng
  - Trường *identifier* được gán bằng PID của tiến trình ping
  - Trường *sequence number* tăng lên một khi gửi gói tin

Các hàm cơ bản trong chương trình ping



đọc tất cả các gói tin nhận được và xuất ra kết quả

gửi gói tin ICMP echo request một lần một giây. Điều khiển bằng tín hiệu SIGALARM gửi mỗi giây một lần

ping.h header

- ping/ping.h
  - Định nghĩa hàm
  - Định nghĩa cấu trúc *proto*

main function

- ping/main.c
- Lấy thông tin về địa chỉ đích từ command line

readloop function

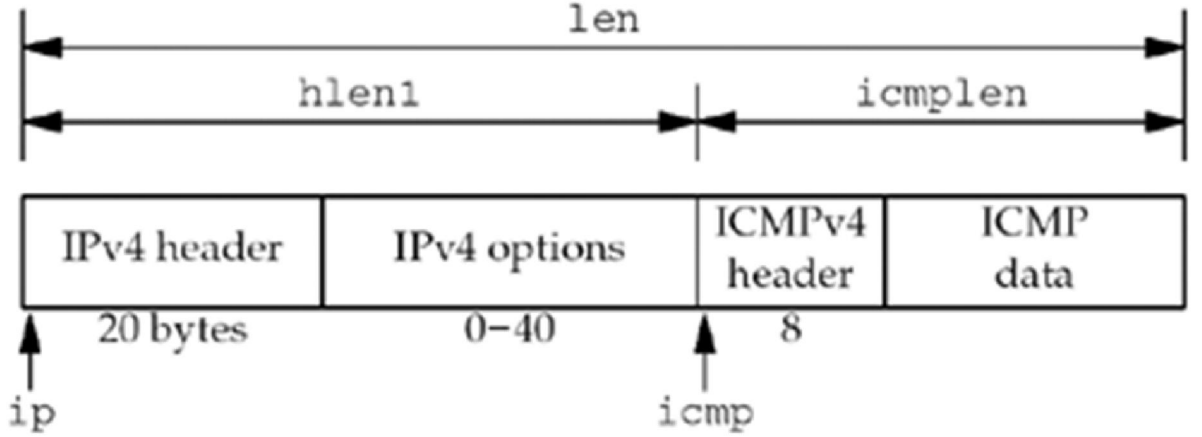
- ping/readloop.c
- Tạo socket
- Thiết lập kích thước bộ đệm nhận của socket
- Gửi gói tin ICMP đầu tiến
- Thực hiện lặp vô hạn để nhận các gói tin ICMP

tv_sub function: tính hiệu thời gian

- lib/tv_sub.c

proc_v4 function: xử lý thông báo ICMPv4

- ping/proc_v4.c
  - Lấy con trỏ trỏ đến ICMP header
  - Kiểm tra ICMP echo reply
  - Xuất tất cả các thông báo ICMP nhận được nếu tùy biến verbose được thiết lập



sig_alrm function: SIGALRM signal handler

- ping/sig_alrm.c

send_v4 function: builds an ICMPv4 echo request message and sends it

- ping/send_v4.c
  - Tạo thông báo ICMPv4
  - Tính toán ICMP checksum
  - Gửi gói tin

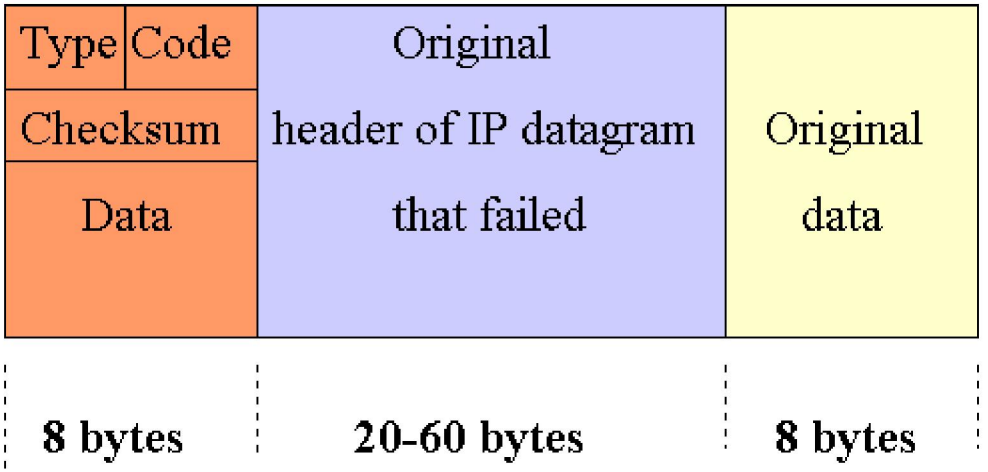
in_cksum function: Tính toán Internet checksum

- libfree/in_cksum.c

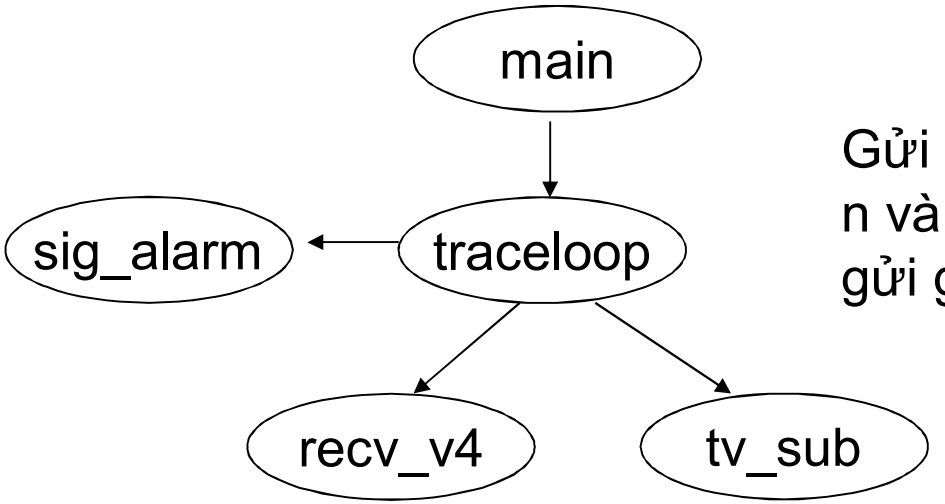
Chương trình traceroute

- Gửi gói tin UDP đến địa chỉ đích với trường TTL (hay hop limit) được gán bởi *n*
- Nhận gói tin ICMP với lỗi "time exceeded in transit"

Format of the ICMP Error Message



Chương trình



Gửi gói tin UDP với TTL = n và dữ liệu là thời gian gửi gói tin

Nhận gói tin ICMP_TIMXCEED hoặc gói tin ICMP_UNREACH và tính toán RTT

Chương trình

- traceroute/trace.h

main function

- traceroute/main.c

traceloop function: main processing loop

- traceroute/traceloop.c

recv_v4 function: reads and processes ICMPv4 messages

- traceroute/recv_v4.c

sig_alrm function

- traceroute/sig_alrm.c

Trả về chuỗi ký tự tương ứng với mã ICMPv6 unreachable

- traceroute/icmpcode_v4.c