

## Introduction to Network Management



Basic Management Architecture

Communication Environment

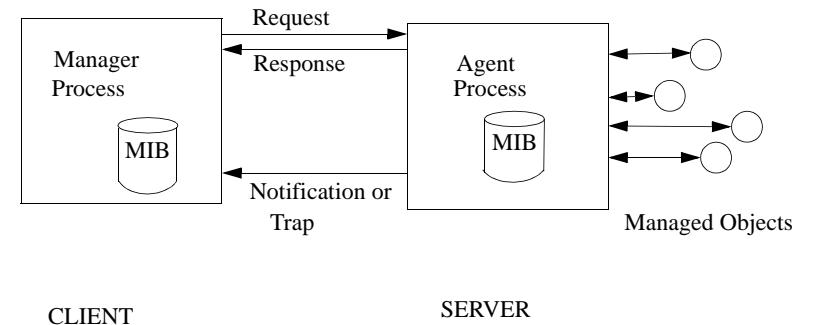
PING Packet Internet Groper

Traceroute

Socket Interprocess Communication

© Bernard Pagurek 2001

## The Manager/Agent Model



MIB (Management Information Base)

The term refers to the definition (or schema) of the managed objects. It is sometimes used loosely to refer to the collection of instances of managed objects.

The protocols do not specify how the agent maintains the managed objects which might very well be distributed.

A Managed Object is an abstraction. It is the managers view of a network element which can be managed.

Manager requests include : Get , GetNext, Set, Create, Delete, etc. depending on the management protocol in use.

Traps represent asynchronous agent initiated notifications.

Notifications can be asynchronous or synchronous.

## **OSI management functional areas (X.700)**

### **Introduction**

OSI management is required for a number of purposes. These requirements are categorized into a number of functional areas:

- a) fault management
- b) accounting management
- c) configuration management
- d) performance management
- e) security management

Specific management functions, within these functional areas, are provided by OSI management mechanisms. Many of the mechanisms are general in the sense that they are used to fulfil requirements in more than one functional area. Similarly, managed objects are general in the sense that they may be common to more than one functional area.

Each of these functional areas is described briefly below. The lists of functions are not necessarily exhaustive.

### **Fault management**

Fault management encompasses fault detection, isolation and the correction of abnormal operation of the OSI environment. Faults cause open systems to fail to meet their operational objectives and they may be persistent or transient. Faults manifest themselves as particular events (e.g. errors) in the operation of an open system. Error detection provides a capability to recognize faults. Fault management includes functions to

- a) maintain and examine error logs;
- b) accept and act upon error detection notifications;
- c) trace and identify faults;
- d) carry out sequences of diagnostic tests; and
- e) correct faults.

### **Accounting management**

Accounting management enables charges to be established for the use of resources in the OSIE, and for costs to be identified for the use of those resources. Accounting management includes functions to

- a) inform users of costs incurred or resources consumed;
- b) enable accounting limits to be set and tariff schedules to be associated with the use of resources; and
- c) enable costs to be combined where multiple resources are invoked to achieve a given communication objective.

### Configuration management

Configuration management identifies, exercises control over, collects data from and provides data to open systems for the purpose of preparing for, initializing, starting, providing for the continuous operation of, and terminating interconnection services. Configuration management includes functions to

- a) set the parameters that control the routine operation of the open system;
- b) associate names with managed objects and sets of managed objects;
- c) initialize and close down managed objects;
- d) collect information on demand about the current condition of the open system;
- e) obtain announcements of significant changes in the condition of the open system;
- f) change the configuration of the open system.

### Performance management

Performance management enables the behaviour of resources in the OSIE and the effectiveness of communication activities to be evaluated. Performance management includes functions to

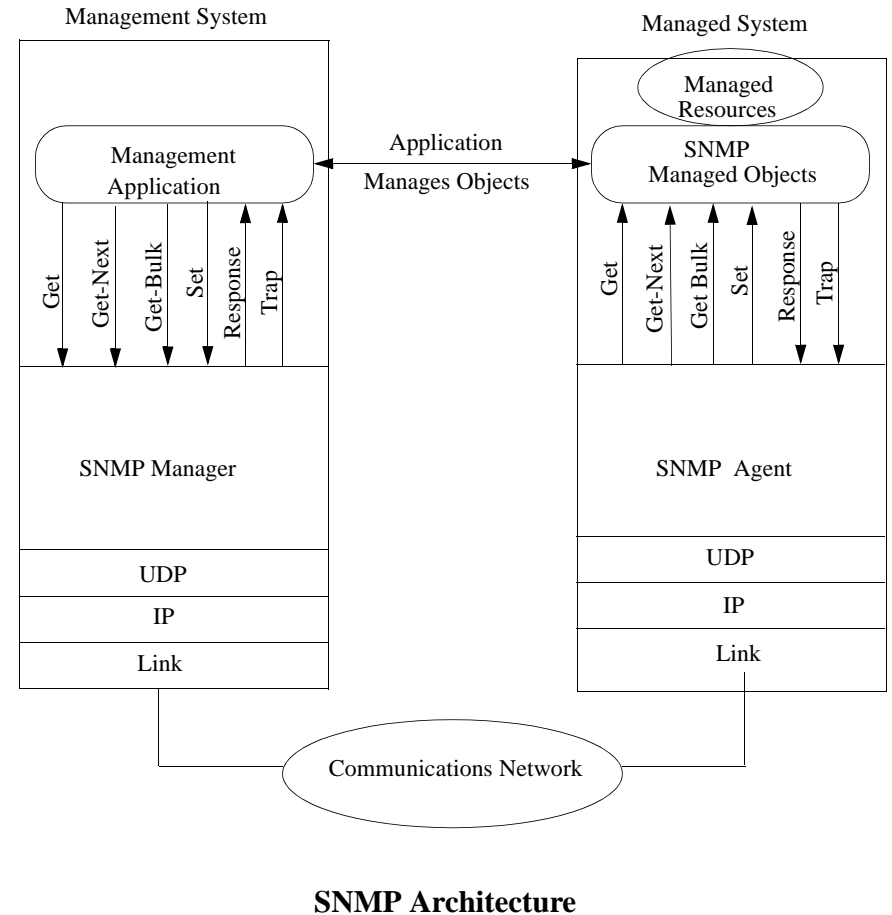
- a) gather statistical information;
- b) maintain and examine logs of system state histories;
- c) determine system performance under natural and artificial conditions; and
- d) alter system modes of operation for the purpose of conducting performance management activities.

### Security management

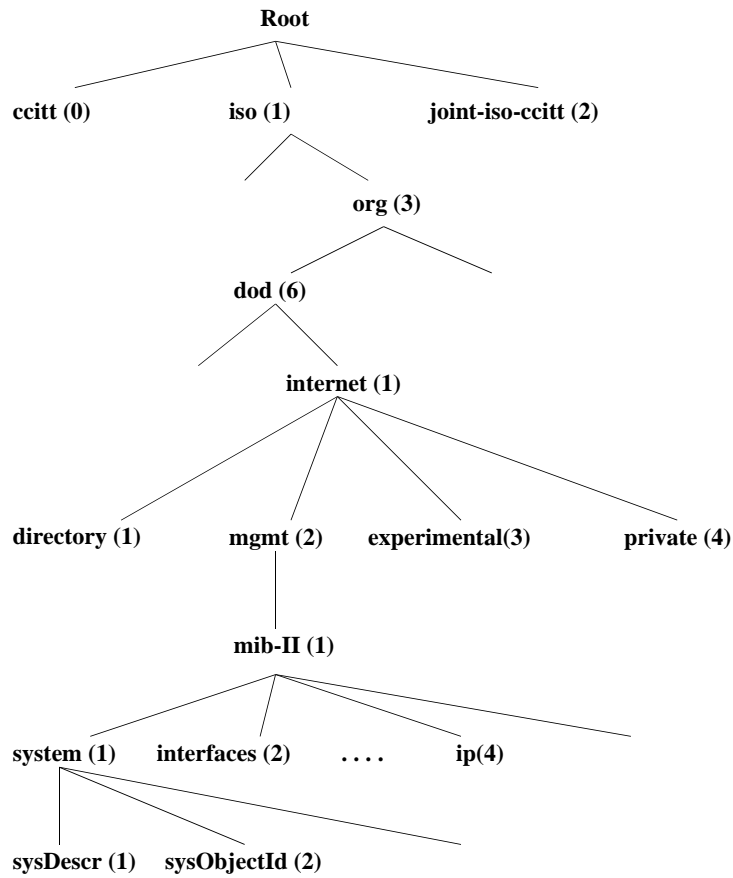
The purpose of security management is to support the application of security policies by means of functions which include

- a) the creation, deletion and control of security services and mechanisms;
- b) the distribution of security-relevant information; and
- c) the reporting of security-relevant events.

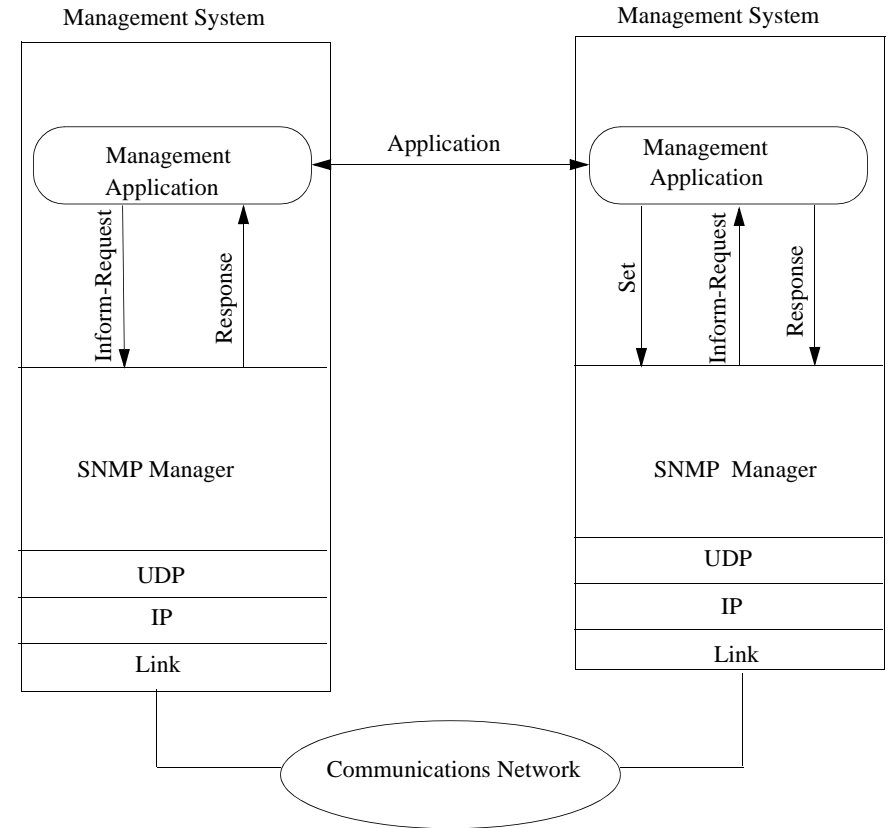
Note - Recommendation X.800 provides further information on the placement of OSI management functions within the overall security architecture.



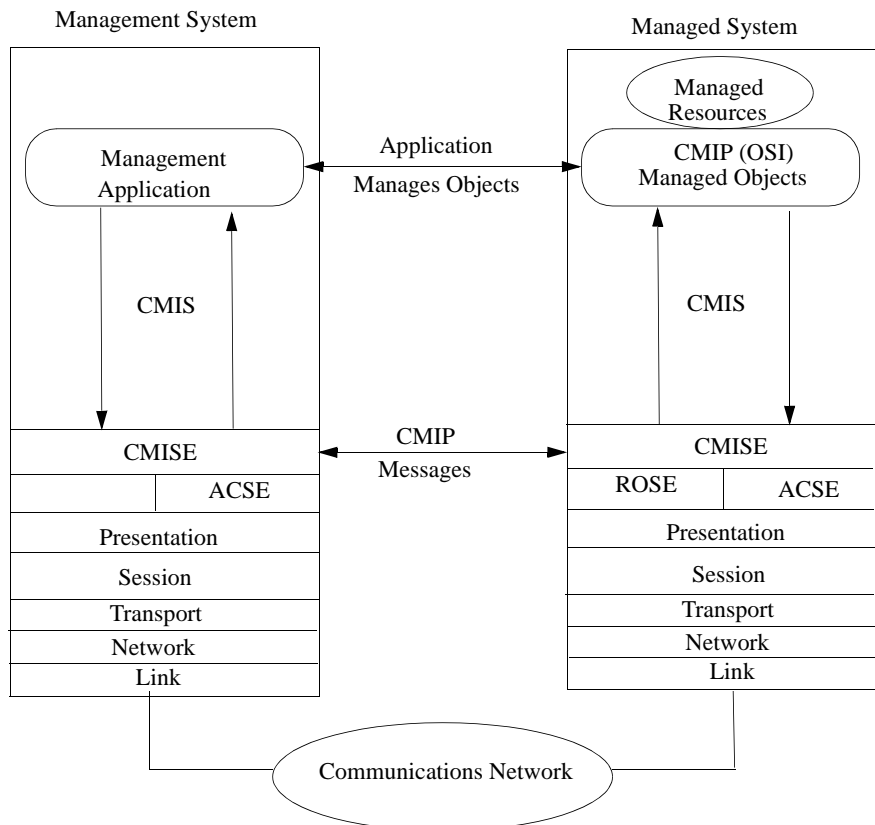
## Object Registration Tree



All objects in the system group have identifiers with prefix 1.3.6.1.2.1.1



## SNMPv2 Management to Management Communication



**CMIP Architecture**

### SNMP Philosophy

#### Simple:

Simple enough to be very widely deployed.  
Hence MIB2 for example reflects a minimal set of objects.  
Uses in band messaging.

Devices are purchased for purposes other than management

Agents must not be burdened with complexity. This is the managers job.

#### Robust:

Must be robust under adverse conditions. Manager must continue to operate if at all possible even if all else fails.

Datagram oriented (CL as opposed to CO)

No setup phase needed

No connection to fail

Minimal dependency on other services.

Manager application has full control over retransmission requests. If not critical, can ignore. If critical can keep requesting retransmission until.....

Messages limited to what can fit into a single UDP datagram.

Must fit into 576 byte IP datagram

#### Polling Oriented: (Request - Response)

Manager can't respond to traps if they are lost and it didn't even know it was sent. It can respond to lack of a reply to a request.

Limited number of trap types. Only for most important situations where increased vigilance is needed.

Concept is trap-directed polling.

**Re SNMP and CMIP coexistence and translation from one to the other.**

The basic philosophy is different.

The Entity models are different.

The information Models are different.

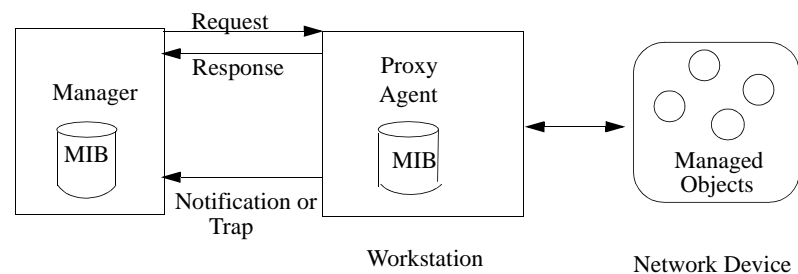
The naming mechanisms are different.

The SMIs are different.

The protocol operations are different.

The transport assumptions are different.

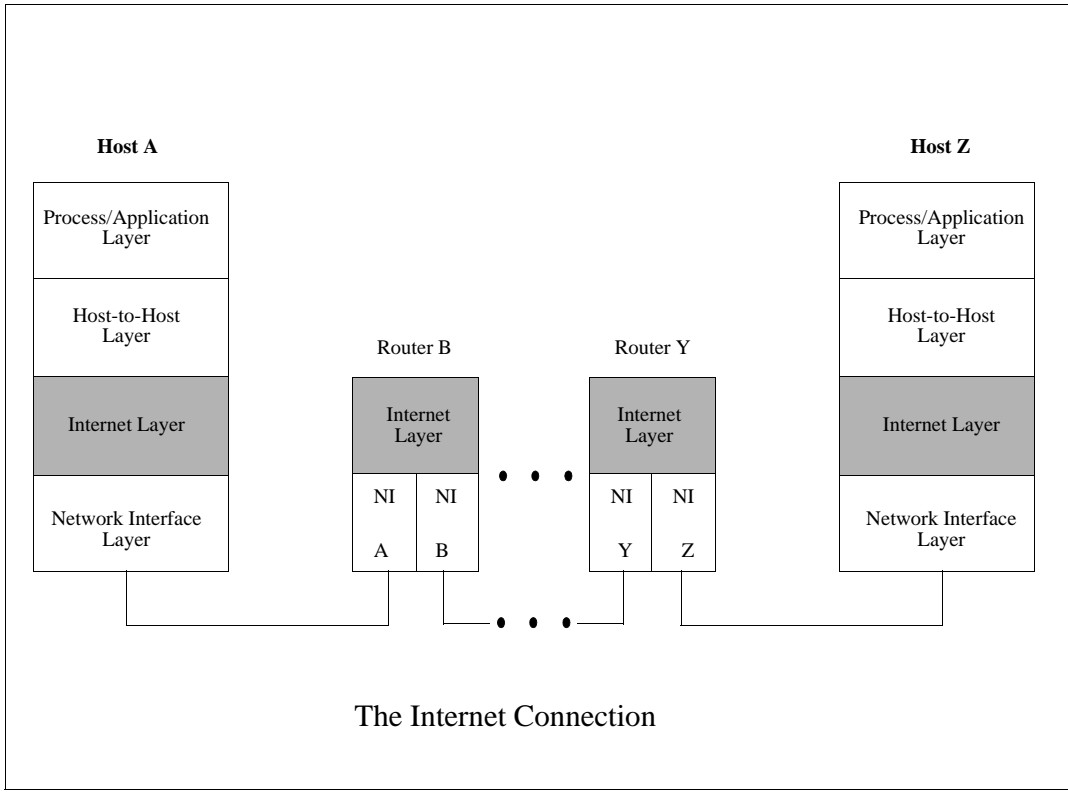
**Other than that, they both have managers and agents.**



Manager Proxy Agent Communication

Note: Network Device may be Native or Foreign

Proxy may communicate with network device using proprietary protocol



### Introduction to Network Management

Basic Management Architecture

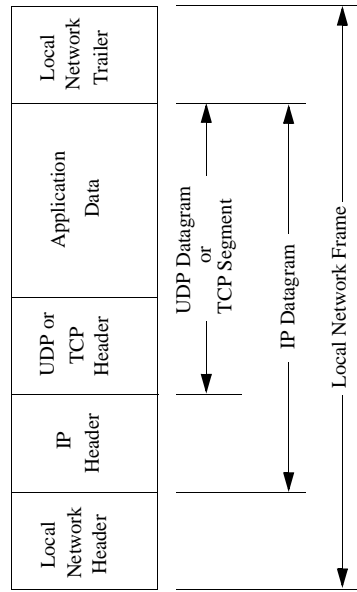
Communication Environment

PING Packet Internet Groper

Traceroute

Socket Interprocess Communication





The Internet Transmission Frame and IP Header Position

DARPA Layer		Example Internet Protocols				Example ISO Protocols				OSI Layer
Process/ Application	FTP SMTP TELNET rFTP SNMP	ISO 9040/9041 VT				ISO 8831/8832 JTM	ISO 8571/8572 FTAM	ISO 9595/9596 CMIP		Application
		ISO 8823/CCITT X.226 Connection-Oriented Presentation Protocol								Presentation
		ISO 8327/CCITT X.225 Connection-Oriented Session Protocol								Session
Host-to-Host	TCP		UDP		ISO 8073/CCITT X.224 Connection-Oriented Transport Protocol				Transport	
Internet	IP				ISO 8473 Connectionless Network Service		ISO 8208/CCITT X.25 Packet Level Protocol			Network
Network Interface	LAN, MAN and WAN Options				ISO 8802-2			ISO 7778 CCITT X.25 LAP/LAPB	ISO 7809 HDLC	Data Link
					ISO 9314-2 FDDI	ISO 8802-3 CSMA/CD BUS	ISO 8802-4 TOKEN BUS	ISO 8802-4 TOKEN RING	Options from EIA, CCITT, IEEE, etc.	

The Internet and the ISO Protocols Suites





## Problems with IPv4

### Not Enough IP Adresses available

IPv4 address has fixed boundary between ID of network and ID of node

e.g. class B allows for 65,536 host addresses  
class C allows for 256 addresses

Most sites are somewhere between so many addresses are wasted

### Routing Memory Requirements Increasing and Router Performance Deteriorating

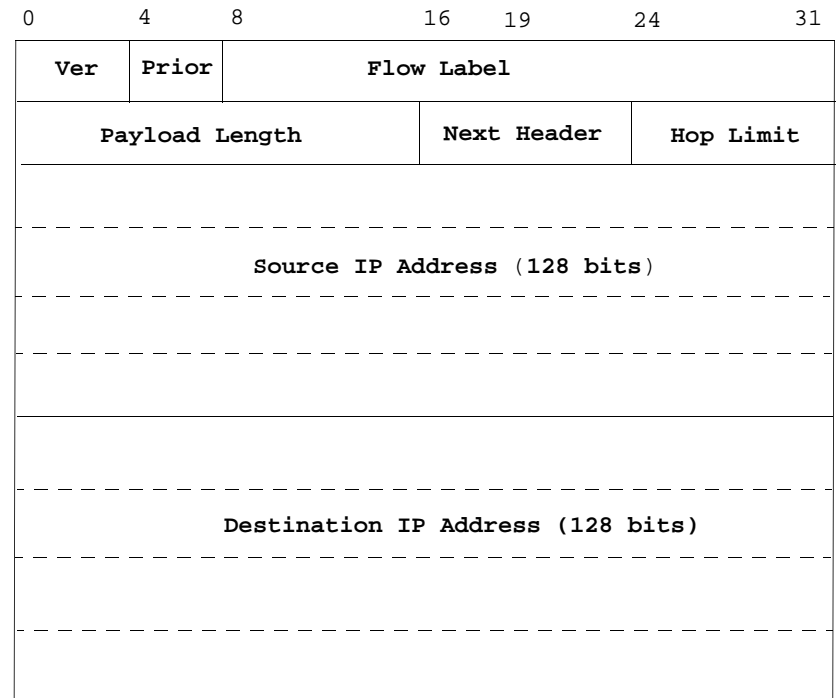
e.g. 16,384 possible class B networks  
over 2,000,000 class C networks

One entry is needed in a backbone router for each existing network  
Subnet masks used for hierarchical routing within network

### CIDR Classless Inter-Domain Routing

Introduced as interim solution to help alleviate problems  
Uses subnet masks for "sharing classes"  
Permits arbitrary aggregation of network and subnetwork numbers  
Requires fewer router table entries

## Internet Protocol (IPv6) Header



**Header** - only 40 bytes. Uses Extension Header for less used options

**Prior** - desired delivery priority relative to other packets from same source

**Flow Label** - Used by source to identify sequence of packets requiring special handling  
e.g. non-default quality of service or "real-time" service.  
Handling may be specified to routers by control protocol like a resource reservation protocol

## IPv6 Addressing

- Basic Addressing Types** - unicast
- multicast
  - anycast

**Unicast address denotes a single host interface**

3 FP	13 TLA ID	32 NLA ID	16 SLA ID	64 Interface ID
---------	-----------------	-----------------	-----------------	--------------------

### IPv6 Aggregatable Global Unicast Address Format

<b>FP Format Prefix</b>	denotes which address format is being used
<b>TLA Top-level Aggregation ID</b>	denotes top-level transit providers who make up backbone network
<b>NLA Next-level Aggregation ID</b>	assigned by transit providers to identify transit networks and sites serviced by provider
<b>SLA Site-level Aggregation ID</b>	assigned by site to individual networks within the site
<b>Interface ID</b>	assigned by site

## IPv6 prefix-based route aggregation

mechanism is based on on CIDR

Hex representation with colon separators (16 bytes - 2 hex digits per byte)

3FFE:0900:0001:0000:0260:97FF:FE6C:57BF

or in compressed format

3FFE:900:1::260:97FF:FE6C:57BF

Standard prefix notation

3FFE:0900::0/24

would mean that the first 24 bits of the address is the prefix

For unicast addresses, the prefix either identifies

a transit network if prefix length is between 1 and 64 bits long

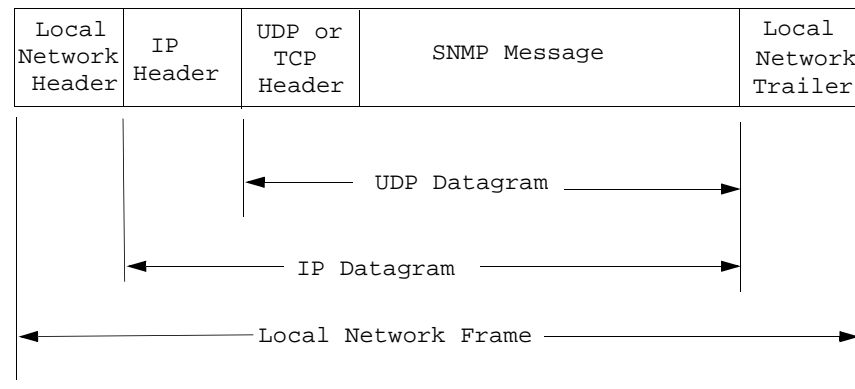
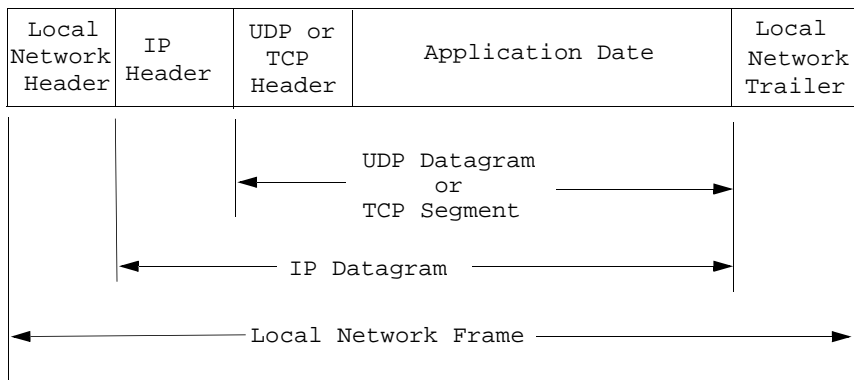
or

a specific address if prefix is 128 bits long.

For other types of address, the prefix has different meanings.

IPv6 prefix routing reduces table size but not algorithm complexity

**INTERNET Transmission Frame and UDP/TCP Header Position**



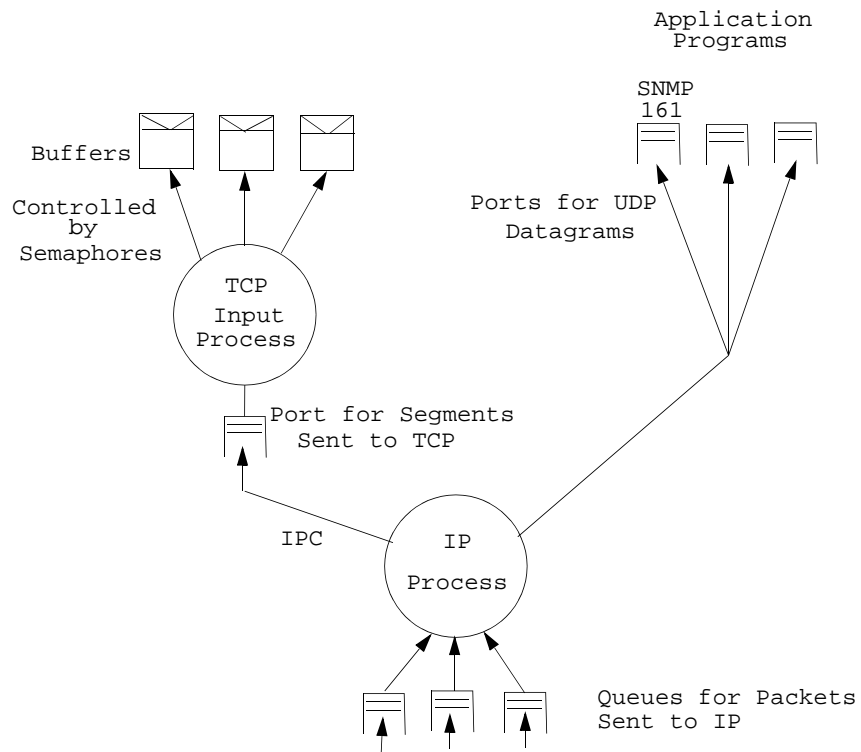
**SNMP Message Within A Transmission Frame**

**User Datagram Protocol (UDP) Fields**

0		16		31
UDP Source Port		UDP Destination Port		
UDP Message Length		UDP Checksum		
Data				
.....				

## Communications Network Management

Notes:



### Internet Packet Flow at Higher Levels

#### Typical UDP Ports

21	FTP	File Transfer (Control)
53	DOMAIN	Domain Name Server
68	TFTP	Trivial File Transfer
111	SUNRPC	SUN Microsystems RPC
161	SNMP	SNMP net monitor
162	SNMP	SNMP traps

## Introduction to Network Management

Basic Management Architecture

Communication Environment

 PING Packet Internet Groper

Traceroute

Socket Interprocess Communication

## Internet Ping Program

"ping" stands for Packet InterNet Groper.

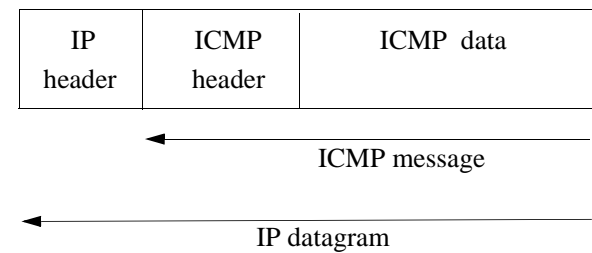
We ping a system by sending it ICMP echo requests that it must respond to with ICMP echo replies.

ICMP is at the same layer as IP.

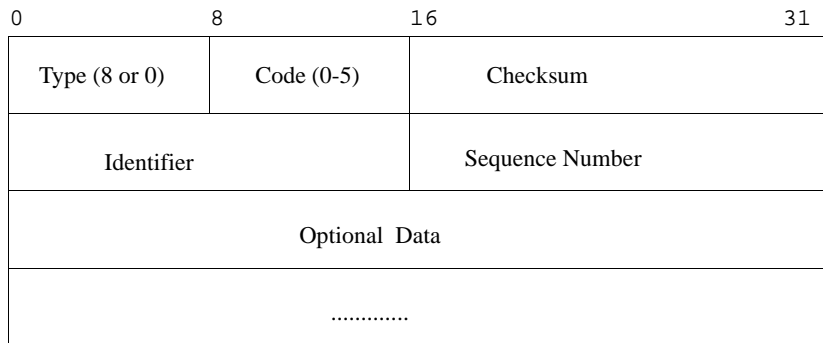
The operation of ICMP doesn't depend on the higher level protocols - TCP and UDP.

The echo request and echo reply messages are only two of the 13 currently defined ICMP messages.

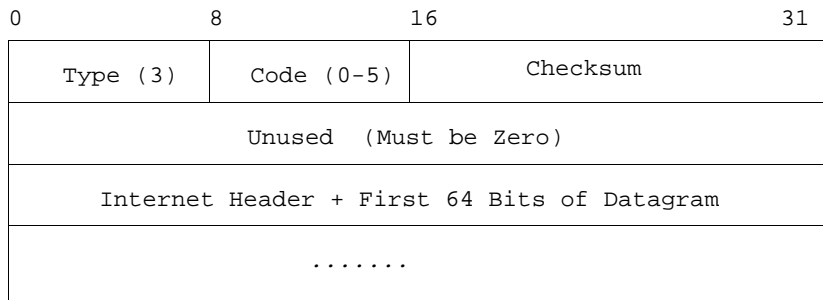
ICMP messages are sent in IP packets:



ICMP Echo Request and Reply Message Format



ICMP Destination Unreachable Message Format



The format of the ICMP message:

```
struct icmp (
    u_char icmp_type; /*type of message */
    u_char icmp_code; /* type of code */
    u_short icmp_cksum; /* checksum of structure */
    u_short icmp_id; /* identifier */
    u_short icmp_seq; /* sequence number */
    char icmp_data[1]; /* start of optional data */
);
```

icmp\_type specifies the type of the ICMP message

The two values used by the ping program are ICMP\_ECHO and ICMP\_ECHOREPLY.

icmp\_code is a subcode for some of the ICMP messages.

Not used by echo request or echo reply.

icmp\_id and icmp\_seq are set by the client, and returned by the server.

The identifier field identifies the sender of the ICMP echo requests; we set it to the Unix process ID of the ping program.

If the ping program is being run multiple times on a given host, each instance generates messages with a unique ID field.

We use `icmp_seq` as a sequence number to identify each message that a client transmits.

There is no guarantee that successive packets travel the same route, so the return packets can arrive in a different order from which they were transmitted.

This sequence number field lets us identify each message.

The ping program stores the time that each message is transmitted in the optional data portion, and uses this when the packet is returned to calculate the round-trip time.

The ping program contains two logical portions; one transmits an ICMP echo request message every second and the other receives any echo reply messages that are returned.

The transmit portion is simple - it uses the Unix alarm function to generate a `SIGALRM` signal every second.

The receive portion is an infinite loop that receives every ICMP message on a socket.

## A Simple Procedure for Connectivity Check Using PING, TRACEROUTE

This procedure is for isolating a TCP/IP connectivity problem. In this procedure, a series of tests methodically examine connectivity from a host, starting with nearby resources and working outward. The steps in our connectivity-testing procedure are:

1. As an initial sanity check, ping your own IP address and the loopback address.
2. Next, try to ping other IP hosts on the local subnet. Use numeric addresses when starting off, since this eliminates the name resolvers and host tables as potential sources of problems. The lack of an answer may indicate either that the destination host did not respond to ARP (if it is used on your LAN), or that a datagram was forwarded (and hence, the destination IP address was resolved to a local media address) but that no ICMP Echo Reply was received. This could indicate a length-related problem, or misconfigured IP Security.
3. If an IP router (gateway) is in the system, ping both its near and far-side addresses.
4. Make sure that your local host recognizes the gateway as a relay. (For BSD hosts, use `netstat`.)
5. Still using numeric IP addresses, try to ping hosts beyond the gateway. If you get no response, run `hop-check` or `traceroute`, if available. Note whether your packets even go to the gateway on their way to the destination. If not, examine the methods used to instruct your host to use this gateway to reach the specified destination net (e.g., is the default route in place? Alternatively, are you successfully wire-tapping the IGP messages broadcast on the net you are attached to?)

If `traceroute` is not available, ping, `netstat`, `arp`, and a knowledge of the IP addresses of all the gateway's interfaces can be used to isolate the cause of the problem. Use `netstat` to determine your next hop to the destination. Ping that IP address to ensure the router is up. Next, ping the router inter-



## Communications Network Management

Notes:

face on the far subnet. If the router returns “network unreachable” or other errors, investigate the router’s routing tables and interface status. If the pings succeed, ping the close interface of the succeeding next hop gateway, and so on. Remember the routing along the outbound and return paths may be different.

6. Once ping is working with numeric addresses, use ping to try to reach a few remote hosts by name. If ping fails when host names are used, check the operation of the local name-mapping system (i.e., with nslookup or DiG). If you want to use “shorthand” forms (“myhost” instead of “myhost.mydomain.com”), be sure that the alias tables are correctly configured.
7. Once basic reachability has been established with ping, try some TCP - based applications: FTP and TELNET are supported on almost all IP hosts, but FINGER is a simpler protocol. The Berkeley-specific protocols (RSH, RCP, REXEC and LPR) require extra configuration on the server host before they can work, and so are poor choices for connectivity testing.

If problems arise in steps 2-7 above, rerunning the tests while executing a line monitor ( etherfind, netwatch, or tcpdump) can help to pinpoint the problem.

The above procedure is sound and useful, especially if little is known about the cause of the connectivity problem. It is not, however, guaranteed to be the shortest path to diagnosis. In some cases, a binary search on the problem might be more effective (i.e., try a test “in the middle,” in a spot where the failure modes are well defined). In other cases, available information might so strongly suggest a particular failure that immediately testing for it is in order. This last “approach,” which might be called “hunting and pecking,” should be used with caution: chasing one will o’ the wisp after another can waste much time and effort.

Note that line problems are still among the most common causes of connectivity loss. Problems in transmission across local media are outside the scope of this note. But, if a host or workstation loses or cannot establish connectivity, check its physical connection.

From RFC1147

## Introduction to Network Management

Basic Management Architecture

Communication Environment

PING Packet Internet Groper

 Traceroute

Socket Interprocess Communication

## Traceroute Program

Originally written by Van Jacobson.

It is used to determine exactly what route a packet takes to a specified destination.

A UDP datagram is sent to the destination, however the first time it is sent, the time-to-live field is set to 1.

This causes the first gateway to discard the packet and return an ICMP "time exceeded".

The ICMP reply will have the gateway's IP address as the source IP address, so we know the identity of the first gateway.

Additionally, the time required to receive the ICMP reply is also measured, to give an estimate of the round-trip time (RTT) to the gateway.

This procedure is repeated two more times, to give three RTT estimates for this gateway.

Next the TTL is set to 2, to determine the identity of the second gateway.

Three measurements are made for this gateway.

When the UDP datagram finally reaches the destination host an ICMP "port unreachable" reply should be returned, since the destination UDP port number is chosen so that it is unlikely that any process on the destination is using that port (UDP port 33434, by default).

Note that the program must be looking for two types of ICMP messages to be returned: "time exceeded" and "port unreachable".

This program requires a kernel modification to run under many Berkeley-derived systems.

An option also exists to specify a loose source route for the datagram, however many gateways don't handle source routing correctly.

An example output is:

```
[yak 71]% traceroute nis.nsf.net
o nis.nsf.net, 30 hops max, 56 byte packet
 1. helios.ee.lbl.gov 19 ms 19 ms 0 ms
 2. lilac-dmc.Berkeley.EDU 39 ms 39 ms 19 ms
 3. lilac-dmc.Berkeley.EDU 39 ms 39 ms 19 ms
 4. ccngw-ner-cc.Berkeley.EDU 39 ms 40 ms 39 ms
 5. ccn-nerif22.Berkely.EDU 39 ms 39 ms 39 ms
 6. 128.32.197.4 40 ms 59 ms 59 ms
 7. 131.119.2.5 59 ms 59 ms 59 ms
 8. 129.140.70.13 99 ms 99 ms 80 ms
 9. 129.140.71.6 139 ms 239 ms 319 ms
10. 129.140.81.7 220 ms 199 ms 199 ms
11. nic.merit.edu 239 ms 239 ms 239 ms
```

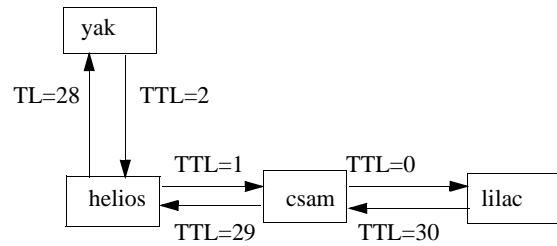
The dotted decimal addresses on each line have been removed to save space

Note that lines 2 and 3 are the same. This is because of a bug on the second system (csam) that forwards packets to the third system (lilac-dmc) even if the TTL field is zero..

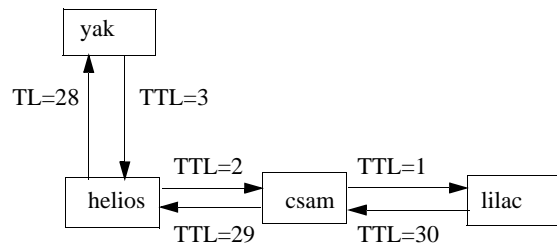
## Communications Network Management

Notes:

When the initial TTL is two, we have



hen the initial TTL is three, we have



This is why both lines of output, for TTL values of two and three, are both returned by lilac.


## Introduction to Network Management

Basic Management Architecture

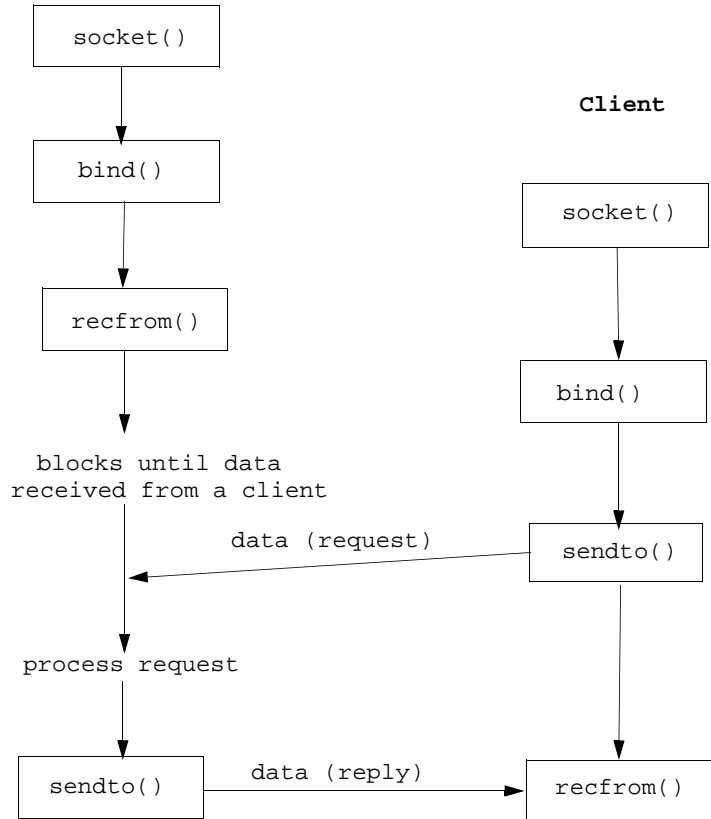
Communication Environment

PING Packet Internet Groper

Traceroute

 Socket Interprocess Communication

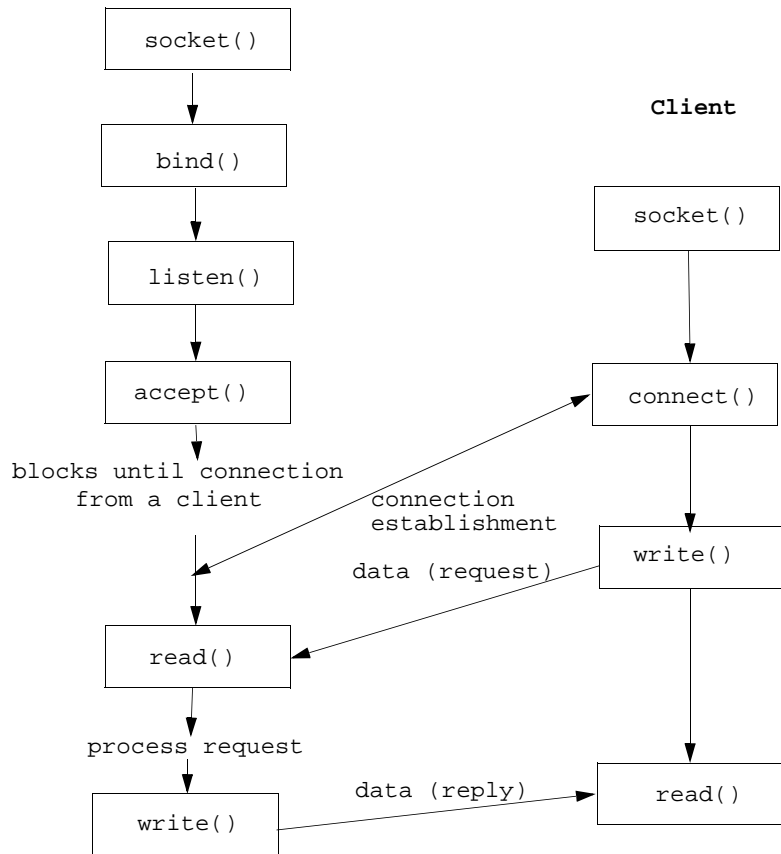
**Server**  
(connectionless protocol)



Socket System calls for connectionless protocol

### Server

(connection-oriented protocol)



Socket System calls for connection oriented protocol

```
/* Example of client using UDP protocol. */
```

```
#include "inet.h"
```

```
main(argc, argv)
int argc;
char *argv[];
{
int sockfd;
struct sockaddr_in, cli_addr, serv_addr;
pname = argv[0];
```

```
/* Fill in the structure "serv_addr" with the address
of the server that we want to send to. */
```

```
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
serv_addr.sin_port = htons(SERV_UDP_PORT);
```

```
/* Open a UDP socket (an Internet datagram socket). */
```

```
if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
err_dump("client: can't open datagram socket");
```

```
/* Bind any local address for us. */
```

```
bzero((char *) &cli_addr, sizeof(cli_addr)); /*zero out*/
cli_addr.sin_family = AF_INET;
cli_addr.sin_addr.s_addr = htonl(INADDR_ANY);
cli_addr.sin_port = htons(0);
if ( bind(sockfd, (struct sockaddr *) &cli_addr,
sizeof(cli_addr)) < 0 )
err_dump("client: can't bind local address");
```

```
dg_cli(stdin, sockfd, (struct sockaddr *) &serv_addr,
sizeof(serv_addr));
close(sockfd);
exit(0);
}
```

```

/* Example of server using UDP protocol. */

#include "inet.h"

main(argc, argv)
int argc;
char *argv[];
{
int sockfd;
struct sockaddr_in, serv_addr, cli_addr;

pname = argv[0];

/* Open a UDP socket (an Internet datagram socket). */

if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
    err_dump("server: can't open datagram socket");

/*Bind our local address so that the client can send to
us. */

bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family      = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port       = htons(SERV_UDP_PORT);

if (bind(sockfd, (struct sockaddr *) &serv_addr,
    sizeof(serv_addr)) < 0)
    err_dump("server: can't bind local address");

dg_echo(sockfd, (struct sockaddr *) &cli_addr,
    sizeof(cli_addr));

/* NOTREACHED */
}

```

```

/* File inet.h */
/* Definitions for TCP and UDP client/server programs.*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SERV_UDP_PORT 6000
#define SERV_HOST_ADDR "192.43.235.6" /*server host addr*/

char *pname;

struct sockaddr_in {

    short sin_family;      /* AF_INET */
    u_short sin_port;      /* 16 bit port # */
    struct sin_addr;       /* 32 bit netid/hostid */
    char sin_zero;        /* unused */

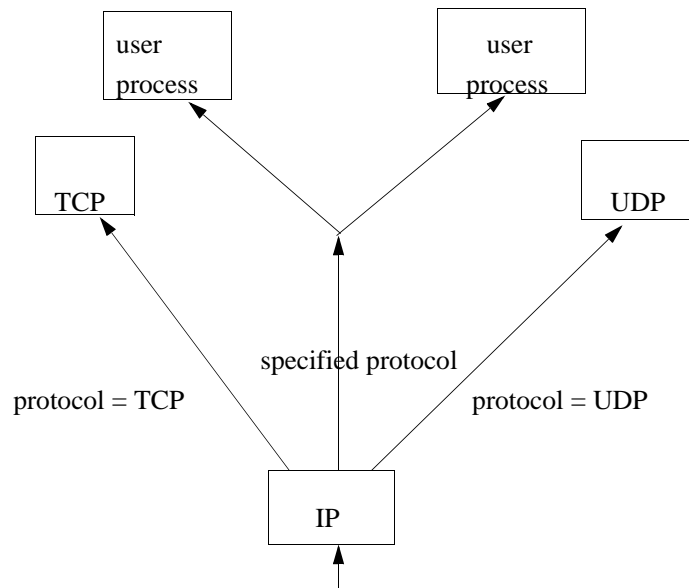
};

```

## Raw Sockets

We create raw internet socket and specify one particular Internet protocol as the final argument to the socket function.

This protocol is used by IP to demultiplex the data it receives:



Only superuser processes can open raw sockets.

IP prepends an appropriate IP header to any data we write.

The protocol field in this IP header is the value specified when the socket was created.

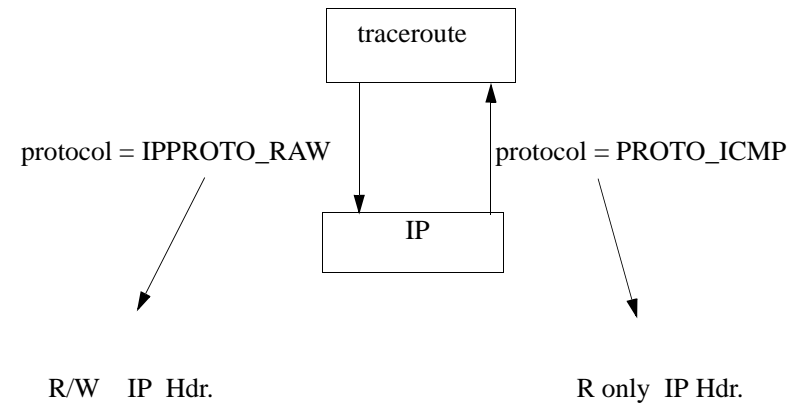
When data is received by the kernel for this protocol, when IP is finished with the data, a copy is passed to *all* processes that have raw sockets for this protocol.

This data contains the received IP header.

Newer kernels (4.3BSD, SunOS 4.1) allow the protocol to be IPPROTO\_RAW.

This allows the user process to build its own IP header.

Traceroute needs this to set its TTL field:





## **Broadcasting**

Broadcasting is only allowed on datagram sockets.

The network being used must support broadcasting, such as an Ethernet or a token ring.

The user process must specify that a socket can be used for broadcasting: the `SO_BROADCAST` socket option must be enabled.

4.3BSD does not allow a broadcast IP datagram to be fragmented.

By convention, an Internet address with a host ID of all 1 bits is considered a broadcast address.

A program should use the `SIOCGIFBRDADDR` `ioctl` to obtain the broadcast address for a given interface.