RAW SOCKETS

Compiled by: kanitnath@gmail.com

INTRODUCTION

- A raw socket is a type of socket that allows access to the underlying transport provider.
- Raw sockets, are those that bypass the TCP and IP layers and pass the ICMPv4, (Internet Control Message Protocol), IGMPv4 ()Internet Group Management Protocol – used with multicasting) and ICMPv6 packets directly to the link layers.
- Raw sockets are not used for most applications. These sockets are the same as the datagram oriented, their characteristics are dependent on the interfaces. They provided support in developing new communication protocols or for access to more facilities of an existing protocol. Only the superusers can access the Raw Sockets.
- The socket type of Raw Socket is SOCK_RAW.
- Winsock service providers for the IP protocol may support a socket *type* of **SOCK_RAW**. The Windows Sockets 2 provider for TCP/IP included on Windows supports this **SOCK_RAW** socket type.

There are two basic types of such raw sockets:

- The ***first type*** uses a known protocol type written in the IP header that is recognized by a Winsock service provider. An example of the first type of socket is a socket for the ICMP protocol (IP protocol type = 1) or the ICMPv6 protocol (IP procotol type = 58).
- The ***second type*** allows any protocol type to be specified. An example of the second type would be an experimental protocol that is not directly supported by the Winsock service provider such as the Stream Control Transmission Protocol (SCTP).

- With raw sockets a process can read and write IPv4 datagram with IPv4 protocol filed (an 8 bit filed in IPv4 packet) that is not processed by the kernel.
- Most kernels process datagrams containing values of 1 (ICMP), 2 (IGMP), 6 (TCP), and 17 (UDP).
- But values like 89 (OSPF) routing protocol does not use TCP or UDP but uses IP directly by setting the protocol field to 89.
- With raw sockets, a process can build its own IPv4 header using the IP_HDRINCL socket option

Raw Socket Creation

1. The socket function creates a raw socket when the second argument is SOCK_RAW. The third argument (the protocol) is normally nonzero.

   Int sockfd;

   Sockfd = socket (AF_INET, SOCK_RAW, protocol);

   where *protocol* is one of the constants, IPPROTO_*xxx*, defined by including the <netinet/in.h> header, such as IPPROTO_ICMP.

   Only the superuser can create a raw socket.

   This prevents normal users from writing their own IP datagrams to the network.

2. The ***IP_HDRINCL*** socket option can be set as follows:

   if (setsocketopt(sockfd, IPPROTO_IP, IP_HDRINCL, &ON, soze0f(ON)) <0) error

3. ***bind*** can be called on the raw socket, but this is rare. This function sets only the local address: There is no concept of a port number with a raw socket. With regard to output, calling bind sets the source IP address that will be used for datagrams sent on the raw socket (but only if the IP_HDRINCL socket option is not set). If bind is not called, the kernel sets the source IP address to the primary IP address of the outgoing interface.

4. ***connect*** can be called on the raw socket, but this is rare. This function sets only the foreign address: Again, there is no concept of a port number with a raw socket. With regard to output, calling connect lets us call write or send instead of sendto, since the destination IP address is already specified.

Raw Socket Output:

The output of raw socket is governed by the following rules:

1. Normal output is performed by calling **sendto** or **sendmsg** and specifying the destination IP address. IN case the socket has been connected, **write** and **send** functions can be used.
2. If the **IP_HDRINCL** option is **not set,** the IP header will be built by the kernal and it will be prepend it to the data.
3. If **IP_HDRINCL** is **set,** the header format will remain the same and the process builds the entire IP header except the IPv4 identification field which is set to 0 by the kernel
4. The kernel fragments the raw packets that exceed the outgoing interface.

Raw Socket Input

The first question that we must answer regarding raw socket input is: Which received IP datagrams does the kernel pass to raw sockets? The following rules apply:

1. Received UDP packets and received TCP packets are *never* passed to a raw socket. If a process wants to read IP datagrams containing UDP or TCP packets, the packets must be read at the datalink layer
2. *Most* ICMP packets are passed to a raw socket after the kernel has finished processing the ICMP message. Berkeley-derived implementations pass all received ICMP packets to a raw socket other than echo request, timestamp request, and address mask request. These three ICMP messages are processed entirely by the kernel.
3. *All* IGMP packets are passed to a raw socket after the kernel has finished processing the IGMP message.
4. *All* IP datagrams with a protocol field that the kernel does not understand are passed to a raw socket. The only kernel processing done on these packets is the minimal verification of some IP header fields: the IP version, IPv4 header checksum, header length, and destination IP address
5. If the datagram arrives in fragments, nothing is passed to a raw socket until all fragments have arrived and have been reassembled.

When the kernel has an IP datagram to pass to the raw sockets, all raw sockets for all processes are examined, looking for all matching sockets.

A copy of the IP datagram is delivered to *each* matching socket.

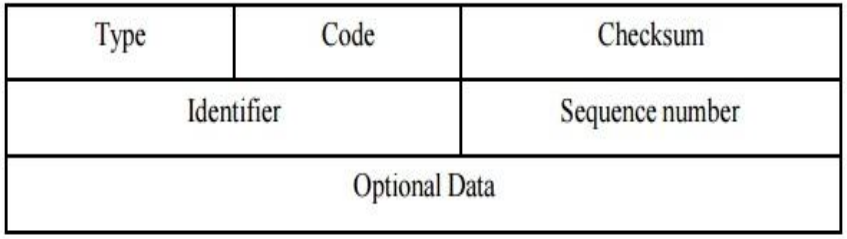
The following tests are performed for each raw socket and only if all three tests are true is the datagram delivered to the socket:

1. If a ***nonzero protocol*** is specified when the raw socket is ***created*** (the third argument to socket), then the received datagram's protocol field must match this value or the datagram is not delivered to this socket.
2. If a ***local IP address*** is bound to the raw socket by ***bind***, then the destination IP address of the received datagram must match this bound address or the datagram is not delivered to this socket.
3. If a ***foreign IP address*** was specified for the raw socket by ***connect***, then the source IP address of the received datagram must match this connected address or the datagram is not delivered to this socket.

Notice that if a raw socket is created with a *protocol* of 0, and ***neither bind nor connect*** is called, then that socket receives a copy of *every* raw datagram the kernel passes to raw sockets.

Ping Program:

- In this ICMP echo request is sent to some IP address and that the node responds with an ICMP echo reply.
- These two ICMP messages are supported under IPv4 and IPv6.
- Following figure shows the format of the ICMP messages.



Type	Code	Description	Handled by error no
0	0	Echo reply	User process (ping)
3		Destination unreachable	
	0	Network unreachable	EHOSTUNREACH
	1	Host unreachable	EHOSTUNREACH
	2	Protocol un reachable	ECONNREFUSED.
	4	port unreachable	ECONNREFUSED
5		REDIRECT	
	0	Redirect for network	Kernel updates routing table
	1		

- set the *identifier* to the PID of the ping process
- increment the *sequence number* by one for each packet we send.
- store the 8-byte timestamp of when the packet is sent as the *optional data*.

The rules of ICMP require that the *identifier, sequence number*, and any *optional data* be returned in the echo reply.

Overview of the functions in our ping program.

- The program operates in two parts:
  - One half reads everything received on a raw socket, printing the ICMP echo replies, and the other half sends an ICMP echo request once per second.
  - The second half is driven by a SIGALRM signal once per second.