NAME

ip, IP – Internet Protocol

SYNOPSIS

#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_RAW, *proto***);**

t = t_open ("/dev/rawip", O_RDWR);

DESCRIPTION

IP is the internetwork datagram delivery protocol that is central to the Internet protocol family. Programs may use IP through higher-level protocols such as the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP), or may interface directly to IP. See **tcp**(7P) and **udp**(7P). Direct access may be via the socket interface (using a "raw socket") or the Transport Level Interface (TLI). The protocol options defined in the IP specification may be set in outgoing datagrams.

The STREAMS driver **/dev/rawip** is the TLI transport provider that provides raw access to IP.

Raw IP sockets are connectionless and are normally used with the **sendto( )** and **recvfrom( )** calls, ((see **send**(3N) and **recv**(3N)) although the **connect**(3N) call may also be used to fix the destination for future datagrams (in which case the **read**(2) or **recv**(3N) and **write**(2) or **send**(3N) calls may be used). If **proto** is **IPPROTO_RAW**, or **IPPROTO_IGMP** the application is expected to include a complete IP header when sending. Otherwise, that protocol number will be set in outgoing datagrams and used to filter incoming datagrams and an IP header will be generated and prepended to each outgoing datagram. In either case received datagrams are returned with the IP header and options intact.

The socket options supported at the IP level are:

IP_OPTIONS IP options for outgoing datagrams. This socket option may be used to set IP options to be included in each outgoing datagram. IP options to be sent are set with **setsockopt( )** (see **getsockopt**(3N)). The **getsockopt**(3N) call returns the IP options set in the last **setsockopt( )** call. IP options on received datagrams are visible to user programs only using raw IP sockets. The format of IP options given in **setsockopt( )** matches those defined in the IP specification with one exception: the list of addresses for the source routing options must include the first-hop gateway at the beginning of the list of gateways. The first-hop gateway address will be extracted from the option list and the size adjusted accordingly before use. IP options may be used with any socket type in the Internet family.

IP_ADD_MEMBERSHIP Join a multicast group.

IP_DROP_MEMBERSHIP Leave a multicast group.

These options take a **struct ip_mreq** as the parameter. The structure contains a multicast address which has to be set to the **CLASS-D** IP multicast address, and an interface address. Normally the interface address is set to **INADDR_ANY** which causes the kernel to choose the interface to join on.

IP_MULTICAST_IF The outgoing interface for multicast packets. This option takes a **struct in_addr** as an argument and it selects that interface for outgoing IP multicast packets. If the address specified is **INADDR_ANY** it will use the unicast routing table to select the outgoing interface (which is the default behavior.)

IP_MULTICAST_TTL Time to live for multicast datagrams. This option takes an unsigned character as an argument. Its value is the TTL that IP will use on outgoing multicast datagrams. The default is 1.

IP_MULTICAST_LOOP Loopback for multicast datagrams. Normally multicast datagrams are delivered to members on the sending host. Setting the unsigned character argument to 0 will cause the opposite behavior.

The multicast socket options can be used with any datagram socket type in the Internet family.

At the socket level, the socket option **SO_DONTROUTE** may be applied. This option forces datagrams being sent to bypass routing and forwarding by forcing the IP Time To Live field to 1 (meaning that the packet will not be forwarded bu routers).

Raw IP datagrams can also be sent and received using the TLI connectionless primitives.

Datagrams flow through the IP layer in two directions: from the network *up* to user processes and from user processes *down* to the network. Using this orientation, IP is layered *above* the network interface drivers and *below* the transport protocols such as UDP and TCP. The Internet Control Message Protocol (ICMP) is logically a part of IP. See **icmp**(7P).

IP provides for a checksum of the header part, but not the data part of the datagram. The checksum value is computed and set in the process of sending datagrams and checked when receiving datagrams.

IP options in received datagrams are processed in the IP layer according to the protocol specification. Currently recognized IP options include: security, loose source and record route (LSRR), strict source and record route (SSRR), record route, and internet timestamp.

The IP layer will normally act as a router (forwarding datagrams that are not addressed to it etc) when the machine has two or more interfaces that are up. This behavior can be overridden by using **ndd**(1M) to to set the **/dev/ip** variable **ip_forwarding**. The value 0 means do not forward, 1 means forward and 2 gives you the default behavior of forwarding when there are two or more "up" interfaces.

The IP layer will send an ICMP message back to the source host in many cases when it receives a datagram that can not be handled. A “time exceeded” ICMP message will be sent if the “time to live” field in the IP header drops to zero in the process of forwarding a datagram. A “destination unreachable” message will be sent if a datagram can not be forwarded because there is no route to the final destination, or if it can not be fragmented. If the datagram is addressed to the local host but is destined for a protocol that is not supported or a port that is not in use, a destination unreachable message will also be sent. The IP layer may send an ICMP “source quench” message if it is receiving datagrams too quickly. ICMP messages are only sent for the first fragment of a fragmented datagram and are never returned in response to errors in other ICMP messages.

The IP layer supports fragmentation and reassembly. Datagrams are fragmented on output if the datagram is larger than the maximum transmission unit (MTU) of the network interface. Fragments of received datagrams are dropped from the reassembly queues if the complete datagram is not reconstructed within a short time period.

Errors in sending discovered at the network interface driver layer are passed by IP back up to the user process.

SEE ALSO

ndd(1M), **read**(2), **write**(2), **connect**(3N), **getsockopt**(3N), **recv**(3N), **send**(3N), **routing**(4), **icmp**(7P), **if_tcp**(7P), **inet**(7P) **tcp**(7P), **udp**(7P)

Postel, Jon, *Internet Protocol - DARPA Internet Program Protocol Specification*, RFC 791, Network Information Center, SRI International, Menlo Park, Calif., September 1981.

DIAGNOSTICS

A socket operation may fail with one of the following errors returned:

EACCES A IP broadcast destination address was specified and the caller was not the privileged user.

EISCONN An attempt was made to establish a connection on a socket which already had one, or to send a datagram with the destination address specified and the socket was already connected.

EMSGSIZE An attempt was made to send a datagram that was too large for an interface, but was not allowed to be fragmented (such as broadcasts).

ENETUNREACH An attempt was made to establish a connection or send a datagram, where there was no matching entry in the routing table, or if an ICMP “destination unreachable” message was received.

ENOTCONN A datagrem was sent, but no destination address was specified, and the socket had not been connected.

ENOBUFS The system ran out of memory for fragmentation buffers or other internal data structure.

EADDRNOTAVAIL An attempt was made to create a socket with a local address that did not match any network interface, or an IP broadcast destination address was specified and the network interface does not support broadcast.

ERRORS

The following errors may occur when setting or getting IP options:

EINVAL An unknown socket option name was given.

EINVAL The IP option field was improperly formed; an option field was shorter than the minimum value or longer than the option buffer provided.

NOTES

Raw sockets should receive ICMP error packets relating to the protocol; currently such packets are simply discarded.

Users of higher-level protocols such as TCP and UDP should be able to see received IP options.