NAME

raw − Linux IPv4 raw sockets

SYNOPSIS

#include <sys/socket.h>
#include <netinet/in.h>
raw_socket = socket(AF_INET, SOCK_RAW, int *protocol***);**

DESCRIPTION

Raw sockets allow new IPv4 protocols to be implemented in user space. A raw socket receives or sends the raw datagram not including link level headers.

The IPv4 layer generates an IP header when sending a packet unless the **IP_HDRINCL** socket option is enabled on the socket. When it is enabled, the packet must contain an IP header. For receiving, the IP header is always included in the packet.

In order to create a raw socket, a process must have the **CAP_NET_RAW** capability in the user namespace that governs its network namespace.

All packets or errors matching the *protocol* number specified for the raw socket are passed to this socket. For a list of the allowed protocols, see the IANA list of assigned protocol numbers at ⟨http://www.iana.org/assignments/protocol−numbers/⟩ and **getprotobyname**(3).

A protocol of **IPPROTO_RAW** implies enabled **IP_HDRINCL** and is able to send any IP protocol that is specified in the passed header. Receiving of all IP protocols via **IPPROTO_RAW** is not possible using raw sockets.

IP Header fields modified on sending by **IP_HDRINCL**	
IP Checksum	Always filled in
Source Address	Filled in when zero
Packet ID	Filled in when zero
Total Length	Always filled in

If **IP_HDRINCL** is specified and the IP header has a nonzero destination address, then the destination address of the socket is used to route the packet. When **MSG_DONTROUTE** is specified, the destination address should refer to a local interface, otherwise a routing table lookup is done anyway but gatewayed routes are ignored.

If **IP_HDRINCL** isn't set, then IP header options can be set on raw sockets with **setsockopt**(2); see **ip**(7) for more information.

Starting with Linux 2.2, all IP header fields and options can be set using IP socket options. This means raw sockets are usually needed only for new protocols or protocols with no user interface (like ICMP).

When a packet is received, it is passed to any raw sockets which have been bound to its protocol before it is passed to other protocol handlers (e.g., kernel protocol modules).

Address format

For sending and receiving datagrams (**sendto**(2), **recvfrom**(2), and similar), raw sockets use the standard *sockaddr_in* address structure defined in **ip**(7). The *sin_port* field could be used to specify the IP protocol number, but it is ignored for sending in Linux 2.2 and later, and should be always set to 0 (see BUGS). For incoming packets, *sin_port* is set to zero.

Socket options

Raw socket options can be set with **setsockopt**(2) and read with **getsockopt**(2) by passing the **IPPROTO_RAW** family flag.

ICMP_FILTER

Enable a special filter for raw sockets bound to the **IPPROTO_ICMP** protocol. The value has a bit set for each ICMP message type which should be filtered out. The default is to filter no ICMP messages.

In addition, all **ip**(7) **IPPROTO_IP** socket options valid for datagram sockets are supported.

Error handling

Errors originating from the network are passed to the user only when the socket is connected or the **IP_RECVERR** flag is enabled. For connected sockets, only **EMSGSIZE** and **EPROTO** are passed for compatibility. With **IP_RECVERR**, all network errors are saved in the error queue.

ERRORS

EACCES

User tried to send to a broadcast address without having the broadcast flag set on the socket.

EFAULT

An invalid memory address was supplied.

EINVAL

Invalid argument.

EMSGSIZE

Packet too big. Either Path MTU Discovery is enabled (the **IP_MTU_DISCOVER** socket flag) or the packet size exceeds the maximum allowed IPv4 packet size of 64 kB.

EOPNOTSUPP

Invalid flag has been passed to a socket call (like **MSG_OOB**).

EPERM

The user doesn't have permission to open raw sockets. Only processes with an effective user ID of 0 or the **CAP_NET_RAW** attribute may do that.

EPROTO

An ICMP error has arrived reporting a parameter problem.

VERSIONS

IP_RECVERR and **ICMP_FILTER** are new in Linux 2.2. They are Linux extensions and should not be used in portable programs.

Linux 2.0 enabled some bug-to-bug compatibility with BSD in the raw socket code when the **SO_BSDCOMPAT** socket option was set; since Linux 2.2, this option no longer has that effect.

NOTES

By default, raw sockets do path MTU (Maximum Transmission Unit) discovery. This means the kernel will keep track of the MTU to a specific target IP address and return **EMSGSIZE** when a raw packet write exceeds it. When this happens, the application should decrease the packet size. Path MTU discovery can be also turned off using the **IP_MTU_DISCOVER** socket option or the */proc/sys/net/ipv4/ip_no_pmtu_disc* file, see **ip**(7) for details. When turned off, raw sockets will fragment outgoing packets that exceed the interface MTU. However, disabling it is not recommended for performance and reliability reasons.

A raw socket can be bound to a specific local address using the **bind**(2) call. If it isn't bound, all packets with the specified IP protocol are received. In addition, a raw socket can be bound to a specific network device using **SO_BINDTODEVICE**; see **socket**(7).

An **IPPROTO_RAW** socket is send only. If you really want to receive all IP packets, use a **packet**(7) socket with the **ETH_P_IP** protocol. Note that packet sockets don't reassemble IP fragments, unlike raw sockets.

If you want to receive all ICMP packets for a datagram socket, it is often better to use **IP_RECVERR** on that particular socket; see **ip**(7).

Raw sockets may tap all IP protocols in Linux, even protocols like ICMP or TCP which have a protocol module in the kernel. In this case, the packets are passed to both the kernel module and the raw socket(s). This should not be relied upon in portable programs, many other BSD socket implementation have limitations here.

Linux never changes headers passed from the user (except for filling in some zeroed fields as described for **IP_HDRINCL**). This differs from many other implementations of raw sockets.

Raw sockets are generally rather unportable and should be avoided in programs intended to be portable.

Sending on raw sockets should take the IP protocol from *sin_port*; this ability was lost in Linux 2.2. The workaround is to use **IP_HDRINCL**.

BUGS

Transparent proxy extensions are not described.

When the **IP_HDRINCL** option is set, datagrams will not be fragmented and are limited to the interface MTU.

Setting the IP protocol for sending in *sin_port* got lost in Linux 2.2. The protocol that the socket was bound to or that was specified in the initial **socket**(2) call is always used.

SEE ALSO

recvmsg(2), **sendmsg**(2), **capabilities**(7), **ip**(7), **socket**(7)

RFC 1191 for path MTU discovery. **RFC 791** and the <*linux/ip.h*> header file for the IP protocol.

COLOPHON

This page is part of release 5.05 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.