[**u32**](http://lxr.free-electrons.com/ident?i=u32)[**mtu**](http://lxr.free-electrons.com/ident?i=mtu)**;**

**struct** [**iphdr**](http://lxr.free-electrons.com/ident?i=iphdr) **\*iph; */\* Our header \*/***

**struct** [**rtable**](http://lxr.free-electrons.com/ident?i=rtable) **\*[rt](http://lxr.free-electrons.com/ident?i=rt); */\* Route we use \*/***

**struct** [**ip\_options**](http://lxr.free-electrons.com/ident?i=ip_options) **\*opt = &(**[**IPCB**](http://lxr.free-electrons.com/ident?i=IPCB)**([skb](http://lxr.free-electrons.com/ident?i=skb))->opt);**

In this firstly we are initalising 32 mtu (**maximum transmission unit** (**MTU**),which is the largest size packet or frame, specified in octets (eight-bit bytes), that can be sent in a packet- or frame-based **network** such as the Internet)

Structures iphdr and rtable deal with initalising structures of header and route we use in forwarding.

***/\* that should never happen \*/***

**if ([skb](http://lxr.free-electrons.com/ident?i=skb)->pkt\_type !=** [**PACKET\_HOST**](http://lxr.free-electrons.com/ident?i=PACKET_HOST)**)**

**goto** [**drop**](http://lxr.free-electrons.com/ident?i=drop)**;**

If packet type in skb is not equal to PACKET\_HOST (when an incoming ethernet frame is to a destination MAC address matching the MAC address of the ethernet device it arrived on, this field will be set to 'PACKET\_HOST') i.e like 'PACKET\_BROADCAST', 'PACKET\_MULTICAST' (we get diffferent packet types from 'linux/if\_packet.h' header file).

**if (**[**unlikely**](http://lxr.free-electrons.com/ident?i=unlikely)**([skb](http://lxr.free-electrons.com/ident?i=skb)->sk))**

**goto** [**drop**](http://lxr.free-electrons.com/ident?i=drop)**;**

In this unlikely is just a prefrocessor micro which takes condition and activate condition i.e (**#define** [**unlikely**](http://lxr.free-electrons.com/ident?i=unlikely)**([cond](http://lxr.free-electrons.com/ident?i=cond)) ([cond](http://lxr.free-electrons.com/ident?i=cond))).** So when there is error associated with socket with respect to skb (generally in memory allocation)then we need to drop it.

**if ([skb\_warn\_if\_lro](http://lxr.free-electrons.com/ident?i=skb_warn_if_lro)([skb](http://lxr.free-electrons.com/ident?i=skb)))**

**goto** [**drop**](http://lxr.free-electrons.com/ident?i=drop)**;**

In this function we send warning by passing true value in return if large receive offload(It works by aggregating multiple incoming [packets](https://en.wikipedia.org/wiki/Packet_(information_technology)) from a single [stream](https://en.wikipedia.org/wiki/Stream_(computing)) into a larger buffer before they are passed higher up the networking stack, thus reducing the number of packets that have to be processed.) sets a incompatible size(i.e gso\_size) and may also set wrong gso\_type(This used for drivers receiveing super-sized skb's. These are indicated to the driver by skb\_shinfo(skb)->gso\_size being non-zero. The gso\_size is the size the hardware should fragment the TCP data. TSO may change how and when TCP decides to send data.)

Drop:

void kfree\_skb(struct sk\_buff \*skb);

void dev\_kfree\_skb(struct sk\_buff \*skb);

void dev\_kfree\_skb\_irq(struct sk\_buff \*skb);

void dev\_kfree\_skb\_any(struct sk\_buff \*skb);

Free a buffer. The *kfree\_skb* call is used internally by the kernel. A driver should use one of the forms of *dev\_kfree\_skb* instead: *dev\_kfree\_skb* for noninterrupt context,*dev\_kfree\_skb\_irq* for interrupt context, or *dev\_kfree\_skb\_any* for code that can run in either context.

**if (!**[**xfrm4\_policy\_check**](http://lxr.free-electrons.com/ident?i=xfrm4_policy_check)**(**[**NULL**](http://lxr.free-electrons.com/ident?i=NULL)**, XFRM\_POLICY\_FWD,** [**skb**](http://lxr.free-electrons.com/ident?i=skb)**))**

**goto** [**drop**](http://lxr.free-electrons.com/ident?i=drop)**;**

This function returns a flag if network device condition is not properly set along with device destination .It will check inside calling another functions from net.h header.So here if not flag, We will drop it.

[**skb\_forward\_csum**](http://lxr.free-electrons.com/ident?i=skb_forward_csum)**([skb](http://lxr.free-electrons.com/ident?i=skb));**

Here we check skb->ip\_summed is CHECKSUM\_COMPLETE or not. Unfortunately we don’t support this one. So we change that value to CHECKSUM\_NONE

**if ([ip\_hdr](http://lxr.free-electrons.com/ident?i=ip_hdr)([skb](http://lxr.free-electrons.com/ident?i=skb))->[ttl](http://lxr.free-electrons.com/ident?i=ttl) <= 1)**

**goto too\_many\_hops;**

ip\_hdr(skb):

Takes skb as parameter and returns skb->head + skb->network\_header

If its ttl is less than or equal to 1,signifies maximum count of hops is attained and we are gonna drop the packet and send the sender an ICMP message.

[**rt**](http://lxr.free-electrons.com/ident?i=rt) **=** [**skb\_rtable**](http://lxr.free-electrons.com/ident?i=skb_rtable)**([skb](http://lxr.free-electrons.com/ident?i=skb));**

skb\_rtable which returns the routing table entry

**if (opt->is\_strictroute &&** [**rt**](http://lxr.free-electrons.com/ident?i=rt)**->rt\_uses\_gateway) goto sr\_failed;**

Strict routing implies that the entire set of SIP nodes which may be

visited is listed, in order of visitation, in the Route header. No other

nodes may be visited by this message, and all the listed nodes MUST be

visited in the given order or the message has "failed".

Loose routing implies that the indicated nodes MUST be visited before the

message can be delivered to the target indicated in the original request

URI. The message may visit other nodes before, between or after any node

specified on the loose route.

opt is pointer object to ip\_options

If opt is strict route and rt uses gateway which is not possible since strict route dosent allow gatewaying. The packet is dropped after sending ICMP message.

**iph =** [**ip\_hdr**](http://lxr.free-electrons.com/ident?i=ip_hdr)**([skb](http://lxr.free-electrons.com/ident?i=skb));**

**What ip\_hdr(skb) returns is**

**return (struct** [**iphdr**](http://lxr.free-electrons.com/ident?i=iphdr) **\*)[skb\_network\_header](http://lxr.free-electrons.com/ident?i=skb_network_header)([skb](http://lxr.free-electrons.com/ident?i=skb));**

**You are assigning this value to iph.**

**if ([skb\_cow](http://lxr.free-electrons.com/ident?i=skb_cow)([skb](http://lxr.free-electrons.com/ident?i=skb),** [**LL\_RESERVED\_SPACE**](http://lxr.free-electrons.com/ident?i=LL_RESERVED_SPACE)**([rt](http://lxr.free-electrons.com/ident?i=rt)->**[**dst**](http://lxr.free-electrons.com/ident?i=dst)**.**[**dev**](http://lxr.free-electrons.com/ident?i=dev)**)+[rt](http://lxr.free-electrons.com/ident?i=rt)- >[dst](http://lxr.free-electrons.com/ident?i=dst).**[**header\_len**](http://lxr.free-electrons.com/ident?i=header_len)**))**

**goto** [**drop**](http://lxr.free-electrons.com/ident?i=drop)**;**

**skb->cow:-**

copy header of skb when it is required.

**Arguments**

*skb*

buffer to cow

*headroom*

needed headroom

If the skb passed lacks sufficient headroom or its data part is shared, data is reallocated. If reallocation fails, an error is returned and original skb is not changed.The result is skb with writable area skb->head...skb->tail and at least *headroom* of space at head.

[**ip\_decrease\_ttl**](http://lxr.free-electrons.com/ident?i=ip_decrease_ttl)**(iph);**

Decrements the ttl after the skb\_cow is done.

**if (**[**IPCB**](http://lxr.free-electrons.com/ident?i=IPCB)**([skb](http://lxr.free-electrons.com/ident?i=skb))->**[**flags**](http://lxr.free-electrons.com/ident?i=flags) **&** [**IPSKB\_DOREDIRECT**](http://lxr.free-electrons.com/ident?i=IPSKB_DOREDIRECT) **&& !opt->[srr](http://lxr.free-electrons.com/ident?i=srr) &&**

**![skb\_sec\_path](http://lxr.free-electrons.com/ident?i=skb_sec_path)([skb](http://lxr.free-electrons.com/ident?i=skb)))**

IPSKB\_DOREDIRECT:-

**#define** [**IPSKB\_DOREDIRECT**](http://lxr.free-electrons.com/ident?i=IPSKB_DOREDIRECT)[**BIT**](http://lxr.free-electrons.com/ident?i=BIT)**(5)**

**sr\_failed:**

[**icmp\_send**](http://lxr.free-electrons.com/ident?i=icmp_send)**([skb](http://lxr.free-electrons.com/ident?i=skb),** [**ICMP\_DEST\_UNREACH**](http://lxr.free-electrons.com/ident?i=ICMP_DEST_UNREACH)**,** [**ICMP\_SR\_FAILED**](http://lxr.free-electrons.com/ident?i=ICMP_SR_FAILED)**, 0);**

**goto** [**drop**](http://lxr.free-electrons.com/ident?i=drop)**;**

Strict routing permits no gateway

So send an icmp message with destination unreachable and source failed set.

[**IPCB**](http://lxr.free-electrons.com/ident?i=IPCB)**([skb](http://lxr.free-electrons.com/ident?i=skb))->**[**flags**](http://lxr.free-electrons.com/ident?i=flags) **|=** [**IPSKB\_FORWARDED**](http://lxr.free-electrons.com/ident?i=IPSKB_FORWARDED)**;**

IPSKB\_FORWARDED:

Preprocessor macro

**#define** [**IPSKB\_FORWARDED**](http://lxr.free-electrons.com/ident?i=IPSKB_FORWARDED)[**BIT**](http://lxr.free-electrons.com/ident?i=BIT)**(0)**

**return** [**NF\_HOOK**](http://lxr.free-electrons.com/ident?i=NF_HOOK)**(NFPROTO\_IPV4, NF\_INET\_FORWARD,** [**NULL**](http://lxr.free-electrons.com/ident?i=NULL)**,** [**skb**](http://lxr.free-electrons.com/ident?i=skb)**,** [**skb**](http://lxr.free-electrons.com/ident?i=skb)**->**[**dev**](http://lxr.free-electrons.com/ident?i=dev)**,** [**rt**](http://lxr.free-electrons.com/ident?i=rt)**->**[**dst**](http://lxr.free-electrons.com/ident?i=dst)**.**[**dev**](http://lxr.free-electrons.com/ident?i=dev)**,** [**ip\_forward\_finish**](http://lxr.free-electrons.com/ident?i=ip_forward_finish)**);**

NF\_HOOK:

* NF\_HOOK macro:
  + NF\_HOOK(pf, hook, skb, indev, outdev, okfn)
* For each registered module under the hook nf\_hooks[pf][hook]
  + Pass the packet to the module and wait for the "verdict"
  + If verdict is drop, drop the packet
  + If verdict is okay, continue next module
* If final verdict is okay, call okfn(skb)