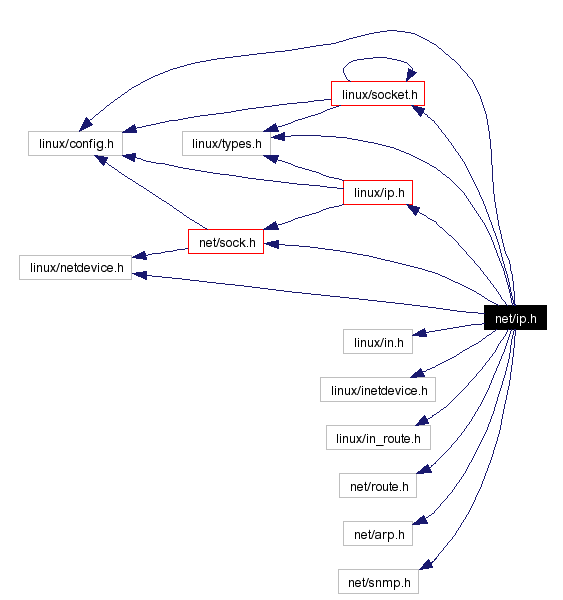
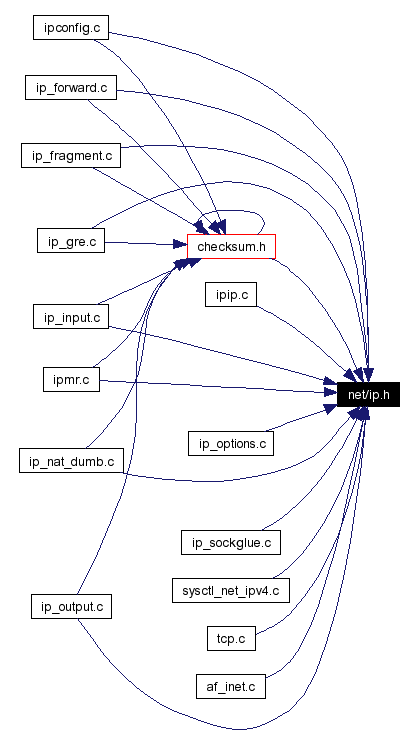
Ip.h important header which is derived from net and deals with inturn connected headers like sock.h,socket.h,lenux/ip.h as shown in figure:



Added to this there are many files directly or indirectly include this file as shown below:



So our file ip\_forward.c is connected directly and internally (via checksum.c) to net/ip.h

Some of important structures which effect ip\_forward.c from ip.h are:

Ip\_options

Ip\_hdr

And functions are:

ip\_call\_ra\_chain

ip\_rt\_send\_redirect

ip\_hdr

ip\_decrease\_ttl

ip\_exceeds\_mtu

ip\_dst\_mtu\_maybe\_forward

**ip\_options:**

struct [ip\_options](http://www.cse.scu.edu/~dclark/am_256_graph_theory/linux_2_6_stack/structip__options.html) {

\_\_u32 faddr; /\* Saved first hop address \*/

unsigned char optlen;

unsigned char srr;

unsigned char rr;

unsigned char ts;

unsigned char is\_setbyuser:1, /\* Set by setsockopt? \*/

is\_data:1, /\* Options in \_\_data, rather than skb \*/

is\_strictroute:1, /\* Strict source route \*/

srr\_is\_hit:1, /\* Packet destination addr was our one \*/

is\_changed:1, /\* IP checksum more not valid \*/

rr\_needaddr:1, /\* Need to record addr of outgoing dev \*/

ts\_needtime:1, /\* Need to record timestamp \*/

ts\_needaddr:1; /\* Need to record addr of outgoing dev \*/

unsigned char router\_alert;

unsigned char \_\_pad1;

unsigned char \_\_pad2;

unsigned char \_\_data[0];

};

“ip\_options” is the structure that holds different ip parameters and options that have to be initialized in order to define the proper ip format for the packets.

Set or get the IP options to be sent with every packet from the socket. The arguments are a pointer to a memory buffer containing the options and the option length. The [**setsockopt**](http://linux.die.net/man/2/setsockopt) call sets the IP options associated with a socket. The maximum option size for IPv4 is 40 bytes. When the initial connection request packet for a **SOCK\_STREAM** socket contains IP options, the IP options will be set automatically to the options from the initial packet with routing headers reversed. Incoming packets are not allowed to change options after the connection is established. The processing of all incoming source routing options is disabled by default and can be enabled by using the *accept\_source\_route /proc* interface. Other options like timestamps are still handled. For datagram sockets, IP options can be only set by the local user. Calling [**getsockopt**](http://linux.die.net/man/2/getsockopt) with **IP\_OPTIONS** puts the current IP options used for sending into the supplied buffer.

**struct** [**iphdr**](http://lxr.free-electrons.com/ident?i=iphdr) **{**

**#if defined([\_\_LITTLE\_ENDIAN\_BITFIELD](http://lxr.free-electrons.com/ident?i=__LITTLE_ENDIAN_BITFIELD))**

[**\_\_u8**](http://lxr.free-electrons.com/ident?i=__u8) **ihl:4,**

[**version**](http://lxr.free-electrons.com/ident?i=version)**:4;**

**#elif defined (**[**\_\_BIG\_ENDIAN\_BITFIELD**](http://lxr.free-electrons.com/ident?i=__BIG_ENDIAN_BITFIELD)**)**

[**\_\_u8**](http://lxr.free-electrons.com/ident?i=__u8)[**version**](http://lxr.free-electrons.com/ident?i=version)**:4,**

**ihl:4;**

**#else**

**#error *"Please fix <asm/byteorder.h>"***

**#endif**

[**\_\_u8**](http://lxr.free-electrons.com/ident?i=__u8) **tos;**

[**\_\_be16**](http://lxr.free-electrons.com/ident?i=__be16) **tot\_len;**

[**\_\_be16**](http://lxr.free-electrons.com/ident?i=__be16)[**id**](http://lxr.free-electrons.com/ident?i=id)**;**

[**\_\_be16**](http://lxr.free-electrons.com/ident?i=__be16) **frag\_off;**

[**\_\_u8**](http://lxr.free-electrons.com/ident?i=__u8)[**ttl**](http://lxr.free-electrons.com/ident?i=ttl)**;**

[**\_\_u8**](http://lxr.free-electrons.com/ident?i=__u8)[**protocol**](http://lxr.free-electrons.com/ident?i=protocol)**;**

[**\_\_sum16**](http://lxr.free-electrons.com/ident?i=__sum16)[**check**](http://lxr.free-electrons.com/ident?i=check)**;**

[**\_\_be32**](http://lxr.free-electrons.com/ident?i=__be32)[**saddr**](http://lxr.free-electrons.com/ident?i=saddr)**;**

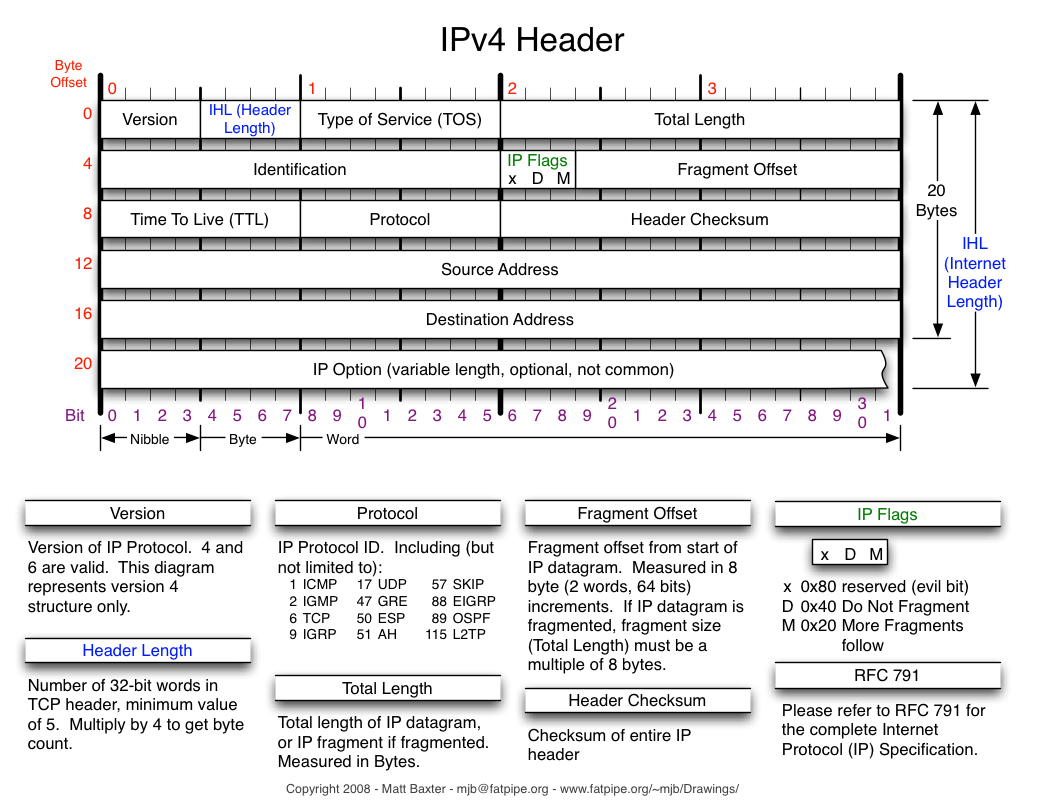
[**\_\_be32**](http://lxr.free-electrons.com/ident?i=__be32)[**daddr**](http://lxr.free-electrons.com/ident?i=daddr)**;**

***/\*The options start here. \*/***

**};**

So this structure defines all the components of structure one header of ip would have.

All the parameter are set to packet derived from it and used all along the trasmission to decode required information.



**Functions:**

ip\_call\_ra\_chain

If the router \_alert option is set , the ip\_call\_ra\_chain() method should be invoked to handle packet.When calling setsocket() with IP\_ROUTER\_Alert on raw socket, socket is added to global list named ip\_ra\_chain(net/ip.h) .This function delivers packets to all raw sockets.

Forwarding

**int ip\_forward(struct sk\_buff \*skb** )

* Decreases the ttl in the ip header; If the ttl is <=1, the methods send ICMP message (ICMP\_TIME\_EXCEEDED) with ICMP\_EXC\_TTL ("TTL count exceeded"), and drops the packet.
* Calls NF\_HOOK(PF\_INET,NF\_IP\_FORWARD, skb, skb->dev,rt- >u.dst.dev, ip\_forward\_finish);

ip\_forward\_finish()

sends the packet out by calling dst\_output(skb).

* dst\_output(skb) is just a wrapper, which calls skb->dst->output(skb).

**Sending the packet:**

We need to perform routing lookup also in the case of transmission.

* There are cases when we perform two lookups, like in ipip tunnels.
* Handling of sending a packet is done by ip\_route\_output\_key().
* In case of a cache miss, we calls ip\_route\_output\_slow(), which looks in the routing table (by calling fib\_lookup(), as also is done in ip\_route\_input\_slow().)
* If the packet is for a remote host, we set dst->output to ip\_output()
* ip\_output() will call ip\_finish\_output() – This is the NF\_IP\_POST\_ROUTING point.

**ip\_finish\_output()** will eventually send the packet from a neighbor by: – dst->neighbour->output(skb) – arp\_bind\_neighbour() sees to it that the L2 address of the next hop will be known.

**ip\_decrease\_ttl()**

Each node that forwards the packet should decrese the ttl andas a result of ttl change , checksum is also updated accordingly in the ip\_decrease\_ttl() method.