FUNCTION :

static int ip\_forward\_finish(struct sock \*sk, struct sk\_buff \*skb)

{

struct ip\_options \*opt = &(IPCB(skb)->opt);

IP\_INC\_STATS\_BH(dev\_net(skb\_dst(skb)->dev), IPSTATS\_MIB\_OUTFORWDATAGRAMS);

IP\_ADD\_STATS\_BH(dev\_net(skb\_dst(skb)->dev), IPSTATS\_MIB\_OUTOCTETS, skb->len);

if (unlikely(opt->optlen))

ip\_forward\_options(skb);

skb\_sender\_cpu\_clear(skb);

return dst\_output\_sk(sk, skb);

}

1. static int ip\_forward\_finish(struct sock \*sk, struct sk\_buff \*skb);

This function is responsible to complete the routing table once it reaches every router. It accepts the arguments :

1. struct sock \*sk : a structure object(pointer) of structure “sock” which is responsible for the data variables managing the socket creation, use and close.
2. struct sk\_buff \*skb : a structure object(pointer) of structure “sk\_buff” which defines a packet structure in the network.

1. struct ip\_options \*opt = &(IPCB(skb)->opt);
2. Here the object of the structure “ip\_options” initializes itself to the address of the data variable “opt” corresponding to the structure “IPCB”.

IPCB is defined as a macro in the header file “net/ip.h”

#define [IPCB](http://lxr.free-electrons.com/ident?i=IPCB)([skb](http://lxr.free-electrons.com/ident?i=skb)) ((struct [inet\_skb\_parm](http://lxr.free-electrons.com/ident?i=inet_skb_parm)\*)(([skb](http://lxr.free-electrons.com/ident?i=skb))->[cb](http://lxr.free-electrons.com/ident?i=cb)))

1. “inet\_skb\_parm” is the structure that holds the details of the flags, ip options and the packet size.

struct [inet\_skb\_parm](http://lxr.free-electrons.com/ident?i=inet_skb_parm)

{

struct ip\_options opt;

unsigned char flags;

#define IPSKB\_FORWARDED BIT(0)

#define IPSKB\_XFRM\_TUNNEL\_SIZE BIT(1)

#define IPSKB\_XFRM\_TRANSFORMED BIT(2)

#define IPSKB\_FRAG\_COMPLETE BIT(3)

#define IPSKB\_REROUTED BIT(4)

#define IPSKB\_DOREDIRECT BIT(5)

#define IPSKB\_FRAG\_PMTU BIT(6)

u16 frag\_max\_size;

};

1. “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.

struct [ip\_options](http://lxr.free-electrons.com/ident?i=ip_options)

{

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

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

unsigned char [optlen](http://lxr.free-electrons.com/ident?i=optlen);

unsigned char [srr](http://lxr.free-electrons.com/ident?i=srr);

unsigned char rr;

unsigned char [ts](http://lxr.free-electrons.com/ident?i=ts);

unsigned char is\_strictroute:1,

srr\_is\_hit:1,

is\_changed:1,

rr\_needaddr:1,

ts\_needtime:1,

ts\_needaddr:1;

unsigned char router\_alert;

unsigned char cipso;

unsigned char \_\_pad2;

unsigned char \_\_data[0];

};

1. IP\_INC\_STATS\_BH(dev\_net(skb\_dst(skb)->dev), IPSTATS\_MIB\_OUTFORWDATAGRAMS);

IP\_INC\_STATS\_BH() is a macro defined in net/ip.h header file and is used to determine amount of padding added to the ethernet frame.

1. IP\_ADD\_STATS\_BH(dev\_net(skb\_dst(skb)->dev), IPSTATS\_MIB\_OUTOCTETS, skb->len);

After determining the amount of padding to be added to the ethernet frame, now this macro adds the determined padding to the packet.

1. if (unlikely(opt->optlen))

ip\_forward\_options(skb);

Here the unlikely() function checks the error range of the given parameter and if that is true, calls the function ip\_forward\_options(skb).

1. ip\_forward\_options(skb)-

void ip\_forward\_options(struct sk\_buff \*skb)

{

struct ip\_options \*opt = &(IPCB(skb)->opt);

unsigned char \*optptr;

struct rtable \*rt = skb\_rtable(skb);

unsigned char \*raw = skb\_network\_header(skb);

if (opt->rr\_needaddr) {

optptr = (unsigned char \*)raw + opt->rr;

ip\_rt\_get\_source(&optptr[optptr[2]-5], skb, rt);

opt->is\_changed = 1;

}

if (opt->srr\_is\_hit) {

int srrptr, srrspace;

optptr = raw + opt->srr;

for ( srrptr = optptr[2], srrspace = optptr[1];

srrptr <= srrspace;

srrptr += 4

) {

if (srrptr + 3 > srrspace)

break;

if (memcmp(&opt->nexthop, &optptr[srrptr-1], 4) == 0)

break;

}

if (srrptr + 3 <= srrspace) {

opt->is\_changed = 1;

ip\_hdr(skb)->daddr = opt->nexthop;

ip\_rt\_get\_source(&optptr[srrptr-1], skb, rt);

optptr[2] = srrptr+4;

} else {

net\_crit\_ratelimited(*"%s(): Argh! Destination lost!\n"*,

\_\_func\_\_);

}

if (opt->ts\_needaddr) {

optptr = raw + opt->ts;

ip\_rt\_get\_source(&optptr[optptr[2]-9], skb, rt);

opt->is\_changed = 1;

}

}

if (opt->is\_changed) {

opt->is\_changed = 0;

ip\_send\_check(ip\_hdr(skb));

}

}

This function is responsible for packet forwarding with the help of routing table but after the hop count of 4, it stops the packet forwarding.