Most SKBs are stored on a list, whose head is implemented by 'struct sk\_buff\_head':

I am just providing just general struct code of it:-

struct sk\_buff\_head {

/\* These two members must be first. \*/

struct sk\_buff \*next;

struct sk\_buff \*prev;

\_\_u32 qlen;

spinlock\_t lock;

};

The first two members implement the doubly linked list handling. The third member, 'qlen', keeps track of how many packets are on this list. The fourth member, 'lock', is used for SMP protection.Here SMP i.e. **Shared Mesh Protection is used** as a **means** of increasing the **network** resiliency without incurring failures.

There are many operations one can perform on SKBs wrt. List tracking.

Here are the primary interfaces used in the code:

int skb\_queue\_empty(const struct sk\_buff\_head \*list);

This function will check that whether the given list is empty?

struct sk\_buff \*skb\_peek(struct sk\_buff\_head \*list\_);

Return, but do not remove, the first SKB on the list, else NULL. We should only use this function on an SKB list we have exclusive access to( I mean our own private acess), else a thread in another context could remove the SKB this function returns.

struct sk\_buff \*skb\_peek\_tail(struct sk\_buff\_head \*list\_);

It is almost Exactly like skb\_peek(), except it returns the last SKB on the list.

\_\_u32 skb\_queue\_len(const struct sk\_buff\_head \*list\_);

It Returns the number of packets on the given list.

void skb\_queue\_head\_init(struct sk\_buff\_head \*list);

Initialize an sk\_buff\_head object. If we dynamically allocate a data structure with an sk\_buff\_head member in it, we should pass it to this function before trying to use it.