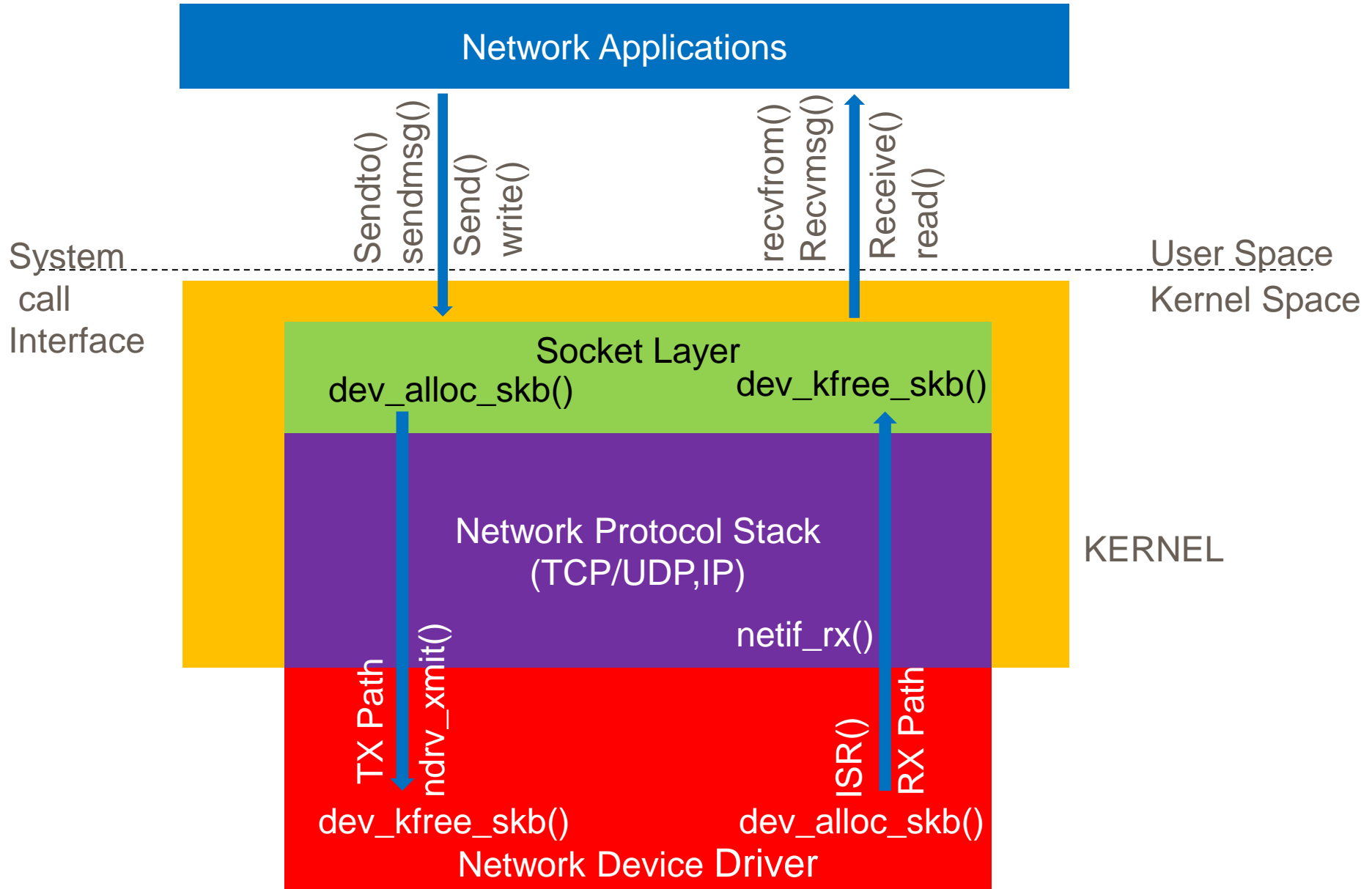# Linux Network Device Driver

Linux Network Device Drivers - 2

By

Jitesh Verma

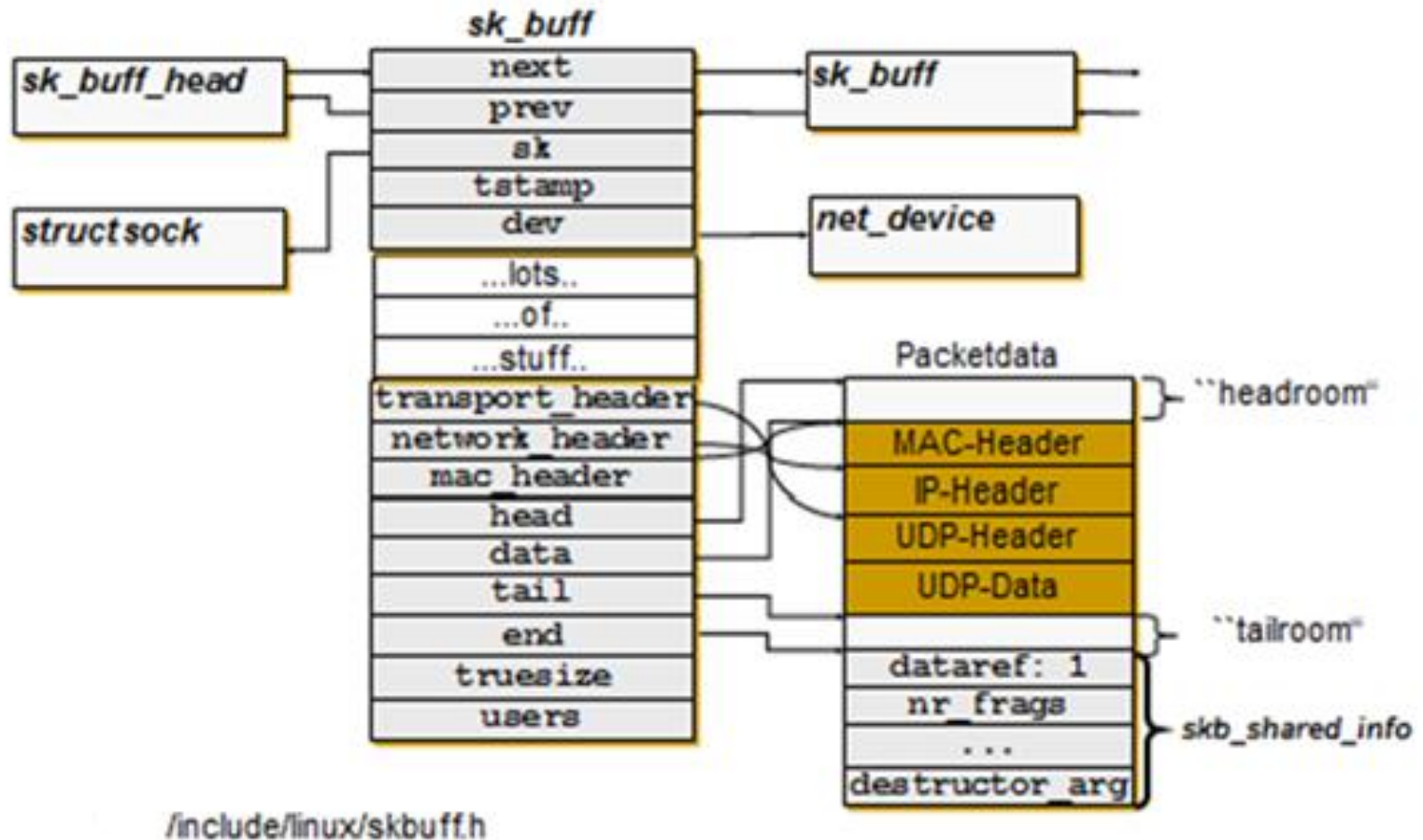# Application – Network Device Driver Interaction
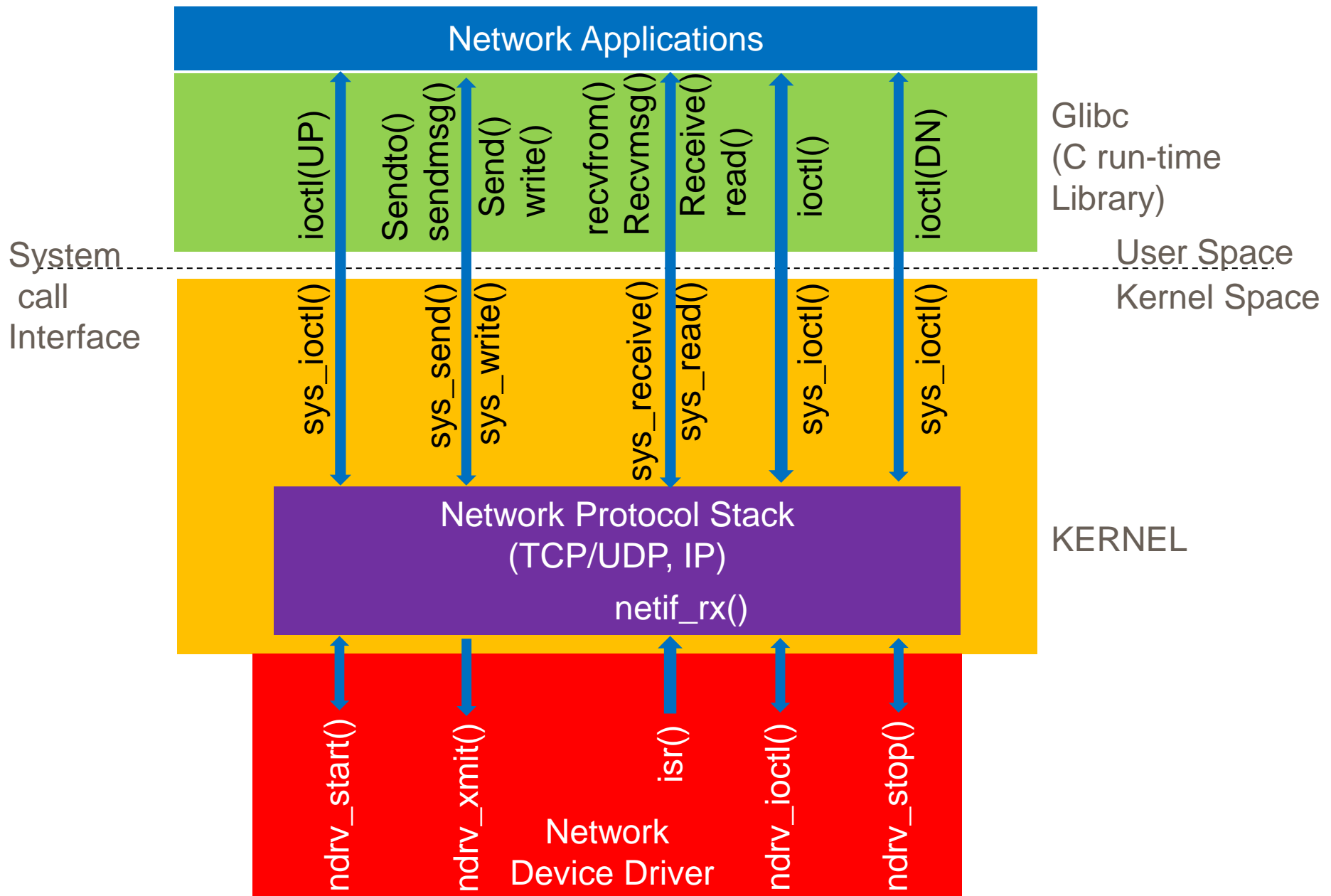
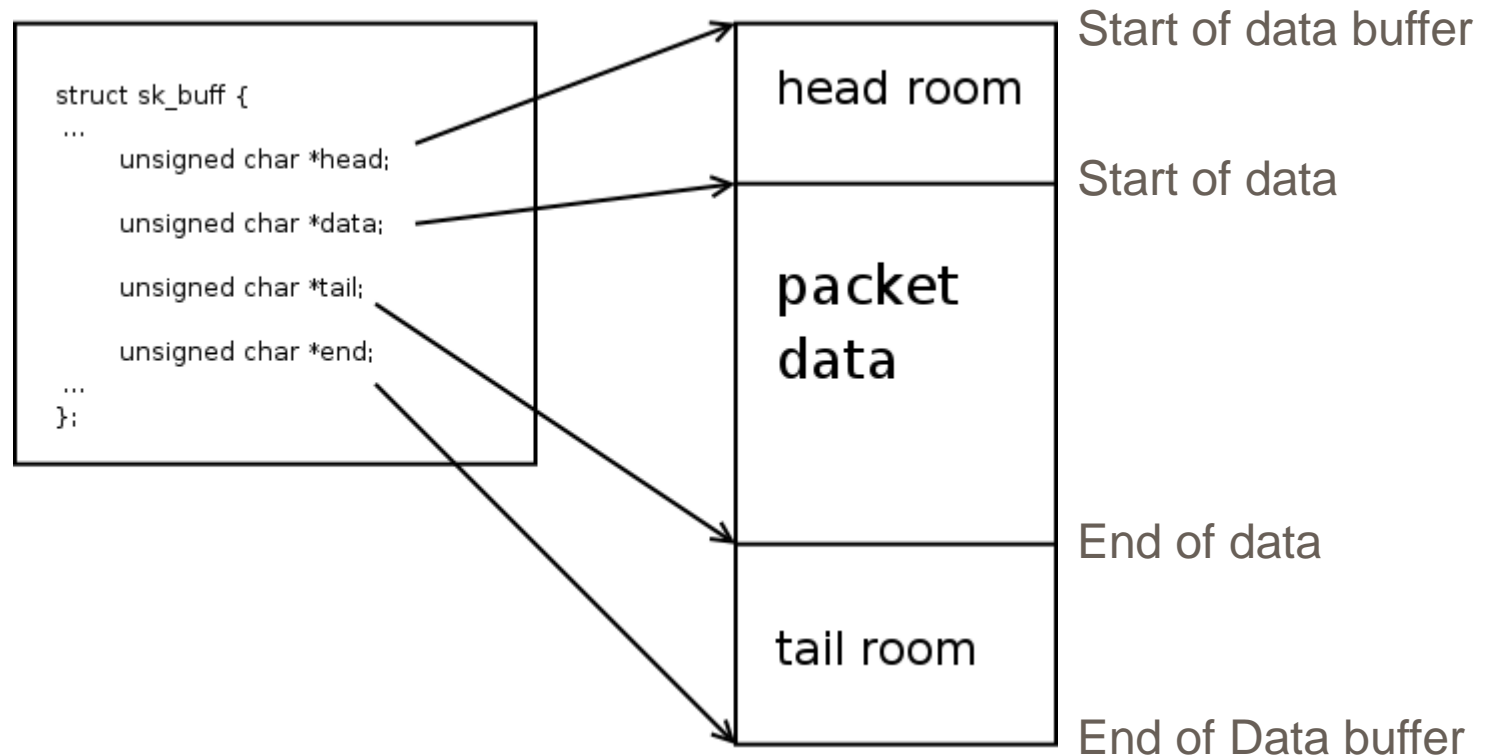# Life-Cycle/Flow of Socket Buffer (SK_BUFF)

struct sk_buff – Detailed View

/include/linux/skbuff.h

struct sk_buff – Simplified View

HEADER FILE: linux/skbuff.h

```
struct sk_buff {
...
    unsigned char *head;

    unsigned char *data;

    unsigned char *tail;

    unsigned char *end;
...
};
```

Start of data buffer

head room

Start of data
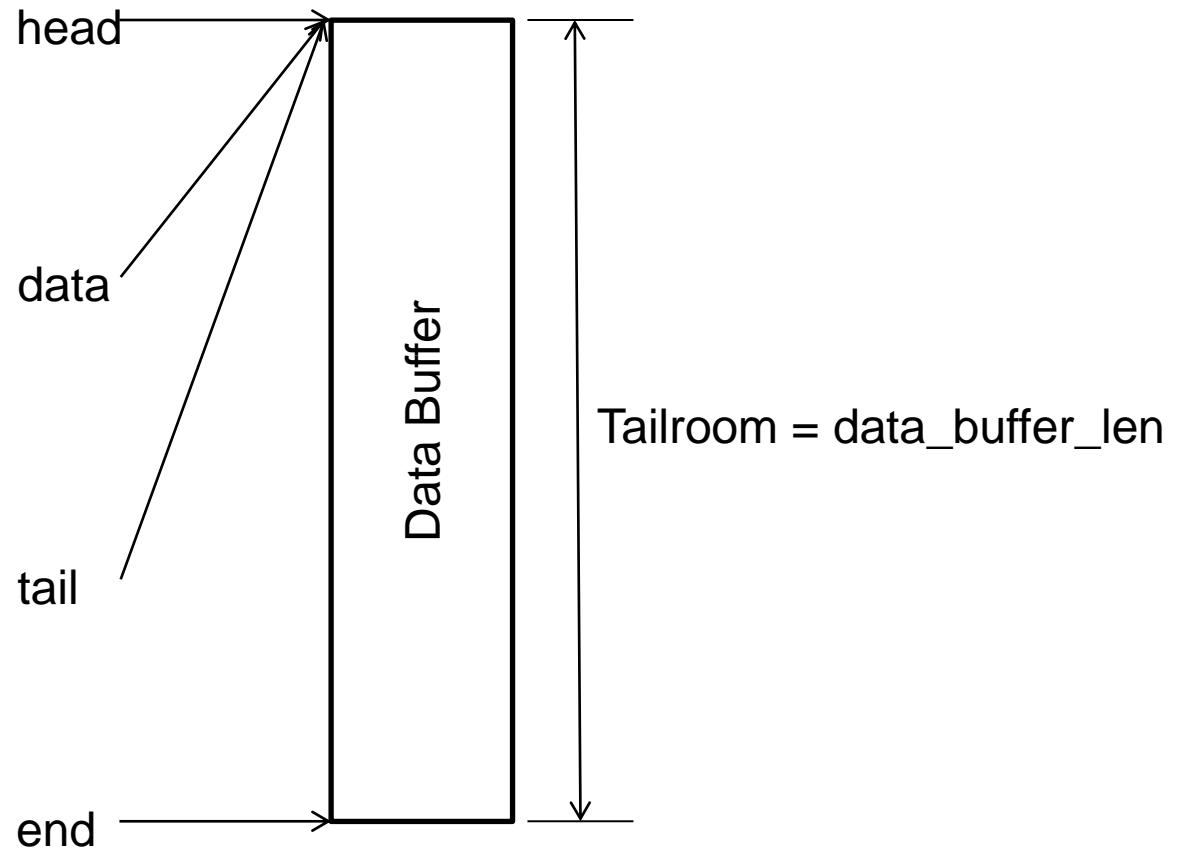
packet
data

End of data

tail room

End of Data buffer

struct sk_buff & its Data Area (Buffer)

TX Processing: sk_buff view after SKB allocation

OPERATION:
struct sk_buff * alloc_skb (int data_buffer_len, GFP_KERNEL)
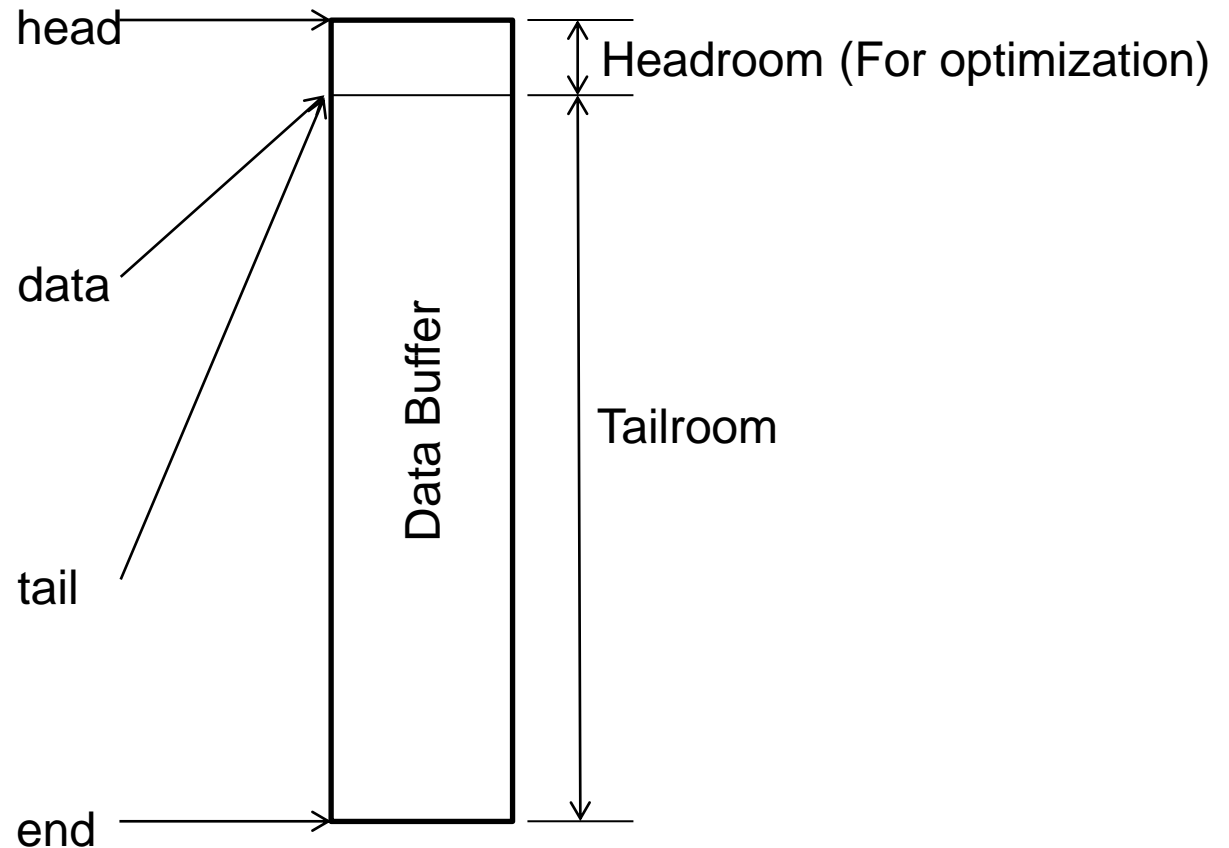
sk_buff->len = 0, Headroom = 0

head

data

Data Buffer

Tailroom = data_buffer_len

tail

end

OPERATION:
struct sk_buff * dev_alloc_skb (int data_buffer_len)

sk_buff->len = 0, data_buffer_len = Headroom + Tailroom

OPERATION:
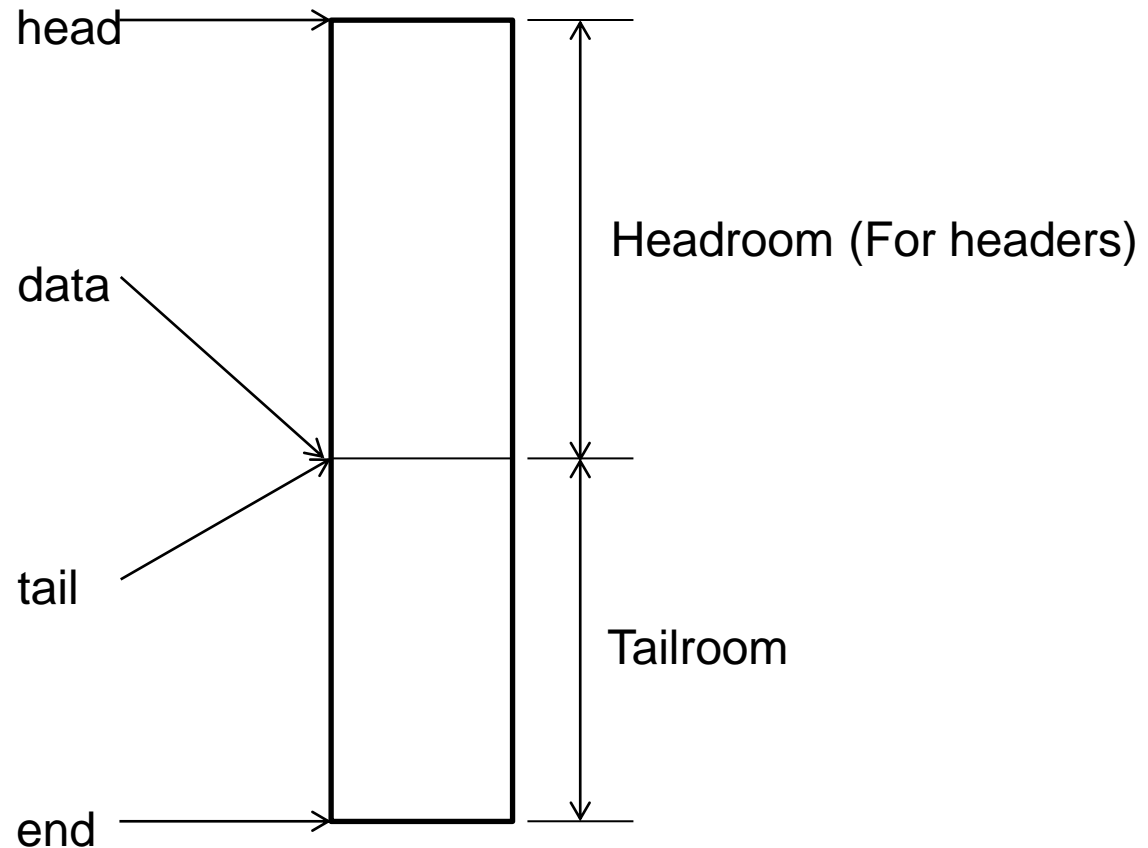void skb_reserve (struct sk_buff *skb, int Total_header_len)

sk_buff->len = 0, Total_header_len = TCP/UDP hdr len + IP hdr len + MAC hdr len
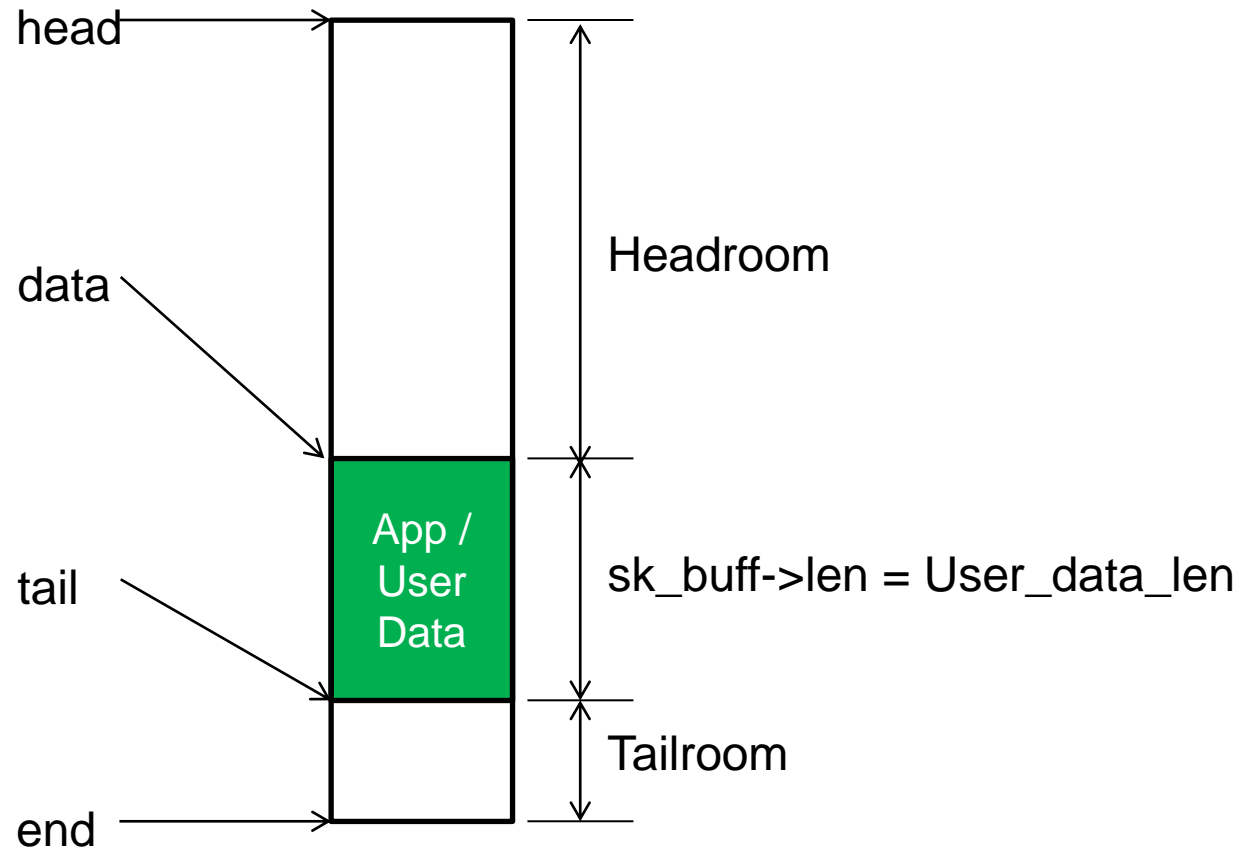
head

Headroom (For headers)

data

tail

Tailroom

end

TX Processing: sk_buff view after adding App/User Data

OPERATION:
void * skb_put (struct sk_buff *skb, int User_data_len)

Returns pointer to user data.

OPERATION:

void * skb_push (struct sk_buff *skb, int TCP/UDP_header_len)

Returns pointer to TCP/UDP header (struct tcphdr * / struct udphdr *)

head → [ ] ⟵ Headroom

data → [TCP / UDP Header]

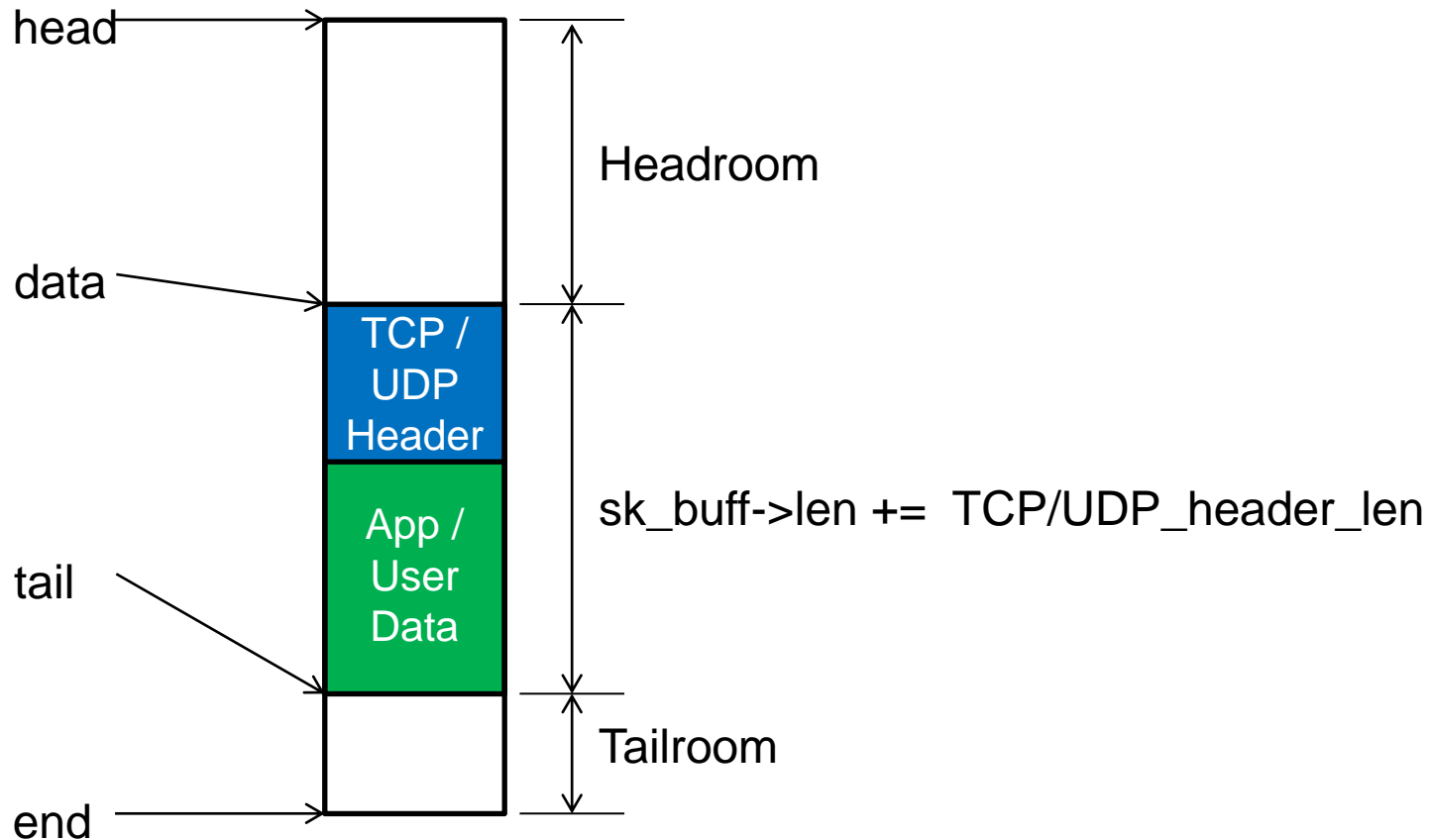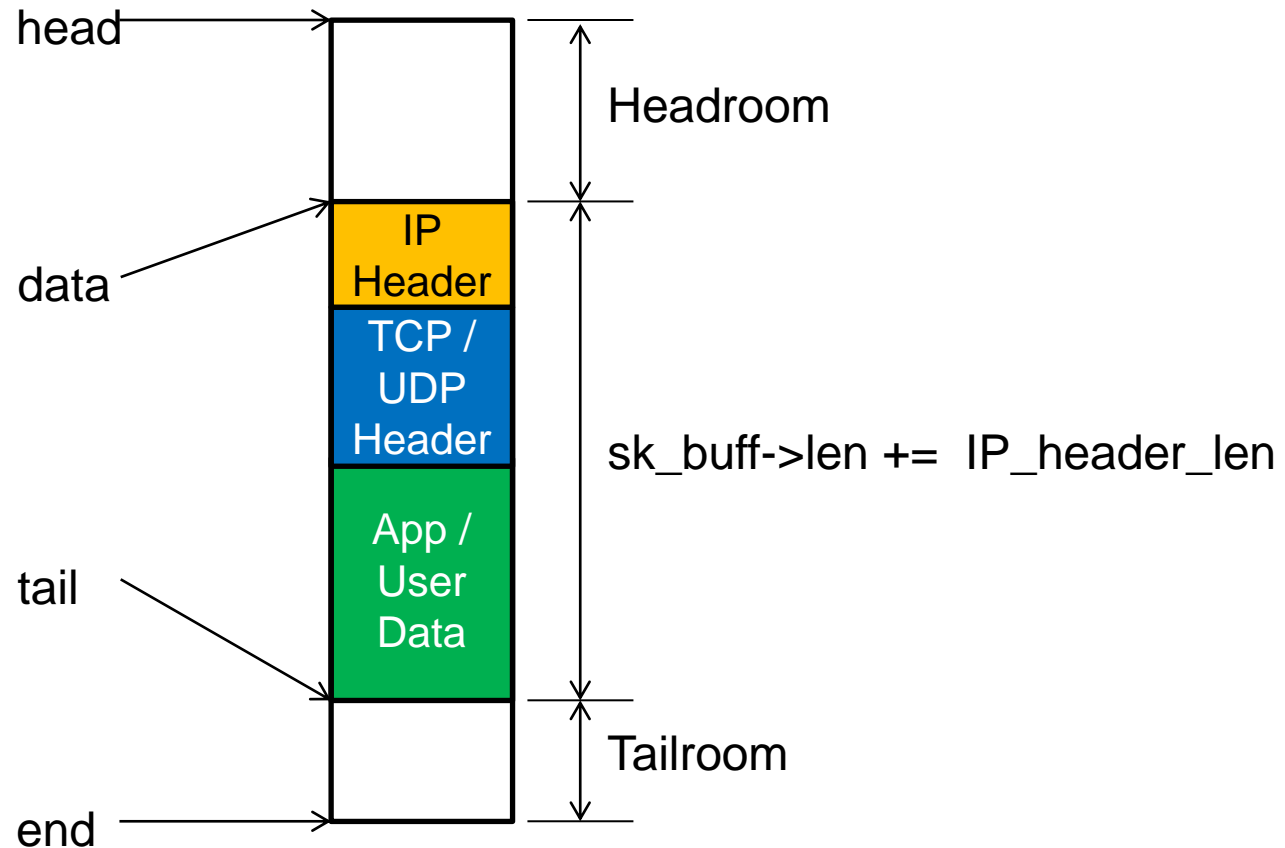[App / User Data] ← sk_buff->len += TCP/UDP_header_len

tail →

end → [ ] Tailroom

TX Processing: sk_buff view after adding IP Header

OPERATION:
void * skb_push (struct sk_buff *skb, int IP_header_len)

Returns pointer to IP header (struct iphdr *)

head → [ ] ⎤
                Headroom
data → [IP Header]

[TCP / UDP Header]

sk_buff->len += IP_header_len
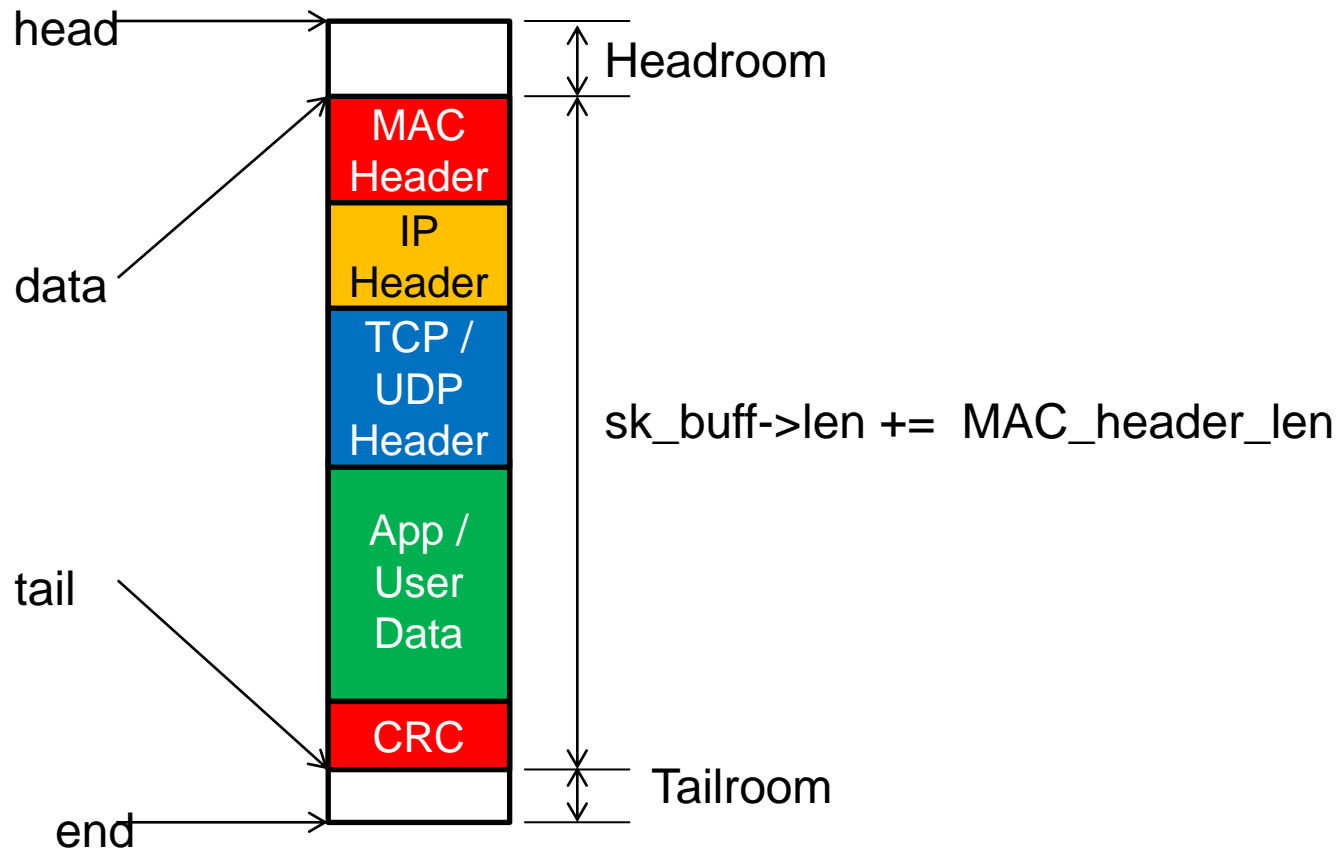
[App / User Data]
tail →

[ ] ⎤
        Tailroom
end →

TX Processing: sk_buff view after adding MAC Header

OPERATION:
void * skb_push (struct sk_buff *skb, int MAC_header_len)

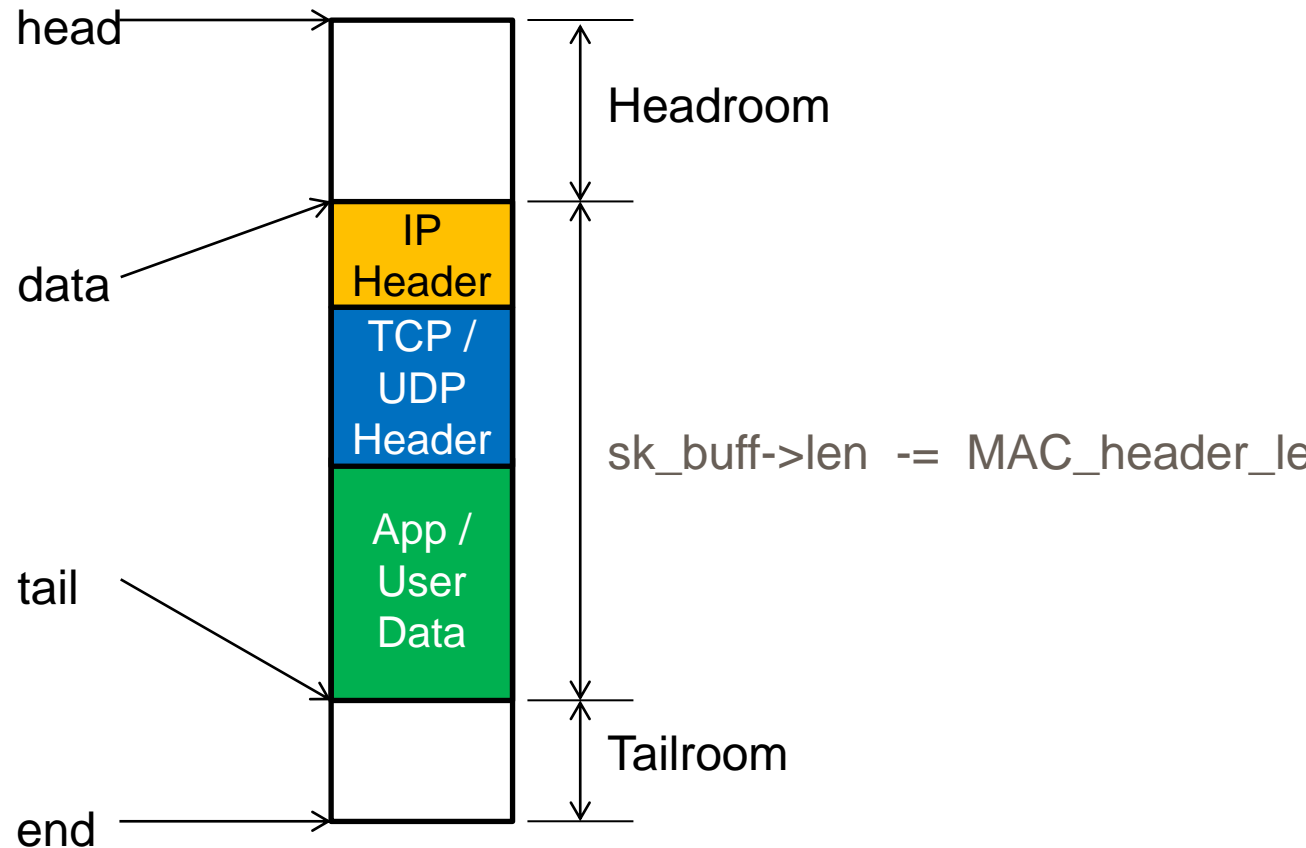Returns pointer to MAC header (struct machdr *)

sk_buff->len += MAC_header_len

NOTE: Separate operation is required for adding CRC to Tailroom.

RX Processing: sk_buff view after removing MAC header

OPERATION:
void * skb_pull (struct sk_buff *skb, int MAC_header_len)

Returns pointer to IP header (struct iphdr *)

head → Headroom

data → IP Header

TCP / UDP Header

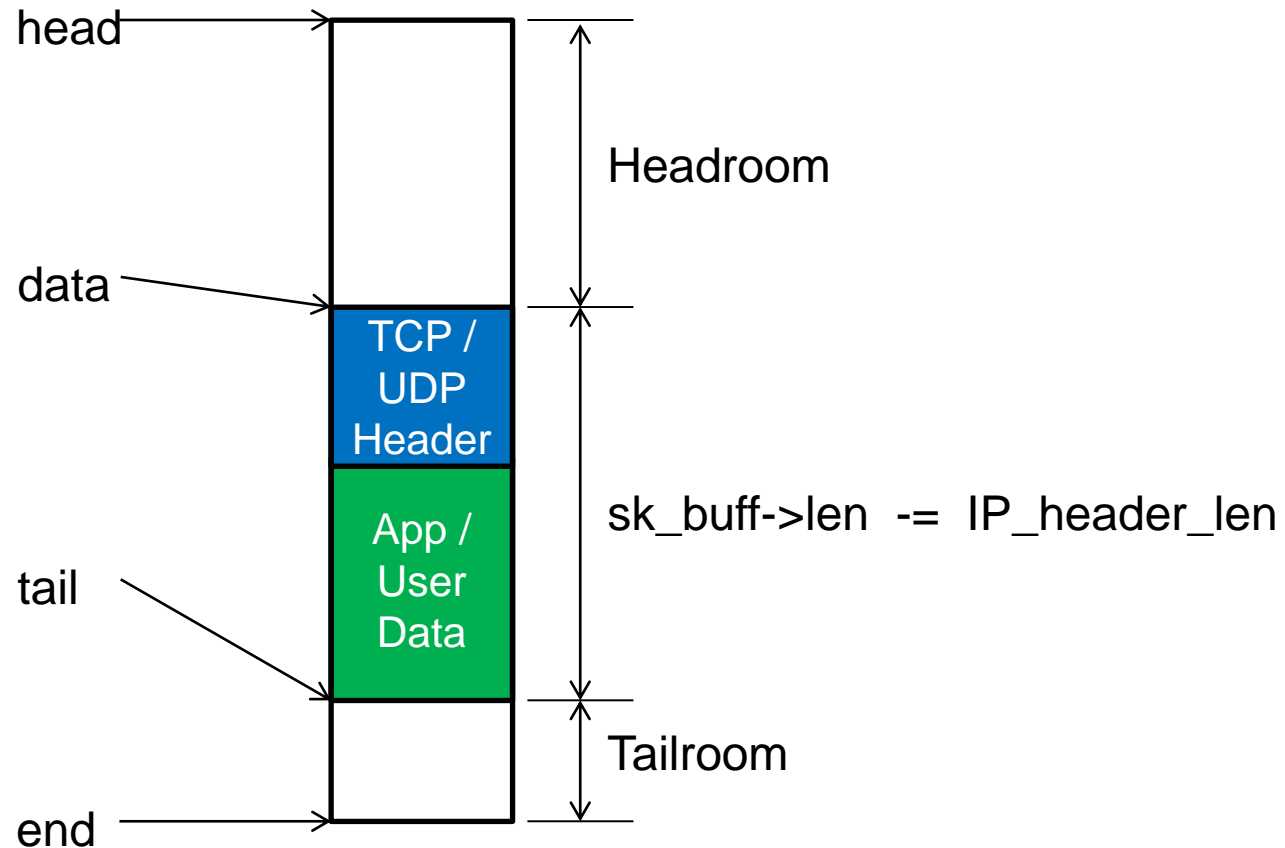sk_buff->len -= MAC_header_le

App / User Data

tail →

Tailroom

end →

NOTE: Separate operation is required for removing CRC from Tailroom.

RX Processing: sk_buff view after removing IP header

OPERATION:
void * skb_pull (struct sk_buff *skb, int IP_header_len)

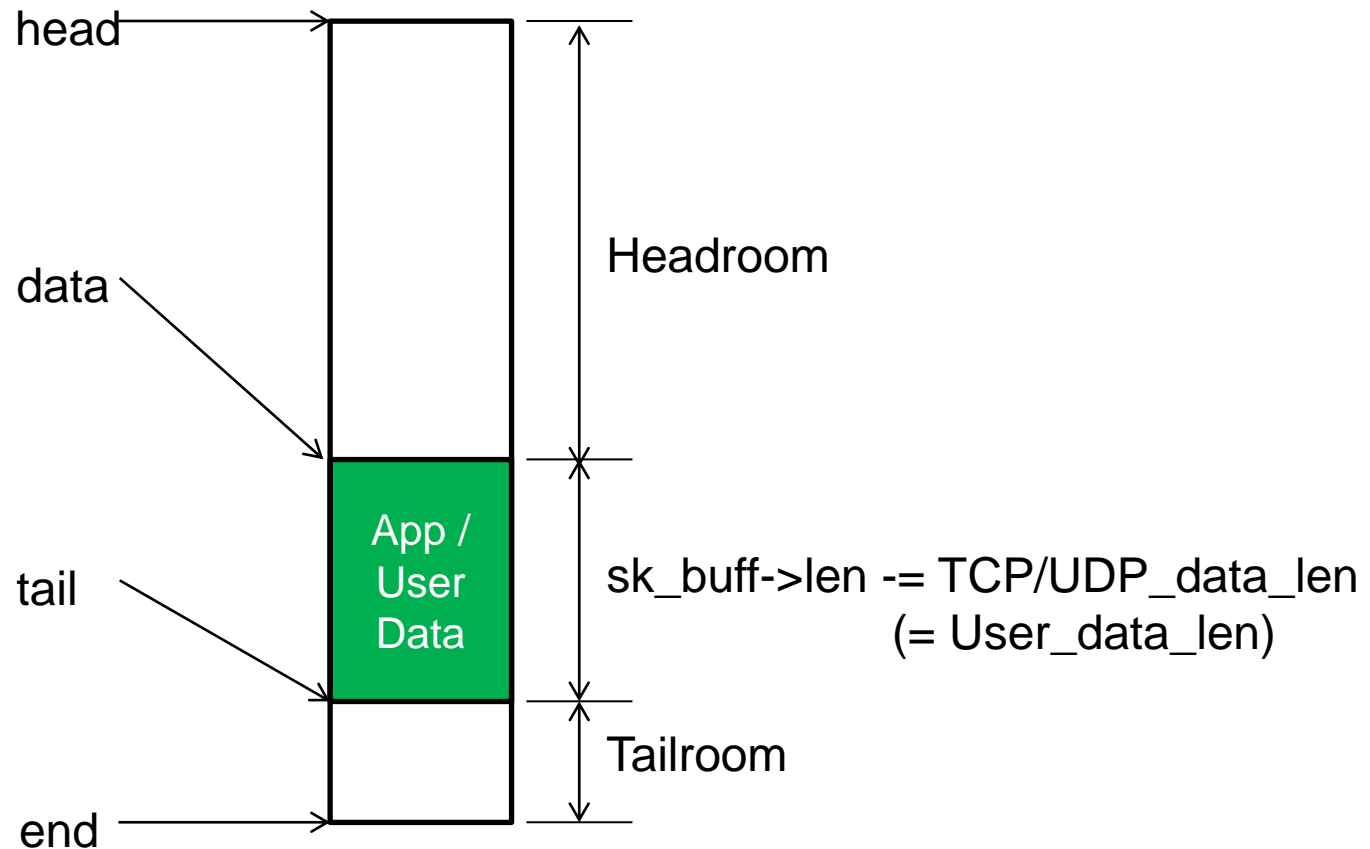Returns pointer to TCP/UDP header (struct tcphdr * / struct udphdr *)

head

Headroom

data

TCP /
UDP
Header

App /
User
Data

sk_buff->len -= IP_header_len

tail

Tailroom

end

Aricent

OPERATION:
void * skb_pull (struct sk_buff *skb, int TCP/UDP_header_len)

Returns pointer to user data.

head

data

App /
User
Data

tail

sk_buff->len -= TCP/UDP_data_len
(= User_data_len)

Headroom

Tailroom
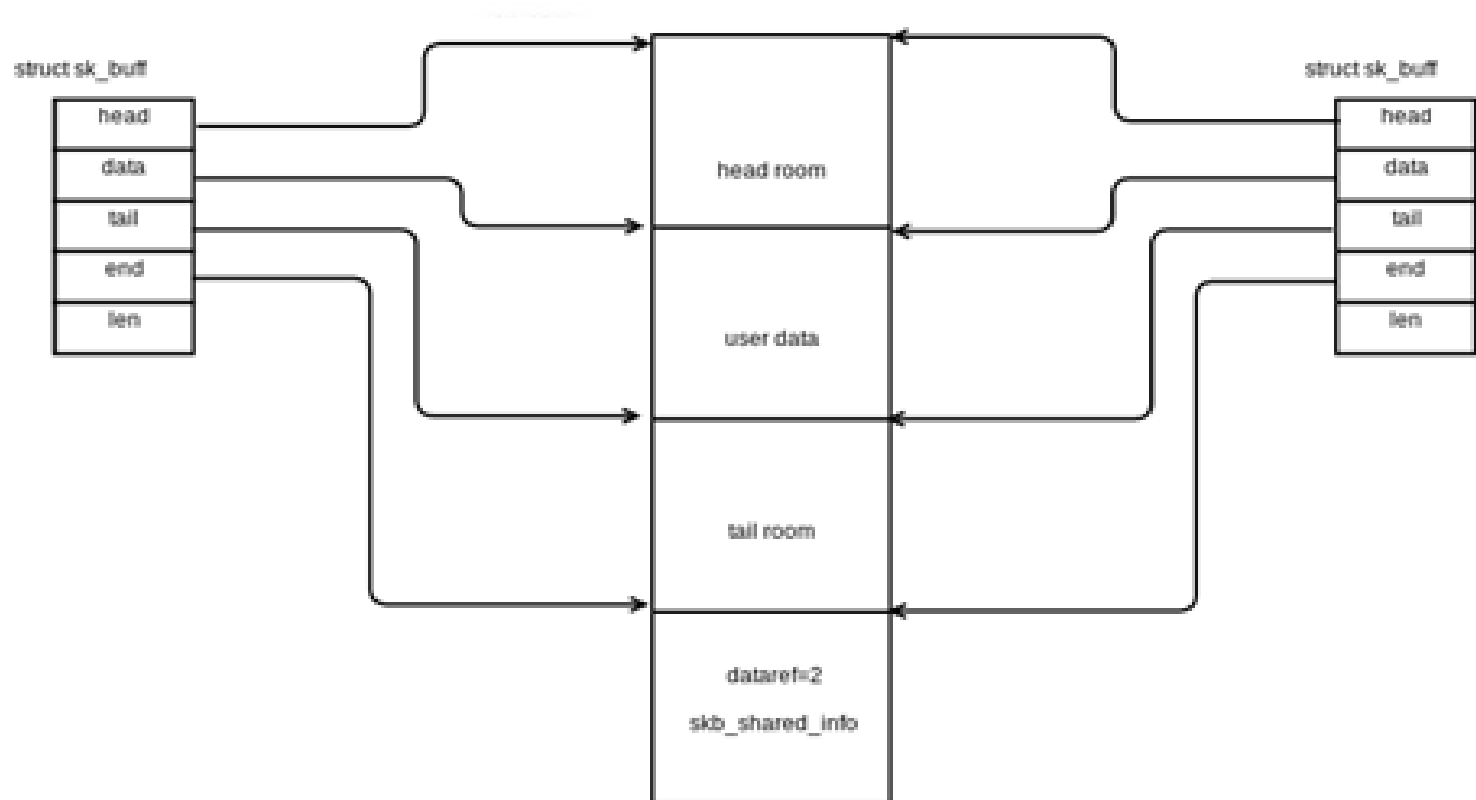
end

sk_buff - Cloning

OPERATION:
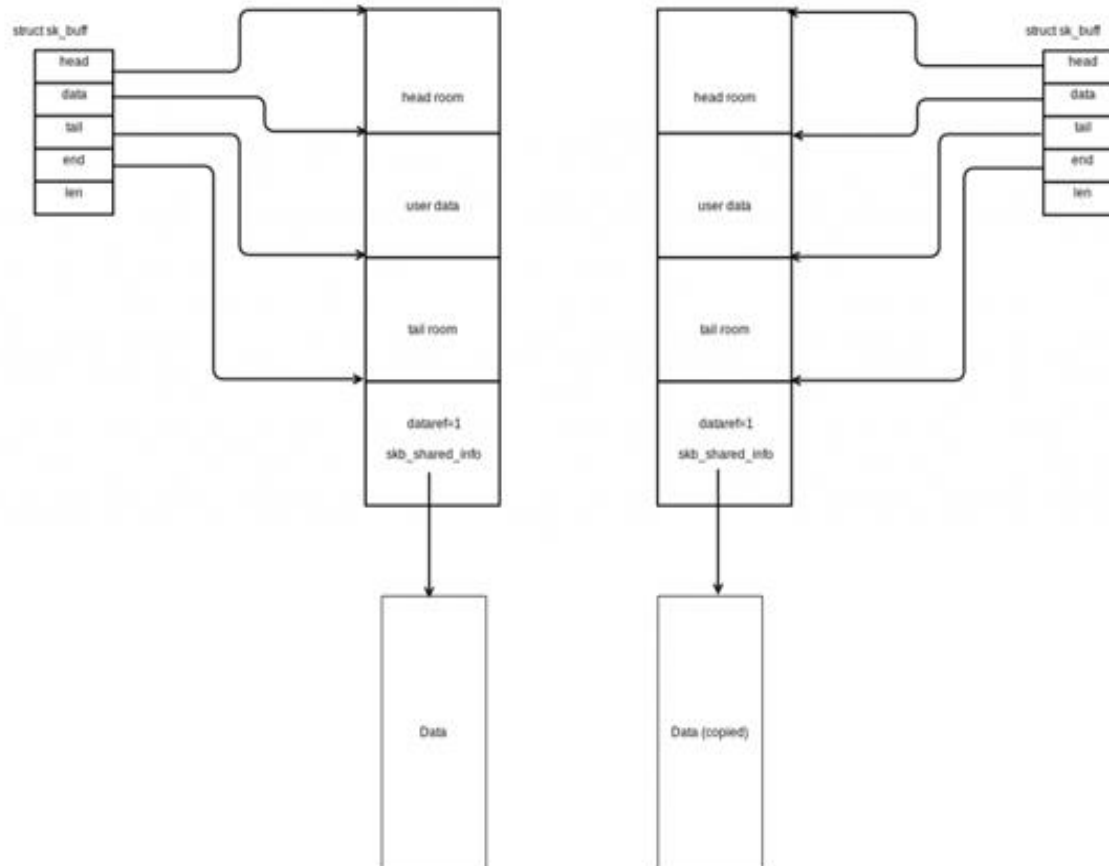struct sk_buff * skb_clone (struct sk_buff *skb, int gfp_mask)

sk_buff - Copying

OPERATION:
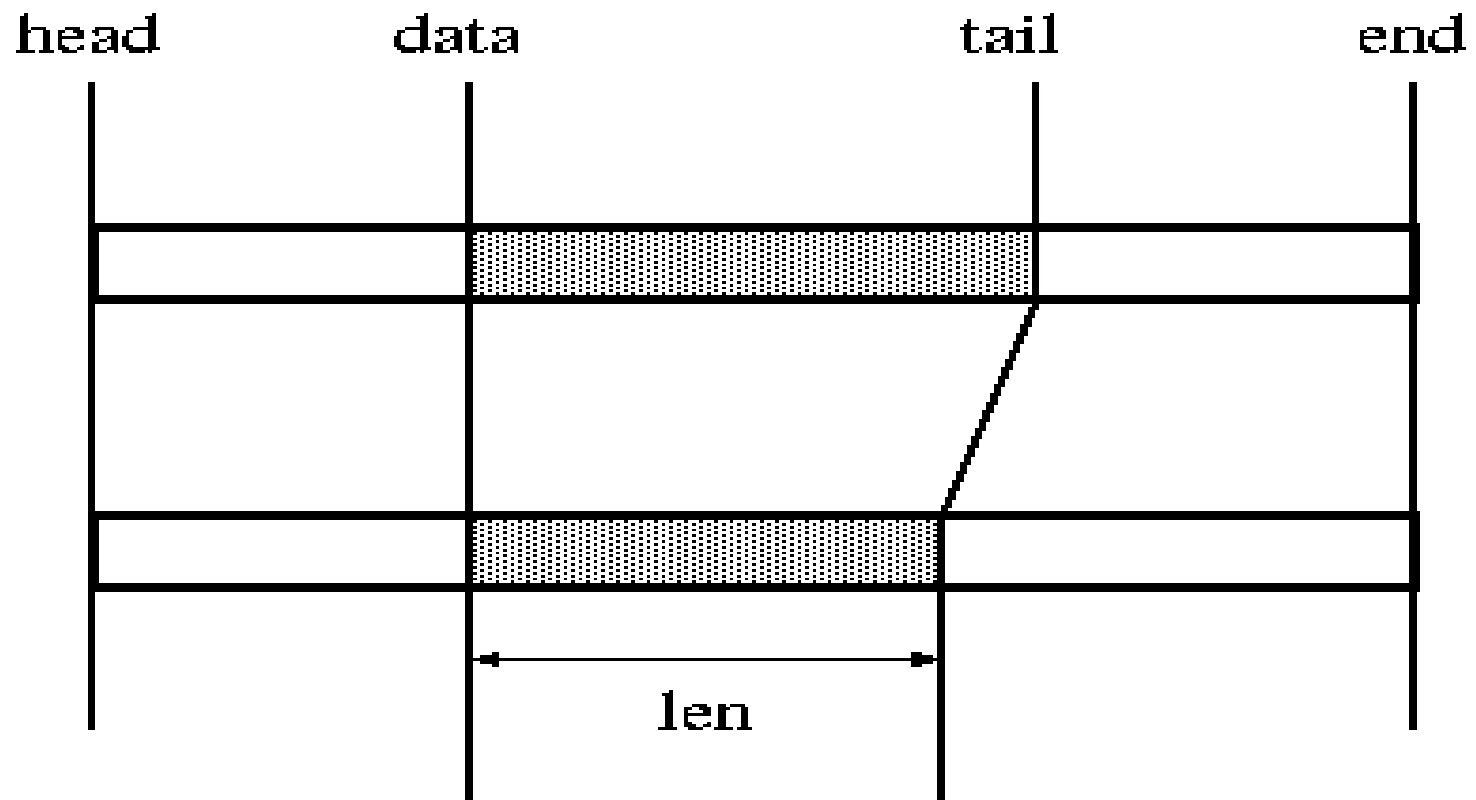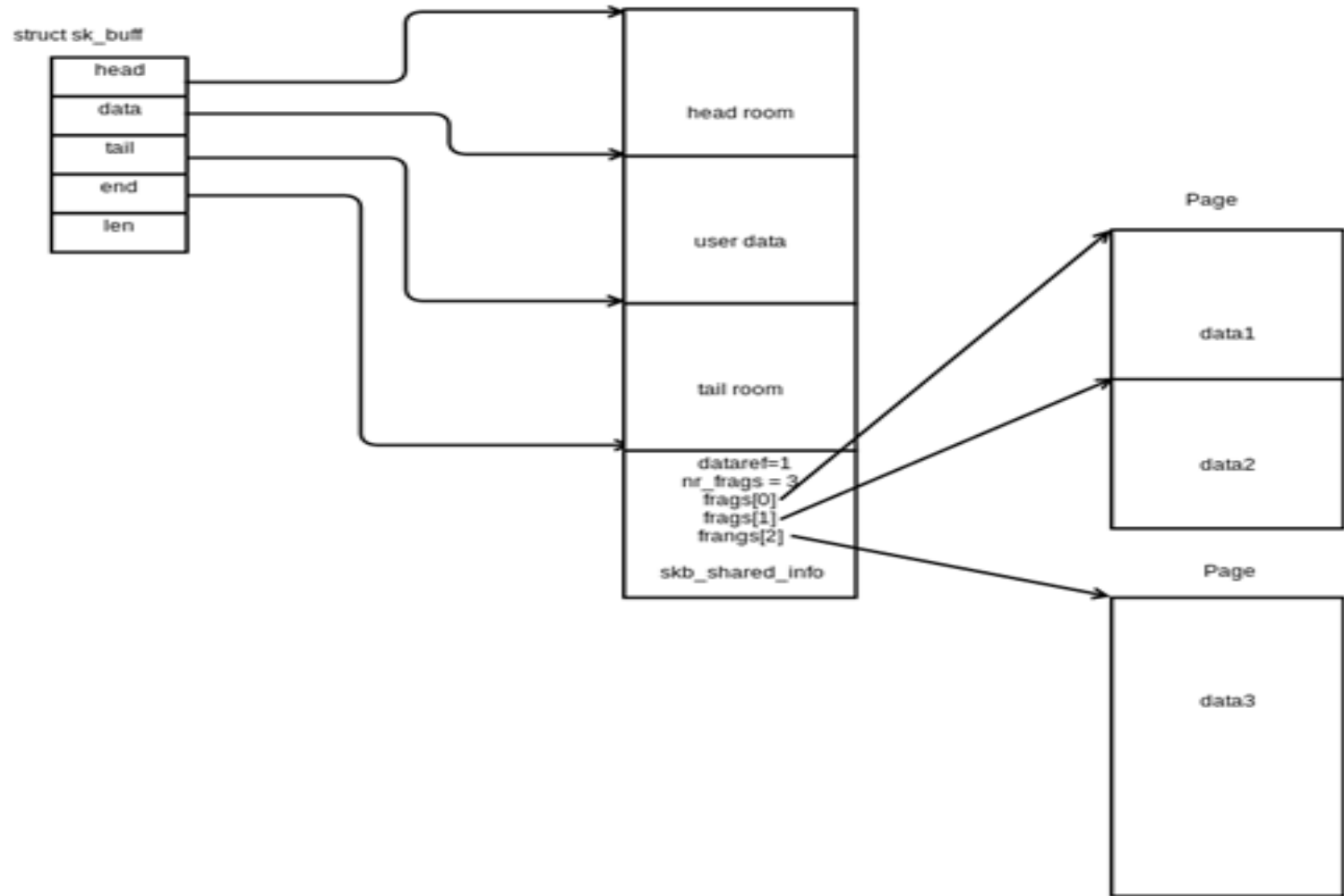struct sk_buff * skb_copy (struct sk_buff *skb, int gfp_mask)

sk_buff – Trimming

OPERATION:
void skb_trim (struct sk_buff *skb, unsigned int len)

Non-Linear (Fragmented) sk_buff

sk_buff - Additional Support Functions

Free SKB in Kernel:
　　　　void kfree_skb (struct sk_buff *skb)

Free SKB in Device Driver:
　　　　void dev_kfree_skb (struct sk_buff *skb)

Return the headroom length:
　　　　unsigned int skb_headroom (struct sk_buff *skb)

Return the tailroom length:
　　　　int skb_tailroom (struct sk_buff *skb)

sk_buff - Additional Support Functions (contd.)

Initialize SKB queue:

        void skb_queue_head _init (struct sk_buff_head *list)

Add SKB at the head of the SKB queue:

        void skb_queue_head (struct sk_buff_head *list, struct sk_buff *skb)

Add SKB at the tail of the SKB queue:

        void skb_queue_tail (struct sk_buff_head *list, struct sk_buff *skb)

Remove SKB from the head of the SKB queue:

        struct sk_buff * skb_dequeue (struct sk_buff_head *list)

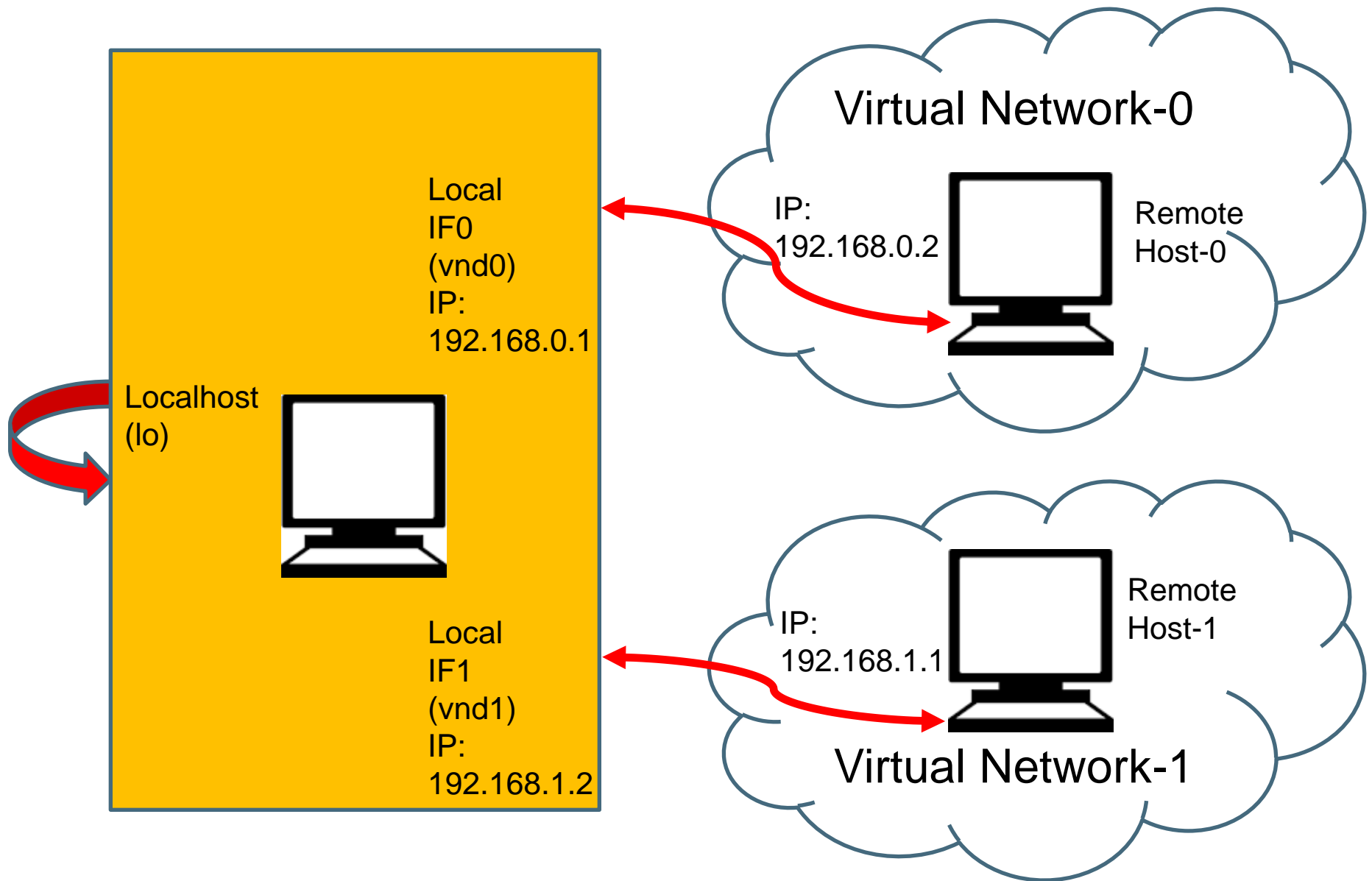**Remove SKB from the tail of the SKB queue:**

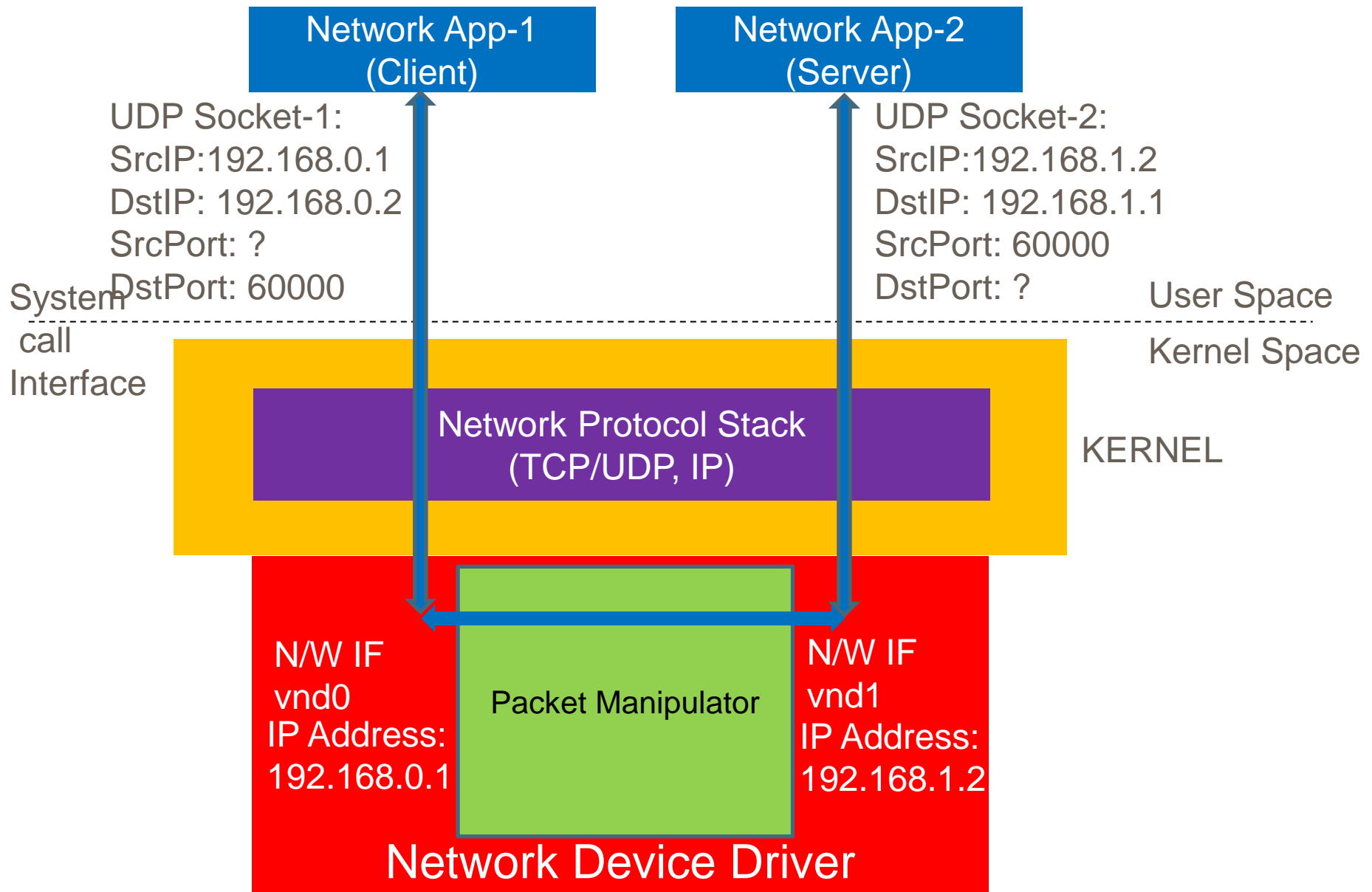   struct sk_buff * skb_dequeue_tail (struct sk_buff_head *list)

Return the length of the SKB queue:

        unsigned int skb_queue_len (struct sk_buff_head *list)
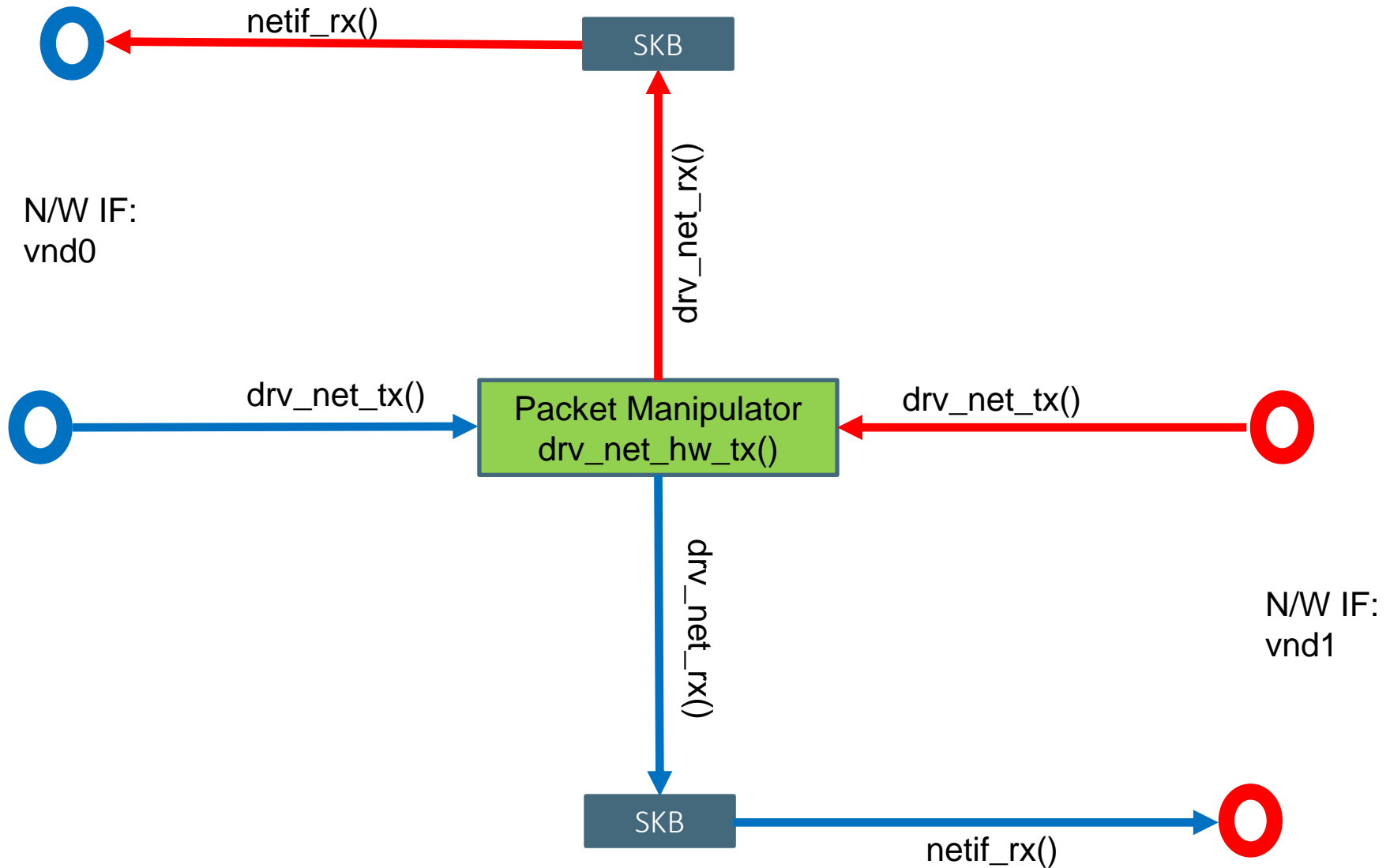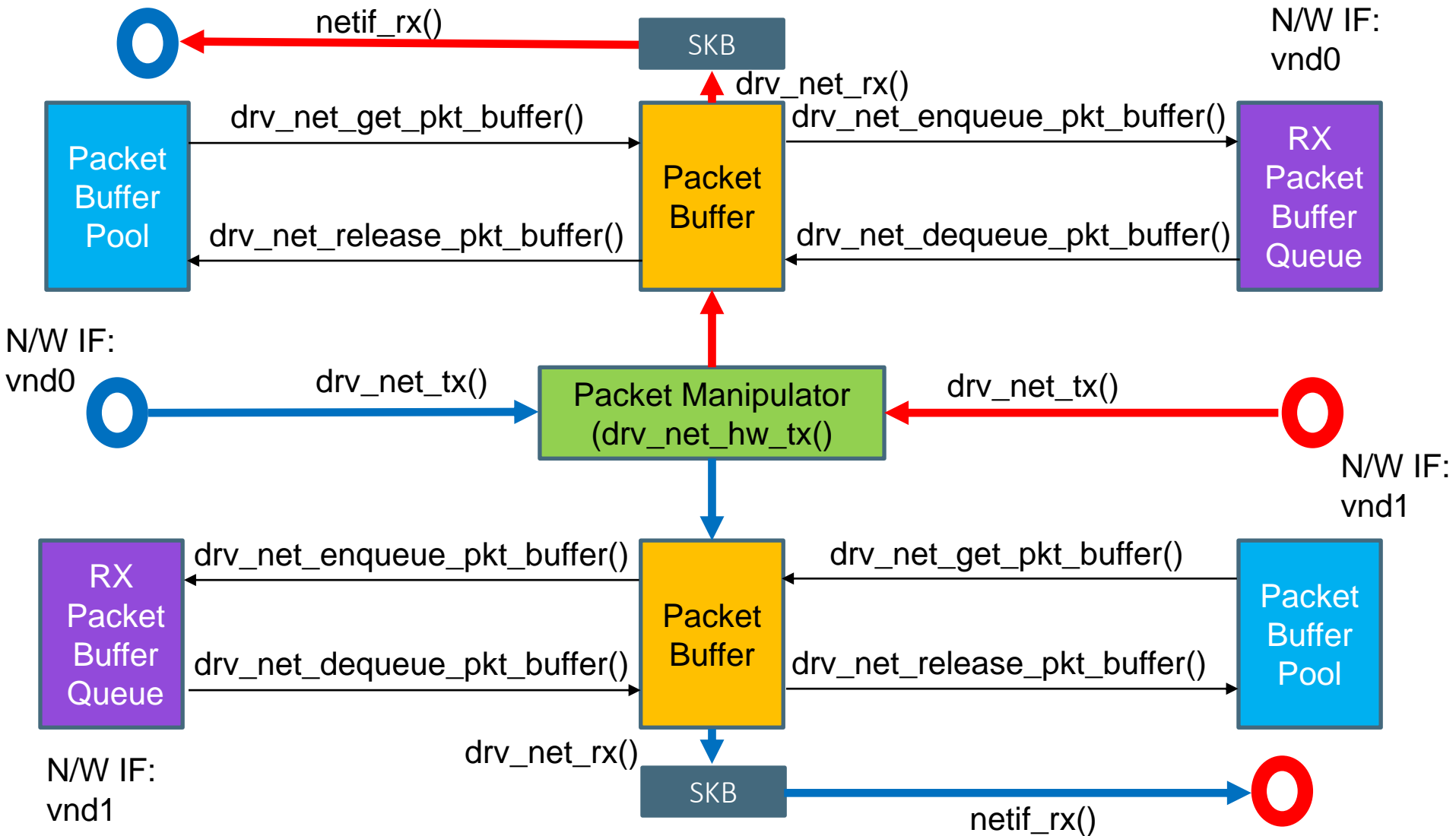
Virtual Network Architecture

Local
IF0
(vnd0)
IP:
192.168.0.1

Localhost
(lo)

Local
IF1
(vnd1)
IP:
192.168.1.2

Virtual Network-0

IP:
192.168.0.2

Remote
Host-0

Remote
Host-1

IP:
192.168.1.1

Virtual Network-1

Virtual (Loopback) N/W Driver – System Architecture

Network App-1
(Client)

Network App-2
(Server)

UDP Socket-1:
SrcIP:192.168.0.1
DstIP: 192.168.0.2
SrcPort: ?
DstPort: 60000

UDP Socket-2:
SrcIP:192.168.1.2
DstIP: 192.168.1.1
SrcPort: 60000
DstPort: ?

System call Interface

User Space

Kernel Space

Network Protocol Stack
(TCP/UDP, IP)

KERNEL

N/W IF
vnd0
IP Address:
192.168.0.1

Packet Manipulator

N/W IF
vnd1
IP Address:
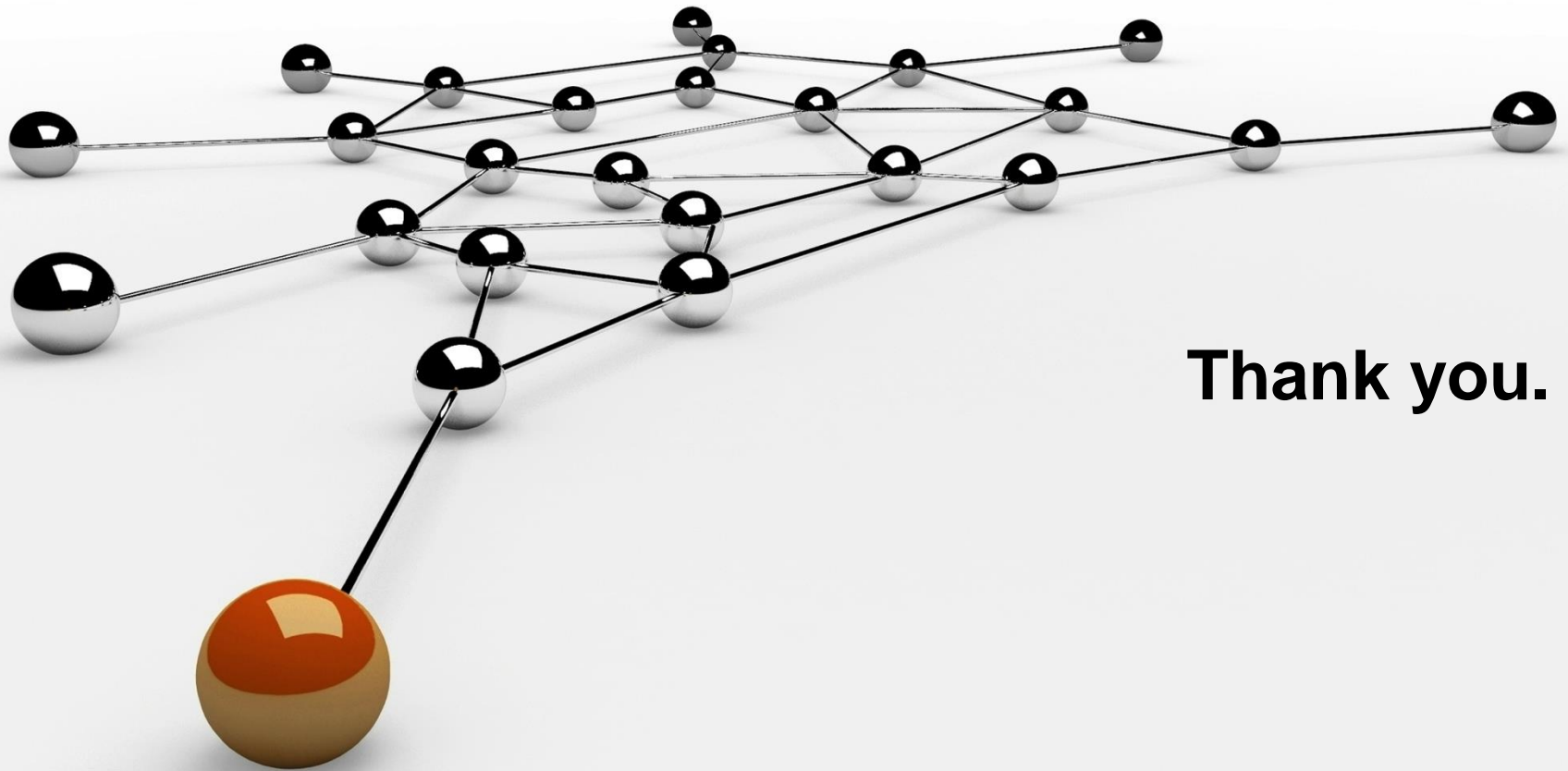192.168.1.2

Network Device Driver

Virtual(Loopback) N/W Driver – Software Architecture (1)

Virtual(Loopback) N/W Driver – Software Architecture (2)

Thank you.