Fall '20 COSE322-00

System Programming

Lecture 11. IP in Linux

2020. 11. 17.

Contents

- **IP (Internet Protocol)**
  - IP Reviews
  - Data Structure for IP
- **IP Main Function: Routing**
  - IP Routing Procedure
  - Data structures for IP Routing
- **Actual Implementation of IP**
  - IP Implementation Architecture
  - Sources of Packets to IP layer
  - IP Output
  - IP Input
  - Packet Forwarding

IP (Internet Protocol)

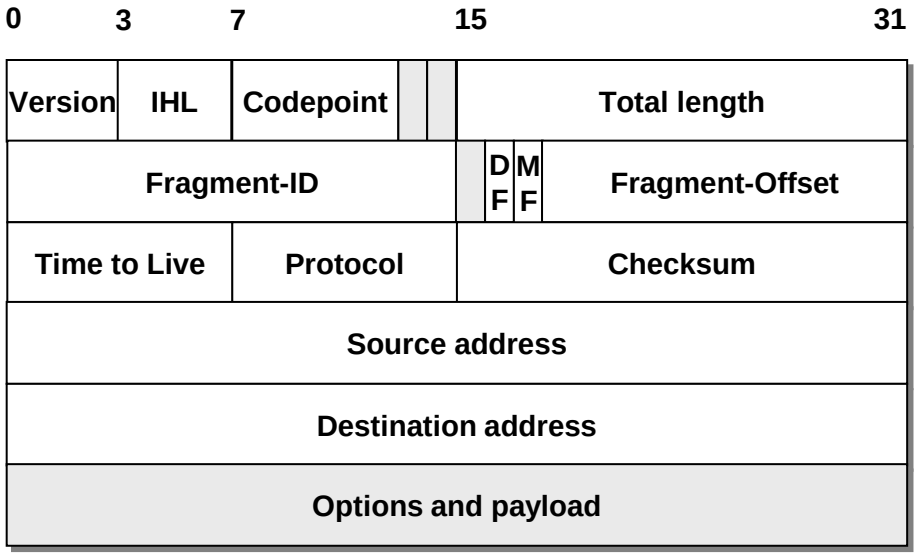
- IP Reviews
- Data Structure for IP

IP Reviews

❖ Functions

- Encapsulate/decapsulate network-layer messages
- Routes datagrams to destination
- Handle routing table updates
- Fragment / reassemble datagrams
- Unreliable

IP-packet format



IP Reviews

- **IP deals with logical address (IP Address)**
  - By logical address, IP determines the route for a packet (end-to-end)
  - In reality, Layer 2 uses physical address to deliver the frame (hop-by-hop)

- **ARP**
  - Translation between IP addresses and MAC layer addresses
  - IP layer sends an ARP request packet with curious IP address by broadcast to its network
  - Given an IP address, corresponding network interface sends its physical address

Where include/linux/ip.h

Data structure for IP

❖ **struct iphdr**

– Represents the IP header of each packet

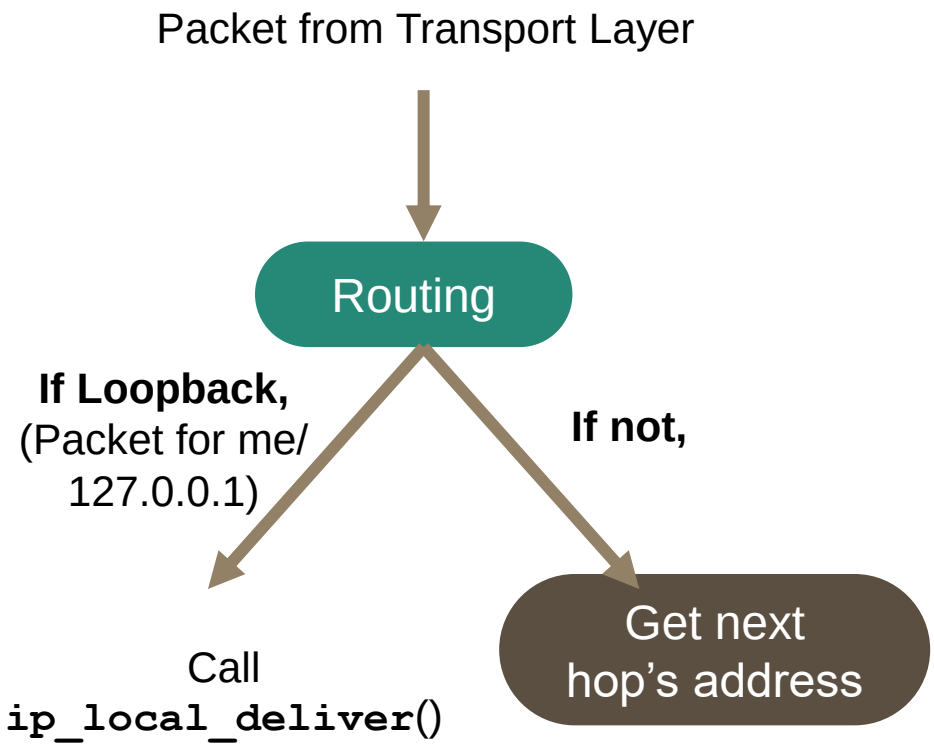
```
 85 struct iphdr {
 86 #if defined(__LITTLE_ENDIAN_BITFIELD)
 87         __u8    ihl:4,
 88                 version:4;
 89 #elif defined (__BIG_ENDIAN_BITFIELD)
 90         __u8    version:4,
 91                 ihl:4;
 92 #else
 93 #error  "Please fix <asm/byteorder.h>"
 94 #endif
 95         __u8    tos;        96          __be16  tot_len;
 97         __be16  id;         98          __be16  frag_off;
 99         __u8    ttl;       100          __u8    protocol;
101         __sum16 check;     102          __be32  saddr;
103         __be32  daddr;
104         /*The options start here. */
105 };
```

IP Main Function : Routing

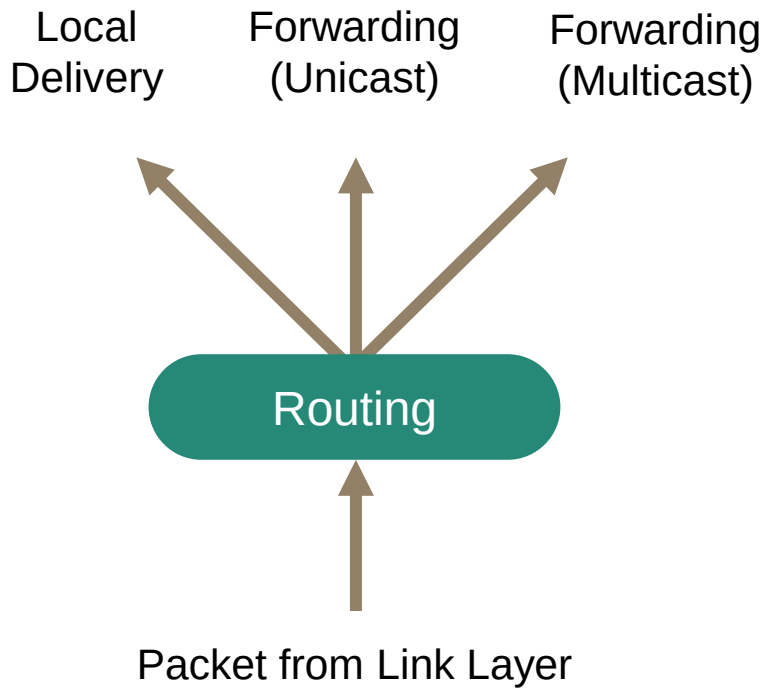
- **IP Routing Procedure**
- **Data structures for IP Routing (forwarding)**
  - Neighbor Table
  - Routing Cache
  - FIB(Forwarding Information Base)

IP Routing Procedure

❖ Sending

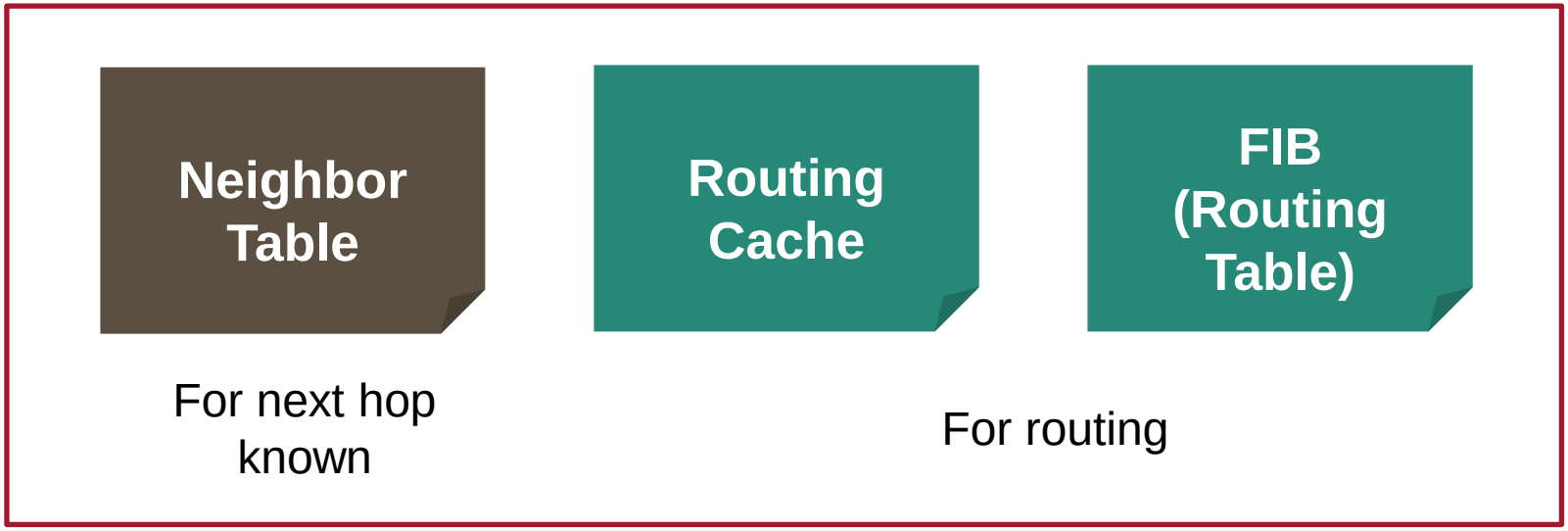


❖ Receiving



Data structures for IP Routing (forwarding)

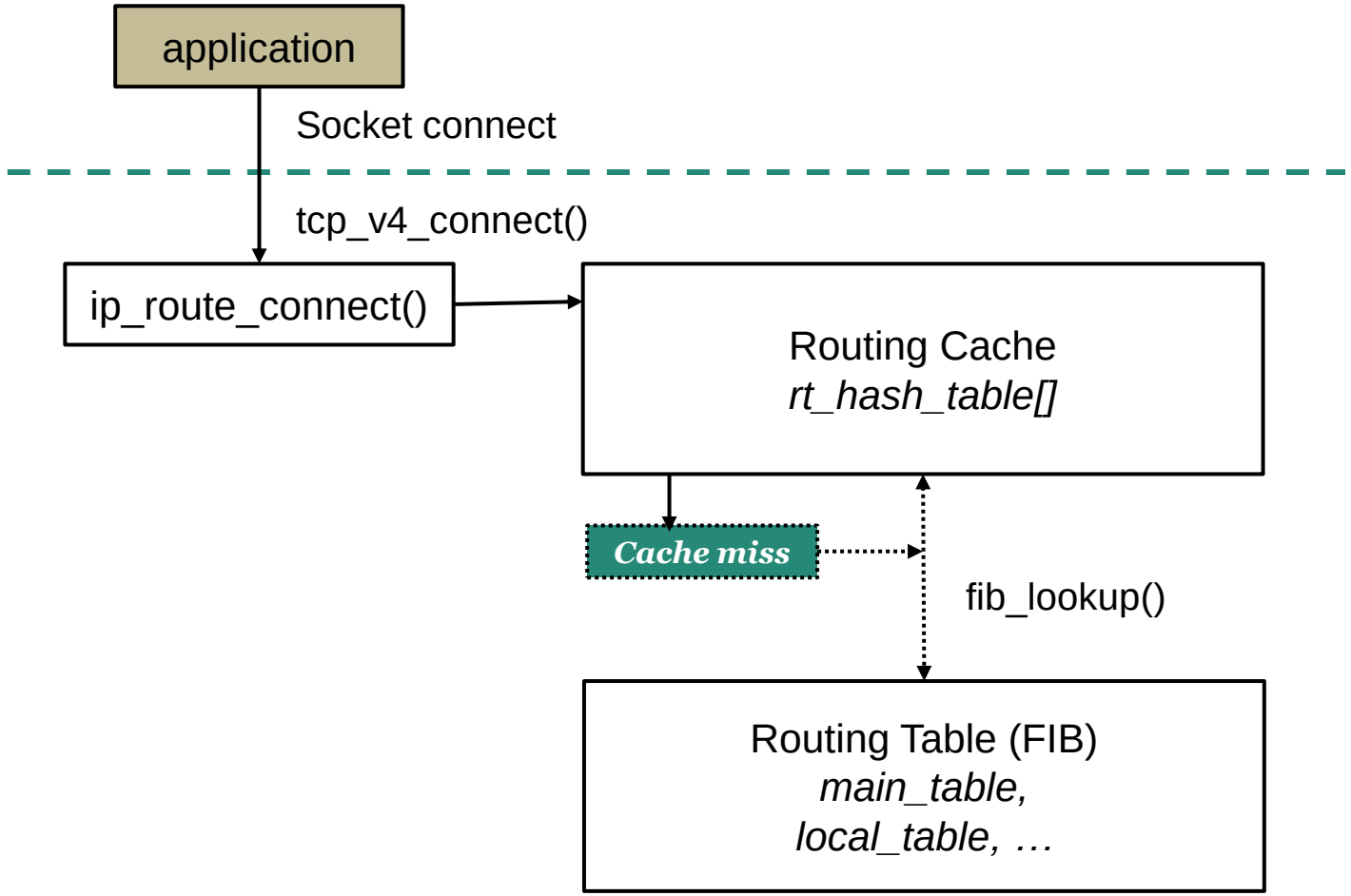
- **Three sets of routing data**
  - One for nodes that are directly connected to the host
    - Neighbor Table (**`struct neigh_table`**)
  - Two for nodes that are only indirectly connected
    - Routing Cache(**`struct rt_hash_table`**), FIB(**`struct fib_tables`**)



Neighbor Table

- **For physically linked nodes with the host**
  - It includes information on which node connects to which neighbor and what protocols to use in exchanging data
  - Linux uses the Address Resolution Protocol (ARP) to maintain and update this table
    - It is dynamic in that entries are added when needed but eventually disappear if not used again within a certain time
    - Administrators can set up entries to be permanent if doing so makes sense

Routing Cache and FIB

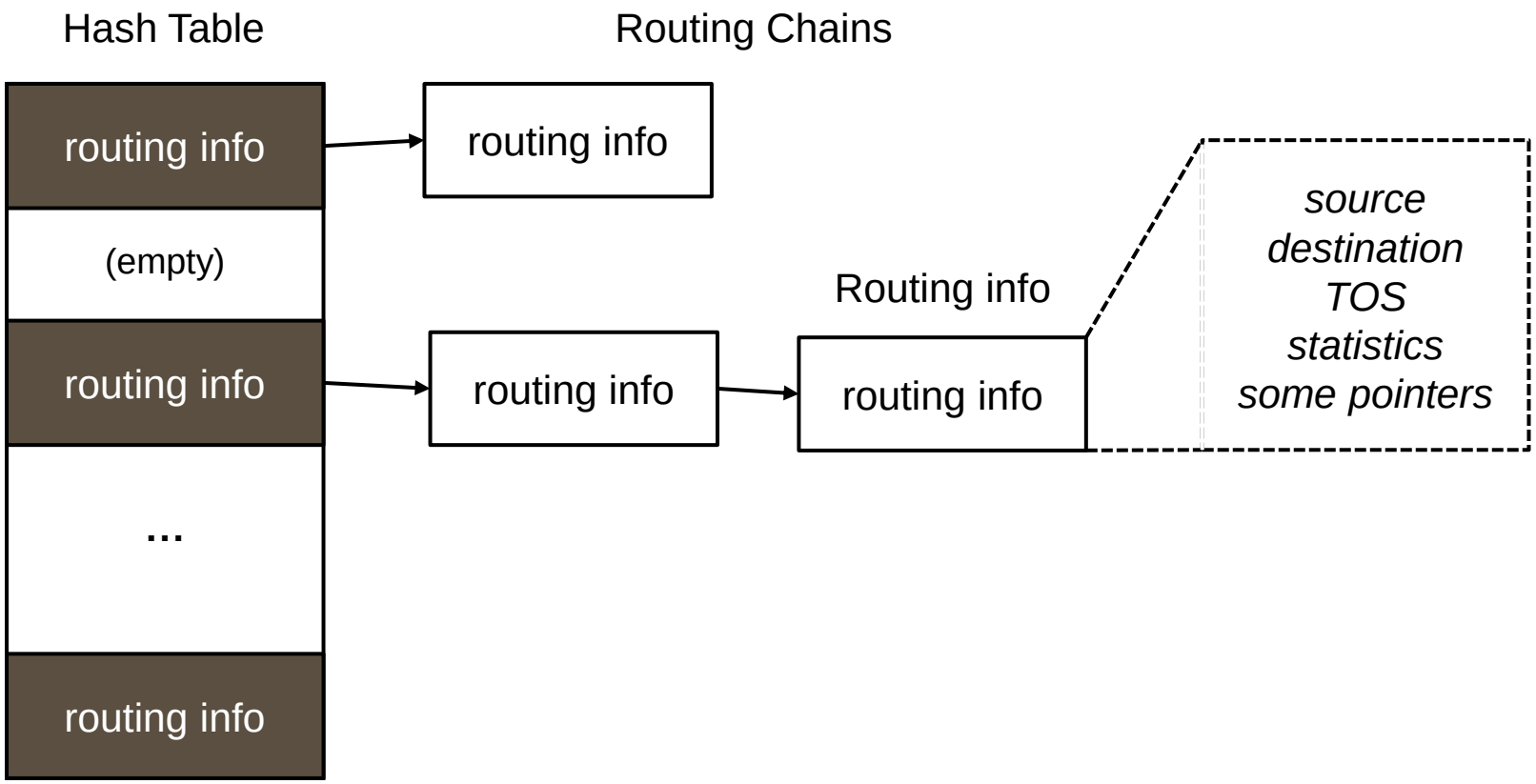


Routing Cache

- **Fastest method to find a route**
- **It keeps every route that is currently in use or has been used recently in a hash table**
- **Entry of routing cache**
  - Each entry is a route
  - Routes are chained in order, most frequently used first
  - Each entry has timer and counter in order to remove routes when they are no longer in use

Routing Cache

❖ Structures of Routing Cache

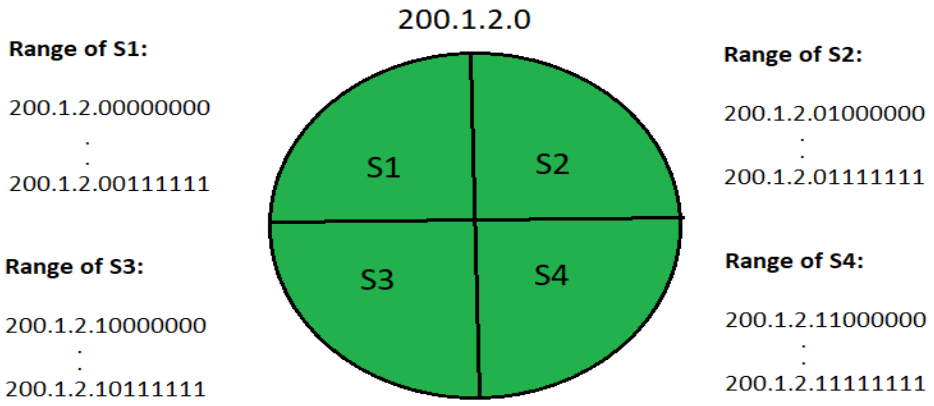


FIB (Forwarding Information Base)

- **Called routing table**
- **Most important routing structure in the kernel**
- **Mechanisms**
  - The IP layer enters the table with the destination address of a packet
  - Compares it to the most specific netmask to see if they match
  - If they do not, IP goes on to the next most general netmask and again compares the two
  - When it finally finds a match, IP copies the address to the distant host into the routing cache and sends the packet on its way

예제

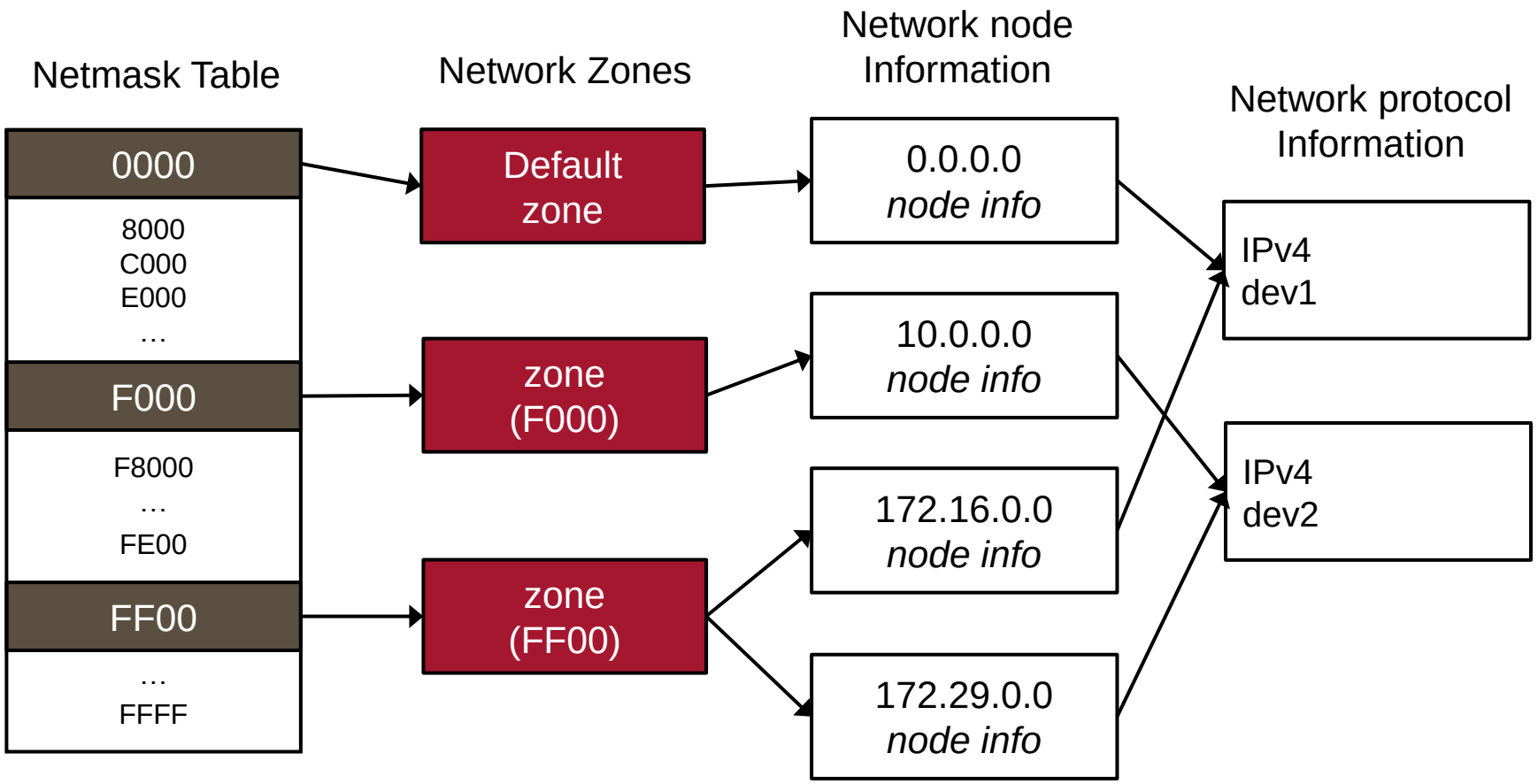
4개 subnet:
S1 – a
S2 – b
S3 – c
S4 - d



해당 FIB:

destination	Subnet mask	Interface	metric
200.1.2.0	255.255.255.192	a	1
200.1.2.64	255.255.255.192	b	10
200.1.2.128	255.255.255.192	c	5
200.1.2.192	255.255.255.192	d	5
default	0.0.0.0	e	10
127.0.0.1	localhost		

FIB 구조


Netmask Table
Network Zones
Network node Information
Network protocol Information
0000
8000
C000
E000
…
F000
F8000
…
FE00
FF00
…
FFFF
Default zone
zone (F000)
zone (FF00)
0.0.0.0
node info
10.0.0.0
node info
172.16.0.0
node info
172.29.0.0
node info
IPv4 dev1
IPv4 dev2

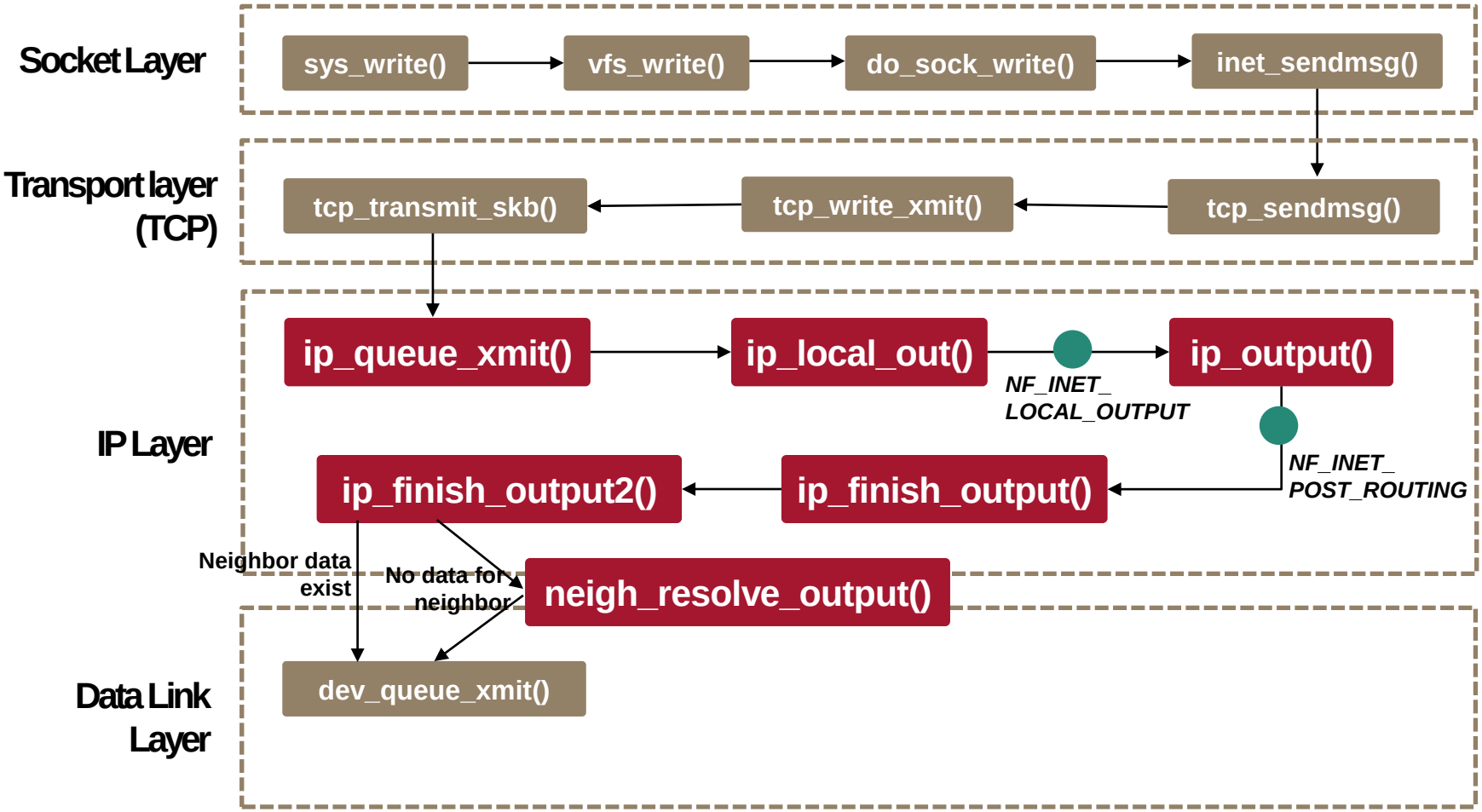
Actual Implementations of IP

- IP Implementation Architecture
- Sources of Packets to IP layer
- IP Output
- IP Input
- Packet Forwarding

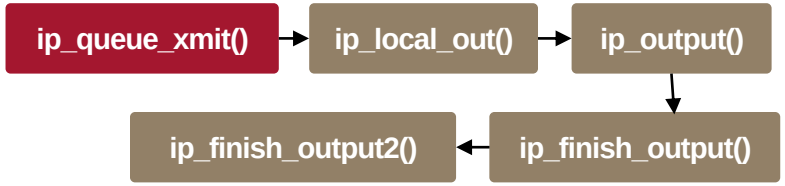
Sources of IP Packets

- **Input (receive)**
  - Packets arrive on a network interface and are passed to the **`ip_rcv`**() function
- **Output (transmit)**
  - TCP/UDP packets are packed into an IP packet and passed down to IP via **`ip_queue_xmit`**()
- **The IP layer generates IP packets itself for**
  - Multicast packets
  - Fragmentation of a large packet
  - ICMP/IGMP packets

IP Output

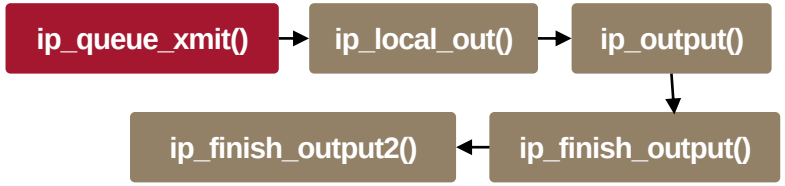

Socket Layer
sys_write()
vfs_write()
do_sock_write()
inet_sendmsg()
Transport layer (TCP)
tcp_transmit_skb()
tcp_write_xmit()
tcp_sendmsg()
IP Layer
ip_queue_xmit()
ip_local_out()
NF_INET_ LOCAL_OUTPUT
ip_output()
NF_INET_ POST_ROUTING
ip_finish_output()
ip_finish_output2()
Neighbor data exist
No data for neighbor
neigh_resolve_output()
Data Link Layer
dev_queue_xmit()

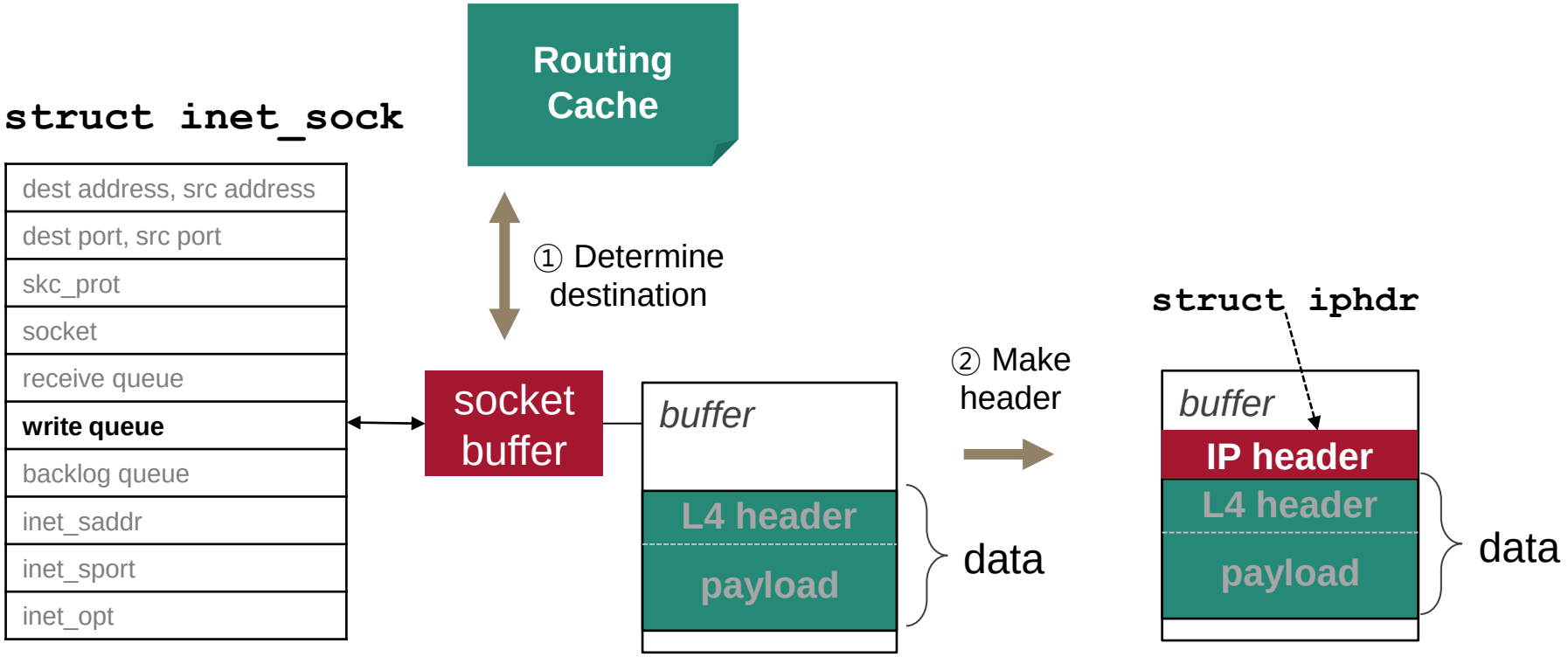
ip_queue_xmit()



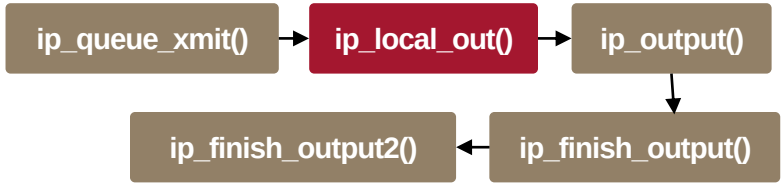
- **Determine destination**
  - Routing cache is used to reduce routing table lookup overhead
- **Initialize IP header**
  - Make space for header in **sk_buff**
  - Initialize header
    - Fill version, header length, TOS, TTL, addresses and protocol
- **Invoke ip_local_out()**

ip_queue_xmit()





ip_local_out()

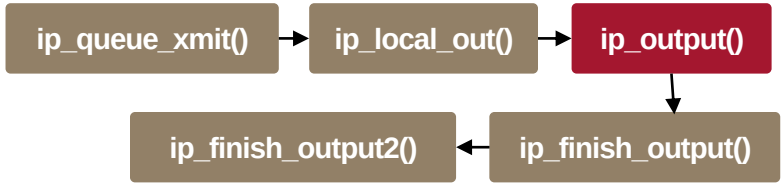


- **Compute checksum**

- **Check whether it is for loopback or not**
  - If destination is my host, call **`ip_local_deliver()`**
  - If not, **`ip_output()`** is called

- **Invoke netfilter hook (NF_INET_LOCAL_OUTPUT)**
  - Netfilter
    - A framework for custom handling of packets
    - represents a set of hooks inside the Linux kernel
      - allowing specific kernel modules to register callback functions

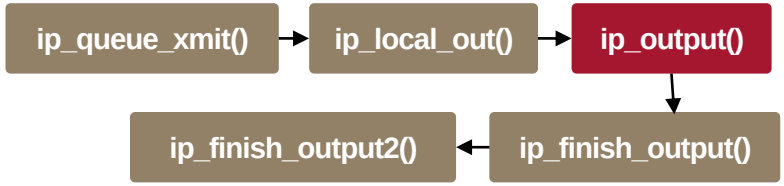
ip_output()

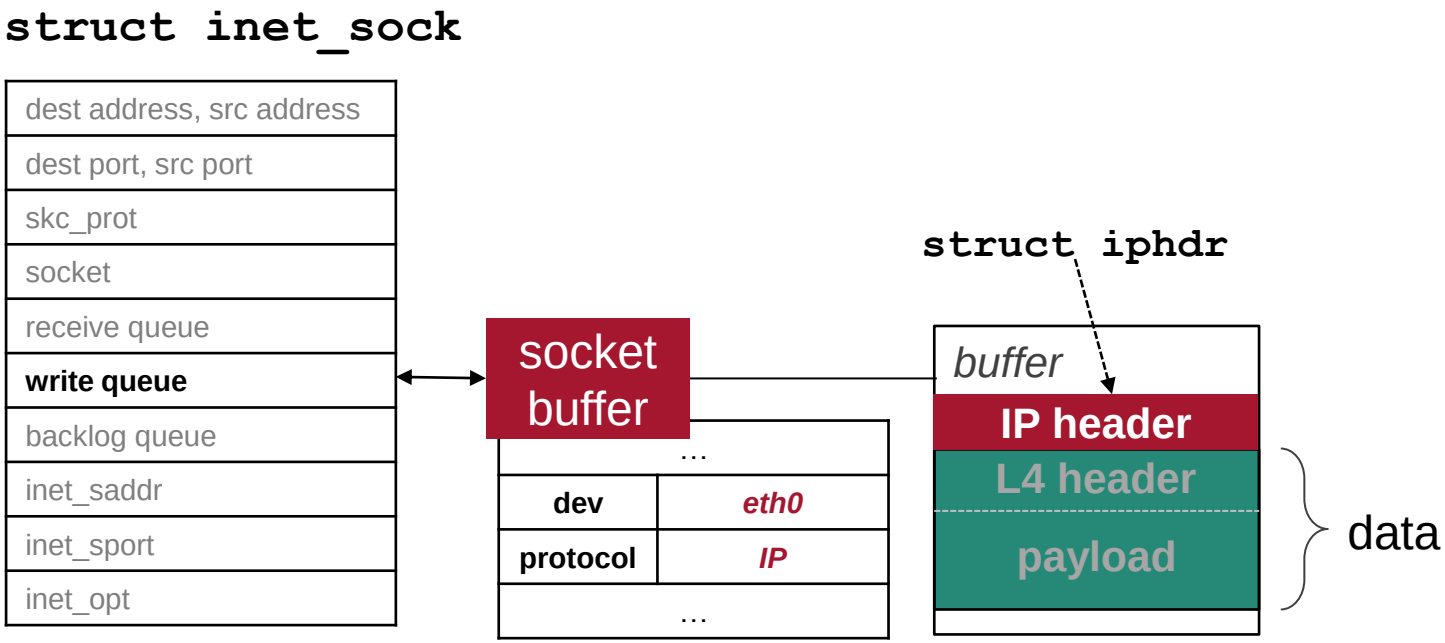


- **Update `sk_buff` fields**
  - Specify which device to use for transmit this packet (**`dev`** field)
  - Specify this packet is with IP (**`protocol`** field)

- **Invoke netfilter hook**
  - NF_INET_POST_ROUTING
  - At last, **`ip_finish_output()`** is called

ip_output()

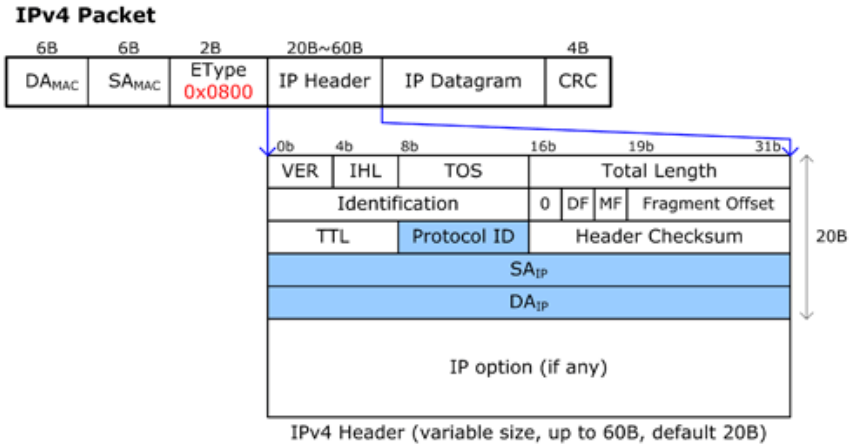

ip_queue_xmit()
ip_local_out()
ip_output()
ip_finish_output()
ip_finish_output2()


struct inet_sock
dest address, src address
dest port, src port
skc_prot
socket
receive queue
write queue
backlog queue
inet_saddr
inet_sport
inet_opt
socket buffer
…
dev
eth0
protocol
IP
…
struct iphdr
buffer
IP header
L4 header
payload
data

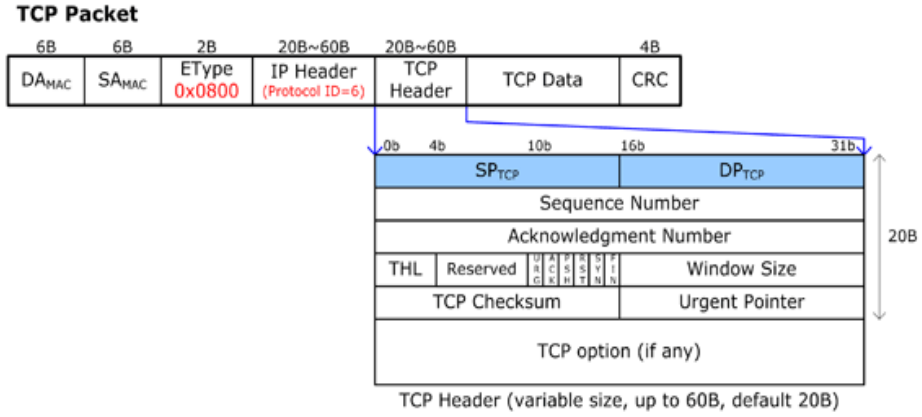
Packet Header: Ethernet, IP & TCP/IP

❖ Ethernet header : 14 bytes

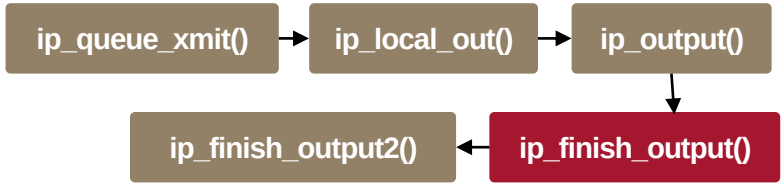
❖ IP header : 20 bytes



❖ TCP header : 20 bytes



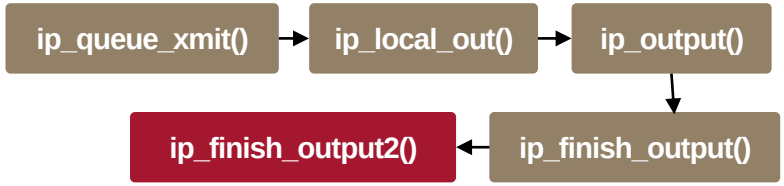
ip_finish_output()



❖ Do fragmentation

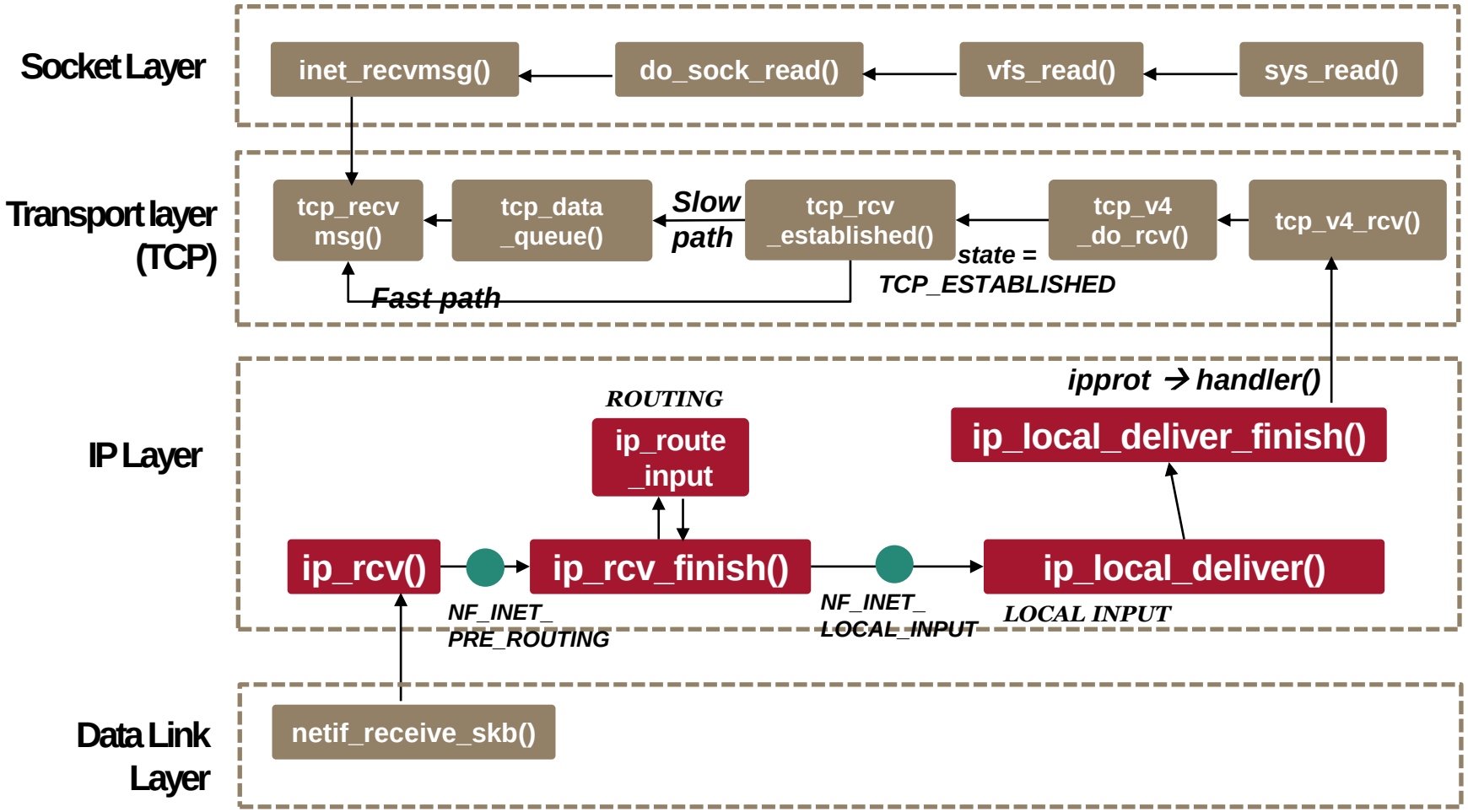
- Get a value for MTU (Maximum Transmission Unit)
  - min(dev→mtu, IP_MAX_MTU)
  - IP_MAX_MTU : 65535
- Check message length against the destination MTU
- Call either
  - Have to fragment → **`ip_fragment()`**
  - No need → **`ip_finish_output2()`**

ip_finish_output2()

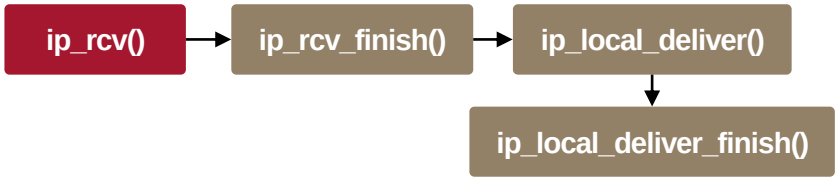


- **Make space for L2 header**
  - If skb has not a sufficient room for L2 header, allocate L2 header's space
- **Determine the `nexthop`**
  - Get a next hop's L2 address (MAC) from **`struct neigh_table`**
  - If no matching, sends an ARP request message (**`neigh_resolve_output()`**)
- **Eventually end up in `dev_queue_xmit()`**
  - Pass the packet down to the device

IP Input


Socket Layer
inet_recvmsg()
do_sock_read()
vfs_read()
sys_read()
Transport layer (TCP)
tcp_recv msg()
tcp_data _queue()
Slow path
tcp_rcv _established()
tcp_v4 _do_rcv()
tcp_v4_rcv()
state = TCP_ESTABLISHED
Fast path
IP Layer
ROUTING
ip_route _input
ipprot → handler()
ip_local_deliver_finish()
ip_rcv()
ip_rcv_finish()
ip_local_deliver()
NF_INET_ PRE_ROUTING
NF_INET_ LOCAL_INPUT
LOCAL INPUT
Data Link Layer
netif_receive_skb()

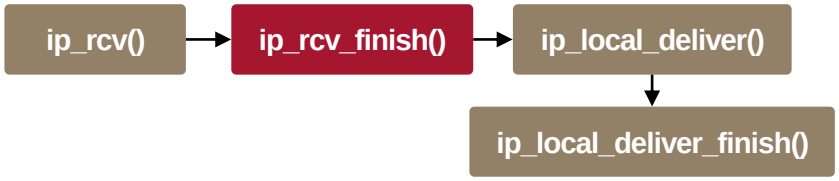
ip_rcv()



- **Do sanity checking**
  - Divide region for header and data
  - Header size, IP Version, checksum, length

- **Invoke Netfilter hook**
  - NF_INET_PRE_ROUTING
  - At last, **`ip_rcv_finish()`** is called

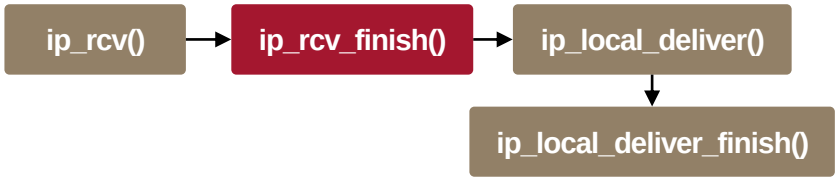
ip_rcv_finish()



- **Find the destination from routing cache**

- **Do the action depending on the routing result**
  - **`ip_local_deliver()`** – For local
  - **`ip_forward()`** – Forwarding (unicast)
  - **`ip_mr_input()`** – Forwarding (multicast)

ip_rcv_finish()



struct iphdr

buffer

header

payload

data

socket buffer

① Decide the processing way for the packet

Has to be locally delivered

ip_local_deliver()

Forwarding (unicast)

ip_forward()

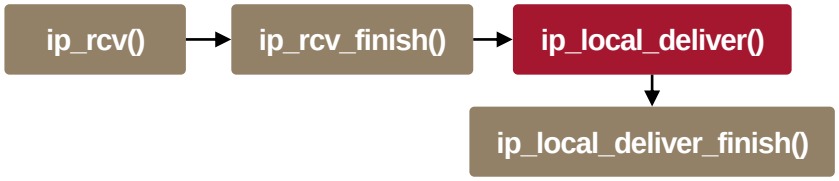
Follow this at the Packet forwarding section

Routing Cache

Forwarding (multicast)

ip_mr_input()

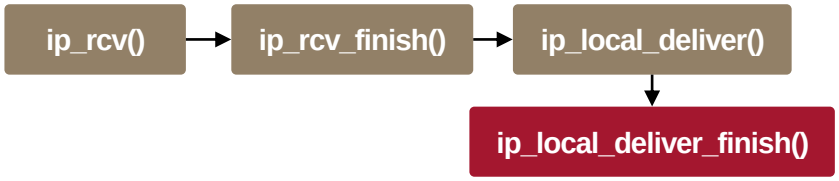
ip_local_deliver()



- **Do re-assemble**
  - If the flag of the IP header indicates the packet is fragmented, the packet is reassembled

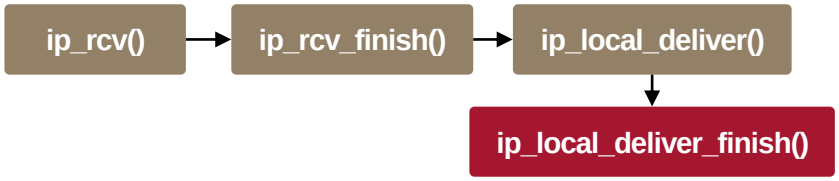
- **Invoke Netfilter hook**
  - NF_INET_LOCAL_IN
  - At last, **`ip_local_deliver_finish()`** is called

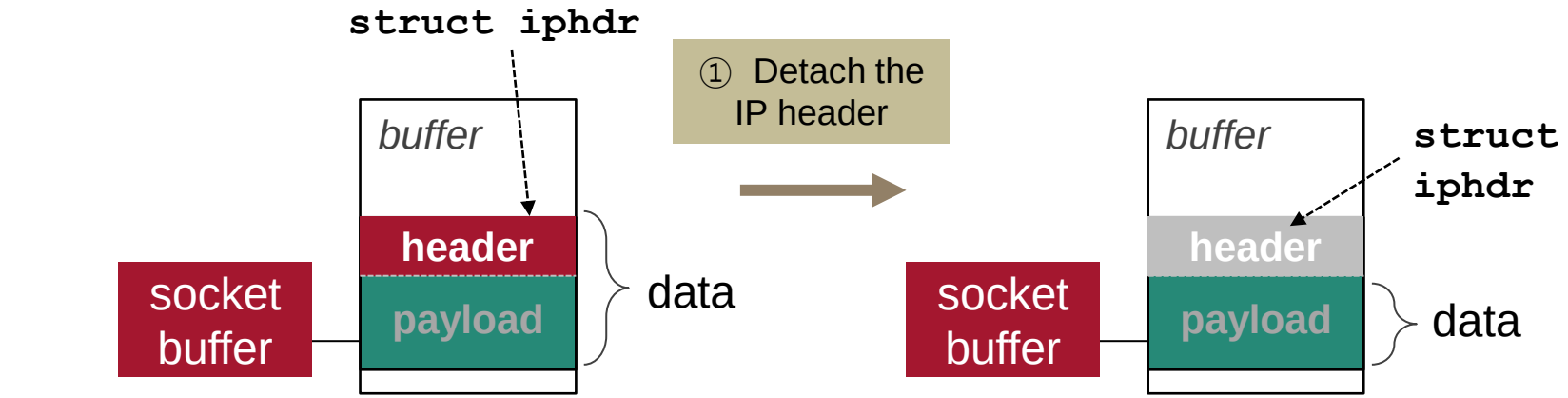
ip_local_deliver_finish()



- **Remove the IP header from skb**
- **Invoke Upper's handler based on protocol field of IP header**
  - `tcp_v4_rcv()` : TCP
  - `udp_rcv()` : UDP
  - `icmp_rcv()` : ICMP
  - `igmp_rcv()` : IGMP
- **If no match**
  - Drop it and send ICMP
    - Destination Unreachable

ip_local_deliver_finish()





② Call upper layer's function depending on the IP header's protocol field

iphdr→protocol == 6 (TCP) → `tcp_v4_rcv()`

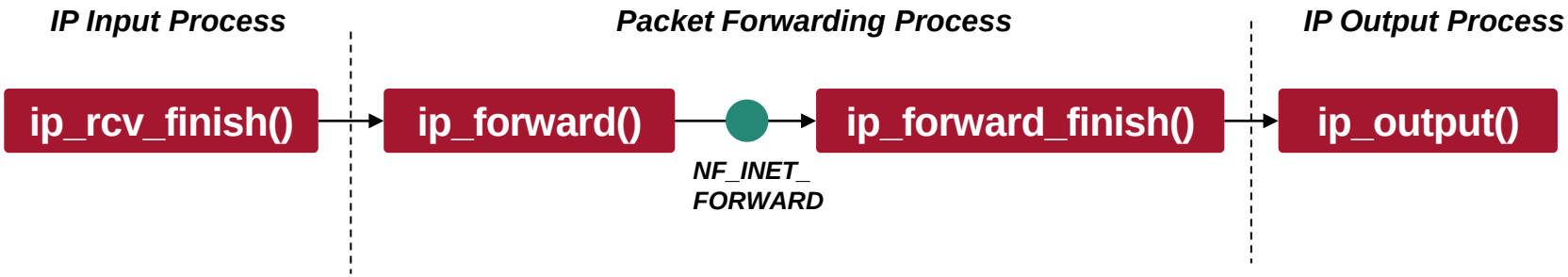
iphdr→protocol == 22 (UDP) → `udp_rcv()`

iphdr→protocol matches nothing → `icmp_send()`

Type : Destination Unreachable

Packet Forwarding

- When the packet is determined to be forwarded, `ip_forward()` is called



ip_forward()

- **Validate and Check**
  - TTL <= 1 : Drop and send ICMP (ICMP_TIME_EXCEEDED)
  - Length > MTU, DF : Drop and send ICMP (ICMP_FRAG_NEEDED)
- **Make space for L2 header**
  - Check skb and allocate L2 header's space
- **Decrease TTL by 1**
- **Invoke netfilter hook**
  - NF_INET_FORWARDING
  - At last, **`ip_forward_finish()`** is called

ip_forward_finish()

- **Handle any IP options if they exist**
- **Call `ip_output()`**
  - the destination output function

IP Implementation Architecture