e1000网卡收包callstack分析

```TXT
/**
 * netif_receive_skb_list - process many receive buffers from
network
 * @head: list of skbs to process.
 *
 * Since return value of netif_receive_skb() is normally
ignored, and
 * wouldn't be meaningful for a list, this function returns
void.
 *
 * This function may only be called from softirq context and
interrupts
 * should be enabled.
 */
```

`void netif_receive_skb_list(struct list_head *head)`: `drivers/net`实现各种硬件设备的驱动，收包后从硬件获取buffer，驱动代码中调用 `netif_receive_skb_list`进入内核网络系统，进行接下来的处理

驱动向内核注册softirq，里面包含回调函数。驱动收到数据触发中断，kernel读取

```c
__netif_receive_skb_list_ptype(struct net_device * orig_dev, struct packet_type * pt_prev, struct list_head * head) (/data00/codes/linux/net/core/dev.c:5533)
__netif_receive_skb_list_core(struct list_head * head, bool pfmemalloc) (/data00/codes/linux/net/core/dev.c:5582)
__netif_receive_skb_list(struct list_head * head) (/data00/codes/linux/net/core/dev.c:5634)
netif_receive_skb_list_internal(struct list_head * head) (/data00/codes/linux/net/core/dev.c:5725)
gro_normal_list(struct napi_struct * napi) (/data00/codes/linux/include/net/gro.h:433)
gro_normal_list(struct napi_struct * napi) (/data00/codes/linux/include/net/gro.h:429)
napi_complete_done(struct napi_struct * n, int work_done) (/data00/codes/linux/net/core/dev.c:6065)
e1000_clean(struct napi_struct * napi, int budget) (/data00/codes/linux/drivers/net/ethernet/intel/e1000/e1000_main.c:3811)
__napi_poll(struct napi_struct * n, bool * repoll) (/data00/codes/linux/net/core/dev.c:6496)
napi_poll(struct list_head * repoll, struct napi_struct * n) (/data00/codes/linux/net/core/dev.c:6563)
net_rx_action(struct softirq_action * h) (/data00/codes/linux/net/core/dev.c:6696)
__do_softirq() (/data00/codes/linux/kernel/softirq.c:571)
do_softirq() (/data00/codes/linux/kernel/softirq.c:472)
do_softirq() (/data00/codes/linux/kernel/softirq.c:459)
```

调试kernel收发包流程

用 `curl http://bilibili.com`测试，在 `__netif_receive_skb_core`下断点

bt如下

TXT

```
Breakpoint 2, __netif_receive_skb_core
(pskb=pskb@entry=0xffffc90000003da0,
    pfmemalloc=pfmemalloc@entry=false,
ppt_prev=ppt_prev@entry=0xffffc90000003da8)
    at net/core/dev.c:5240
5240    static int __netif_receive_skb_core(struct sk_buff
**pskb, bool pfmemalloc,
(gdb) frame
#0  __netif_receive_skb_core
(pskb=pskb@entry=0xffffc90000003da0,
    pfmemalloc=pfmemalloc@entry=false,
ppt_prev=ppt_prev@entry=0xffffc90000003da8)
    at net/core/dev.c:5240
5240    static int __netif_receive_skb_core(struct sk_buff
**pskb, bool pfmemalloc,
(gdb) bt
#0  __netif_receive_skb_core
(pskb=pskb@entry=0xffffc90000003da0,
    pfmemalloc=pfmemalloc@entry=false,
ppt_prev=ppt_prev@entry=0xffffc90000003da8)
    at net/core/dev.c:5240
#1  0xffffffff81bac595 in __netif_receive_skb_list_core (
    head=head@entry=0xffff888003ddccb8,
pfmemalloc=pfmemalloc@entry=false)
    at net/core/dev.c:5528
#2  0xffffffff81bacd3b in __netif_receive_skb_list
(head=0xffff888003ddccb8)
    at net/core/dev.c:5595
#3  netif_receive_skb_list_internal
(head=head@entry=0xffff888003ddccb8)
    at net/core/dev.c:5686
#4  0xffffffff81bad4be in gro_normal_list
(napi=0xffff888003ddcbb0)
    at ./include/net/gro.h:439
#5  gro_normal_list (napi=0xffff888003ddcbb0) at
./include/net/gro.h:435
#6  napi_complete_done (n=n@entry=0xffff888003ddcbb0,
work_done=<optimized out>)
    at net/core/dev.c:6026
```

```
#7  0xffffffff819b8863 in e1000_clean (napi=0xffff888003ddcbb0,
budget=64)
    at drivers/net/ethernet/intel/e1000/e1000_main.c:3811
#8  0xffffffff81bad627 in __napi_poll
(n=n@entry=0xffff888003ddcbb0,
    repoll=repoll@entry=0xffffc90000003f47) at
net/core/dev.c:6460
#9  0xffffffff81badcb2 in napi_poll (repoll=0xffffc90000003f58,
n=0xffff888003ddcbb0)
    at net/core/dev.c:6527
#10 net_rx_action (h=<optimized out>) at net/core/dev.c:6660
#11 0xffffffff81f0a5a0 in __do_softirq () at
kernel/softirq.c:553
#12 0xffffffff81091a8e in do_softirq () at kernel/softirq.c:454
#13 do_softirq () at kernel/softirq.c:441
```

inet_init 添加 ipv4 的 packet handler

`receive_skb*` 有list和非list版本，list可以做GRO合并，根据条件将多个skb合并，减少重复处理次数

调用顺序

> netif_receive_skb_list
> __netif_receive_skb_list_core
> __netif_receive_skb_core
> deliver_skb

> 回调packet_type的handler
> 5.

路由选择

__fib_lookup

fib_lookup

struct rtable

代表一条路由规则，路由结束绑定到skb#_skb_refdst

- struct dst_entry dst;
  - 下面详细介绍。
- _u16 rt_type
  - 本路由类型

字段	值	ip route	含义
RTN_UNICAST	1	unicast	默认值，如果skb#_skb_refdst是此值，数据需要被转发
RTN_LOCAL	2	local	本机路由
RNT_BROADCAST	3	广播	需要转发
RTN_MULTICAST	5	多播	多播
RNT_UNREACHABLE	7	unreachable	丢弃，返回ICMP network unreachable

- __u8 rt_uses_gateway
  - bool，路由的下一跳是网关(ip route 返回类似 `via 10.0.0.1` 的格式)，那么 `rt_gw4`包含网关IP地址。
- u8 rt_gw_family
  - 如果rt_uses_gateway是0，那rt_gw_family是0。如果网关地址是IPV4，=AF_INET。IPV6, =AF_INET6
- union {__be32 rt_gw4; struct in6_addr rt_gw6;}
  - 如果是网关，根据IP类型使用 `rt_gw4` 或 `rt_gw6` 字段

struct dst_entry

- struct net_device *dev
  - 发送数据的网络设备，数据包最终从此设备发送
- struct xfrm_state *xfrm
  - 和IPsec 相关，一般是NULL
- int (_input)(struct sk_buff_skb)
  - 根据路由结果，input是不同值。决定数据包接下来如果处理。dest ip本机则继续向上传递。不是本机的ip则转发

可选值	备注
dst_discard	默认值
ip_local_deliver	目的地址是本机，或者广播数据包
ip_forward	单播，但不是本机，需要转发
ip_error	没有找到匹配路由，*unreachable*，数据包丢弃，返回ICMP host unreachable

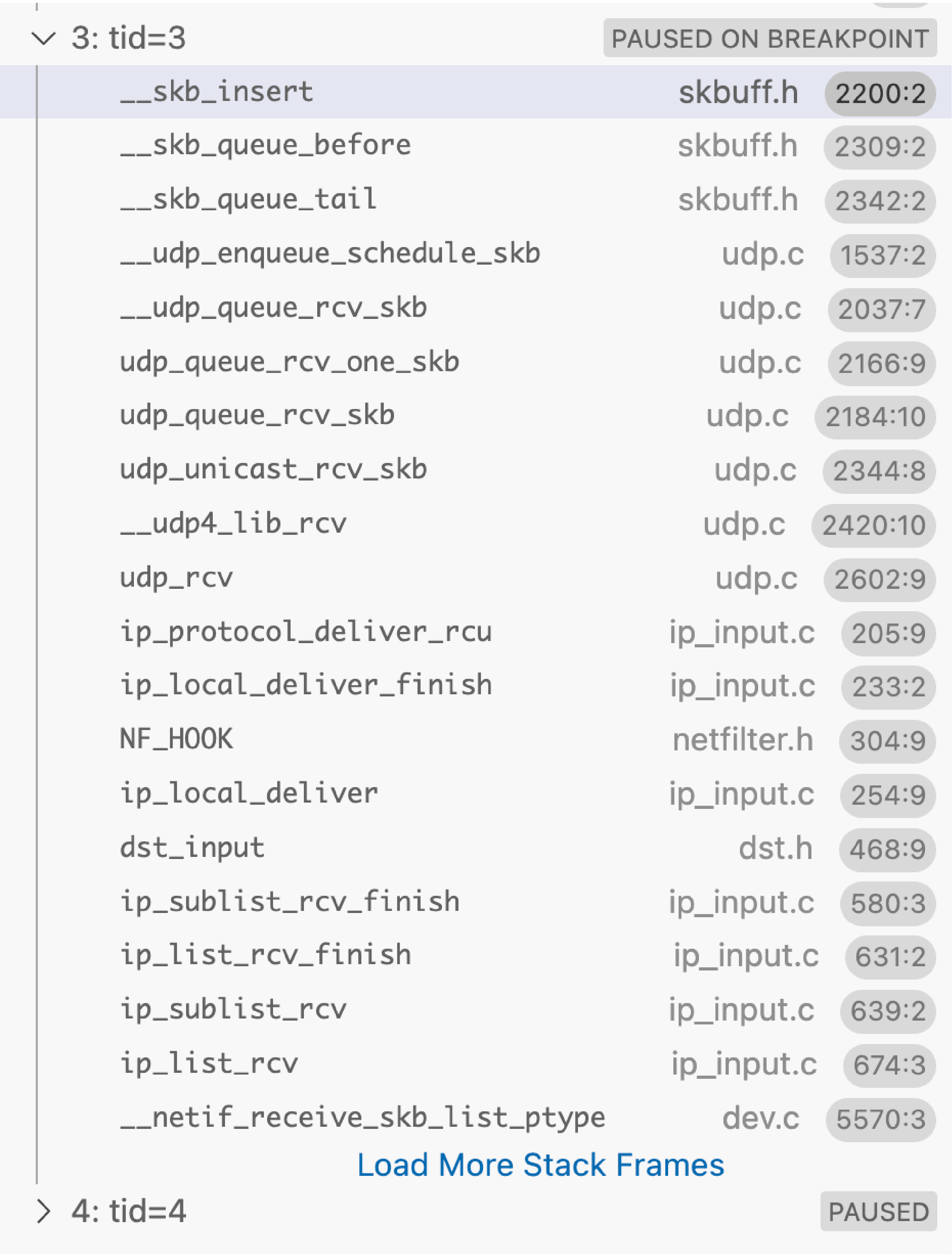
可选值	备注
lwtunnel_input	Kconfig - net/Kconfig - Linux source code (v5.14.7) - Bootlin

- int (_output)(struct net_net, struct sock _sk, struct sk_buff_skb);
    - 根据路由结果output指向不同函数。决定数据包如何发送

可选值	备注
dst_discard_out()	默认值
ip_output	本机生成，需要发送的单播包
ip_rt_bug()	bug?
xfrm4_output	要转发的数据包
ip_mc_output	本机生成，需发送的多播（multicast）包

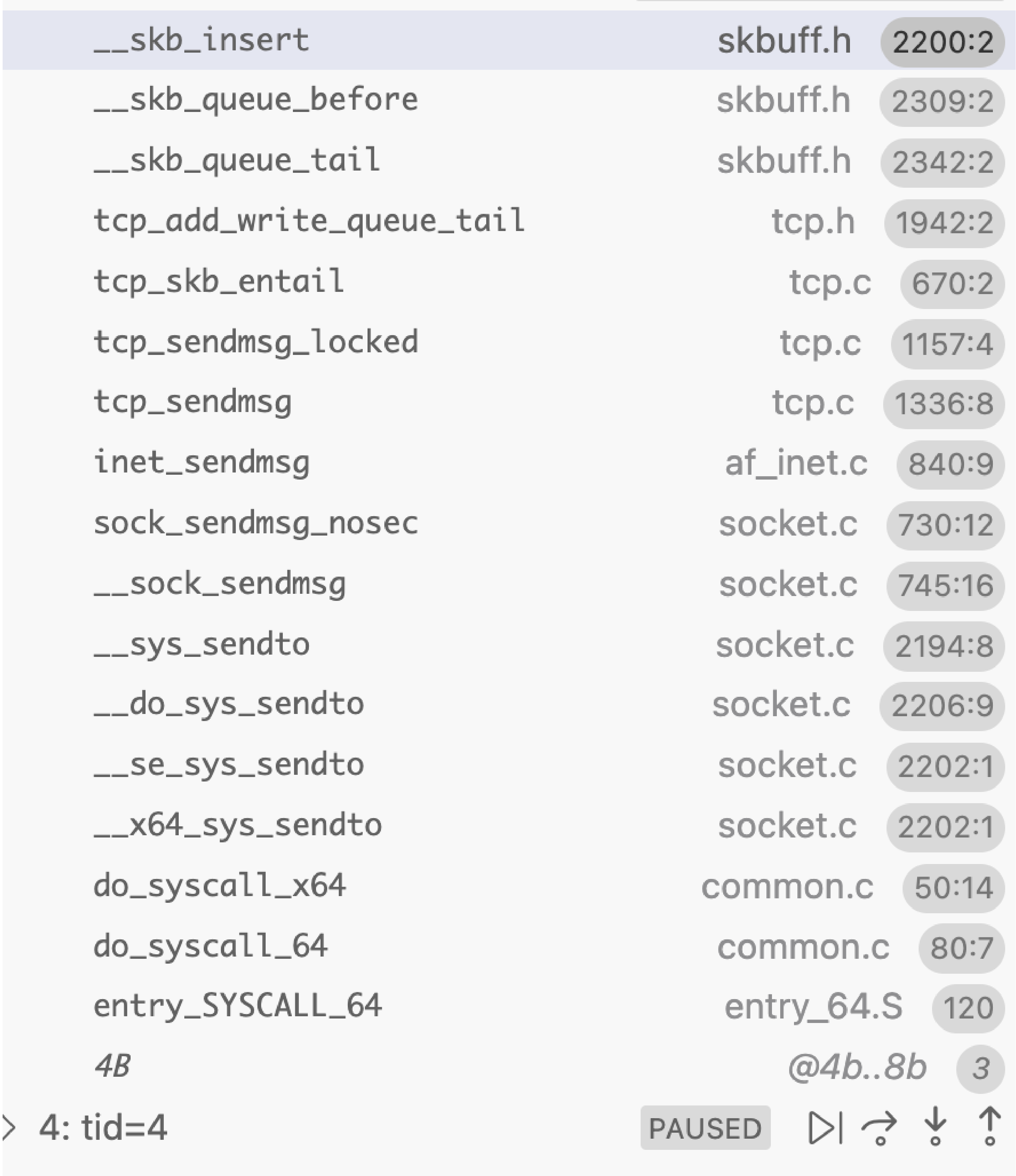
skb 到 sock

UDP sock send to skb



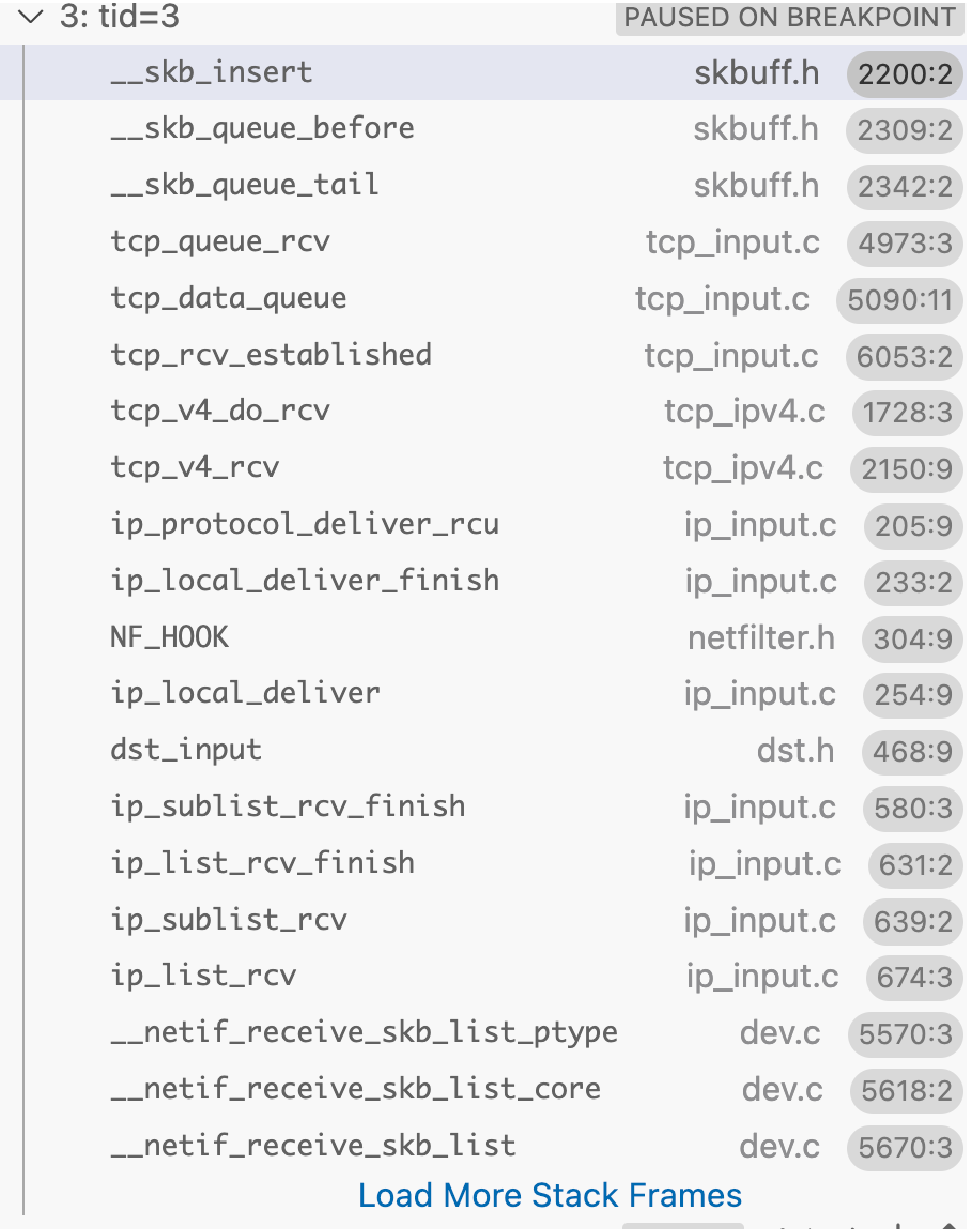
```
∨ 3: tid=3                                    PAUSED ON BREAKPOINT
    __skb_insert                              skbuff.h    2200:2
    __skb_queue_before                        skbuff.h    2309:2
    __skb_queue_tail                          skbuff.h    2342:2
    __udp_enqueue_schedule_skb                udp.c       1537:2
    __udp_queue_rcv_skb                       udp.c       2037:7
    udp_queue_rcv_one_skb                     udp.c       2166:9
    udp_queue_rcv_skb                         udp.c       2184:10
    udp_unicast_rcv_skb                       udp.c       2344:8
    __udp4_lib_rcv                            udp.c       2420:10
    udp_rcv                                   udp.c       2602:9
    ip_protocol_deliver_rcu                   ip_input.c  205:9
    ip_local_deliver_finish                   ip_input.c  233:2
    NF_HOOK                                   netfilter.h 304:9
    ip_local_deliver                          ip_input.c  254:9
    dst_input                                 dst.h       468:9
    ip_sublist_rcv_finish                     ip_input.c  580:3
    ip_list_rcv_finish                        ip_input.c  631:2
    ip_sublist_rcv                            ip_input.c  639:2
    ip_list_rcv                               ip_input.c  674:3
    __netif_receive_skb_list_ptype            dev.c       5570:3
                     Load More Stack Frames
> 4: tid=4                                                  PAUSED
```

tcp sock send



```
__skb_insert                    skbuff.h    2200:2
__skb_queue_before              skbuff.h    2309:2
__skb_queue_tail                skbuff.h    2342:2
tcp_add_write_queue_tail        tcp.h       1942:2
tcp_skb_entail                  tcp.c       670:2
tcp_sendmsg_locked              tcp.c       1157:4
tcp_sendmsg                     tcp.c       1336:8
inet_sendmsg                    af_inet.c   840:9
sock_sendmsg_nosec              socket.c    730:12
__sock_sendmsg                  socket.c    745:16
__sys_sendto                    socket.c    2194:8
__do_sys_sendto                 socket.c    2206:9
__se_sys_sendto                 socket.c    2202:1
__x64_sys_sendto                socket.c    2202:1
do_syscall_x64                  common.c    50:14
do_syscall_64                   common.c    80:7
entry_SYSCALL_64                entry_64.S  120
4B                              @4b..8b     3
> 4: tid=4
```

tcp sock receive from skb



```
3: tid=3                                          PAUSED ON BREAKPOINT
    __skb_insert                         skbuff.h      2200:2
    __skb_queue_before                   skbuff.h      2309:2
    __skb_queue_tail                     skbuff.h      2342:2
    tcp_queue_rcv                        tcp_input.c   4973:3
    tcp_data_queue                       tcp_input.c   5090:11
    tcp_rcv_established                  tcp_input.c   6053:2
    tcp_v4_do_rcv                        tcp_ipv4.c    1728:3
    tcp_v4_rcv                           tcp_ipv4.c    2150:9
    ip_protocol_deliver_rcu              ip_input.c    205:9
    ip_local_deliver_finish              ip_input.c    233:2
    NF_HOOK                              netfilter.h   304:9
    ip_local_deliver                     ip_input.c    254:9
    dst_input                            dst.h         468:9
    ip_sublist_rcv_finish                ip_input.c    580:3
    ip_list_rcv_finish                   ip_input.c    631:2
    ip_sublist_rcv                       ip_input.c    639:2
    ip_list_rcv                          ip_input.c    674:3
    __netif_receive_skb_list_ptype       dev.c         5570:3
    __netif_receive_skb_list_core        dev.c         5618:2
    __netif_receive_skb_list             dev.c         5670:3
```

参考

> Routing Decisions in the Linux Kernel - Part 1: Lookup and packet flow [Thermalcircle.de]
> networking:kernel_flow [Wiki]