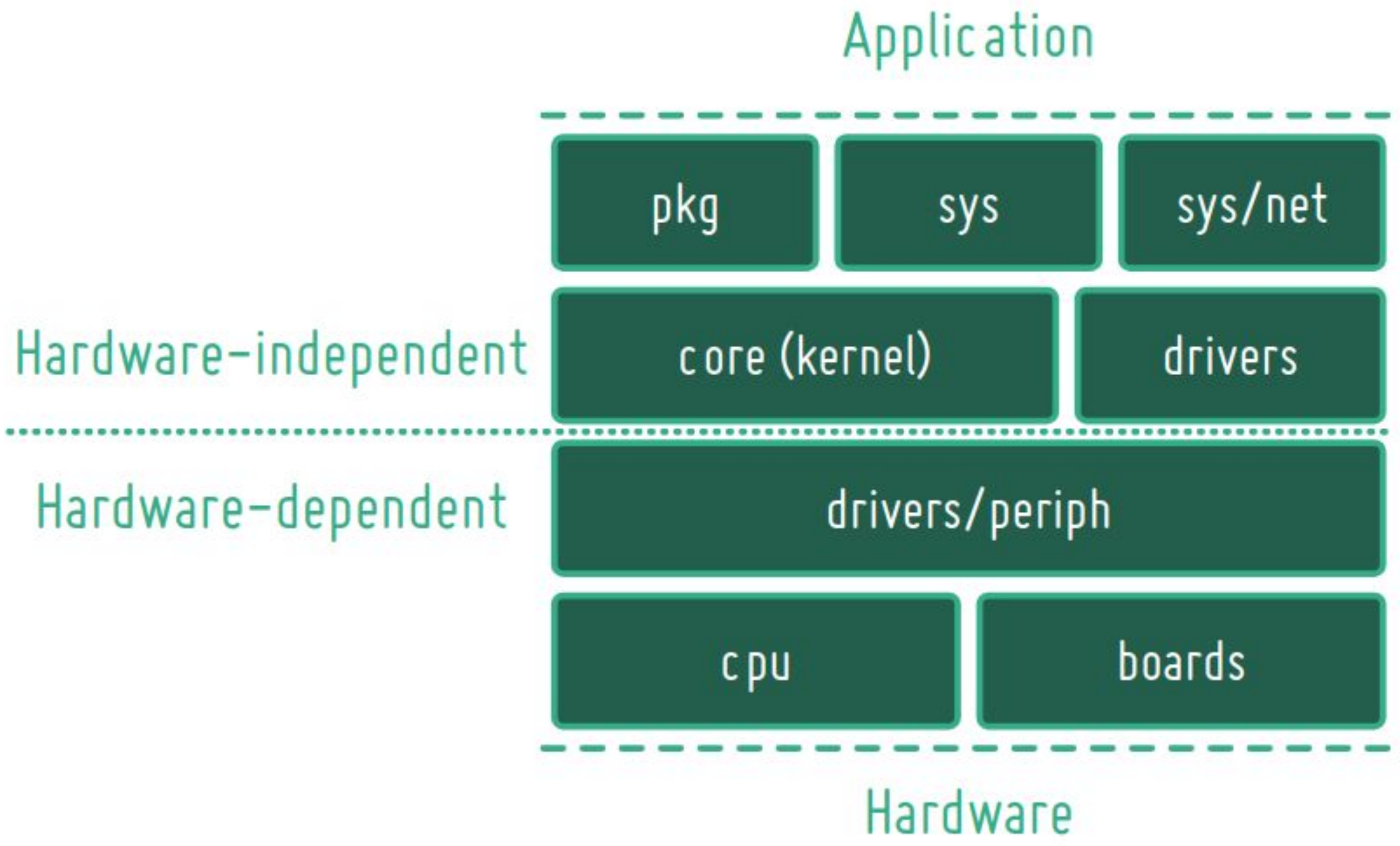
Interrupções no ~~Nanvix~~ RIOT-OS

Ad Nunes Ribeiro (11200620)
Eduardo Dias Defreyn (11203780)
Ricardo do Nascimento Boing (14200760)
UFSC | INE | CCO

Sumário

- Interrupt
  - Interrupt Handling
  - Interrupt Handling na CPU Native
  - Interrupt handler thread
  - IRQ Handling
- GNRC - Generic Network Stack
  - Packet buffer, netapi, netreg, sock, netdev
  - Propagation
- Demo case

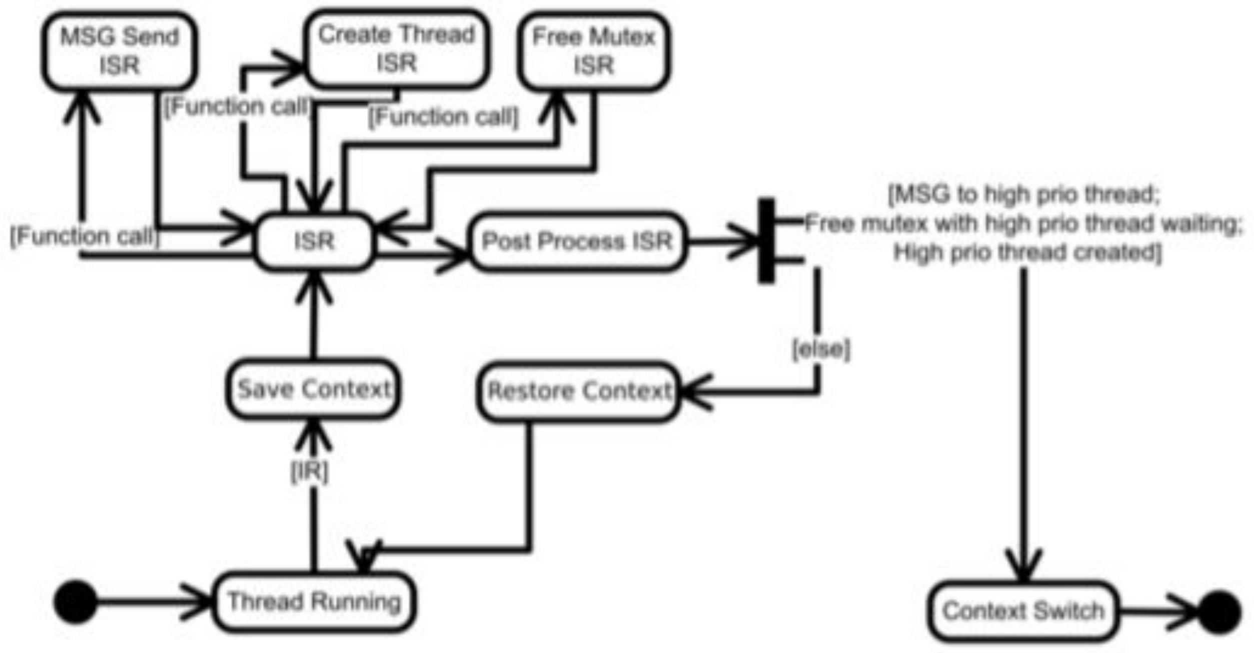
Arquitetura



Interrupt Handling

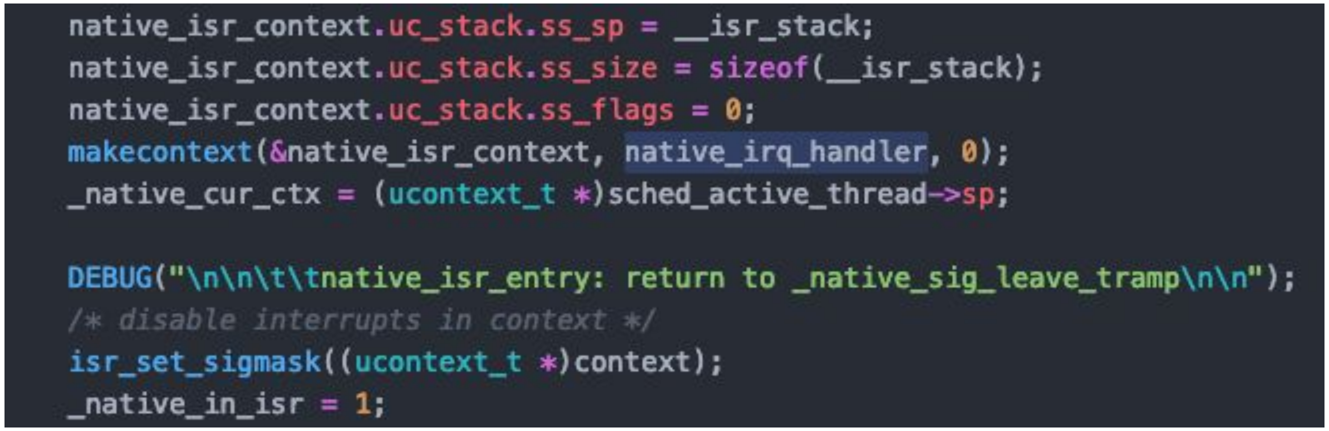
- Dever do Kernel:
  - Salvamento do Contexto
  - Chamada do ISR correspondente
  - Tomada de decisão sobre troca de contexto
- Dever do ISR:
  - Processamento da interrupção
  - Operações de retorno ao kernel

Interrupt Handling



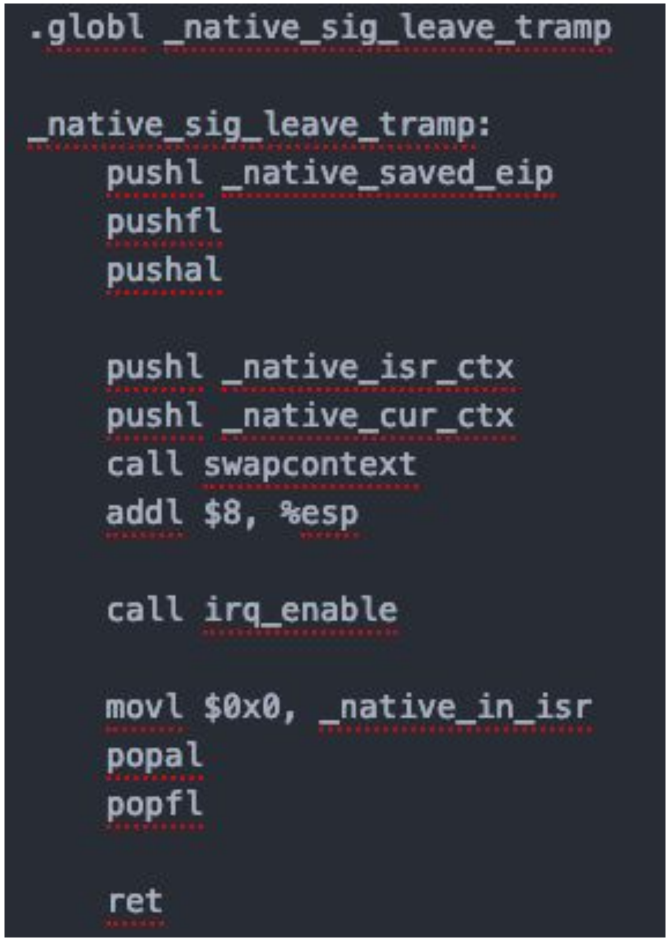
CPU Native

- Signal Handler
  - FIFO (First In FIrst Out)
  - Salva contexto da thread
  - Cria contexto do ISR
  - Chama a função trampolim



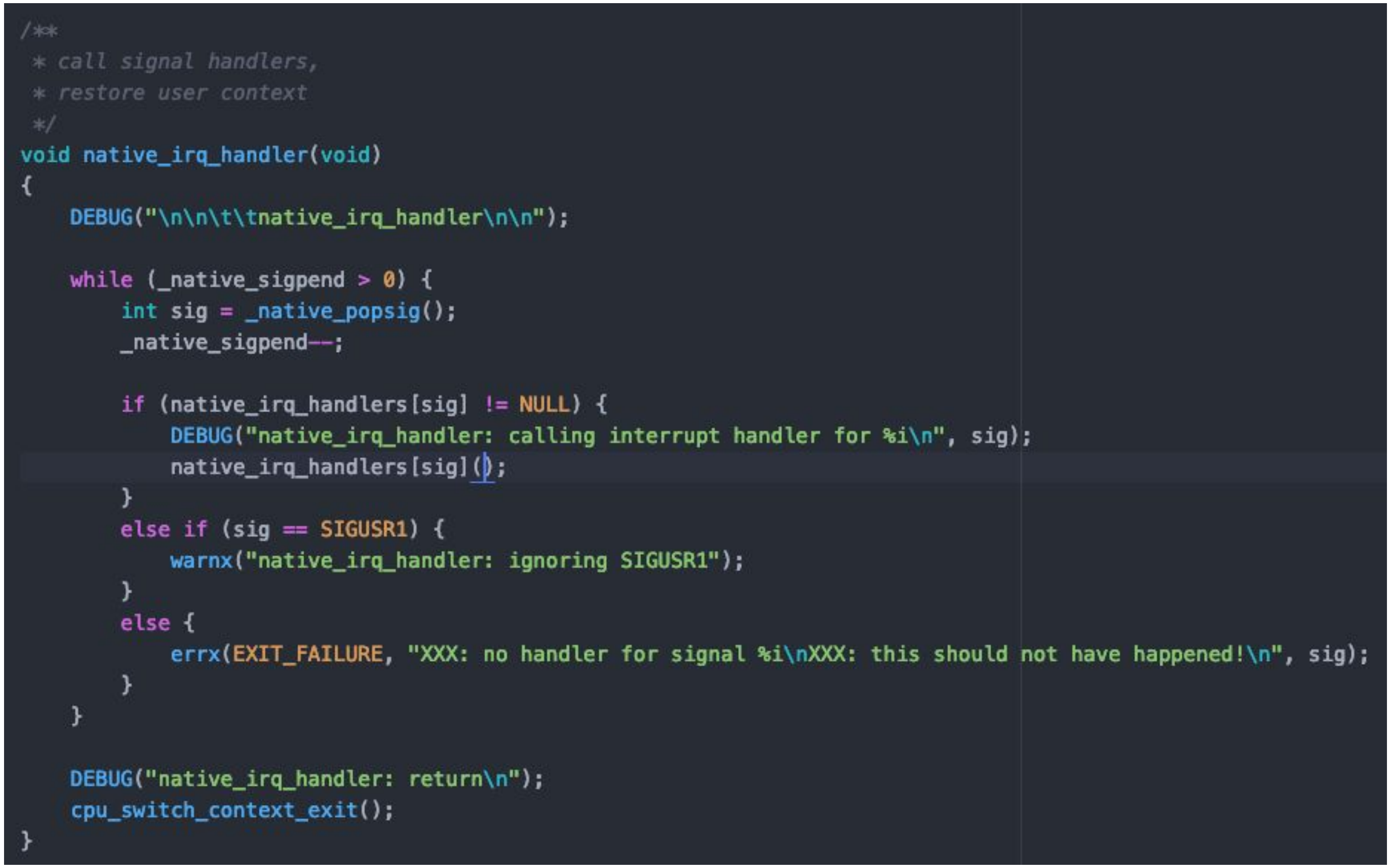
```
    native_isr_context.uc_stack.ss_sp = __isr_stack;
    native_isr_context.uc_stack.ss_size = sizeof(__isr_stack);
    native_isr_context.uc_stack.ss_flags = 0;
    makecontext(&native_isr_context, native_irq_handler, 0);
    _native_cur_ctx = (ucontext_t *)sched_active_thread->sp;

    DEBUG("\n\n\t\tnative_isr_entry: return to _native_sig_leave_tramp\n\n");
    /* disable interrupts in context */
    isr_set_sigmask((ucontext_t *)context);
    _native_in_isr = 1;
```

Função trampolim



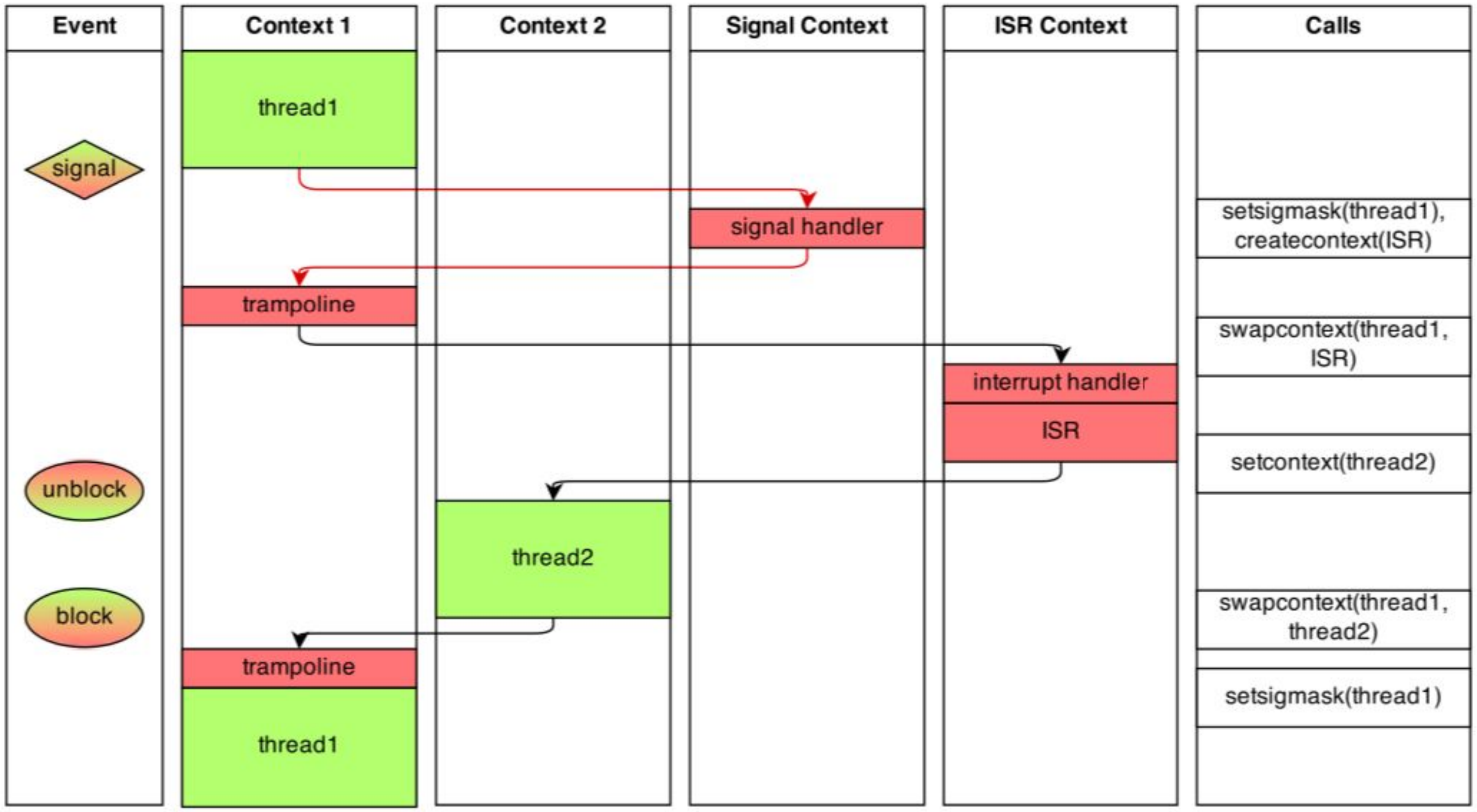
```
.globl _native_sig_leave_tramp

_native_sig_leave_tramp:
    pushl _native_saved_eip
    pushfl
    pushal

    pushl _native_isr_ctx
    pushl _native_cur_ctx
    call swapcontext
    addl $8, %esp

    call irq_enable

    movl $0x0, _native_in_isr
    popal
    popfl

    ret
```

IRQ Handler

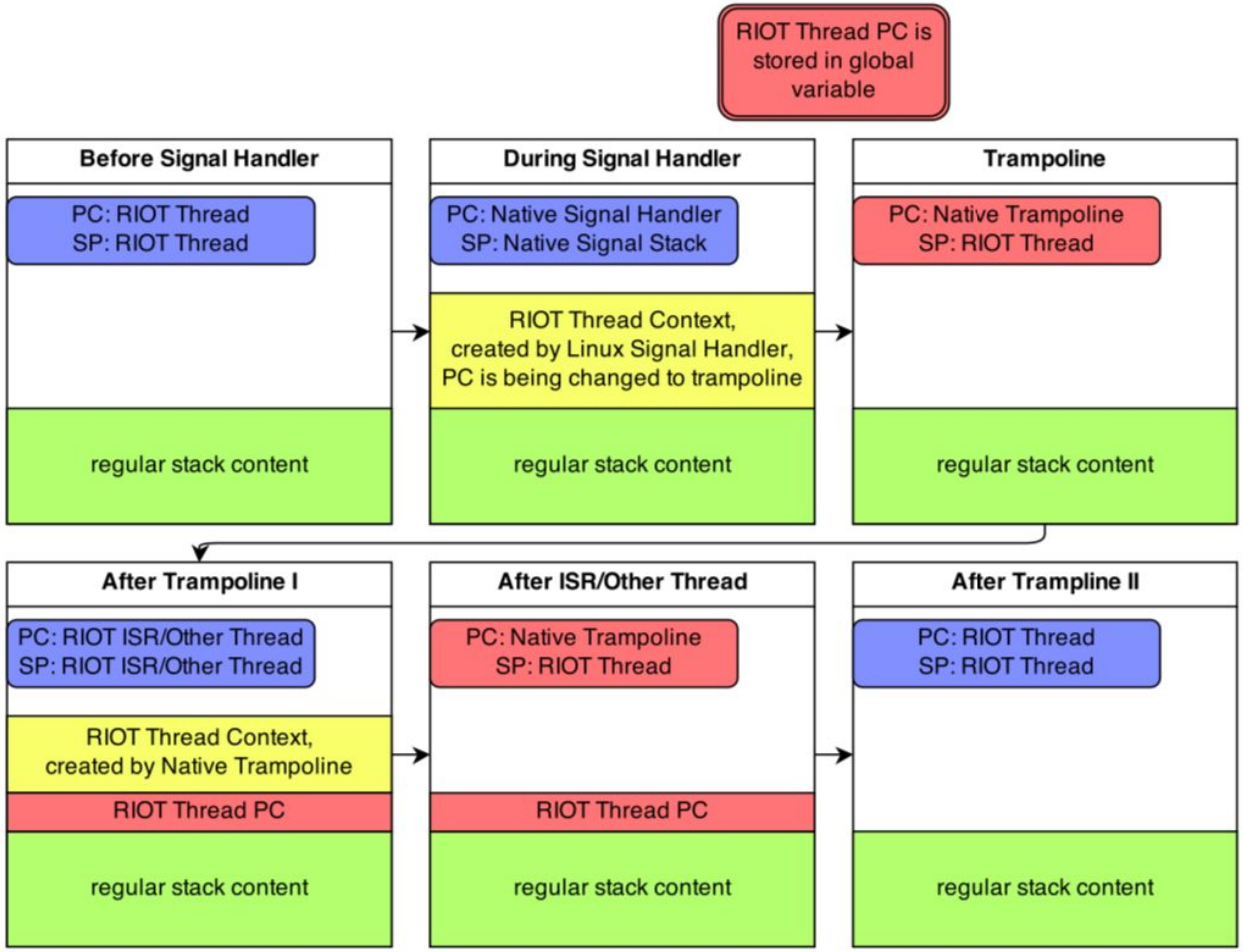


```
/**
 * call signal handlers,
 * restore user context
 */
void native_irq_handler(void)
{
    DEBUG("\n\n\t\tnative_irq_handler\n\n");

    while (_native_sigpend > 0) {
        int sig = _native_popsig();
        _native_sigpend--;

        if (native_irq_handlers[sig] != NULL) {
            DEBUG("native_irq_handler: calling interrupt handler for %i\n", sig);
            native_irq_handlers[sig]();
        }
        else if (sig == SIGUSR1) {
            warnx("native_irq_handler: ignoring SIGUSR1");
        }
        else {
            errx(EXIT_FAILURE, "XXX: no handler for signal %i\nXXX: this should not have happened!\n", sig);
        }
    }

    DEBUG("native_irq_handler: return\n");
    cpu_switch_context_exit();
}
```

Handling

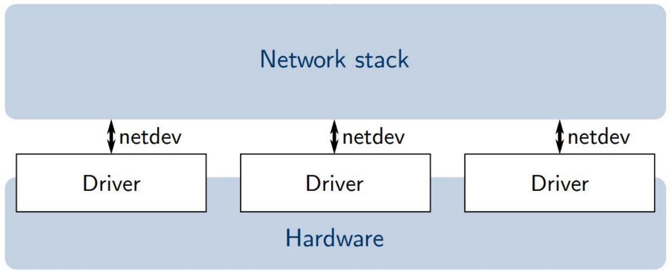

Event
Context 1
Context 2
Signal Context
ISR Context
Calls
thread1
signal
signal handler
setsigmask(thread1), createcontext(ISR)
trampoline
swapcontext(thread1, ISR)
interrupt handler
ISR
setcontext(thread2)
unblock
thread2
block
swapcontext(thread1, thread2)
trampoline
setsigmask(thread1)
thread1

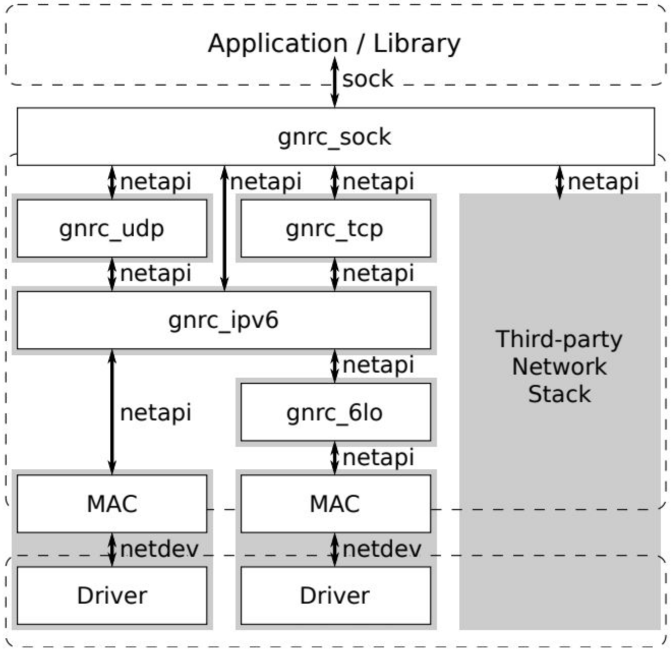
Estados da Pilha



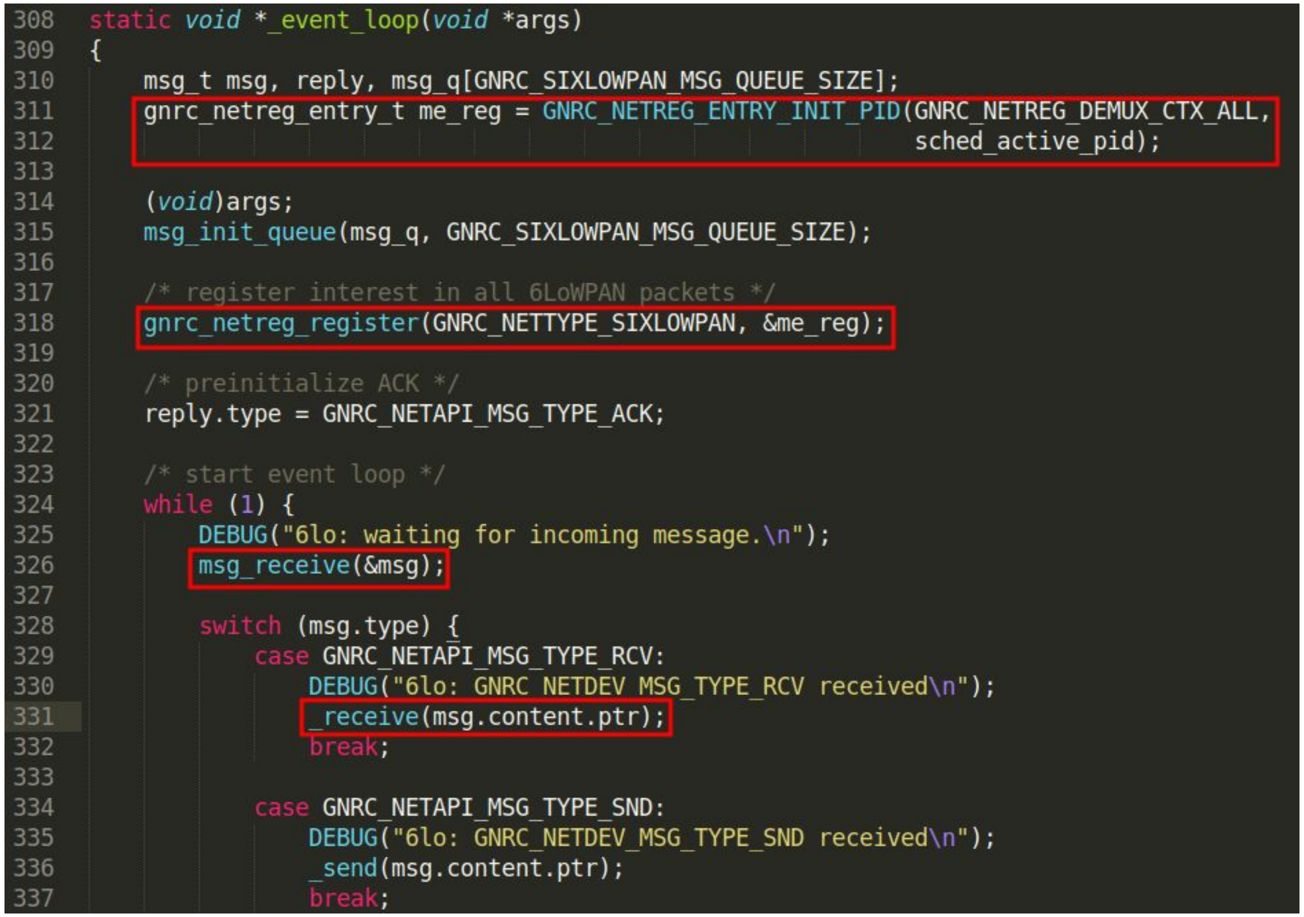
GNRC - Generic Network Stack

- Sock
- Netapi
- Netdev
- Netreg
- Buffer Packet



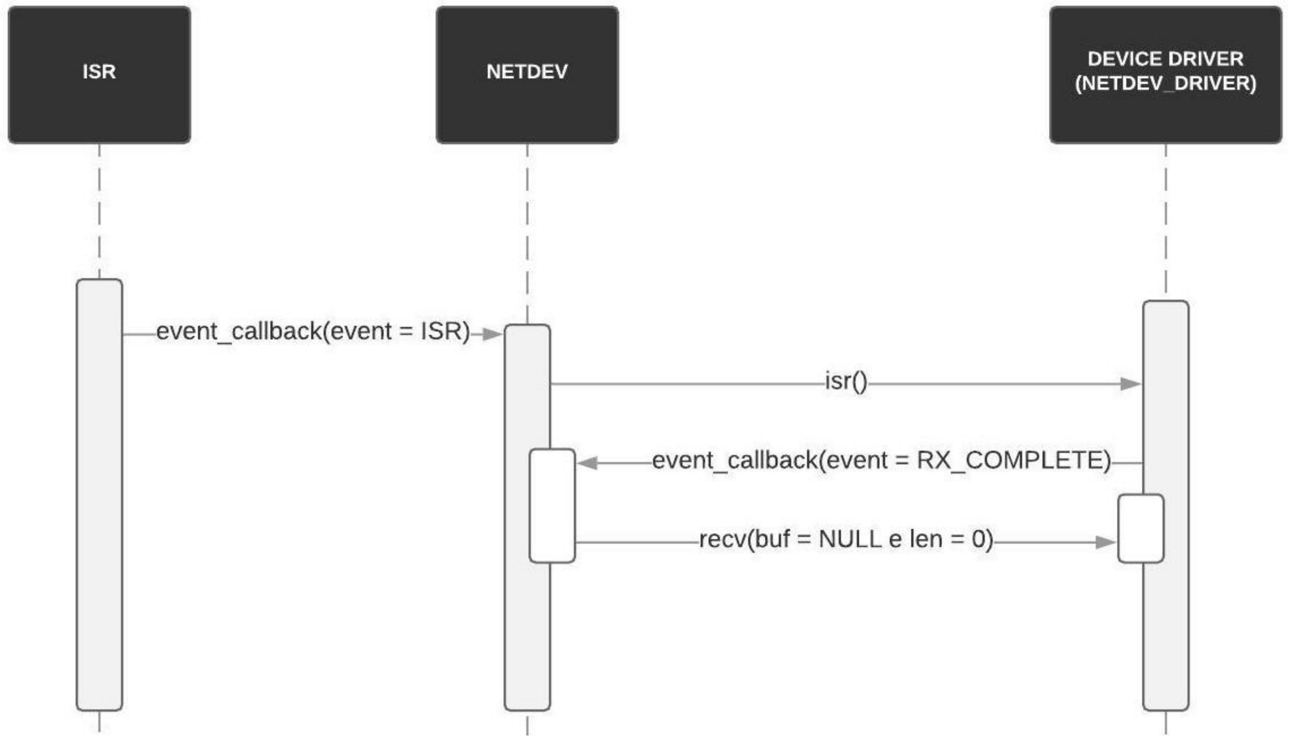


GNRC - Recebendo pacotes



```c
static void *_event_loop(void *args)
{
    msg_t msg, reply, msg_q[GNRC_SIXLOWPAN_MSG_QUEUE_SIZE];
    gnrc_netreg_entry_t me_reg = GNRC_NETREG_ENTRY_INIT_PID(GNRC_NETREG_DEMUX_CTX_ALL,
                                                            sched_active_pid);

    (void)args;
    msg_init_queue(msg_q, GNRC_SIXLOWPAN_MSG_QUEUE_SIZE);

    /* register interest in all 6LoWPAN packets */
    gnrc_netreg_register(GNRC_NETTYPE_SIXLOWPAN, &me_reg);

    /* preinitialize ACK */
    reply.type = GNRC_NETAPI_MSG_TYPE_ACK;

    /* start event loop */
    while (1) {
        DEBUG("6lo: waiting for incoming message.\n");
        msg_receive(&msg);

        switch (msg.type) {
            case GNRC_NETAPI_MSG_TYPE_RCV:
                DEBUG("6lo: GNRC_NETDEV_MSG_TYPE_RCV received\n");
                _receive(msg.content.ptr);
                break;

            case GNRC_NETAPI_MSG_TYPE_SND:
                DEBUG("6lo: GNRC_NETDEV_MSG_TYPE_SND received\n");
                _send(msg.content.ptr);
                break;
```

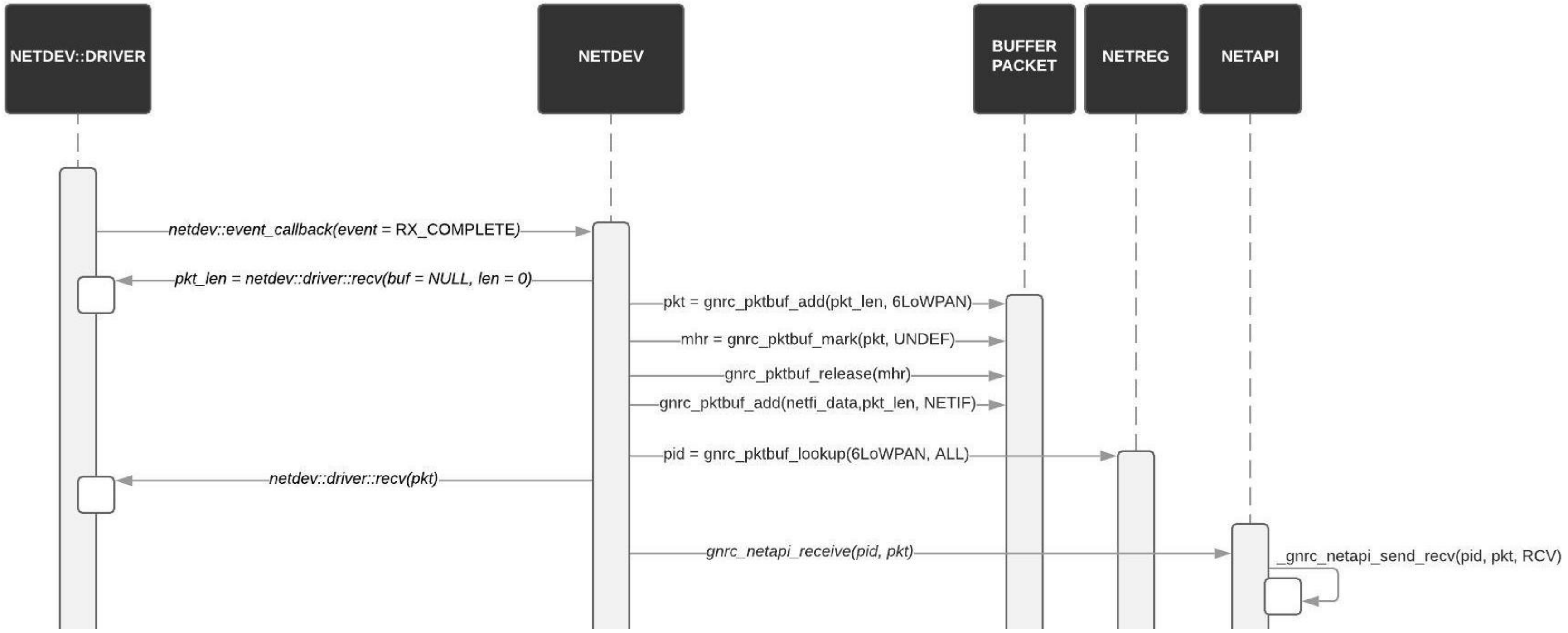
GNRC - Recebendo pacotes


ISR
NETDEV
DEVICE DRIVER (NETDEV_DRIVER)
event_callback(event = ISR)
isr()
event_callback(event = RX_COMPLETE)
recv(buf = NULL e len = 0)

No lugar de ISR: **NETDEV_EVENT_ISR**

No lugar de RX_COMPLETE: **NETDEV_EVENT_TX_COMPLETE**

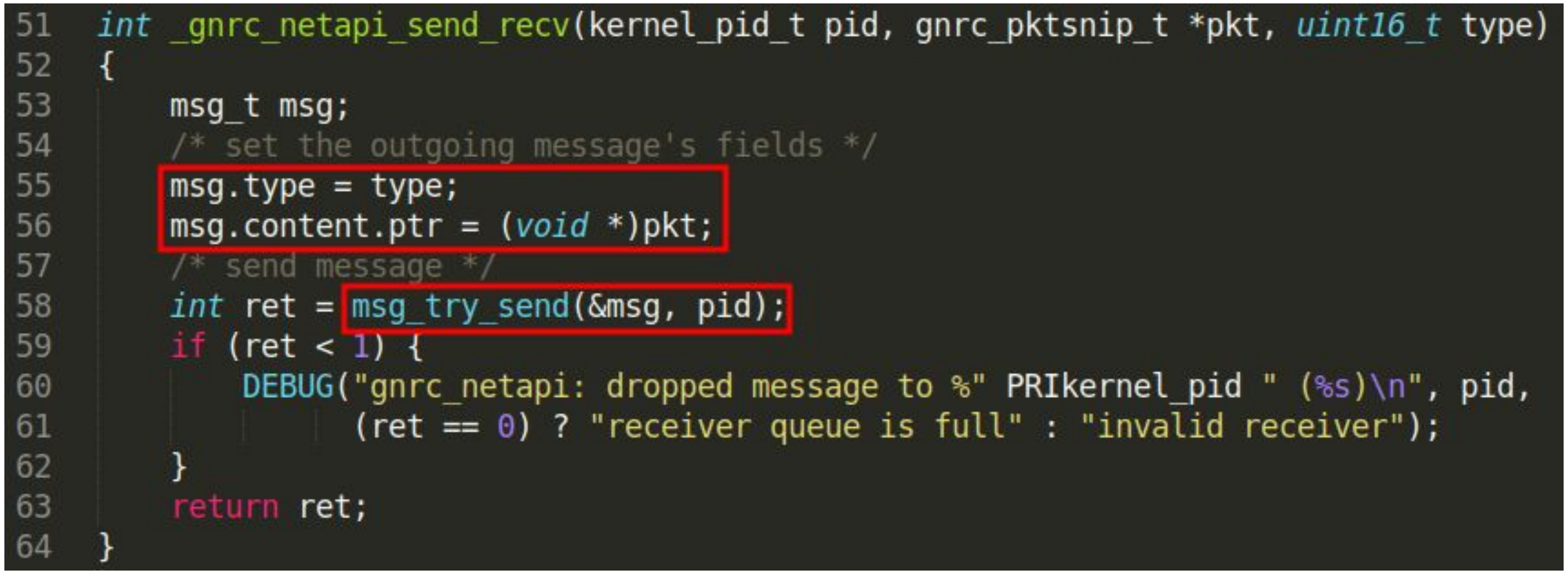
GNRC - Recebendo pacotes


NETDEV::DRIVER
NETDEV
BUFFER PACKET
NETREG
NETAPI
netdev::event_callback(event = RX_COMPLETE)
pkt_len = netdev::driver::recv(buf = NULL, len = 0)
pkt = gnrc_pktbuf_add(pkt_len, 6LoWPAN)
mhr = gnrc_pktbuf_mark(pkt, UNDEF)
gnrc_pktbuf_release(mhr)
gnrc_pktbuf_add(netfi_data,pkt_len, NETIF)
pid = gnrc_pktbuf_lookup(6LoWPAN, ALL)
netdev::driver::recv(pkt)
gnrc_netapi_receive(pid, pkt)
_gnrc_netapi_send_recv(pid, pkt, RCV)

No lugar de RX_COMPLETE: **NETDEV_EVENT_RX_COMPLETE**

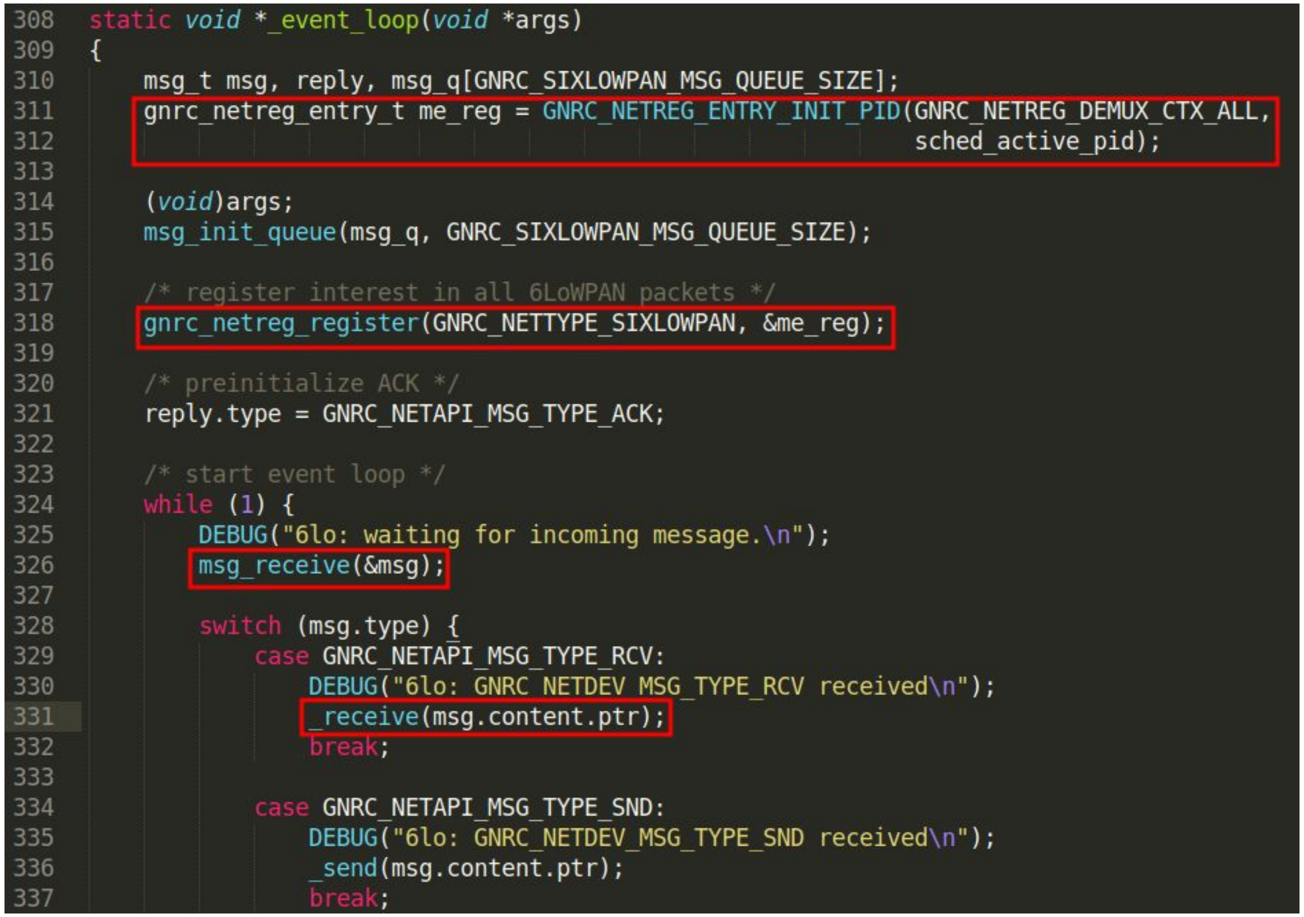
No lugar de RCV: **GNRC_NETAPI_MSG_TYPE_RCV**

GNRC - Recebendo pacotes



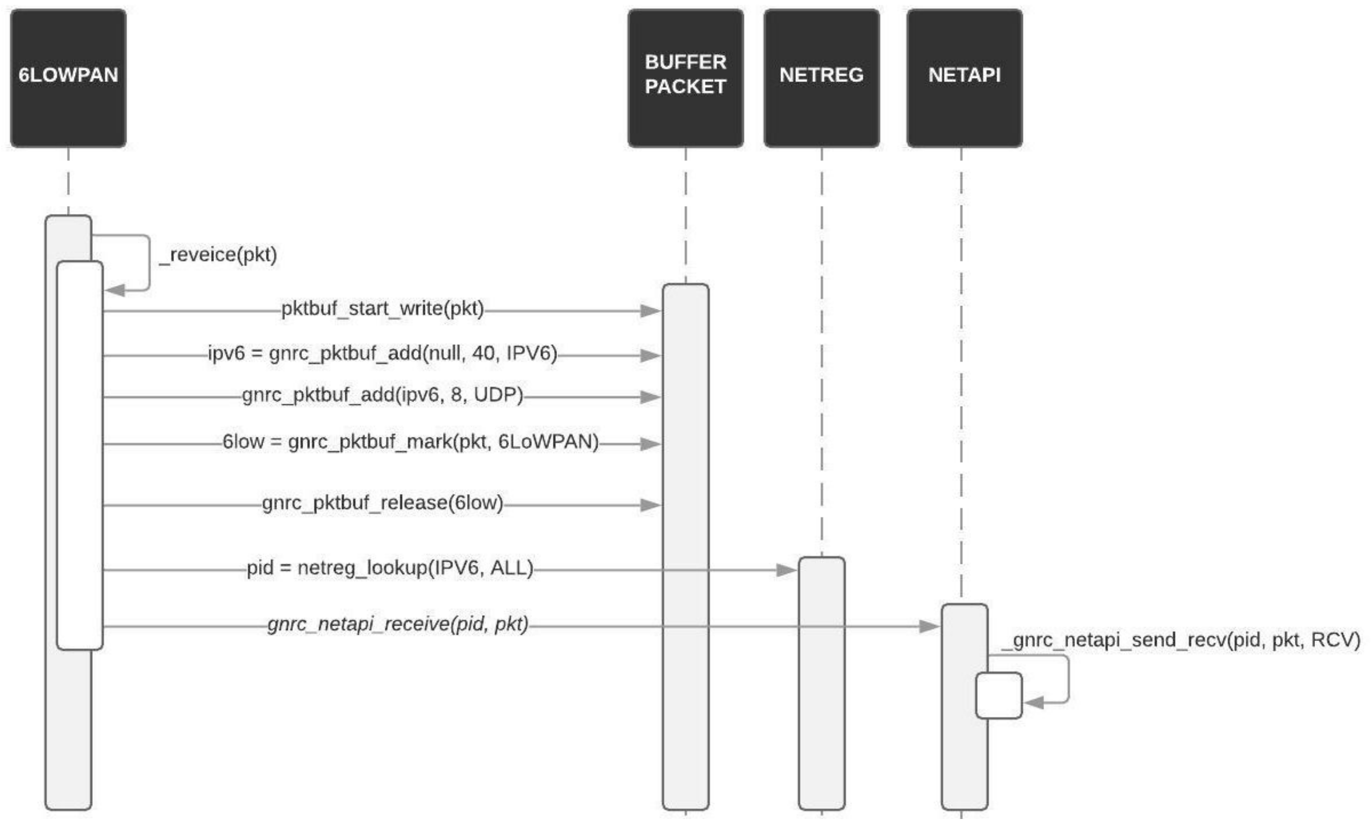
```c
int _gnrc_netapi_send_recv(kernel_pid_t pid, gnrc_pktsnip_t *pkt, uint16_t type)
{
    msg_t msg;
    /* set the outgoing message's fields */
    msg.type = type;
    msg.content.ptr = (void *)pkt;
    /* send message */
    int ret = msg_try_send(&msg, pid);
    if (ret < 1) {
        DEBUG("gnrc_netapi: dropped message to %" PRIkernel_pid " (%s)\n", pid,
                (ret == 0) ? "receiver queue is full" : "invalid receiver");
    }
    return ret;
}
```

GNRC - Recebendo pacotes



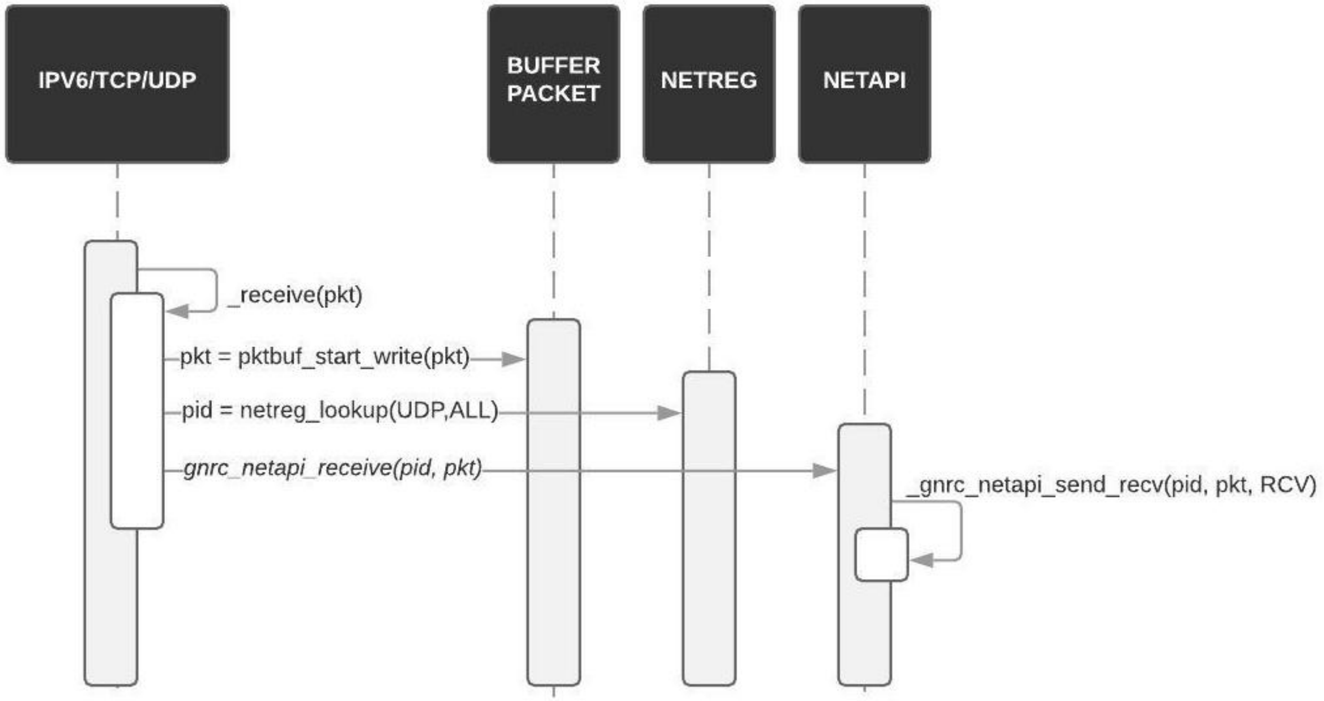
```
static void *_event_loop(void *args)
{
    msg_t msg, reply, msg_q[GNRC_SIXLOWPAN_MSG_QUEUE_SIZE];
    gnrc_netreg_entry_t me_reg = GNRC_NETREG_ENTRY_INIT_PID(GNRC_NETREG_DEMUX_CTX_ALL,
                                                            sched_active_pid);

    (void)args;
    msg_init_queue(msg_q, GNRC_SIXLOWPAN_MSG_QUEUE_SIZE);

    /* register interest in all 6LoWPAN packets */
    gnrc_netreg_register(GNRC_NETTYPE_SIXLOWPAN, &me_reg);

    /* preinitialize ACK */
    reply.type = GNRC_NETAPI_MSG_TYPE_ACK;

    /* start event loop */
    while (1) {
        DEBUG("6lo: waiting for incoming message.\n");
        msg_receive(&msg);

        switch (msg.type) {
            case GNRC_NETAPI_MSG_TYPE_RCV:
                DEBUG("6lo: GNRC_NETDEV_MSG_TYPE_RCV received\n");
                _receive(msg.content.ptr);
                break;

            case GNRC_NETAPI_MSG_TYPE_SND:
                DEBUG("6lo: GNRC_NETDEV_MSG_TYPE_SND received\n");
                _send(msg.content.ptr);
                break;
```

GNRC - Recebendo pacotes


6LOWPAN
BUFFER PACKET
NETREG
NETAPI
_reveice(pkt)
pktbuf_start_write(pkt)
ipv6 = gnrc_pktbuf_add(null, 40, IPV6)
gnrc_pktbuf_add(ipv6, 8, UDP)
6low = gnrc_pktbuf_mark(pkt, 6LoWPAN)
gnrc_pktbuf_release(6low)
pid = netreg_lookup(IPV6, ALL)
gnrc_netapi_receive(pid, pkt)
_gnrc_netapi_send_recv(pid, pkt, RCV)

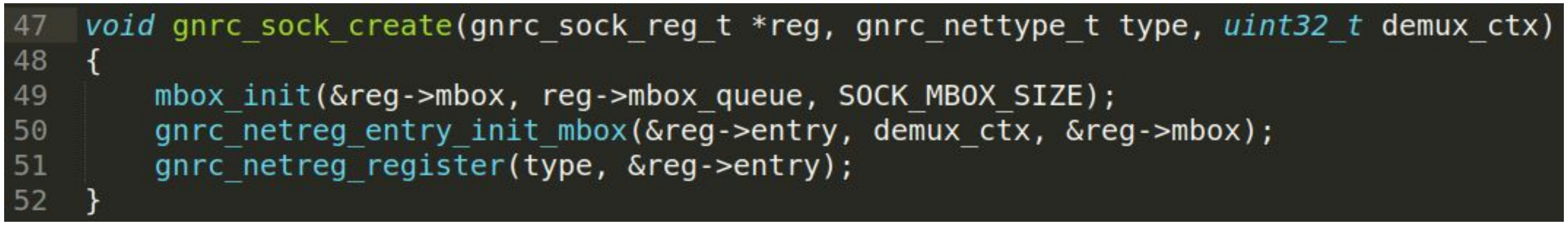
No lugar de RCV: **GNRC_NETAPI_MSG_TYPE_RCV**

GNRC - Recebendo pacotes

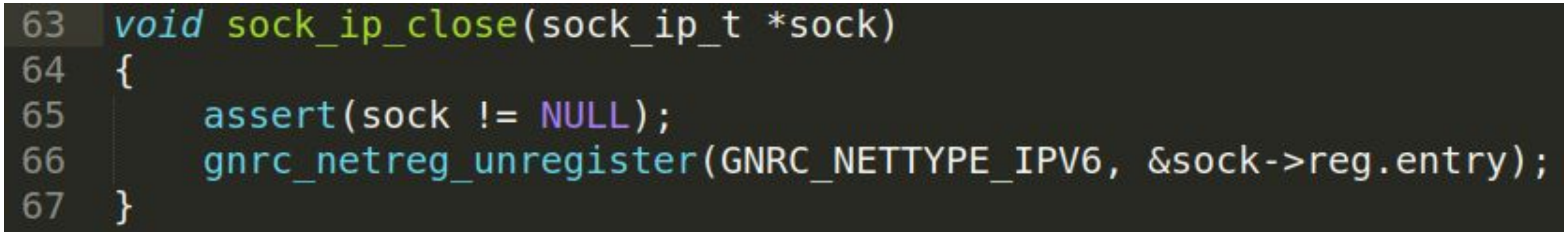

IPV6/TCP/UDP
BUFFER PACKET
NETREG
NETAPI
_receive(pkt)
pkt = pktbuf_start_write(pkt)
pid = netreg_lookup(UDP,ALL)
gnrc_netapi_receive(pid, pkt)
_gnrc_netapi_send_recv(pid, pkt, RCV)

No lugar de RCV: **GNRC_NETAPI_MSG_TYPE_RCV**

GNRC - Recebendo pacotes

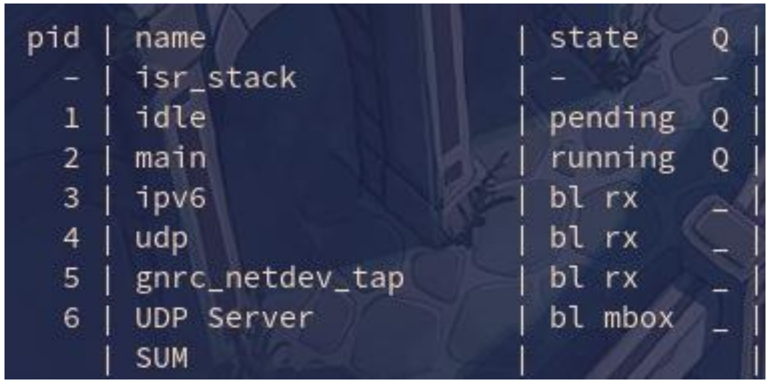


```c
void gnrc_sock_create(gnrc_sock_reg_t *reg, gnrc_nettype_t type, uint32_t demux_ctx)
{
    mbox_init(&reg->mbox, reg->mbox_queue, SOCK_MBOX_SIZE);
    gnrc_netreg_entry_init_mbox(&reg->entry, demux_ctx, &reg->mbox);
    gnrc_netreg_register(type, &reg->entry);
}
```

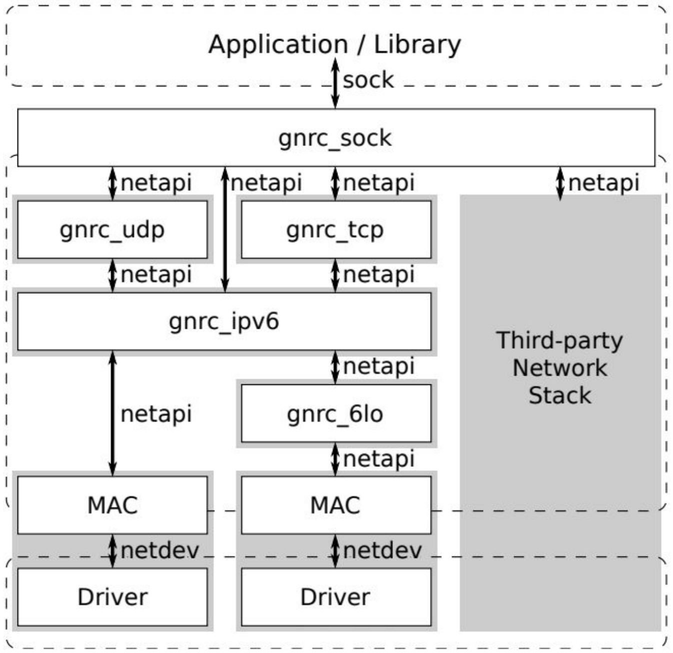


```c
void sock_ip_close(sock_ip_t *sock)
{
    assert(sock != NULL);
    gnrc_netreg_unregister(GNRC_NETTYPE_IPV6, &sock->reg.entry);
}
```

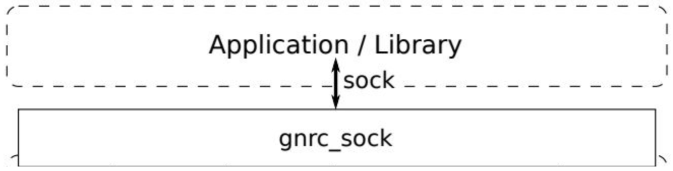
Demo - Recebimento de pacotes UDP

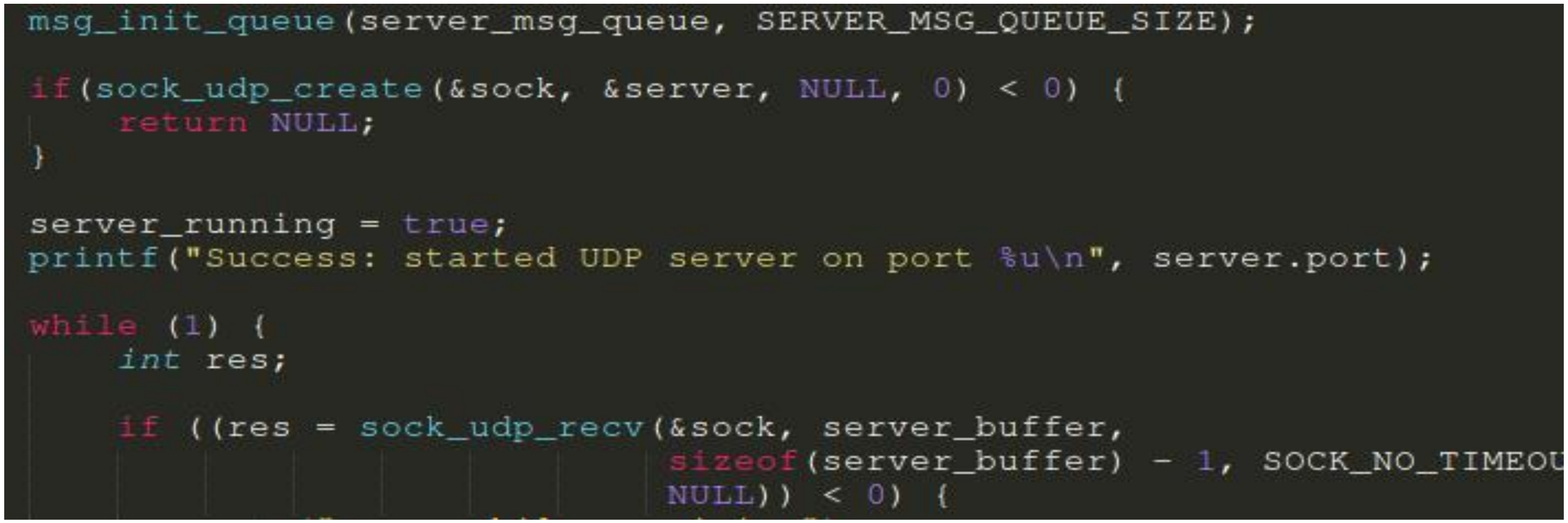


```
pid | name                  | state     Q |
  - | isr_stack             | -         - |
  1 | idle                  | pending   Q |
  2 | main                  | running   Q |
  3 | ipv6                  | bl rx     _ |
  4 | udp                   | bl rx     _ |
  5 | gnrc_netdev_tap       | bl rx     _ |
  6 | UDP Server            | bl mbox   _ |
    | SUM                   |             |
```



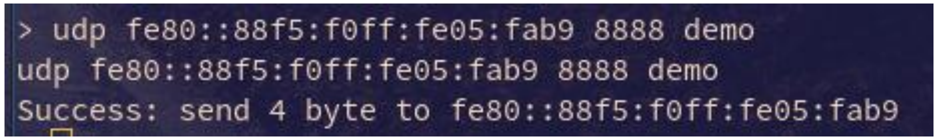
Demo - Recebimento de pacotes UDP



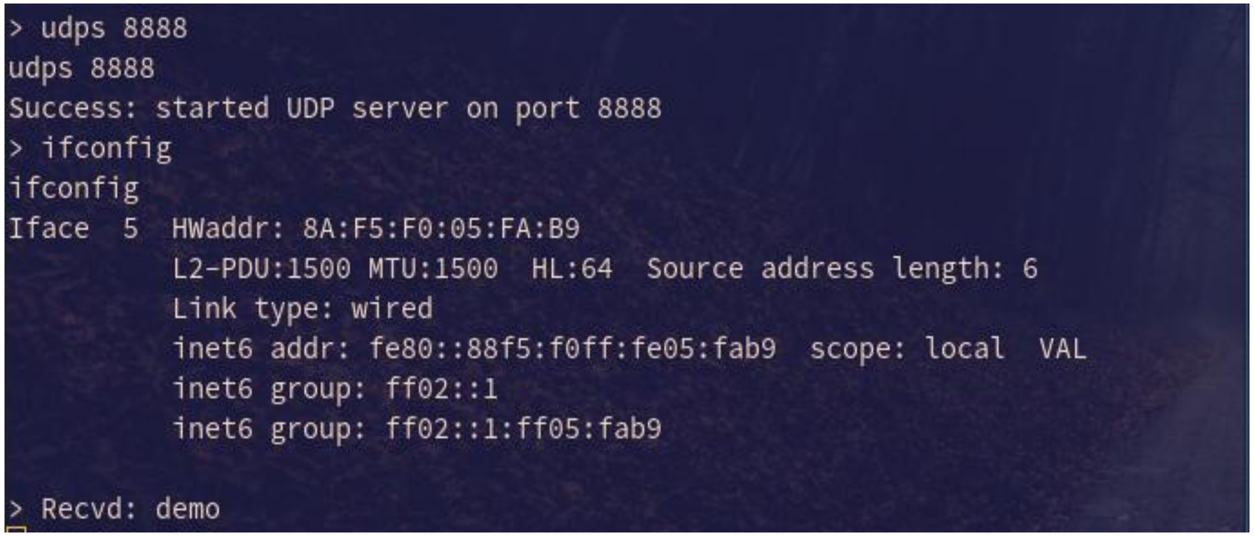


```
msg_init_queue(server_msg_queue, SERVER_MSG_QUEUE_SIZE);

if(sock_udp_create(&sock, &server, NULL, 0) < 0) {
    return NULL;
}

server_running = true;
printf("Success: started UDP server on port %u\n", server.port);

while (1) {
    int res;

    if ((res = sock_udp_recv(&sock, server_buffer,
                             sizeof(server_buffer) - 1, SOCK_NO_TIMEOU
                             NULL)) < 0) {
```

Demo - Recebimento de pacotes UDP



```
> udp fe80::88f5:f0ff:fe05:fab9 8888 demo
udp fe80::88f5:f0ff:fe05:fab9 8888 demo
Success: send 4 byte to fe80::88f5:f0ff:fe05:fab9
```



```
> udps 8888
udps 8888
Success: started UDP server on port 8888
> ifconfig
ifconfig
Iface  5  HWaddr: 8A:F5:F0:05:FA:B9
          L2-PDU:1500 MTU:1500  HL:64  Source address length: 6
          Link type: wired
          inet6 addr: fe80::88f5:f0ff:fe05:fab9  scope: local  VAL
          inet6 group: ff02::1
          inet6 group: ff02::1:ff05:fab9

> Recvd: demo
```

OBRIGADO!

adnunesribeiro@gmail.com
eduardo_dududex@hotmail.com
ricardoboing.ufsc@gmail.com