# Linux IP Networking

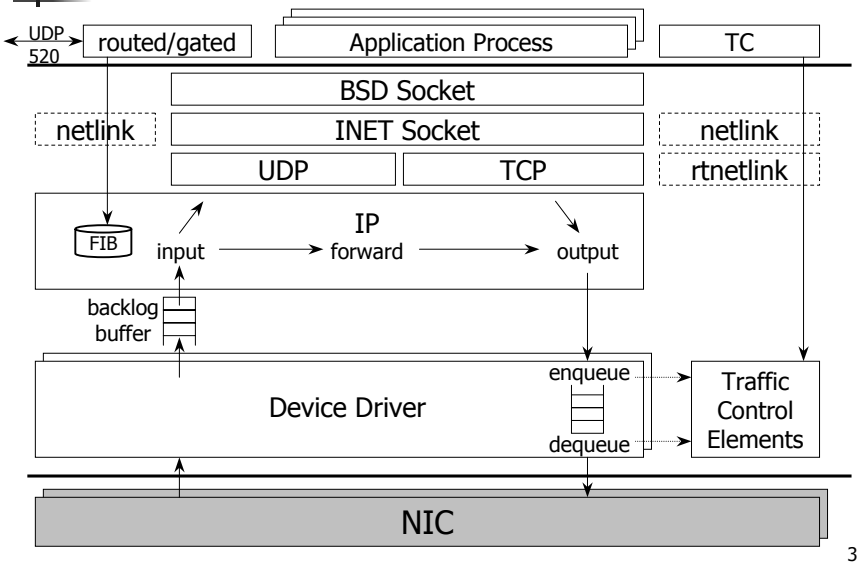
Antonio Salueña
<saluena@lut.fi>

## Preface

We will study linux networking for the following case:

- Intel x86 architecture
- IP packets
- Recent stable linux kernel series 2.4.x

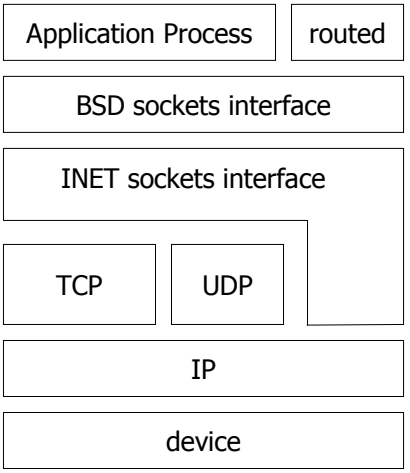
# Overview






# Linux Network Layers

- BSD sockets layer provides standard UNIX sockets API.
- INET layer manages communication end points for the IP based protocols TCP and UDP.
- Device – physical transmission.



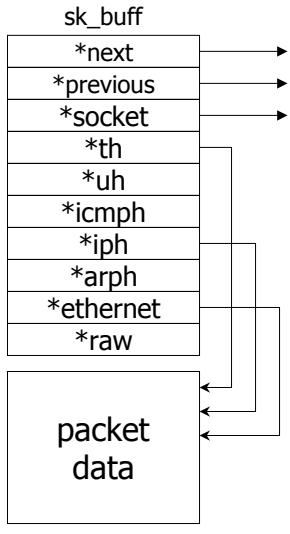
# Glossary

- Hardware Interrupt (Hardware IRQ).
- softirq – one of the 32 software interrupts which can run on multiple CPU at once.
- tasklet – a dinamicaly registrable software interrupt, which is guaranteed to only run on one CPU at time.
- Bottom Half – like softirq but only one bh can be running at any time on all CPUs, **deprecated**.
- Software Interrupt (Software IRQ) – general term for softirqs, tasklets and bottom halves.
- User Context – kernel executing on behalf of a particular process or kernel thread. Can be interrupted by software or hardware interrupt.
- Userspace – a process executing its own code outside the kernel.

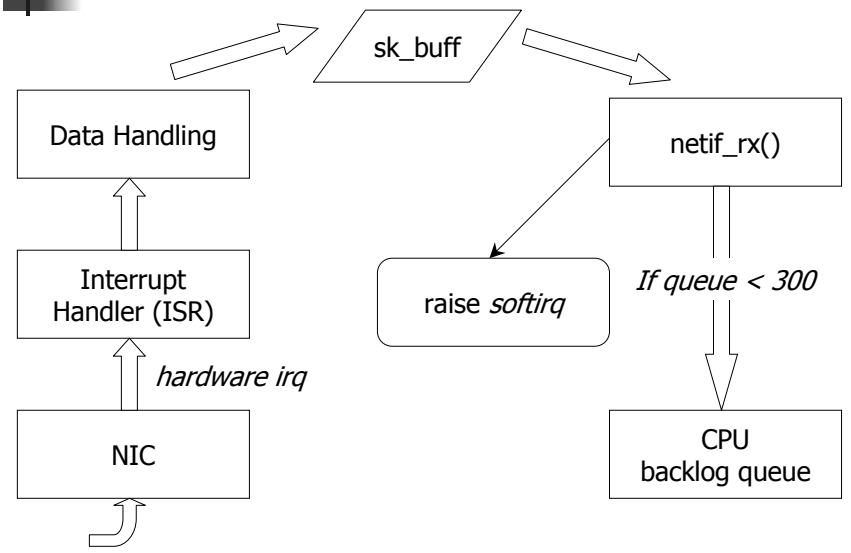


# Socket Buffer – sk_buff

- The key structure of linux networking code: common packet data structure for all protocol layers.
- Contains pointers to all protocol headers and length field that allow each protocol layer to manipulate data via standard functions (methods).
- Data is copied only twice:
  - from user space to kernel space
  - from kernel space to output medium (in case of an outbound packet)



# Packet Reception: ISR





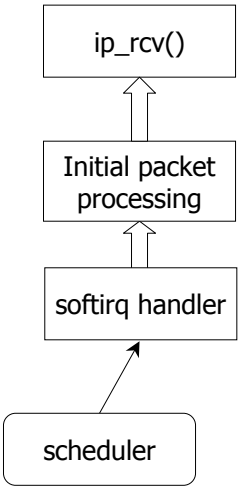

# Packet Reception: ISR

- Packet arrives on medium.
- NIC checks, stores packet – issues *hardware interrupt.*
- Network driver for this card handles irq: DEV_rx().
- Status checks, allocate sk_buff: dev_skb_alloc().
- Packet data is put from the system bus to sk_buff.
- Protocol type determined: eth_type_trans().
- Skb gets posted to network code incoming queue: netif_rx().
- The card status is updated and ISR is finished.
- In netif_rx() if queue is congested (max_backlog_queue=300) packet is dropped, otherwise skb get enqueued and receive *network softirq* is raised.

# Packet Reception: RX softirq

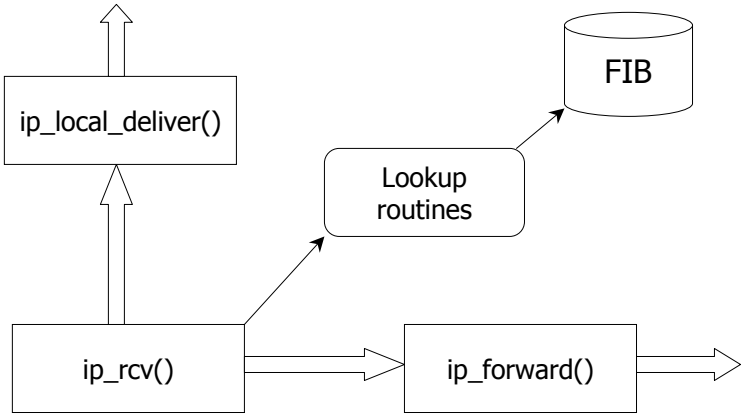
- *Network RX softirq* handler is called by scheduler
- Lock CPU device queue.
- Loop thru queue:
  - dequeue skb
  - find appropriate protocol handler
  - call protocol handler; ip_rcv() in case of IPv4

If queue is empty or time slice expired raise network softirq again and exit loop.






# Packet Reception: IP Layer



# Packet Reception: IP Layer

- ip_rcv() checks packet header for correctness, failed packets are dropped.
- First netfilter hook (NF_IP_PRE_ROUTING).
- After successful netfilter traversal: ip_rcv_finish().
- Packets destination is determined: ip_route_input() to access the FIB for route information.
- If the packet is for local host then it is passed to the upper layer: ip_local_deliver().
- Otherwise ip_forward() is called to rebuild and forward the packet to next destination.
- If the routing error occurred: ip_error().
- If it is a *multicast* packet and we have to do some multicast routing: ip_mr_input().



# Packet Reception: TCP Layer

- Ether tcp_rcv() or udp_rcv() is called to handle local packets.
- For the TCP protocol get_tcp_sock() is called to extract the port number and INET socket from the packet.
- tcp_data() is called to check that the packet is new data and discard duplicates.
- Finally, a hash lookup in the socket hash table is performed in order to forward the received packed to the correct INET socket.

# Packet Reception: Sockets

- After the protocol layer have finished with the received packet, INET socket interface will pass it to the BSD socket interface.
- A function data_ready() will wake up any process that is waiting at the BSD socket for the arriving packet.

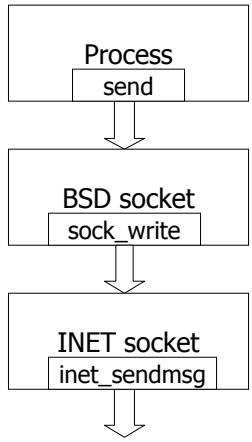


# Packet Forwarding

- ip_forward() is called.
- Check TTL field (drop packet if TTL ≤ 1).
- Check for skb space for destination device link header and expand if nessesary.
- Decrement TTL by one.
- Drop if "*don't fragment*" bit is set and packet needs fragmentation.
- Send ICMP message back to sender if there any problems.
- Netfilter hook (NF_IP_FORWARD).
- If netfilter accepts packet: ip_forward_finish().
- If we need to set additional ip options: ip_forward_options().
- ip_send(); ip_fragment() if packet is bigger than the destination device MTU.
- ip_finish_output(): netfilter (NF_IP_POST_ROUTING).
- If accepted, ip_finish_output2() prepends link header to skb and calls ip_output().

# Packet Sending: Sockets

- Application send data to the other side by using write() C function.
- sock_write() decides the next function to be called according to the address family the socket is associated with. I.E.: In case of INET socket address family sock_write() calls inet_sendmsg().






# Packet Sending: TCP Layer

- tcp_sendmsg() checks for a number of error conditions.
- do_tcp_sendmsg() allocates memory to hold skb.
- build_header() initialize TCP header.
- tcp_send_skb() add a variety of protocol-specific information to the header.
- Call queue_xmit() in skb structure which points to ip_queue_xmit().

# Packet Transmission

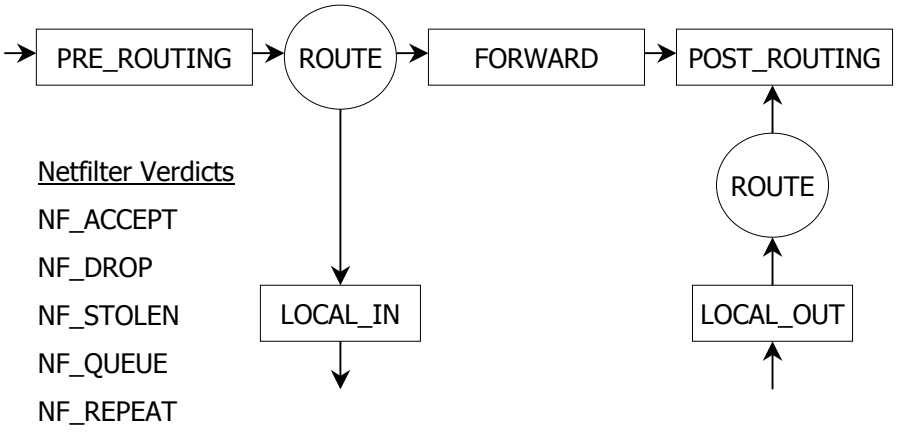
- If the destination interface is up and running send the packet by dev_send_xmit().
- Enqueue skb to the tail of the device output queue (priorities).
- Wake up device.
- Scheduler run device driver: hard_start_xmit().
- Test the medium.
- Send the link header.
- Tell the bus to transmit the packet over the medium.



# Netfilter Framework

- Netfilter is a framework for packet mangling for linux, outside the normal BSD sockets interface.
- Netfilter has three parts
  - Each protocol defines "hooks" well-defined points in a packets traversal of that protocol stack (IPv4 defines 5, IPv6 and DECnet hooks are similar).
  - Parts of the kernel can register to listen to the different hooks of each protocol (it is possible to examine, alter, discard, allow to pass or queue packet for userspace).
  - Packets that have been queued are collected for sending to userspace by the ip_queue driver.

# Netfilter Architecture: IP






# Packet Selection

- Packet selection system called IP Tables has been build over netfilter framework.
  - Kernel modules can register a new table and ask for packet to traverse a given table.
  - 'filter' table: hooks into local_in, forward and local_out points. For any given packet the one (and only one) place to filter it.
  - 'nat' table: network address translation table in pre_routing, post_routing and local_out.
  - Netfilter implements connection tracking mechanism in separate module using local_out and pre_routing.

# References


1. D. Karpov. Linux IP stack. Notes on a packet's traversal path trough the linux-2.4.0 network protocol stack. 2001.
2. M. Feng, R. Leung, A. Do-Sung Jun. Linux Networking Functions. 1999.
3. G. Herrin. Linux IP Networking. 2000.
4. H. Welte. Packet Jorney v1.4. <http://www.gnumonks.org/ftp/pub/doc/packet-journey-2.4.html>.
5. H. Welte. Skb – Linux network buffers. 2000. <http://www.gnumonks.org/ftp/pub/doc/skb-doc.html>.