Network packet capture in Linux kernelspace

An overview of the network stack in the Linux kernel

Beraldo Leal

beraldo@ime.usp.br
http://www.ime.usp.br/~beraldo/

Institute of Mathematics and Statistics - IME
University of Sao Paulo - USP

25th October 2011

Outline

Introduction

Network stack

Packet ingress flow

Methods to capture packets

Introduction

- Sniffers;
- Improvements in packet reception;
- Linux kernel network subsystem;

Sniffers

- tcpdump, wireshark, snort, etc;
- Using the well-known library libpcap;
- Not suitable for > 10 Gbps;
- Packet loss;

Improvements in packet reception

- Commodity hardware for packet capture;
  - 3COM
  - Intel
  - endace, ...
- Many Interruptions
- NEW API or NAPI (interruption coalescence)
- zero-copy
  - Direct Memory Access - DMA
  - mmap()

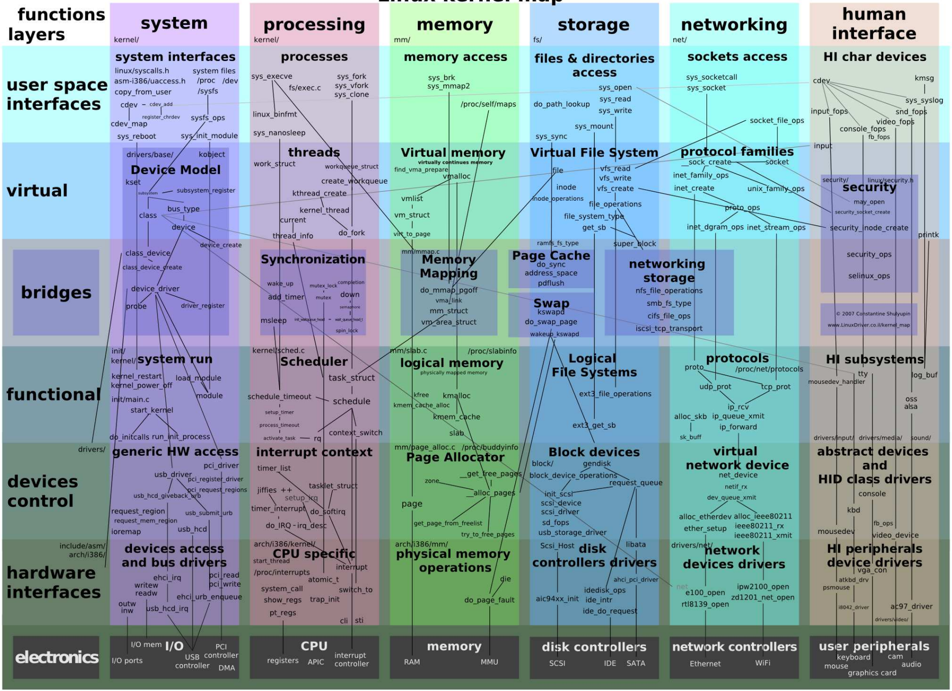
Linux kernel network subsystem

- Kernel number of files: 36.680 [1] [2]
- net/ number of files: 1.293 (3.5%)
- drivers/net/ number of files: 1.935 (5.27%)
- Kernel SLOC: 9.723.525
- net/ SLOC: 480.928 (5%)
- drivers/net/ SLOC: 1.155.317 (12%)

[1] kernel 3.0.0

[2] source: wc, find, cat, etc..

Linux kernel map



functions
layers

system
kernel/

processing
kernel/

memory
mm/

storage
fs/

networking
net/

human
interface

user space
interfaces

system interfaces
linux/syscalls.h
asm-i386/uaccess.h
copy_from_user
system files
/proc /dev
/sysfs
cdev – cdev_add
register_chrdev
cdev_map
sysfs_ops
sys_reboot
sys_init_module

processes
sys_execve
fs/exec.c
sys_fork
sys_vfork
sys_clone
linux_binfmt
sys_nanosleep

memory access
sys_brk
sys_mmap2
/proc/self/maps

files & directories
access
sys_open
sys_read
sys_write
do_path_lookup
sys_mount
sys_sync

sockets access
sys_socketcall
sys_socket
socket_file_ops

HI char devices
cdev
kmsg
sys_syslog
input_fops
snd_fops
video_fops
console_fops
fb_fops

virtual

drivers/base/
kobject
Device Model
kset
subsystem
subsystem_register
bus_type
class
device

threads
work_struct
workqueue_struct
create_workqueue
kthread_create
kernel_thread
current
thread_info
do_fork

Virtual memory
virtually continues memory
find_vma_prepare
vmalloc
vmlist
vm_struct
virt_to_page

Virtual File System
file
inode
inode_operations
vfs_read
vfs_write
vfs_create
file_operations
file_system_type
get_sb

protocol families
input
__sock_create
socket
inet_family_ops
inet_create
unix_family_ops
proto_ops
inet_dgram_ops
inet_stream_ops

security/
linux/security.h
security
may_open
security_socket_create
security_inode_create
printk

bridges

device_create
class_device
class_device_create
device_driver
probe
driver_register

Synchronization
wake_up
mutex_lock
completion
add_timer
mutex
down
semaphore
msleep
init_waitqueue_head – wait_queue_head_t
spin_lock

mm/mmap.c
Memory
Mapping
do_mmap_pgoff
vma_link
mm_struct
vm_area_struct

ramfs_fs_type
super_block
Page Cache
do_sync
address_space
pdflush

Swap
kswapd
do_swap_page
wakeup_kswapd

networking
storage
nfs_file_operations
smb_fs_type
cifs_file_ops
iscsi_tcp_transport

security_ops
selinux_ops

© 2007 Constantine Shulyupin
www.LinuxDriver.co.il/kernel_map

functional

init/
kernel/
system run
kernel_restart
kernel_power_off
load_module
module
init/main.c
start_kernel
do_initcalls
run_init_process

kernel/sched.c
Scheduler
task_struct
schedule_timeout
schedule
setup_timer
process_timeout
activate_task — rq
context_switch

mm/slab.c
/proc/slabinfo
logical memory
physically mapped memory
kfree
kmalloc
kmem_cache_alloc
kmem_cache
slab

Logical
File Systems
ext3_file_operations
ext3_get_sb

protocols
proto
/proc/net/protocols
udp_prot
tcp_prot
ip_rcv
alloc_skb
ip_queue_xmit
ip_forward
sk_buff

HI subsystems
tty
mousedev_handler
log_buf
oss
alsa
drivers/input/
drivers/media/
sound/

drivers/
devices
control

generic HW access
pci_driver
usb_driver
pci_register_driver
pci_request_regions
usb_hcd_giveback_urb
request_region
request_mem_region
usb_submit_urb
ioremap
usb_hcd

interrupt context
timer_list
tasklet_struct
jiffies ++
setup_irq
timer_interrupt
do_softirq
do_IRQ - irq_desc

mm/page_alloc.c
/proc/buddyinfo
Page Allocator
__get_free_pages
zone
__alloc_pages
page
get_page_from_freelist
try_to_free_pages

Block devices
block/
gendisk
block_device_operations
request_queue
init_scsi
scsi_device
scsi_driver
sd_fops
usb_storage_driver

virtual
network device
net_device
netif_rx
dev_queue_xmit
alloc_etherdev
alloc_ieee80211
ether_setup
ieee80211_rx
ieee80211_xmit

abstract devices
and
HID class drivers
console
kbd
fb_ops
mousedev
video_device

include/asm/
arch/i386/
hardware
interfaces

devices access
and bus drivers
ehci_irq
pci_read
pci_write
writew
readw
ehci_urb_enqueue
outw
inw
usb_hcd_irq

arch/i386/kernel/
CPU specific
start_thread
interrupt
/proc/interrupts
atomic_t
system_call
switch_to
show_regs
trap_init
pt_regs
cli sti

arch/i386/mm/
physical memory
operations
die
do_page_fault

Scsi_Host
disk
controllers drivers
libata
ahci_pci_driver
idedisk_ops
aic94xx_init
ide_intr
ide_do_request

drivers/net/
network
devices drivers
net
e100_open
ipw2100_open
zd1201_net_open
rtl8139_open

HI peripherals
device drivers
vga_con
atkbd_drv
psmouse
i8042_driver
ac97_driver
drivers/video/

electronics

I/O mem
I/O
I/O ports
USB
controller
PCI
controller
DMA

CPU
registers
APIC
interrupt
controller

memory
RAM
MMU

disk controllers
SCSI
IDE
SATA

network controllers
Ethernet
WiFi

user peripherals
keyboard
mouse
cam
audio
graphics card

Network stack

L5: Application

http, ftp, ssh, telnet, ... (message)

L4: Transport

tcp, udp, ... (segment)

L3: Network

ipv4, ipv6, ... (datagram/packet)

L1/2: Link / host-to-network

ethernet, token ring, ... (frame)

Important data structs:

- net_device
    - include/linux/netdevice.h
- sk_buff
    - include/linux/skbuff.h

Important data structs:

- net_device (include/linux/netdevice.h)
  - unsigned int mtu
  - unsigned int flags
  - unsigned char dev_addr[MAX_ADDR_LEN]
  - int promiscuity

Important data structs:

- sk_buff (include/linux/skbuff.h)
  - struct sk_buff *next;
  - struct sk_buff *prev;
  - ktime_t tstamp;
  - struct net_device *dev;
  - unsigned int len;
  - unsigned int data_len;
  - __u16 mac_len;
  - __u8 pkt_type;
  - __be16 protocol;
  - sk_buff_data_t transport_header; (old h)
  - sk_buff_data_t network_header; (old nh)
  - sk_buff_data_t mac_header; (old mac)

Important sk_buff routines

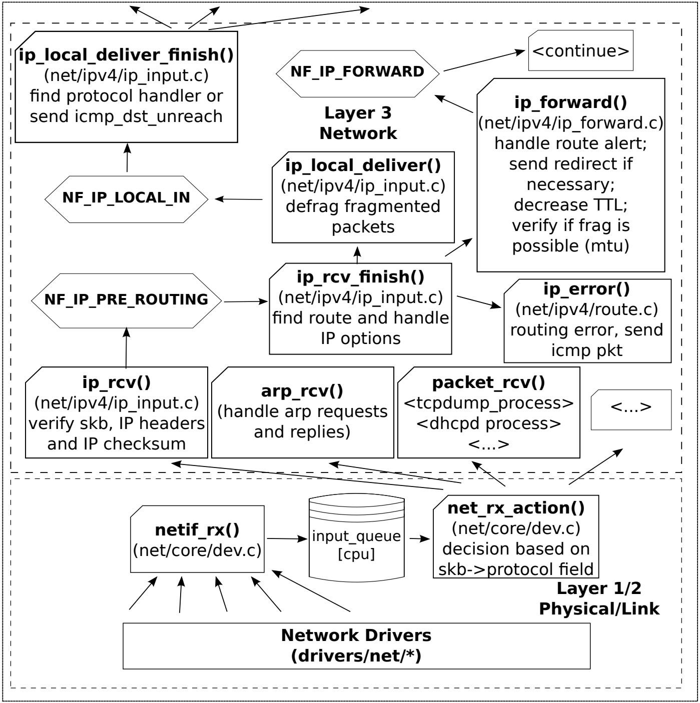
- `alloc_skb();`
- `dev_alloc_skb();`
- `kfree_skb();`
- `dev_kfree_skb();`
- `skb_clone();`
- `skb_network_header(skb);`
- `skb_transport_header(skb);`
- `skb_mac_header(skb);`

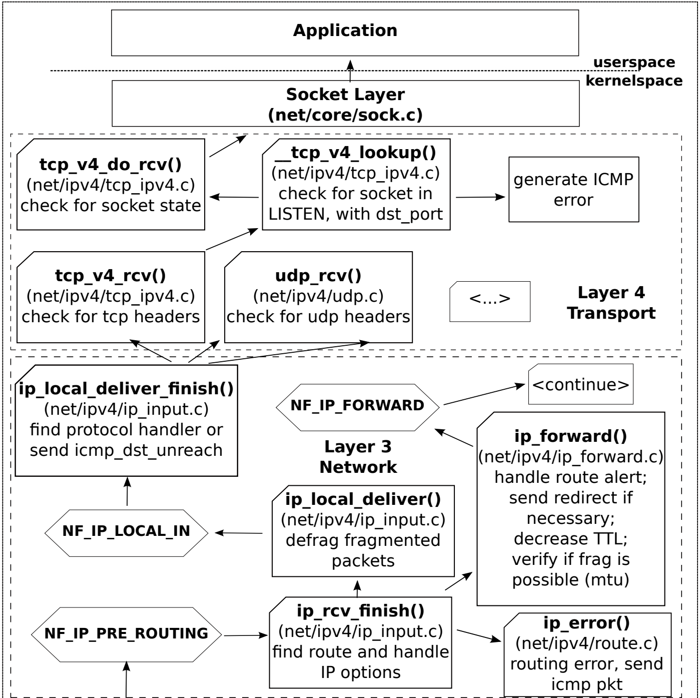
Packet ingress flow

- When working in interrupt driven model, the nic registers an interrupt handler;
- This interrupt handler will be called when a frame is received;
- Typically in the handler, we allocate `sk_buff` by calling `dev_alloc_skb()`;
- Copies data from nic's buffer to this struct just created;
- nic call generic reception routine `netif_rx()`;
- `netif_rx()` put frame in per cpu queue;
- if queue is full, drop!
- `net_rx_action()` decision based on `skb->protocol`;
- This function basically dequeues the frame and delivery a copy for every protocol handler;
    - `ptype_all` and `ptype_base` queues

Packet ingress flow

- `ip_v4_rcv()` will receive the ip datagram (if is a ipv4 packet);
- ip checksum, check ip headers, ...
- `ip_rcv_finish()` makes route decision (`ip_forward()` or `ip_local_delivery()`)
- `ip_local_delivery()` defrag fragmented packets, and call `ip_local_deliver_finish()`
- `ip_local_deliver_finish()` find protocol handler again;
- `tcp_v4_rcv()`, `udp_rcv()`, or other L4 protocol handler
- ...


ip_local_deliver_finish()
(net/ipv4/ip_input.c)
find protocol handler or
send icmp_dst_unreach
NF_IP_FORWARD
<continue>
Layer 3
Network
ip_forward()
(net/ipv4/ip_forward.c)
handle route alert;
send redirect if
necessary;
decrease TTL;
verify if frag is
possible (mtu)
ip_local_deliver()
(net/ipv4/ip_input.c)
defrag fragmented
packets
NF_IP_LOCAL_IN
NF_IP_PRE_ROUTING
ip_rcv_finish()
(net/ipv4/ip_input.c)
find route and handle
IP options
ip_error()
(net/ipv4/route.c)
routing error, send
icmp pkt
ip_rcv()
(net/ipv4/ip_input.c)
verify skb, IP headers
and IP checksum
arp_rcv()
(handle arp requests
and replies)
packet_rcv()
<tcpdump_process>
<dhcpd process>
<...>
<...>
netif_rx()
(net/core/dev.c)
input_queue
[cpu]
net_rx_action()
(net/core/dev.c)
decision based on
skb->protocol field
Layer 1/2
Physical/Link
Network Drivers
(drivers/net/*)


Application
userspace
kernelspace
Socket Layer
(net/core/sock.c)
tcp_v4_do_rcv()
(net/ipv4/tcp_ipv4.c)
check for socket state
__tcp_v4_lookup()
(net/ipv4/tcp_ipv4.c)
check for socket in
LISTEN, with dst_port
generate ICMP
error
tcp_v4_rcv()
(net/ipv4/tcp_ipv4.c)
check for tcp headers
udp_rcv()
(net/ipv4/udp.c)
check for udp headers
<...>
Layer 4
Transport
ip_local_deliver_finish()
(net/ipv4/ip_input.c)
find protocol handler or
send icmp_dst_unreach
NF_IP_FORWARD
<continue>
Layer 3
Network
ip_forward()
(net/ipv4/ip_forward.c)
handle route alert;
send redirect if
necessary;
decrease TTL;
verify if frag is
possible (mtu)
ip_local_deliver()
(net/ipv4/ip_input.c)
defrag fragmented
packets
NF_IP_LOCAL_IN
NF_IP_PRE_ROUTING
ip_rcv_finish()
(net/ipv4/ip_input.c)
find route and handle
IP options
ip_error()
(net/ipv4/route.c)
routing error, send
icmp pkt

Methods to capture packets

- protocol handler
  - register a function to handler packets with `dev_add_pack()`
- netfilter hooks
- userspace tools;
  - socket `AF_PACKET`, libpcap, ...

```
struct packet_type my_proto;

int my_packet_rcv(struct sk_buff *skb, struct net_device *dev,
                  struct packet_type *pt, struct net_device *orig_dev) {

    printk(KERN_ERR "+ 1!\n");

    kfree_skb(skb);
    return 0;
}

static int hello_init(void) {
        printk("<1> Hello world!\n");

        my_proto.type = htons(ETH_P_ALL);
        my_proto.dev  = NULL;
        my_proto.func = my_packet_rcv;

        dev_add_pack(&my_proto);
        return 0;
}

static void hello_exit(void) {
        dev_remove_pack(&my_proto);
        printk("<1> Bye, cruel world\n");
}
module_init(hello_init);
module_exit(hello_exit);
```

```c
int my_packet_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type *pt, struct net_device
        *orig_dev)
{
   switch (skb->pkt_type) {
          case PACKET_HOST:
               printk("PACKET_HOST - ");
               break;
          case PACKET_BROADCAST:
               printk("PACKET_BROADCAST - ");
               break;
          case PACKET_MULTICAST:
               printk("PACKET_MULTICAST - ");
               break;
          case PACKET_OTHERHOST:
               printk("PACKET_OTHERHOST - ");
               break;
          case PACKET_OUTGOING:
               printk("PACKET_OUTGOING - ");
               break;
          case PACKET_LOOPBACK:
               printk("PACKET_LOOPBACK - ");
               break;
          case PACKET_FASTROUTE:
               printk("PACKET_FASTROUTE - ");
               break;
   }
   printk("%s 0x%.4X 0x%.4X \n", skb->dev->name, ntohs(skb->protocol), ip_hdr(skb)->protocol)

   kfree_skb(skb);
   return 0;
}
```

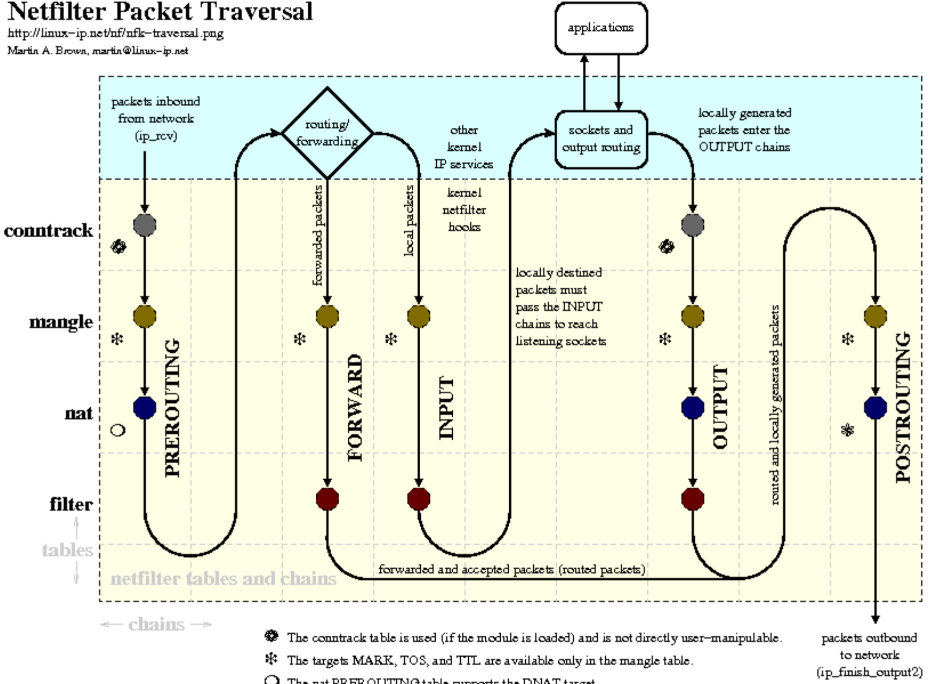
Netfilter hooks

- iptables = userspace;
- netfilter = kernelspace;
- Netfilter is merely a series of hooks in various points in a protocol stack;
- packet filtering, network address [and port] translation (NA[P]T) and other packet mangling;
- `www.netfilter.org`

Netfilter Packet Traversal

http://linux-ip.net/nf/nfk-traversal.png

Martin A. Brown, martin@linux-ip.net



❂ The conntrack table is used (if the module is loaded) and is not directly user-manipulable.

✻ The targets MARK, TOS, and TTL are available only in the mangle table.

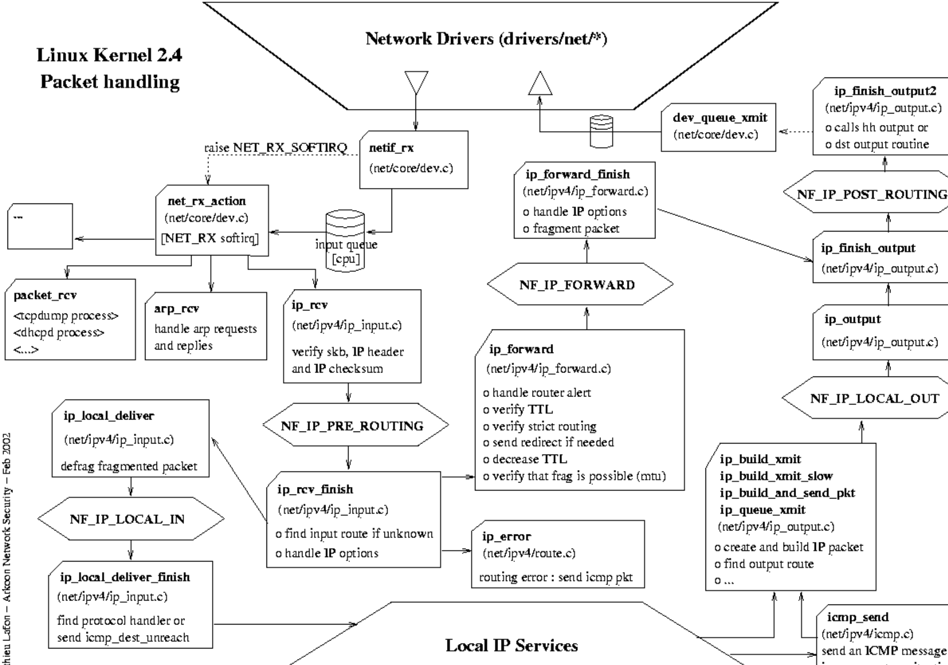
○ The nat PREROUTING table supports the DNAT target.

❃ The nat POSTROUTING table supports SNAT and MASQUERADE targets.

cf. http://www.docum.org/qos/kptd/

cf. http://open-source.arkoon.net/kernel/kernel_net.png

Linux Kernel 2.4
Packet handling



thieu Lafon – Arkoon Network Security – Feb 2002

References

- br.kernelnewbies.org/node/150 has many links

Thankyou! Question?

Network packet capture in Linux kernelspace

An overview of the network stack in the Linux kernel

Beraldo Leal

beraldo@ime.usp.br
http://www.ime.usp.br/~beraldo/

Institute of Mathematics and Statistics - IME
University of Sao Paulo - USP

25th October 2011