

Hack3rcon^3: A PCAP Workshop

Who am I?

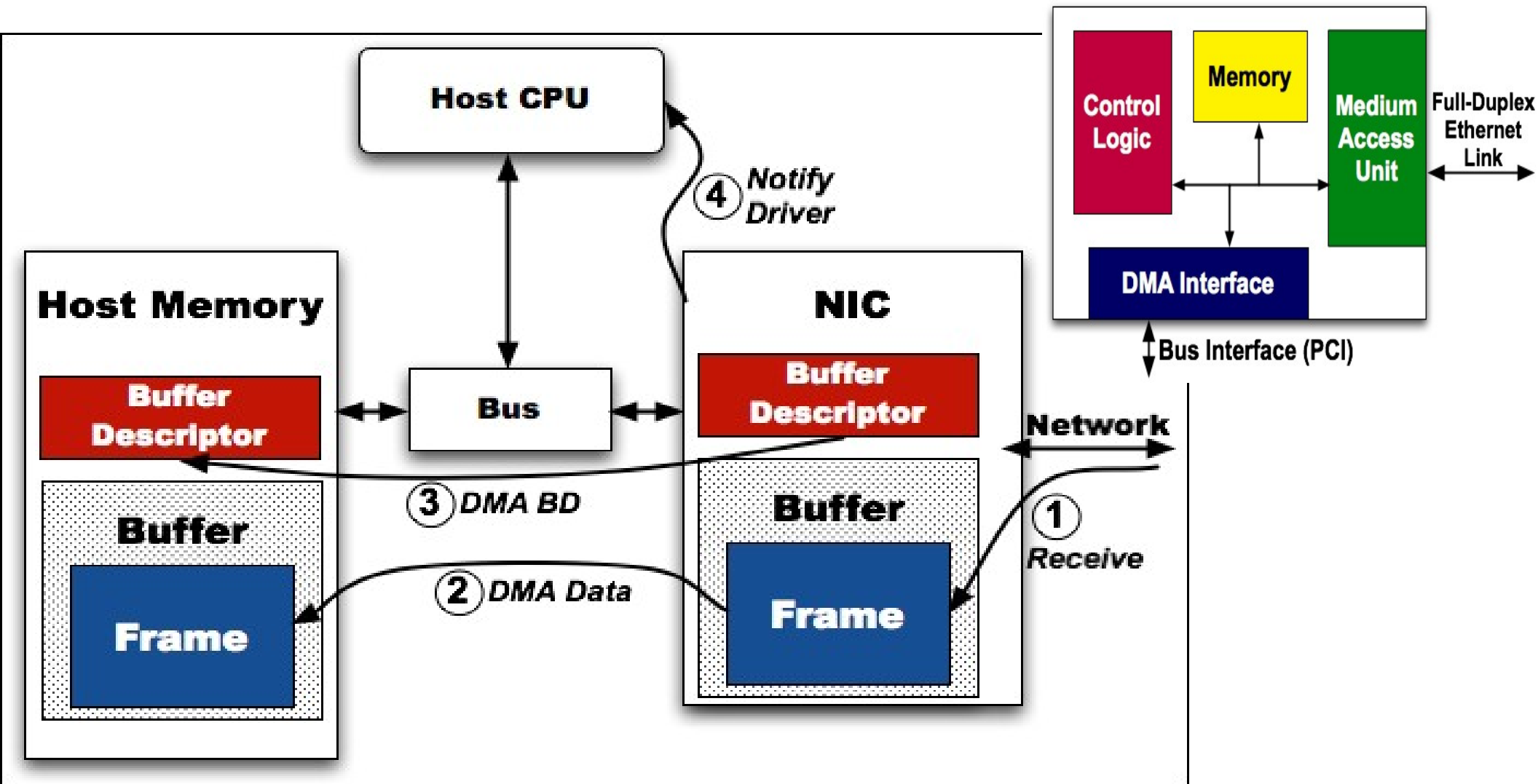
- ✧ Jon Schipp
- ✧ Unix Admin
- ✧ Linux & Unix User Group
- ✧ Southern Indiana Computer Klub



sickbits.net

30+ articles on packet capture and other security related things

Frame Processing



[http://www.ece.rice.edu/~willmann/teng_nics_hownicswork.html]

e1000 driver

- *InterruptThrottleRate*: the maximum number of interrupts per second. It is implemented by limiting the minimum time interval between consecutive interrupts.
- *(Rx/Tx)AbsIntDelay*: the delay between the arrival of the first packet after the last interrupt and the generation of a new interrupt. It controls the maximum queuing delay of a packet at the NIC buffers.
- *(Rx/Tx)IntDelay*: the delay between the last arrival of a packet and the generation of a new interrupt. It controls the minimum queuing delay of a packet at the NIC buffers.

Note that the previous parameters can be combined to meet constraints on the maximum interrupt rate and on the maximum/minimum IC-induced queuing delays. In case of conflict, *InterruptThrottleRate* has a higher precedence than *(Rx,Tx)AbsIntDelay* and *(Rx,Tx)IntDelay* parameters.

InterruptThrottleRate

limits the number of interrupts per second generated by the card. Values ≥ 100 are interpreted as the maximum number of interrupts per second. The default value used to be 8'000 up to and including kernel release 2.6.19. A value of zero (0) disabled interrupt moderation completely. Above 2.6.19, some values between 1 and 99 can be used to select *adaptive* interrupt rate control. The first adaptive modes are "dynamic conservative" (1) and dynamic with reduced latency (3). In conservative mode (1), the rate changes between 4'000 interrupts per second when only bulk traffic ("normal-size packets") is seen, and 20'000 when small packets are present that might benefit from lower latency. The more aggressive mode (3), "low-latency" traffic may drive the interrupt rate up to 70'000 per second. This mode is supposed to be useful for cluster communication in grid applications.

RxIntDelay

specifies, in multiples of 1'024 microseconds, the time after reception of a frame to wait for another frame to arrive before sending an interrupt.

RxAbsIntDelay

bounds the delay between reception of a frame and generation of an interrupt. It is specified in units of 1'024 microseconds. Note that *InterruptThrottleRate* overrides *RxAbsIntDelay*, so even when a very short *RxAbsIntDelay* is specified, the interrupt rate should never exceed the rate specified (either directly or by the dynamic algorithm) by *InterruptThrottleRate*

RxDescriptors

specifies the number of descriptors to store incoming frames on the adapter. The default value is 256, which is also the maximum for some types of E1000-based adapters. Others can allocate up to 4'096 of these descriptors. The size of the receive buffer associated with each descriptor varies with the [MTU](#) configured on the adapter. It is always a power-of-two number of bytes. The number of descriptors available will also depend on the per-buffer size. When all buffers have been filled by incoming frames, an interrupt will have to be signaled in any case.

Receive Path

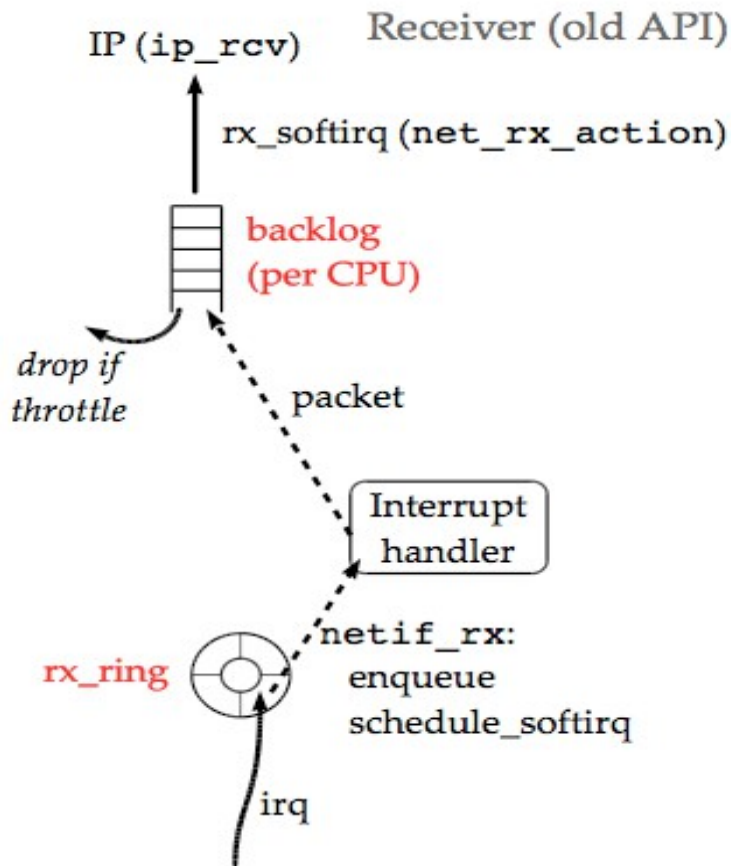


Figure 2 - Packet Reception with the old API

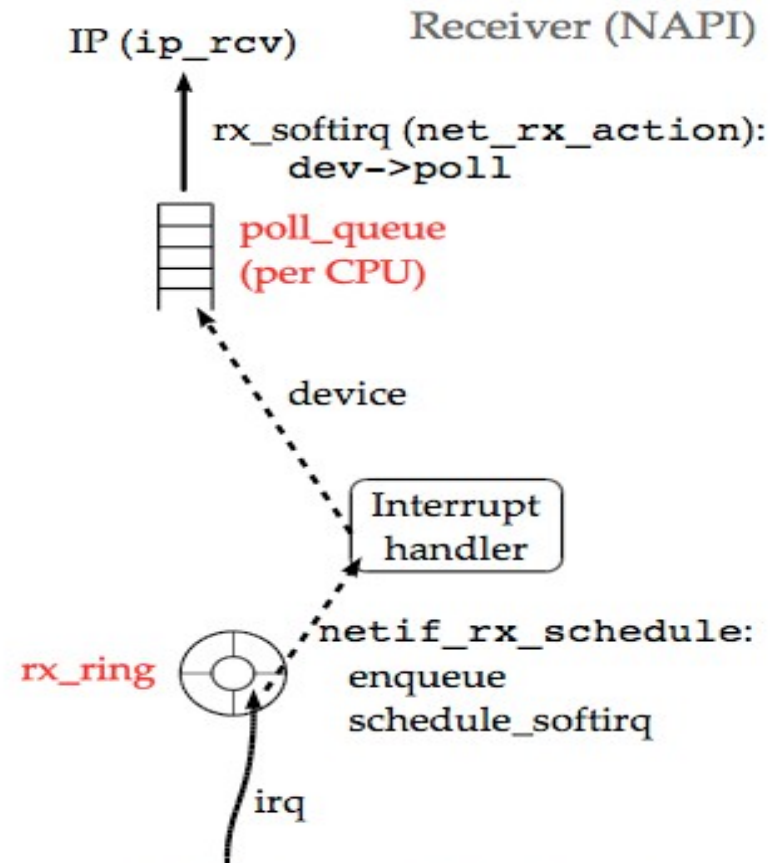
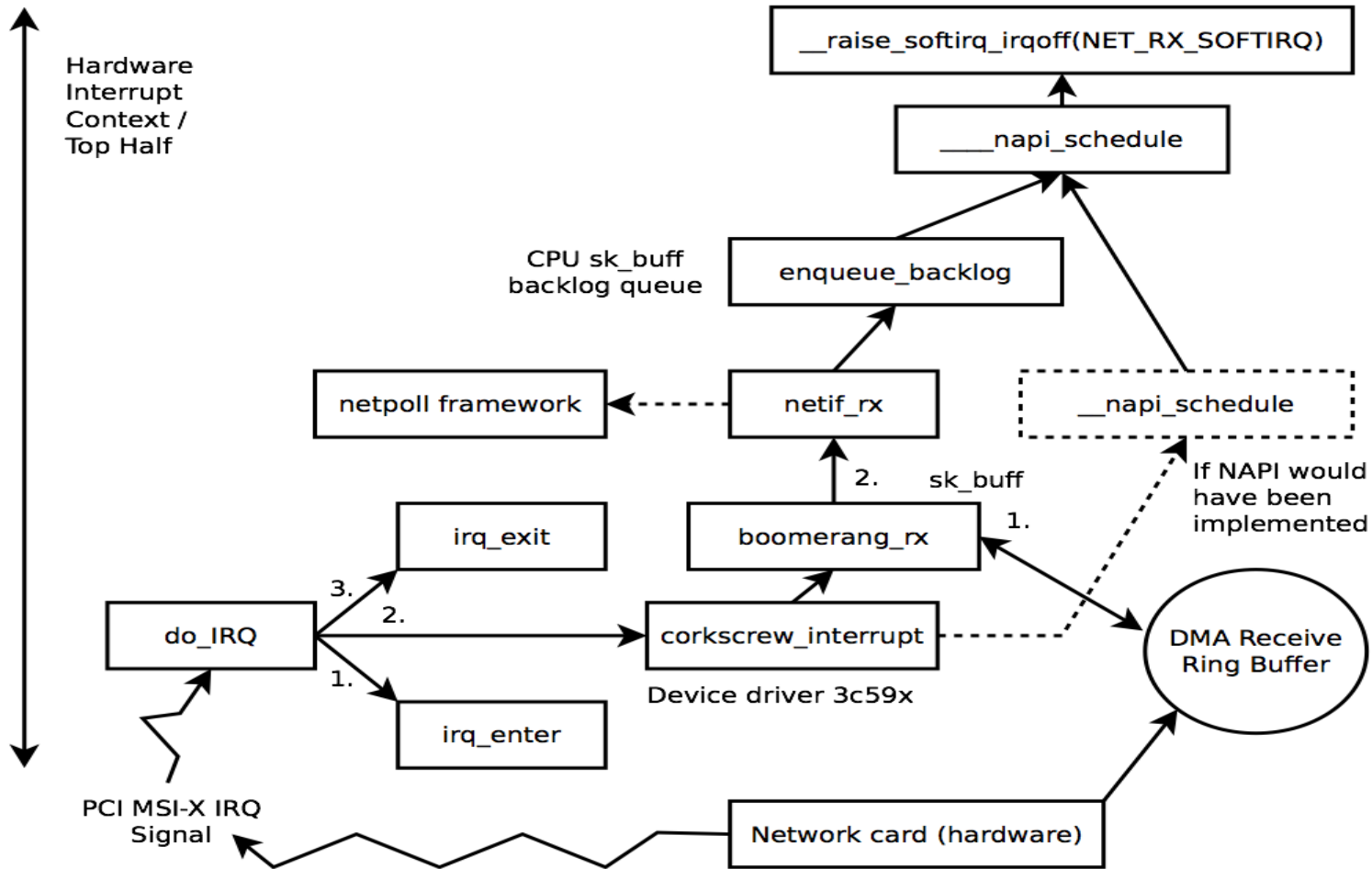


Figure 3 - Packet reception with the new API

Packet Path Ingress Direction



Linux, packet drops

ifconfig

Drops reported by kernel (out of space):

```
root@nms:~# awk '{ print $1,$5 }' /proc/net/dev | grep eth
eth0: 4555795
eth1: 0
eth2: 0
eth3: 0
eth4: 0
eth5: 0
```

← Drop Column, the 5th

```
root@nms:~# ifconfig -a | grep -E '(^eth|RX.*dropped)'
eth0      Link encap:Ethernet  HWaddr aa:00:04:00:0a:04
          RX packets:403386126 errors:0 dropped:4555795 overruns:0 frame:0
eth1      Link encap:Ethernet  HWaddr aa:00:04:00:0a:04
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
eth2      Link encap:Ethernet  HWaddr aa:00:04:00:0a:04
          RX packets:10853853 errors:0 dropped:0 overruns:0 frame:0
eth3      Link encap:Ethernet  HWaddr aa:00:04:00:0a:04
          RX packets:136168419 errors:15677 dropped:0 overruns:0 frame:15677
eth4      Link encap:Ethernet  HWaddr aa:00:04:00:0a:04
          RX packets:10949878 errors:0 dropped:0 overruns:0 frame:0
eth5      Link encap:Ethernet  HWaddr aa:00:04:00:0a:04
          RX packets:3184783 errors:0 dropped:0 overruns:0 frame:0
```

[me]

Drops reported by NIC, NIC dependent:
\$ ethtool -S eth0

```
static int get_dev_fields(char *bp, struct interface *ife)
{
    switch (procnetdev_vsn) {
    case 3:
        sscanf(bp,
            "%llu %llu %lu %lu %lu %lu %lu",
            &ife->stats.rx_bytes,
            &ife->stats.rx_packets,
            &ife->stats.rx_errors,
            &ife->stats.rx_dropped,
            ...
        );
    }
}
```


snort



[www.snort.org/]

```
# snort -r 05-11-2012_12\30_eth3.pcap -c /etc/snort/snort.read.conf -l .
```

```
# snort -r 05-11-2012_12\30_eth3.pcap -c /etc/snort/snort.read.conf -l .
```

```
root@nms:/home/jon/mypcaps# head -21 alert
[**] [1:2002087:10] ET POLICY Inbound Frequent Emails - Possible Spambot Inbound [**]
[Classification: Misc activity] [Priority: 3]
05/11-12:40:27.856276 213.5.178.106:25059 -> 208.110.191.226:25
TCP TTL:109 TOS:0x0 ID:20451 IpLen:20 DgmLen:101 DF
***AP*** Seq: 0x6507C2BE Ack: 0xB122E3E9 Win: 0xFC TcpLen: 20
[Xref => http://doc.emergingthreats.net/2002087]

[**] [1:2002087:10] ET POLICY Inbound Frequent Emails - Possible Spambot Inbound [**]
[Classification: Misc activity] [Priority: 3]
05/11-12:46:54.421213 198.87.2.43:37396 -> 208.110.191.226:25
TCP TTL:112 TOS:0x0 ID:1896 IpLen:20 DgmLen:101 DF
***AP*** Seq: 0x27B738ED Ack: 0x1CE8E9E5 Win: 0xFC TcpLen: 20
[Xref => http://doc.emergingthreats.net/2002087]

[**] [1:2406106:283] ET RBN Known Russian Business Network IP TCP (54) [**]
[Classification: Misc Attack] [Priority: 2]
05/11-13:13:02.428316 178.18.245.71:38076 -> 208.110.191.226:25
TCP TTL:49 TOS:0x0 ID:9765 IpLen:20 DgmLen:44 DF
*****S* Seq: 0xE64D315B Ack: 0x0 Win: 0x16D0 TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://doc.emergingthreats.net/bin/view/Main/RussianBusinessNetwork]
```

[http://sickbits.networklabs.org/snort-offline-analysis/]

Read file (**-r**),
use configuration file (**-c**),
write alerts to the cwd (**-l**),

Files:

```
root@nms:/home/jon/mypcaps# ls
05-11-2012_12:30_eth3.pcap alert flows tcpdump.log.1337615272
```

Ethtool, lost packets

```
$ ethtool -S etho | egrep (rx_missed|no_buffer)'
```

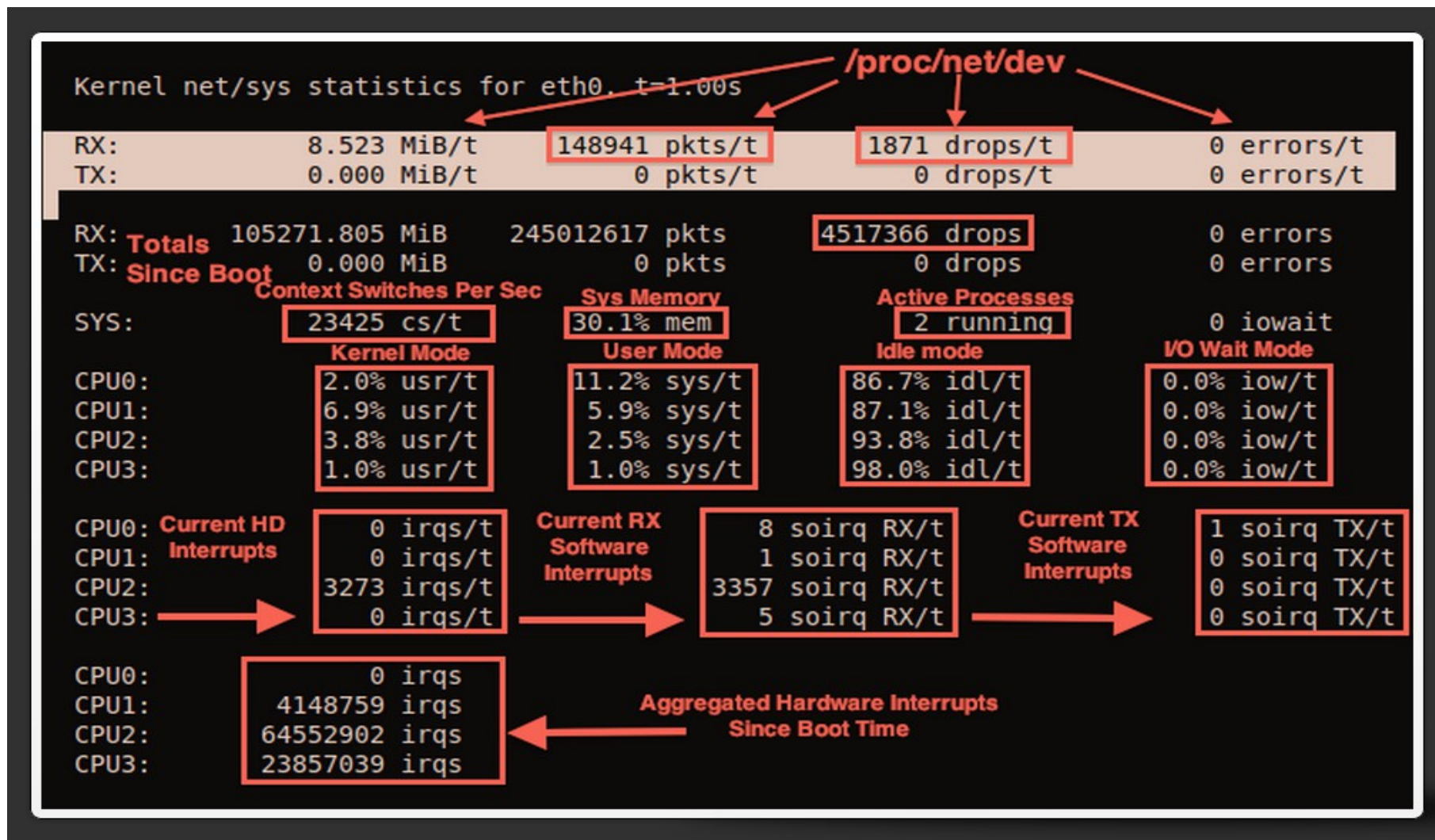
rx_missed_errors indicates frames that were dropped due to the e1000 adapter's fifo getting full and overflowing.

```
> rx_no_buffer_count: 310
```

```
> rx_missed_errors: 5865
```

rx_no_buffer_count indicates that the driver didn't return buffers to the hardware soon enough, but the hardware was able to store the packet (at the time of reception) in the fifo to try again.

ifpps - Statistics



[<http://sickbits.networklabs.org/ifpps-top-like-network-statistic-tool/>]

ifpps – Get RX Pkts

1.) Get values from RX Pkts/t field from the eth0 sample file:

```
awk -F \, '{ print "RX Packets: " $2 }' stats/ifpps_eth0.txt
```

```
root@nms:/pf_ring_apps# awk -F \, '{ print "RX Packets: " $2 }' stats/ifpps_eth0.txt | head
RX Packets: 12376
RX Packets: 2283
RX Packets: 2545
RX Packets: 16548
RX Packets: 2634
RX Packets: 4309
RX Packets: 2964
RX Packets: 2291
RX Packets: 2309
RX Packets: 2075
```

[<http://sickbits.networklabs.org/ifpps-top-like-network-statistic-tool/>]

ifpps – High Interrupts

Let's try to find high interrupt usage across CPU's:

```
gawk -F \, '{ if ( ( $11 > 10000 ) || ( $14 > 10000 ) || ( $17 > 10000 ) || \
( $20 > 10000 ) ) print strftime("%x-%X",$1) "\t|Stats:\tPPS: " $3, "\tRX \
Errors: " $5, "\tRX Drops: " $4, "\t|IRQs: \tCPU0 RX:" $11, "\tCPU1 RX: \
" $14, "\tCPU2 RX: " $17, "\tCPU3 RX: " $20 }' ifpps_eth0.txt
```

Let's do another example, this time, with a header describing the fields/columns:

```
gawk -F \, 'BEGIN { printf("%15s%20s%15s%15s%15s%15s%15s\n", "Time:", "PPS:", "RX Drops:", "CPU0 RX:", "CPU1 RX:", "CPU2 RX:", "CPU3 RX:"
```

Time:	PPS:	RX Drops:	CPU0 RX:	CPU1 RX:	CPU2 RX:	CPU3 RX:
07/16/2012-04:47:47 PM	401	0	16032	117	1640	318
07/16/2012-04:53:57 PM	3058	0	21819	53	544	1014
07/16/2012-05:00:07 PM	8185	0	21232	12	993	4482
07/16/2012-05:06:16 PM	4318	0	29689	33	4305	4164
07/16/2012-05:12:26 PM	229	0	20574	56	46	215
07/16/2012-05:18:36 PM	773	0	20112	41	3203	775
07/16/2012-05:24:46 PM	30908	0	20985	16	2465	30343
07/16/2012-05:30:56 PM	543	0	281	59	674	520
07/16/2012-05:37:06 PM	203	0	163	47	246	196

ifpps – Drops and Errors > 0

Our goal is to find the times of the day when the values of RX drops and RX errors are greater than 0.
Also, print some other information that may help us in determining excessive drops e.g. an *interrupt storm* (excessive interrupts)

```
$ gawk -F \, '{ if (( $3 > 0 ) && ( $4 > 0 )) print strftime("%x-%X",$1) \
"\t|Stats:\tPPS: " $3, "\tRX Errors: " $5, "\tRX Drops: " $4, \
"\t|IRQs: \tCPU0 RX:" $11, "\tCPU1 RX: " $14, "\tCPU2 RX: " $17, \
"\tCPU3 RX: " $20 }' ifpps_eth0.txt
```

```
root@nms:/pf_ring_apps/stats# gawk -F \, '{ if (( $3 > 0 ) && ( $4 > 0 )) print strftime("%x-%X",$1) "\t|Stats:\tPPS: " $3, "\tRX Errors: " $5, "\tRX Drops: " $4, \
"\t|IRQs: \tCPU0 RX:" $11, "\tCPU1 RX: " $14, "\tCPU2 RX: " $17, \
"\tCPU3 RX: " $20 }' ifpps_eth0.txt
```

10/02/1976-08:12:59 PM	Stats: PPS: 2165	RX Errors: 0	RX Drops: 75	IRQs: CPU0 RX:4
06/15/2012-04:08:53 PM	Stats: PPS: 63016	RX Errors: 8	RX Drops: 3290	IRQs: CPU0 RX:32634
06/18/2012-12:21:15 PM	Stats: PPS: 357177	RX Errors: 107	RX Drops: 2466	IRQs: CPU0 RX:1518
06/18/2012-02:30:26 PM	Stats: PPS: 26749	RX Errors: 8	RX Drops: 1351	IRQs: CPU0 RX:39967
06/18/2012-02:51:06 PM	Stats: PPS: 22799	RX Errors: 5	RX Drops: 3597	IRQs: CPU0 RX:18891
06/18/2012-04:24:07 PM	Stats: PPS: 26218	RX Errors: 1	RX Drops: 5406	IRQs: CPU0 RX:25909

Capture

Collecting the data and writing it to disk

Can we handle it all?

Tools:

bpf filters - a packet filtering language

netsniff-ng - a high-performance zero-copy capturing program

tcpdump – the *de facto* command-line packet capturing tool

BPF filters

Examples:

Basic Filters:

Hosts:

ether aa:bb:cc:dd:ee
ether src aa:bb:cc:dd:ee
ether dst aa:bb:cc:dd:ee
host 192.168.1.1
src host 192.168.1.1
dst host 192.168.1.1

Size:

less 64
greater 500

Ports:

port 80
src port 80
dst port 25
portrange 0-1023

Protocol:

arp
ip
ip6
tcp
udp
icmp

Network:

net 192.168.1.0/24
src net 192.168.1.0/24
dst net 192.168.1.0/24

Advanced Filters:

tcp[13] = 0x02
tcp[13] & 2 = 2
ip[12:4] = ip[16:4]
port 25 and tcp[20:4] = 0x4d41494c
port 80 and tcp[32:4] = 0x47455420

Combinations:

icmp and ether dst host 00:01:02:03:04:05
ip and tcp and port 80 and dst host (192.168.1.1 or 192.168.1.2)
udp port 53 and not src net (192.168.1.0/24 or 192.168.2.0/24)

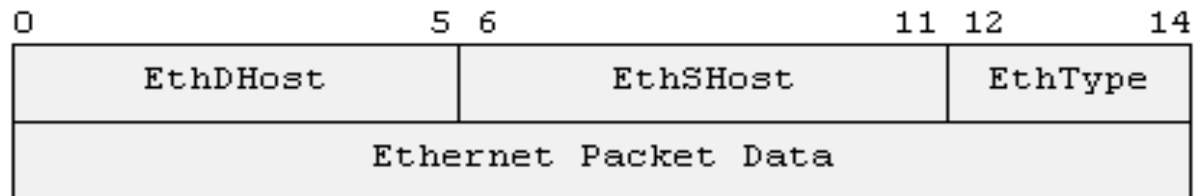
BPF Filters – 1

```
# arp example

# tcpdump -d arp
(000) ldh      [12]
(001) jeq      #0x806          jt 2    jf 3
(002) ret      #65535
(003) ret      #0

# tcpdump -dd arp
{ 0x28, 0, 0, 0x0000000c },
{ 0x15, 0, 1, 0x00000806 },
{ 0x6, 0, 0, 0x0000ffff },
{ 0x6, 0, 0, 0x00000000 },
```

[me]



[<http://www.security-freak.net/raw-sockets/EthernetHeaderStructure.png>]

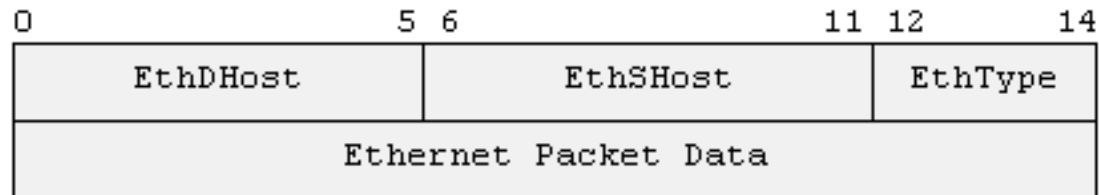
BPF Filters – 2

```
# ether src hw addr example

# tcpdump -s 1514 -d ether src aa:bb:cc:dd:ee:ff
(000) ld      [8]
(001) jeq     #0xccddeeff      jt 2      jf 5
(002) ldh     [6]
(003) jeq     #0xaabb          jt 4      jf 5
(004) ret     #1514
(005) ret     #0

# tcpdump -s 1514 -dd ether src aa:bb:cc:dd:ee:ff
{ 0x20, 0, 0, 0x00000008 },
{ 0x15, 0, 3, 0xccddeeff },
{ 0x28, 0, 0, 0x00000006 },
{ 0x15, 0, 1, 0x0000aabb },
{ 0x6, 0, 0, 0x000005ea },
{ 0x6, 0, 0, 0x00000000 },
```

[me]



[<http://www.security-freak.net/raw-sockets/EthernetHeaderStructure.png>]

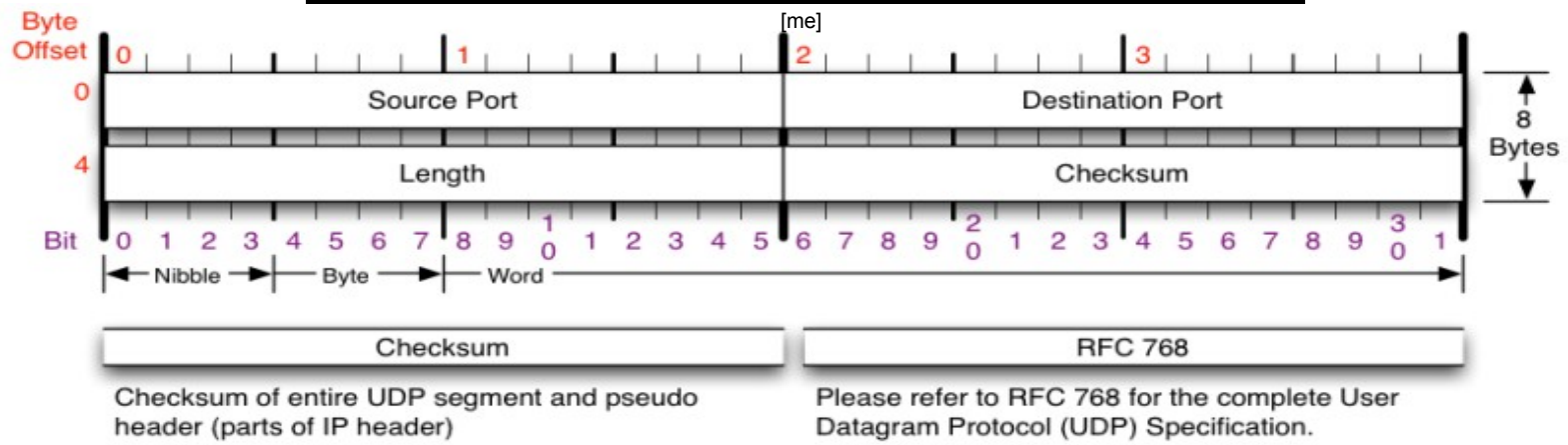
BPF Filters – 3

```
# udp dst port 53

# tcpdump -s 574 -d ip and udp dst port 53
(000) ldh      [12]
(001) jeq      #0x800          jt 2    jf 10
(002) ldb      [23]
(003) jeq      #0x11          jt 4    jf 10
(004) ldh      [20]
(005) jset     #0x1fff        jt 10   jf 6
(006) ldx      4*([14]&0xf)
(007) ldh      [x + 16]
(008) jeq      #0x35          jt 9    jf 10
(009) ret      #574
(010) ret      #0
```

Compare protocol field (udp = 0x11)
 Load Halfword from IP ID field (bitwise & to detect fragmentation)
 Load 1 byte from offset 14 (IHL)
 Calculate IP header length
 0101 AND
 1111 (0x0f)

 0101 (5 * 4 = 20 bytes) IP header size
 ^ value stored as x
 ld halfword, dst port = [x + 16] = [36] byte offset
 [36] = 0x35 = 53 decimal



[<http://www.visi.com/~mjb/Drawings>]

Capture SYN &

```
# tcpdump -nnr 05-11-2012_12\30_eth3.pcap -c 3 'tcp[13] & 2 = 2' | grep -E '(S|S\.)'
```

```
tcpdump -nnr 05-11-2012_12\30_eth3.pcap -c 3 'tcp[13] & 2 = 2' | grep -E '(S|S\.)'  
2_12:30_eth3.pcap, link-type EN10MB (Ethernet)  
191.250.65514 > 17.172.34.95.143: Flags [S], seq 1211190387, win 65535, options [mss 1  
ecr 0,sackOK,eol], length 0  
4.95.143 > 208.110.191.250.65514: Flags [S.], seq 1383665187, ack 1211190388, win 8190  
191.250.38654 > 208.111.168.7.80: Flags [S], seq 633041479, win 65535, options [mss 14  
t 0,sackOK,eol], length 0
```

```
# tcpdump -d 'tcp[13] & 2 = 2' >/dev/null | grep -B 1 -A 2 0x2
```

```
root@nms:~# tcpdump -d 'tcp[13] & 2 = 2' >/dev/null | grep -B 1 -A 2 0x2  
(007) ldb      [x + 27]  
(008) and      #0x2  
(009) jeq      #0x2          jt 10   jf 11  
(010) ret      #8192  
(011) ret      #0
```

[me]

IP Options: RR Example

```
# ping -R 192.168.1.1 -c 1
```

```
root@nms:/home/jon/Desktop# ping -R 192.168.1.1 -c 1
PING 192.168.1.1 (192.168.1.1) 56(124) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.334 ms
RR:    192.168.1.6
       192.168.1.1
       192.168.1.6
```

```
# tcpdump -Xvvnni eth5 'ip[0] & 0x0f > 5'
```

```
root@nms:~# tcpdump -Xvvnni eth5 'ip[0] & 0x0f > 5'
pfring-tcpdump: listening on eth5, link-type EN10MB (Ethernet), capture size 8192 bytes
15:49:09.316246 IP (tos 0x0, ttl 64, id 10265, offset 0, flags [DF], proto ICMP (1), length 124, options (RR 192.168.1.6, 192.168.1.1, 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0, EOL))
192.168.1.1 > 192.168.1.6: ICMP echo reply, id 4381, seq 1, length 64
 0x0000: 4f00 007c 2819 4000 4001 1966 c0a8 0101  0..|(.@.@..f....
 0x0010: c0a8 0106 0727 0cc0 a801 06c0 a801 0100  .....'.
 0x0020: 0000 0000 0000 0000 0000 0000 0000 0000  .....
 0x0030: 0000 0000 0000 0000 0000 0000 0000 0b51  .....Q
 0x0040: 111d 0001 359c ba4f 0000 0000 30d2 0400  ....5..0....0...
 0x0050: 0000 0000 1011 1213 1415 1617 1819 1a1b  .....
 0x0060: 1c1d 1e1f 2021 2223 2425 2627 2829 2a2b  .....! "$%&'()*+
 0x0070: 2c2d 2e2f 3031 3233 3435 3637  ,-. /01234567
```

[me]

Capture HTTP GET Method

```
# tcpdump -Xnr ~/mycaps/05-11-2012\12:30_eth3.pcap -c 3 -s 96 'port 80 and tcp[32:4] = 0x47455420'
```


```
root@nms:~# tcpdump -Xnr ~/mycaps/05-11-2012_12:30_eth3.pcap -c 3 -s 96 'port 80 and tcp[32:4] = 0x47455420' | grep GET
reading from file /home/jon/mycaps/05-11-2012_12:30_eth3.pcap, link-type EN10MB (Ethernet)
0x0030: 6936 996d 4745 5420 2f31 3030 3132 352f i6.mGET./100125/
0x0030: ca9a 2daf 4745 5420 2f69 6d61 6765 732f ..GET./images/
0x0030: 1be0 3284 4745 5420 2f69 6d61 6765 732f ..2.GET./images/
```

```
# printf '\x47\x45\x54\x20\n' | hexdump -c
```

```
root@nms:~# printf '\x47\x45\x54\x20\n' | hexdump -c
00000000  G  E  T      \n
00000005
```

[me]

netsniff-ng toolkit



- **netsniff-ng**, a high-performance zero-copy analyzer, pcap capturing and replaying tool
- **trafgen**, a high-performance zero-copy network traffic generator
- **mausezahn**, a packet generator and analyzer for HW/SW appliances with a Cisco-CLI
- **bpfc**, a Berkeley Packet Filter (BPF) compiler with Linux extensions
- **ifpps**, a top-like kernel networking and system statistics tool
- **flowtop**, a top-like netfilter connection tracking tool
- **curvetun**, a lightweight multiuser IP tunnel based on elliptic curve cryptography
- **astraceroute**, an autonomous system (AS) trace route utility

What's next in netsniff-ng?

- The usual: cleanups, extend documentation, man-pages
- `bpf-h1a`, high-level language for filtering
- DNS traceroute to detect malicious DNS injections on transit traffic
- Compressed on-the-fly bitmap indexing for large PCAP files
- New protocol dissectors/generators for netsniff-ng/mausezahn
- Further performance optimizations (OProfile is your friend)
- Hack `net/packet/af_packet.c` for a better performance

netsniff-ng: a quick look

```
$ netsniff-ng -dev -num 1 -ring-size 50MB -b 0 -H
```

```
root@nms:~# netsniff-ng --dev eth0 --num 1 --ring-size 50MB -b 0 -H
netsniff-ng 0.5.6
RX: 50.00 MB ← 25600 Frames each 2048 Byte allocated
IRQ: eth0:64 > CPU0
PROMISC
BPF:
(000) ret # -1
MD: RX

P 2 974 1337040582.065985
[ Eth MAC (c4:2c:03:10:07:4a => 00:1b:63:22:0e:88), Proto (0x86dd, IPv6) ]
[ Vendor (Unknown => Apple Computer Inc.) ]
[ IPv6 Addr (fe80::c62c:3ff:fe10:74a => fe80::21b:63ff:fe22:e88), Version (6), TrafficClass (128), FlowLabel (4018), Len (920), NextHdr (17), HopLimit (2) ]
[ UDP Port (52375 => 6000, X11), Len (920), CSum (0x814c) ]
[ Payload hex 80 60 20 d7 57 81 64 12 0f d9 49 73 e5 e1 e9 b5 8c 06 78 01 92 5d 68 fd 4b 52 6f 04 47 80 70 37 96 59 59 6c aa 0f 38 f5 0e 51 e1 bf 27 5f 5f e2 09 43 df 59 a9 2f e5 6c 16
ad 9f ea eb 0b 0c ed 0e 60 45 09 ab 2f c1 00 81 a5 35 ee 92 62 7a 1a 4b d6 ba 96 1e b5 c2 e5 37 09 ea 24 01 bd d1 cc 5e 72 16 ef 7a 0f 87 8c e0 9c 54 ce 33 f5 54 85 2d f9 61 05 6a f6
d2 3f b6 7e 94 fd b3 c2 48 ed af e9 ae ac dc 16 37 bd e4 b0 81 eb e7 18 96 56 3c 30 25 36 7e 12 66 c0 ee 13 36 04 59 e2 47 1f f3 7d c9 d2 81 e4 39 b4 a8 6c 73 89 2d 20 c0 ef 91 78 e7
d2 f6 91 5e b2 c6 4d 0b 4e e7 ca 11 e5 1e 1d 5e ea 9d f2 dc 05 a9 2c 1f 00 ea 29 11 09 3fa 1b c8 69 1f e4 59 92 41 57 f2 17 b5 09 06 b7 4b 3a 6e d4 5e 34 1c 78 f4 96 79 cf 15 5c ab 2
3 22 79 58 1f 7c c2 0c 15 c1 f2 9b e0 0d b3 a9 c3 49 b2 4e bb 37 3f 96 87 12 32 98 19 66 69 8a 34 7b e1 49 73 ed 7c fb a9 76 2d 6d 68 33 e0 51 a6 8b 38 41 6a d7 00 21 21 4d cc 8d af
9a 8c 6b 65 7f 93 66 52 11 8f 60 95 bf ea e4 55 e3 af c8 6f be 46 9b a8 c7 50 a8 7a 41 8e 52 1e 2e 6f 37 34 3b 2f dc b3 f4 a6 73 ac 59 97 3b 1c 9e c6 de 19 7b 3e 6f d8 10 b4 5e bd 85
7e 78 ac 94 aa f9 fd 4e b3 84 8b 01 e6 b8 7a 4b 75 a5 a5 c9 bd 32 e5 37 ef e2 f8 5a 74 90 c0 c3 95 e7 d0 99 f6 81 79 79 7c 29 bf c0 f3 c6 eb 66 6e 41 37 cd 8f da bb 1b c8 7a 7b b0 fe
4a 98 a8 f5 f4 5a 9b b0 2f 31 02 a4 27 bb 62 27 92 ca 2b dc fe 96 11 01 70 8b ea 64 57 36 7c 38 11 0c 1c 30 1c 11 fa 66 65 79 40 0c d9 17 a2 01 16 b6 a4 a9 6b 75 7a 50 3d 58 c3 a8 f8
c3 a5 77 10 90 58 16 61 86 1e b8 20 d1 18 9d f1 42 2c 36 fb 84 05 7e 7c 2f b7 08 2c 37 8d 22 bb e3 a5 39 5e 28 a2 21 5b cf 79 78 b7 12 58 7c a3 ed c7 38 c3 53 b3 1d 4c 2c 53 a0 2e 36
5e 65 f8 0a 3fa 2d 59 0d 18 47 83 e4 bd 74 97 e3 fb ba 02 64 c3 03 50 43 c7 d8 b0 5d 1d 1f 2b 9c a7 a9 cb 97 4f d9 51 d0 41 85 de c5 1c c9 db 50 06 2d fd 26 f9 18 3d 5a fb 4e 50 3c 6
3 91 04 61 1b af 47 4e e1 2e 43 92 88 47 4b ab ee c8 0a f5 65 76 c9 a2 85 90 ed aa 01 7d fe 76 cc 58 9d c9 5f 4e 60 19 3e 6e 6d 8e fd 7d d2 95 08 e3 32 10 be c9 4a de 4b ae 3a 1c 90
6d 01 57 50 83 06 aa 39 33 f3 ab bb 41 18 2f 43 2c 77 56 8a 18 3b 95 30 ae 76 28 df 8c 1c 2d eb 3d 59 64 9d e1 21 51 6a a0 b7 61 8c c5 3d 02 0c f7 7a db 84 74 e6 53 f5 bc 22 57 93 df
ad 31 5a cb b6 01 ab e2 98 7b 74 d9 f4 cd e0 9b bc 9d 3f 68 e9 f5 3c 46 a8 ec 60 ac 11 02 09 d4 91 01 66 09 ab ed 7d 7c e6 67 dd f5 5d 24 45 ec 3d b8 6e d9 ac 1f 94 3c 03 5d df ce fe
68 20 9a a1 3d 2a 54 31 02 e3 84 9d eb c9 b6 a5 23 06 2f 97 3c f6 91 e1 4c ee f0 19 24 e8 5a c6 30 c0 e2 4a 90 c6 76 bd f7 43 e3 09 df bf c4 5b 85 c4 e9 e8 ce 77 96 60 01 4c f2 1f 97
16 90 83 f1 d3 ec 15 39 5a 1b 14 af de de 6d eb 9b 12 fc e3 83 58 a6 52 0d 36 71 54 8e 49 8c b7 6b c0 0b 42 6a 8b 52 53 3fa 3b 69 c8 e0 ee c6 c0 a5 0c f0 61 ab 3f 0a 80 b8 e5 01 87 c
f b4 ac ae ]
[ Payload chr . . . W . d . . . I s . . . . . x . . . ] h . K R o . G . p 7 . Y Y l . . . 8 . . . Q . . . _ . . . C . Y . / . l . . . E . . . / . . . . 5 . . . b z . K . . . .
. . . 7 . . . $ . . . ^ r . . . z . . . T . 3 . T . . . . a . j . ? . ~ . . . H . . . . . V < 0 % 6 ~ . . . f . . . . 6 . . . Y . G . . . } . . . . 9 . . . l s . - . . .
. . . x . . . ^ . . . M . N . . . . . ^ . . . . . i . . . Y . A W . . . . . K : n . ^ 4 . x . . . y . < . \ . # " y X . | . . . . I . N . 7 ? . . . 2 . . .
. . . f i . 4 { . I s . | . . v - m h 3 . Q . . . 8 A j . ! ! M . . . . . k e . . . f R . . . . . U . . . o . F . . . P . z A . R . . . o 7 4 ; / . . . . s . Y . ; . . . . { > o . . .
. . . ^ . . . x . . . N . . . . . z K u . . . . . 2 . 7 . . . . Z t . . . . . y y | ) . . . . . f n A 7 . . . . . z { . . . J . . . . . Z . . . / 1 . . . ' . b ' . . . + . . . . p . . .
d W 6 | 8 . . . . . 0 . . . . . f e y @ . 0 . . . . . k u z P = X . . . . . w . X . a . . . . . B , 6 . . . . . ~ | / . . . . . 7 . . . . . 9 ^ ( . ! | . y x . . X | . . . . 8 . S . L ,
S . . . 6 ^ e . K - Y . G . . . . . t . . . . . d . P C . . . . . ] . . . . . + . . . . . 0 . Q . A . . . . . P . . . . . & . = Z . N P < c . . . . . a . G N . C . . . G K . . . . . e v . . .
. . . } . v . X . . . _ N . . . > n m . } . . . . . 2 . . . . . J . K . . . . . m . W P . . . . . 9 3 . . . . . A . / C . w V . . . . . 0 . v ( . . . . . = Y d . . . ! Q j . . . a . . . = . . . z . t . S . .
" W . . . 1 Z . . . [ t . . . . . ? h . . . < F . . . . . f . . . . . ] | . g . . . | $ E . = . n . . . < . . . ] . . . h . . . = * T 1 . . . . . # . / . < . . . L .
. . . $ . Z . 0 . . . J . . . v . . . C . . . . . [ . . . . . w . . . . . L . . . . . . . 9 Z . . . . . m . . . . . X . R . 6 q T . I . . . k . . . B j . R S . ; i . . . . . a . ? . .
. . . . . ]

39 frames incoming
39 frames passed filter
0 frames failed filter (out of space)
0.0000% frame droprate
```

[http://sickbits.networklabs.org/netsniff-ng-performant-packet-sniff/]

Netsniff-ng - Examples

- `netsniff-ng --in eth0 --out dump.pcap -s -b 0`
- `netsniff-ng --in wlan0 --rfraw --out dump.pcap -s -b 0`
- `netsniff-ng --in dump.pcap --mmap --out eth0 -s -b 0`
- `netsniff-ng --in eth1 --out /opt/probe/ -s -m -J
--interval 30 -b 0`
- `netsniff-ng --in any --filter ip4tcp.bpf --ascii`

netsniff-ng toolkit 

netsniff-ng: writing to disk

Write a new pcap to disk every 60 seconds:

```
root@nms:/home/jon# netsniff-ng --in eth0 --out mypcaps/ -F 60 -s -H -b 2,3
netsniff-ng 0.5.6
RX: 23.83 MB, 12200 Frames each 2048 Byte allocated
IRQ: eth0:64 > CPU2
PROMISC
BPF:
(000) ret      #-1
MD: RX SCATTER/GATHER

.(+51287/-0).(+33437/-0).(+25328/-0).(+15913/-0).(+173867/-0).(+25828/-0).(+46049/-0).(+60400/-0).
).(+42054/-0).(+32169/-0).(+25294/-0).(+100832/-0).(+32623/-0).(+16836/-0).(+23035/-0).(+16333/-0).
0).(+35279/-0).(+89906/-0).(+54044/-0).(+6761/-0).(+10907/-0).(+14638/-0).(+22135/-0).(+23908/-0).
).(+27414/-0).(+292356/-0).(+61768/-0).(+94747/-0).(+88450/-0)^C
```

Output directory (-out) Interval in Seconds (-F) Silent Mode (-s) High Priority Mode (-H) Bind to CPU 2 & 3 (-b)

Pcaps are written to disk in unix epoch time:

```
root@nms:/home/jon/mypcaps# ls
1336763069.pcap 1336763249.pcap 1336763429.pcap 1336763609.pcap
1336763129.pcap 1336763309.pcap 1336763489.pcap 1336763669.pcap
1336763189.pcap 1336763369.pcap 1336763549.pcap 1336763729.pcap
root@nms:/home/jon/mypcaps# date -d @1336763069 +"%X-%X"
05/11/2012-03:04:29 PM
```

[<http://sickbits.networklabs.org/netsniff-ng-performant-packet-sniff/>]

bpfc

NETSNIFF-NG

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

netsniff-ng, Filtering

```
ldh [12] ; Load Ethernet type field
jeq #0x800, Cont, Drop ; Check value against 0x800
Cont: ldb [23] ; Load IPv4 proto
jeq #0x6, Keep, Drop ; Check against 0x6 (TCP)
Keep: ret #0xffffffff ; Return packet
Drop: ret #0 ; Discard packet
```

- `bpfc ip4tcp.bpfa > ip4tcp.bpf`, then pass it to `--filter`
- Or abuse `tcpdump`: `tcpdump -dd my-filter`
- Filtering done in the Linux kernel (BPF virtual machine)
- Newer kernels: BPF JIT for x86/x86_64, powerpc, sparc

netsniff-ng: creating filters

1.)

```
root@nms:~# tcpdump -dd arp 2>/dev/null 1>arp.bpf
root@nms:~# tcpdump -dd arp 2>/dev/null
{ 0x28, 0, 0, 0x0000000c },
{ 0x15, 0, 1, 0x00000806 },
{ 0x6, 0, 0, 0x00002000 },
{ 0x6, 0, 0, 0x00000000 },
```

↑
Write stdout to file

2.)

```
root@nms:~# netsniff-ng --in eth0 --out test.pcap -f arp.bpf --num 1
netsniff-ng 0.5.6
RX: 23.83 MB, 12200 Frames each 2048 Byte allocated
PROMISC
BPF:
(000) ldh      [12]
(001) jeq      #0x806      jt 2      jf 3
(002) ret      #8192
(003) ret      #0
MD: RX SCATTER/GATHER

B 2 60 1337045245.869889
[ Eth MAC (40:6c:8f:04:11:35 => ff:ff:ff:ff:ff:ff), Proto (0x0806, ARP) ]
[ Vendor (Unknown => Unknown) ]
[ ARP Format HA (1), Format Proto (2048), HA Len (1536), Proto Len (1024), Opcode (1 => ARP request) ]
[ Payload hex 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ]
[ Payload chr . . . . . ]

      2 frames incoming
      2 frames passed filter
      0 frames failed filter (out of space)
0.0000% frame droprate
```

↑
Load BPF filter from file:
arp.bpf

Netsniff-ng - Statistics

```
568     ret = getsockopt(sock, SOL_PACKET, PACKET_STATISTICS, &kstats,&slen);
569     if (ret > -1) {
570         printf("\r%12ld frames incoming\n",
571             1UL * kstats.tp_packets);
572         printf("\r%12ld frames passed filter\n",
573             1UL * kstats.tp_packets - kstats.tp_drops - skipped);
574         printf("\r%12ld frames failed filter (out of space)\n",
575             1UL * kstats.tp_drops + skipped);
576         if (kstats.tp_packets > 0)
577             printf("\r%12.4f%% frame droprate\n", 1.f * kstats.tp_drops / kstats.tp_packets * 100.f)
```

`tp_drops` from `PF_PACKET`'s `PACKET_STATISTICS` holds the number of *packets* which were dropped due to a lack of buffer space in `PF_PACKET`. This value is used in calculating the drop percentage displayed in the above snippet at line 578. Drops reported by `tp_drops` are due to a high load. The *skipped* variable (line 575), shown in the "frames failed filter (out of space)" output, holds the number of packets received but were too large for the allocated ring buffer's frame size and thus not processed. By default the ring buffer is divided into 2048 byte slots, if you have a frame that exceeds this amount, say a Jumbo Frame of 9000 bytes, it will not fit into this slot and will be dropped. These drops are reported in the *skipped* variable. If you need jumbo support start `netsniff-ng` with the Jumbo option (`-J`).

netsniff-ng – Statistics cont.

Each time a new file is created (`-F / -interval`) stats for the previous file will be written to stdout.

```
root@nms:~# /usr/sbin/netsniff-ng --bind-cpu 3 --in eth2 --out /mnt/pcaps/eth2/ --ring-size 128MB -Q -s
netsniff-ng 0.5.8
RX: 128.00 MiB, 65536 Frames, each 2048 Byte allocated
PROMISC
BPF:
  L0: ret #0xffffffff
  MD: RX scatter-gather lf64 realtime: prio 4
.(+23775/-0).(+9656/-0).(+822/-0).(+933/-0)
```

Default interval of 60 seconds

netsniff-ng_dump_stats

The default value is 60 seconds. We can use an awk script to organize the data so it isn't so hard on the eyes.

```
awk 'BEGIN { RS="."; FS="/"; ORS="\n" } { print }' eth0.txt | tail
```

```
(+6986726/-0)
(+3995027/-0)
(+3580777/-0)
(+8869777/-0)
(+10382901/-0)
(+3081145/-0)
(+528693/-0)
(+458763/-0)
(+615587/-0)
(+127489/-0)
```

[<http://sickbits.networklabs.org/netsniff-ng-performant-packet-sniff/>]

To only print lines that have a loss count greater than 0, try this:

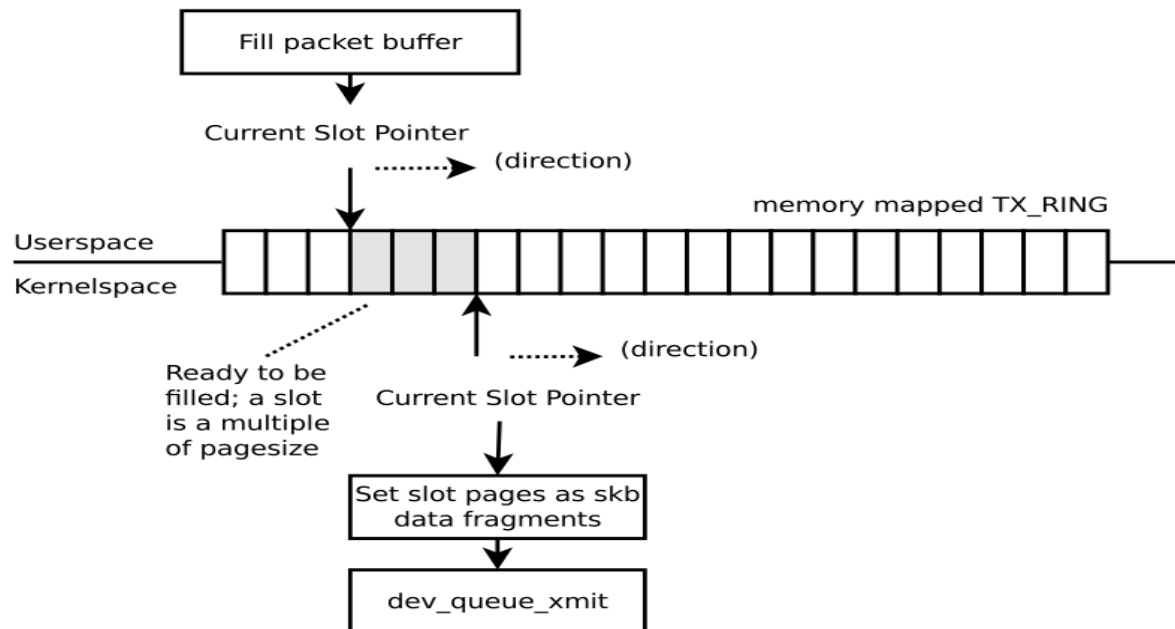
```
awk 'BEGIN { RS="."; FS="/"; ORS="\n" } { if( $0 !~ /netsniff/ && substr( $2,2,length($2)-2 ) > 0 ) print }' eth0.txt
```

```
(+4715313/-147096)
(+8343155/-9325)
```

[<http://sickbits.networklabs.org/netsniff-ng-performant-packet-sniff/>]

trafgen

- Uses PF_PACKET sockets with mmap(2)'ed TX_RING
- Users have reported wire-rate performance from user space
- Low-level packet configuration, more flexible than pktgen



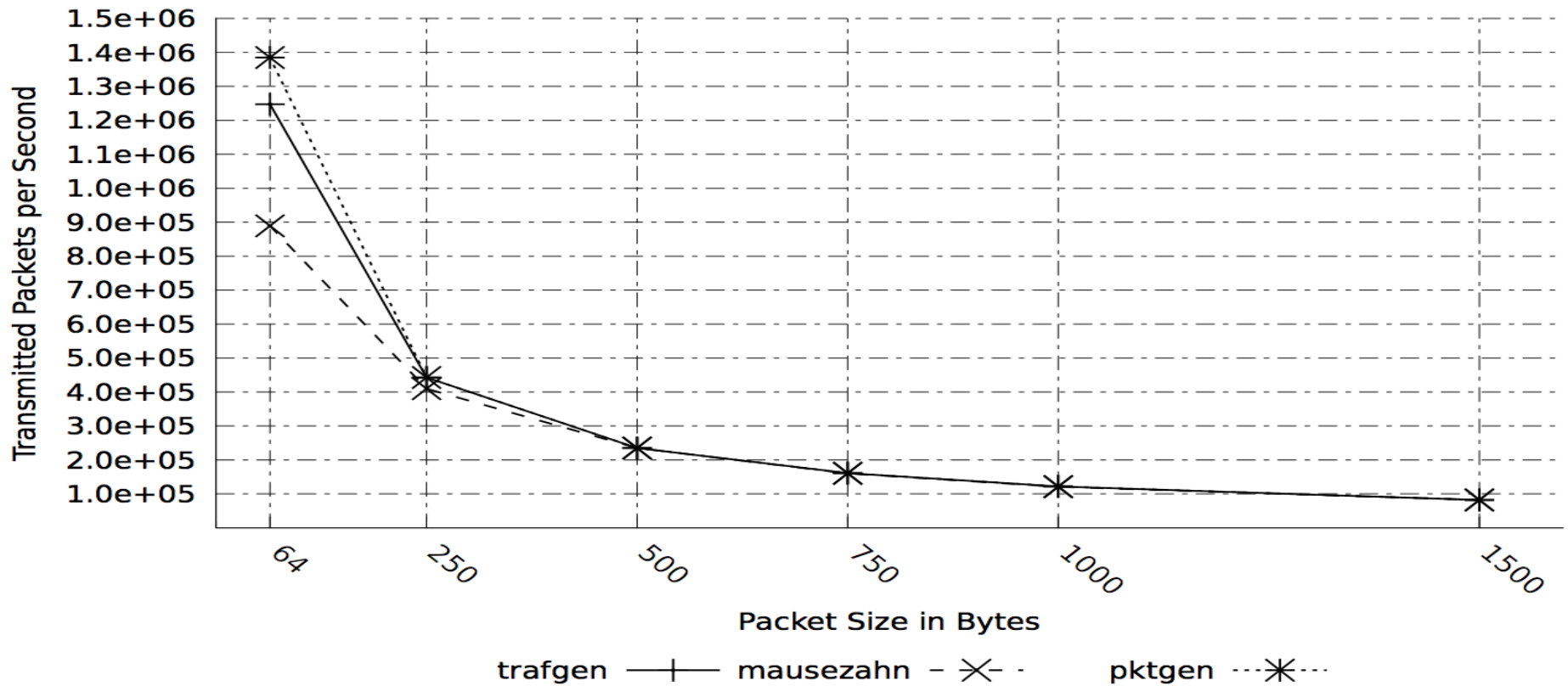
[https://github.com/gnumaniacs/netsniff-ng/blob/master/Documentation/Talks/gtalug_2012.pdf]

trafgen, mausezahn, pktgen



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Comparison of Traffic Generators



[https://github.com/gnumaniacs/netsniff-ng/blob/master/Documentation/Talks/gtalug_2012.pdf]

trafgen config files

```
$P1 {  
# Dst MAC -> work 192.168.1.3  
 0xc4,0x2c,0x03,0x0b,0x65,0x80  
# Src MAC  
 0x90,0xe6,0xba,0x70,0xbd,0x0a  
# Proto  
 0x08, 0x00,  
# Network Layer - IPv4  
# IP Version & IHL  
 0x45,  
# Type of Service  
 0x00,  
# Total Length  
 0x00,0x1c,  
# Identification  
 0x6a,0xae,  
# IP Flags (3 bits) & Fragment Offset  
 0x00,0x00,  
# TTL  
 0x40,  
# Protocol  
 0x11,  
# Header Checksum  
 0x8c,0xa6,  
# IP Source Address  
 0xc0,0xa8,0x01,0x29,  
# IP Destination Address  
 0xc0,0xa8,0x01,0x03,  
# UDP - Source Port  
 0x05,0x32,  
# UDP Destination Port  
 0x05,0x39,  
# Length  
 0x00,0x08,  
# UDP Checksum  
 0x71,0xf6,  
# Padding & Data  
 0x00,0x00,0x00,0x00,0x00,0x00,  
 0x00,0x00,0x00,0x00,0x00,0x00,  
}
```

```
root@bt:~# hping3 --udp -p 1337 -c 1 192.168.1.3  
HPING 192.168.1.3 (eth0 192.168.1.3): udp mode set, 28 headers + 0 data bytes  
ICMP Port Unreachable from ip=192.168.1.3  
  
--- 192.168.1.3 hping statistic ---  
1 packets tramitted, 1 packets received, 0% packet loss  
round-trip min/avg/max = 0.0/0.0/0.0 ms  
root@bt:~#
```

```
root@bt:~# tcpdump -exxnr udp.pcap  
reading from file udp.pcap, link-type EN10MB (Ethernet)  
21:07:19.617389 90:e6:ba:70:bd:0a > c4:2c:03:0b:65:80, ethertype IPv4 (0x0800), length 42:  
192.168.1.41.1790 > 192.168.1.3.1337: UDP, length 0  
      0x0000:  c42c 030b 6580 90e6 ba70 bd0a 0800 4500  
      0x0010:  001c 1ffe 0000 4011 d756 c0a8 0129 c0a8  
      0x0020:  0103 06fe 0539 0008 702a  
root@bt:~#
```

[me]

trafgen config files

```
$P1 {  
# Dst MAC -> work 192.168.1.3  
0xc4,0x2c,0x03,0x0b,0x65,0x80  
# Src MAC  
0x90,0xe6,0xba,0x70,0xbd,0x0a  
# Proto  
0x08, 0x00,  
# IP Version & IHL  
0x45,  
# Type of Service  
0x00,  
# Total Length  
0x00,0x1c,  
# Identification  
0x6a,0xae,  
# IP Flags (3 bits) & Fragment Offset  
0x00,0x00,  
# TTL  
0x40,  
# Protocol  
0x06,  
# Header Checksum  
0x40,0x46,  
# IP Source Address  
0xc0,0xa8,0x01,0x29,  
# IP Destination Address  
0xc0,0xa8,0x01,0x03,  
# TCP - Source Port  
0x08,0x7f,  
# TCP Destination Port  
0x00,0x50,  
# Sequence Number  
0x59,0x3d,0xa6,0xde,  
# Acknowledgement Number  
0x2e,0x5c,0x0d,0xae,  
# Offset & Reserved  
0x50,  
# TCP Flags  
0x02,  
# Window  
0x02,0x00,  
# Checksum  
0xe5,0x70,  
# Urgent Pointer  
0x00,0x00,  
# Padding & Data  
# 0x00,0x00,0x00,0x00,0x00,0x00  
}
```

1.)

```
root@bt:~# tcpdump -w tcpsyn80.pcap -nni eth0 ip dst 192.168.1.3 and dst port 80  
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes  
^C1 packet captured  
1 packet received by filter  
0 packets dropped by kernel  
root@bt:~#
```

2.)

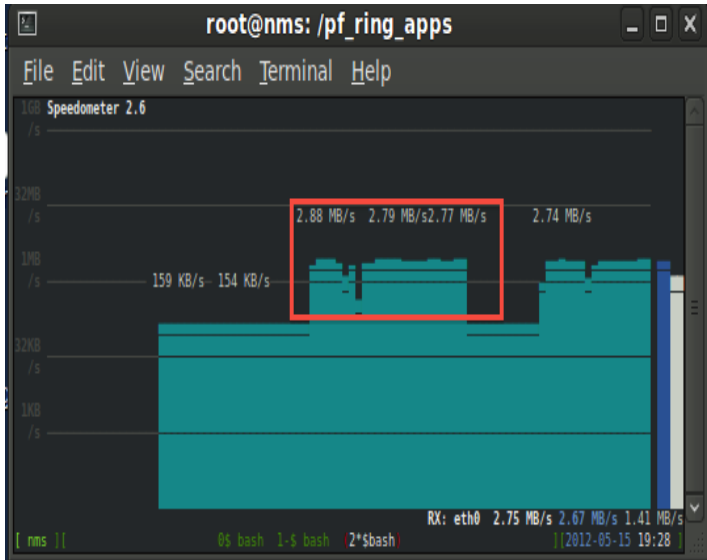
```
root@bt:~# hping3 -S -p 80 -c 1 192.168.1.3  
HPING 192.168.1.3 (eth0 192.168.1.3): S set, 40 headers + 0 data bytes  
len=46 ip=192.168.1.3 ttl=64 id=36357 sport=80 flags=RA seq=0 win=0 rtt=4.1 ms  
  
--- 192.168.1.3 hping statistic ---  
1 packets tramitted, 1 packets received, 0% packet loss  
round-trip min/avg/max = 4.1/4.1/4.1 ms
```

3.)

```
root@bt:~# netsniff-ng --in tcpsyn80.pcap --out tcpsyn80.txf  
netsniff-ng 0.5.7  
BPF:  
L0: ret #0xffffffff  
MD: RD scatter-gather realtime: prio 4  
  
54 1336613281.072639  
[ Eth MAC (90:e6:ba:70:bd:0a => c4:2c:03:0b:65:80), Proto (0x0800, IPv4) ]  
[ Vendor (ASUSTek COMPUTER INC. => Unknown) ]  
[ IPv4 Addr (192.168.1.41 => 192.168.1.3), Proto (6), TTL (64), TOS (0), Ver (4), IHL  
(5), Tlen (40), ID (58279), Res (0), NoFrag (0), MoreFrag (0), FragOff (0), CSum (0  
x13ac) is ok ]  
[ TCP Port (2487 => 80, http), SN (0x3205fd4d), AN (0x4ace708d), DataOff (5),  
Res (0), Flags (SYN ), Window (512), CSum (0x35b0), UrgPtr (0) ]  
  
1 frames outgoing  
0 frames truncated (larger than mtu)  
54 bytes outgoing
```

[me]

trafgen – packet generation



```
root@bt:~# trafgen --dev eth0 --conf udp.txf --bind-cpu 0 -S 100MB --num 1000000
trafgen 0.5.7
CFG:
n 1000000, gap 0 us, pkts 1
[0] pkt
len 58 cnts 0 rnds 0
payload c4 2c 03 0b 65 80 90 e6 ba 70 bd 0a 08 00 45 00 00 1c 6a ae 00 00 40 11 8c
a6 c0 a8 01 29 c0 a8 01 03 05 32 05 39 00 08 71 f6 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00
TX: 100.00 MiB, 51200 Frames, each 2048 Byte allocated
IRQ: eth0:45 > CPU0
MD: FIRE RR 10us

Running! Hang up with ^C!

1000000 frames outgoing
58000000 bytes outgoing
```

Annotations in the terminal output:

- Bind to CPU 0 (-bind-cpu)
- Set Size of TX_RING (-S)
- Send million frames (-num)

File	Edit	View	Search	Terminal	Help
Time: 1337124508	PPS: 14963.60	Pkt Cnt: 74818	Net Load: 0.06	Bps: 829702.20	
Time: 1337124513	PPS: 43583.40	Pkt Cnt: 217917	Net Load: 0.07	Bps: 2145593.40	
Time: 1337124518	PPS: 35708.80	Pkt Cnt: 178544	Net Load: 0.07	Bps: 1785313.20	
Time: 1337124523	PPS: 44178.20	Pkt Cnt: 220891	Net Load: 0.08	Bps: 2174198.80	
Time: 1337124528	PPS: 46422.40	Pkt Cnt: 232112	Net Load: 0.09	Bps: 2278165.40	
Time: 1337124533	PPS: 15972.00	Pkt Cnt: 79860	Net Load: 0.09	Bps: 878501.60	
Time: 1337124538	PPS: 132.80	Pkt Cnt: 664	Net Load: 0.08	Bps: 150248.00	

[me]

tcpdump

TCPDUMP & LIBPCAP

Write 10 packet to disk (`-c`) and do not resolve port numbers and name (`-nn`), write to file test.pcap (`-w`):

```
root@nms:~# /usr/sbin/tcpdump -nni eth0 -s 1514 -c 10 -w test.pcap
tcpdump: WARNING: eth0: no IPv4 address assigned
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
10 packets captured
11 packets received by filter
0 packets dropped by kernel
```

↑ ↑
Write 10 packets to disk

Dump ethernet header (`-e`), everything in hex and ascii (`-XX`) and grab only the first 96 bytes of each by setting the snap length (`-s`):

```
root@nms:~# /usr/sbin/tcpdump -eXXvnni eth0 -s 96 -c 1 ip and udp port 53
tcpdump: WARNING: eth0: no IPv4 address assigned
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
20:35:22.136687 c4:2c:03:2f:0f:7d > c4:2c:03:0b:66:40, ethertype IPv4 (0x0800), length 100
: (tos 0x0, ttl 255, id 8867, offset 0, flags [none], proto UDP (17), length 86)
 192.168.1.195.63619 > 192.168.1.2.53: 63441+[|domain]
    0x0000:  c42c 030b 6640 c42c 032f 0f7d 0800 4500  ....f@.../..E.
    0x0010:  0056 22a3 0000 ff11 14de c0a8 01c3 c0a8  .V".....
    0x0020:  0102 f883 0035 0042 4dd4 f7d1 0100 0001  .....5.BM.....
    0x0030:  0000 0000 0000 026c 6207 5f64 6e73 2d73  .....lb_dns-s
    0x0040:  6404 5f75 6470 0130 0331 3239 0233 3702  d_udp.0.129.37.
    0x0050:  3130 0769 6e2d 6164 6472  ....in-addr
1 packets captured
655 packets received by filter
0 packets dropped by kernel
```

[<http://www.tcpdump.org/>]

Analysis

Analyzing the data that we have collected

Making sense of it

Tools:

ntop – a web-based traffic monitoring tool with many graphs

iftop – shows data rate and other metrics per connection

tcpflow – a tcp/ip session reassembler

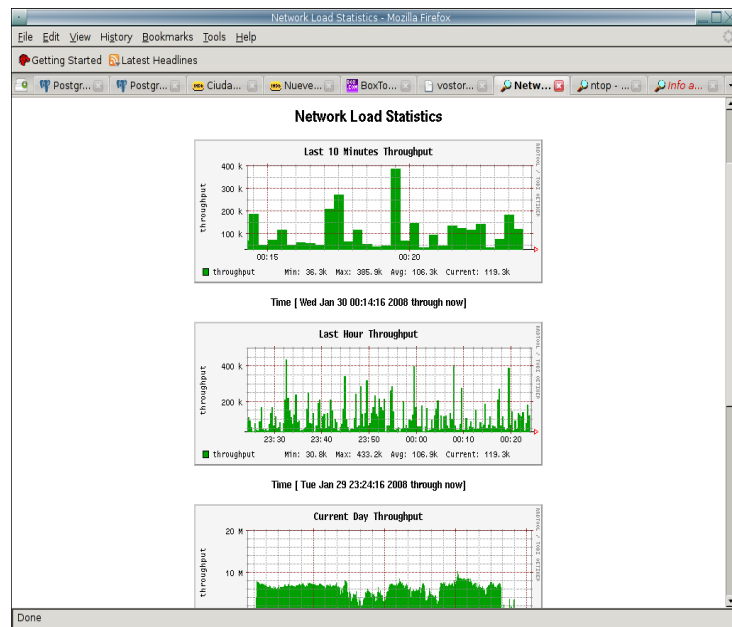
tcpick – a tcp stream sniffer and connection tracker

speedometer – measure and display rate of data across an interface

snort – A free lightweight network intrusion detection system

ntop

Global TCP/UDP Protocol Distribution



ntop -d -L -u ntop --access-log-file=/var/log/ntop/access.log -b -C --output-packet-path=/var/log/ntop-suspicious.log --local-subnets 192.168.1.0/24,192.168.2.0/24,192.168.3.0/24 -o -M -p /etc/ntop/protocol.list -i br0,eth0,eth1,eth2,eth3,eth4,eth5 -o /var/log/ntop

[<http://www.ntop.org/products/ntop/>]

iftop

iftop - display bandwidth usage on an interface by host

Find bandwidth hogs

Per connection bandwidth statistics

BPF filters via libpcap and an easy to use regex screen filter

Fault: inability to read pcaps

iftop

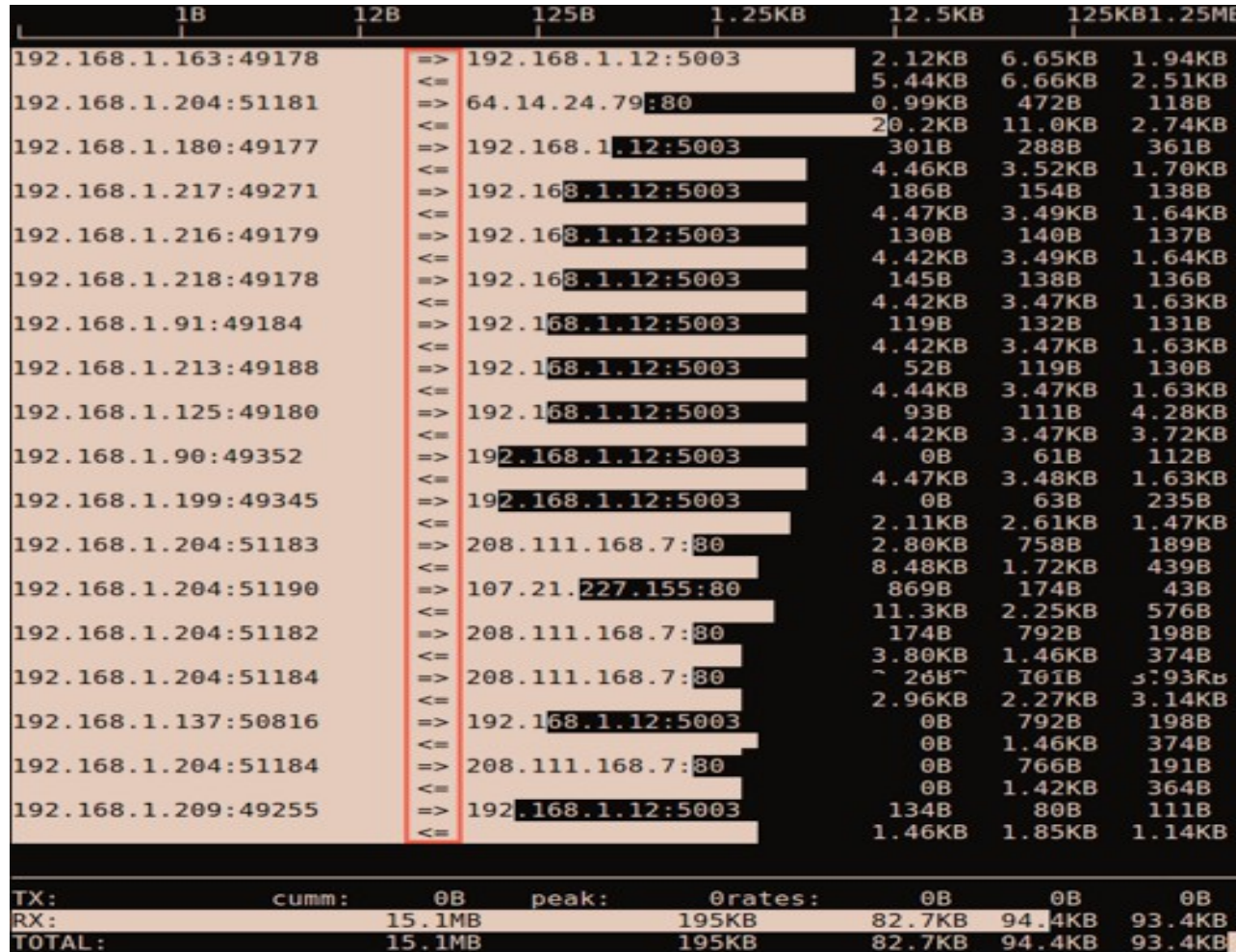
iftop - display bandwidth usage on an interface by host
One connection displayed per line

	1B	12B	125B		1.25KB			
192.168.1.216:49304	<=>	64.14.24.79:80	45.9KB	0B	4.59KB	1.53KB		
192.168.1.125:49260	<=>	74.125.225.68:80	17.9KB	0B	1.79KB	611B		
192.168.1.216:49307	<=>	208.111.168.7:80	5.36KB	0B	549B	183B		
192.168.1.216:49306	<=>	208.111.168.7:80	3.22KB	52B	330B	110B		
192.168.1.216:49308	<=>	208.111.168.7:80	3.12KB	0B	319B	106B		
192.168.1.60:5353	<=>	224.0.0.251:5353	2.84KB	0B	291B	97B		
192.168.1.125:49259	<=>	74.125.225.104:80	2.40KB	0B	246B	82B		
192.168.1.216:49305	<=>	208.111.168.7:80	1.77KB	52B	181B	60B		
192.168.1.216:49310	<=>	74.125.225.98:80	1.62KB	0B	166B	55B		
192.168.1.216:49309	<=>	23.0.32.80:80	0.98KB	0B	101B	34B		
192.168.1.5:5353	<=>	224.0.0.251:5353	449B	0B	32B	15B		
192.168.1.208:5353	<=>	224.0.0.251:5353	1.46KB	0B	30B	50B		
192.168.1.195:55453	<=>	209.85.145.141:80	156B	0B	16B	5B		
192.168.1.195:55457	<=>	209.85.145.141:80	156B	0B	16B	5B		
192.168.1.218:49555	<=>	63.84.95.65:80	156B	78B	16B	5B		
192.168.1.162:17500	<=>	255.255.255.255:17500	147B	74B	15B	5B		
192.168.1.5:626	<=>	224.0.0.1:626	140B	70B	14B	5B		
192.168.1.4:626	<=>	224.0.0.1:626	136B	68B	14B	5B		
192.168.1.10:626	<=>	224.0.0.1:626	134B	0B	13B	4B		
192.168.1.2:626	<=>	224.0.0.1:626	134B	0B	13B	4B		
192.168.1.226:49292	<=>	239.255.255.253:427	128B	0B	13B	4B		
192.168.1.226:49293	<=>	239.255.255.253:427	128B	0B	13B	4B		
192.168.1.226:49294	<=>	239.255.255.253:427	128B	0B	13B	4B		
192.168.1.226:49295	<=>	239.255.255.253:427	128B	0B	13B	4B		
192.168.1.189:49368	<=>	63.84.95.40:80	104B	0B	10B	3B		
192.168.1.181:53709	<=>	224.0.0.1:8612	88B	44B	9B	3B		
192.168.1.181:55502	<=>	224.0.0.1:8612	88B	0B	9B	3B		
			cumulative	2 sec	10 sec	40 sec		
TX:	cumm:	33.8KB	peak:	6.82KB	rates:	458B	2.36KB	1.13KB
RX:		66.1KB		23.2KB		78B	6.45KB	2.20KB
TOTAL:		99.9KB		30.0KB		536B	8.81KB	3.33KB

[<http://sickbits.networklabs.org/iftop-finding-traffic-hogs/>]

iftop

Interactive: press "h" to cycle through views, traffic show in both directions, per connection, one line each



[<http://sickbits.networklabs.org/iftop-finding-traffic-hogs/>]

iftop

Show traffic originating from network 192.168.1.0/24 to any *not* from 192.168.1.0/24

```
$ iftop -i eth0 -F 192.168.1.0/255.255.255.0
```

	1B	12B	125B	1.25KB	
192.168.1.216:49304	<=>	64.14.24.79:80	45.9KB	0B	4.59KB 1.53KB
192.168.1.125:49260	<=>	74.125.225.68:80	17.9KB	0B	1.79KB 611B
192.168.1.216:49307	<=>	208.111.168.7:80	5.36KB	0B	549B 183B
192.168.1.216:49306	<=>	208.111.168.7:80	3.22KB	52B	330B 110B
192.168.1.216:49308	<=>	208.111.168.7:80	3.12KB	0B	319B 106B
192.168.1.60:5353	<=>	224.0.0.251:5353	2.84KB	0B	291B 97B
192.168.1.125:49259	<=>	74.125.225.104:80	2.40KB	0B	246B 82B
192.168.1.216:49305	<=>	208.111.168.7:80	1.77KB	52B	181B 60B
192.168.1.216:49310	<=>	74.125.225.98:80	1.62KB	0B	166B 55B
192.168.1.216:49309	<=>	23.0.32.80:80	0.98KB	0B	101B 34B
192.168.1.5:5353	<=>	224.0.0.251:5353	449B	0B	32B 15B
192.168.1.208:5353	<=>	224.0.0.251:5353	1.46KB	0B	30B 50B
192.168.1.195:55453	<=>	209.85.145.141:80	156B	0B	16B 5B
192.168.1.195:55457	<=>	209.85.145.141:80	156B	0B	16B 5B
192.168.1.218:49555	<=>	63.84.95.65:80	156B	78B	16B 5B
192.168.1.162:17500	<=>	255.255.255.255:17500	147B	74B	15B 5B
192.168.1.5:626	<=>	224.0.0.1:626	140B	70B	14B 5B
192.168.1.4:626	<=>	224.0.0.1:626	136B	68B	14B 5B
192.168.1.10:626	<=>	224.0.0.1:626	134B	0B	13B 4B
192.168.1.2:626	<=>	224.0.0.1:626	134B	0B	13B 4B
192.168.1.226:49292	<=>	239.255.255.253:427	128B	0B	13B 4B
192.168.1.226:49293	<=>	239.255.255.253:427	128B	0B	13B 4B
192.168.1.226:49294	<=>	239.255.255.253:427	128B	0B	13B 4B
192.168.1.226:49295	<=>	239.255.255.253:427	128B	0B	13B 4B
192.168.1.189:49368	<=>	63.84.95.40:80	104B	0B	10B 3B
192.168.1.181:53709	<=>	224.0.0.1:8612	88B	44B	9B 3B
192.168.1.181:55502	<=>	224.0.0.1:8612	88B	0B	9B 3B
<hr/>					
TX:	cumm:	33.8KB	peak:	6.82KB	rates: 458B 2.36KB 1.13KB
RX:		66.1KB		23.2KB	78B 6.45KB 2.20KB
TOTAL:		99.9KB		30.0KB	536B 8.81KB 3.33KB

[<http://sickbits.networklabs.org/iftop-finding-traffic-hogs/>]

iftop

Example BPF filters

`$ iftop -i eth0 -f 'port (80 or 443)'`

`$ iftop -i eth0 -f 'ip dst 192.168.1.5'`

	1B	12B	125B	1.25KB	12.5KB	125KB	1.25MB
192.168.1.2:51521<=> 208.111.168.6:80				425KB	0B	36.9KB	10.6KB
192.168.1.2:51531<=> 208.111.168.6:80				248KB	124KB	24.8KB	6.21KB
192.168.1.2:51520<=> 208.111.168.6:80				121KB	770B	8.99KB	3.02KB
192.168.1.2:51525<=> 208.111.168.6:80				74.5KB	770B	6.02KB	1.86KB
192.168.1.2:51528<=> 208.111.168.6:80				61.5KB	770B	5.94KB	1.54KB
192.168.1.2:51524<=> 208.111.168.6:80				115KB	770B	5.20KB	2.87KB
192.168.1.2:51449<=> 64.14.24.79:80				335KB	4.21KB	4.96KB	7.17KB
192.168.1.2:51522<=> 208.111.168.6:80				121KB	770B	3.42KB	3.01KB
192.168.1.2:51479<=> 107.21.227.155:80				120KB	11.8KB	2.40KB	2.40KB
192.168.1.2:51530<=> 208.111.168.6:80				4.63KB	2.31KB	474B	118B
192.168.1.2:51492<=> 74.125.225.65:80				8.45KB	708B	142B	181B
192.168.1.2:51526<=> 23.0.32.80:80				1.79KB	400B	80B	46B
192.168.1.2:51527<=> 23.0.32.80:80				1.79KB	400B	80B	46B
199.47.217.148:80<=> 192.168.1.1:49221				581B	0B	58B	15B
192.168.1.2:49498<=> 63.236.253.98:80				208B	0B	21B	5B
192.168.1.2:49603<=> 199.47.216.17:443				104B	0B	5B	1B
192.168.1.2:51516<=> 208.111.168.6:80				1.75MB	0B	0B	44.9KB
192.168.1.2:51510<=> 208.111.168.6:80				1.10MB	0B	0B	28.1KB
TOTAL:							
TX:	cumm:	0B	peak:	rates:	0B	0B	0B
RX:		4.78MB		363KB	148KB	99.5KB	120KB
TOTAL:		4.78MB		363KB	148KB	99.5KB	120KB

	1B	12B	125B	1.25KB			
192.168.1.206:49193<=> 192.168.1.5:548			15.2KB	0B	1.18KB	778B	
192.168.1.189:49156<=> 192.168.1.5:548			5.10KB	0B	269B	261B	
192.168.1.128:49160<=> 192.168.1.5:548			25.7KB	0B	225B	1.29KB	
192.168.1.180:49156<=> 192.168.1.5:548			2.71KB	0B	139B	139B	
192.168.1.208:49154<=> 192.168.1.5:548			2.14KB	0B	110B	110B	
192.168.1.197:49165<=> 192.168.1.5:548			2.09KB	0B	107B	107B	
192.168.1.91:49162 <=> 192.168.1.5:548			1.88KB	0B	96B	96B	
192.168.1.201:49182<=> 192.168.1.5:548			1.82KB	0B	93B	93B	
192.168.1.90:49224 <=> 192.168.1.5:548			1.82KB	0B	93B	93B	
192.168.1.163:49156<=> 192.168.1.5:548			1.81KB	0B	93B	93B	
192.168.1.137:49200<=> 192.168.1.5:548			1.40KB	158B	87B	72B	
192.168.1.148:49157<=> 192.168.1.5:548			1.17KB	0B	68B	60B	
192.168.1.204:49165<=> 192.168.1.5:548			1.29KB	0B	66B	66B	
192.168.1.87:49158 <=> 192.168.1.5:548			1.29KB	0B	66B	66B	
192.168.1.138:49465<=> 192.168.1.5:548			792B	0B	40B	40B	
192.168.1.217:49177<=> 192.168.1.5:548			782B	0B	39B	39B	
192.168.1.125:49157<=> 192.168.1.5:548			782B	0B	39B	39B	
TOTAL:							
TX:	cumm:	0B	peak:	rates:	0B	0B	0B
RX:		75.3KB		19.5KB	192B	3.08KB	3.77KB
TOTAL:		75.3KB		19.5KB	192B	3.08KB	3.77KB

[<http://sickbits.networklabs.org/iftop-finding-traffic-hogs/>]

iftop – screen filter & config file

Press the “f” key to set a screen filter with regex

```
Screen filter> 192.168.1.2([0-4][0-9]|5[0-6])
```

	1B	12B	125B	1.25KB	12.5KB	125KB
192.168.1.212	49282	<=>	192.168.11.11:5222	20.6KB	0B	332B 189B
192.168.1.217	49268	<=>	199.47.219.151:80	580B	0B	58B 14B
192.168.1.208	5353	<=>	224.0.0.251:5353	412B	0B	41B 10B
192.168.1.217	5353	<=>	224.0.0.251:5353	407B	0B	41B 10B
192.168.1.219	5353	<=>	224.0.0.251:5353	5.30KB	0B	36B 39B
192.168.1.204	5353	<=>	224.0.0.251:5353	355B	0B	36B 9B
192.168.1.209	5353	<=>	224.0.0.251:5353	344B	0B	34B 9B
192.168.1.216	5353	<=>	224.0.0.251:5353	326B	0B	33B 8B
192.168.1.215	5353	<=>	224.0.0.251:5353	313B	0B	31B 8B
192.168.1.218	5353	<=>	224.0.0.251:5353	309B	0B	31B 8B
192.168.1.211	5353	<=>	224.0.0.251:5353	301B	0B	30B 8B
192.168.1.214	5353	<=>	224.0.0.251:5353	299B	0B	30B 7B
192.168.1.203	5353	<=>	224.0.0.251:5353	252B	0B	25B 6B
192.168.1.250	5353	<=>	224.0.0.251:5353	1.51KB	0B	0B 8B
192.168.1.246	138	<=>	255.255.255.255:138	229B	0B	0B 6B
192.168.1.217	17500	<=>	255.255.255.255:17500	560B	0B	0B 4B
192.168.1.212	34316	<=>	192.168.11.11:5060	128B	0B	0B 1B

TX:	cumm:	1.43MB	peak:	46.0KB	rates:	29.9KB	23.1KB	17.7KB
RX:		8.11MB		268KB		82.6KB	118KB	83.4KB
TOTAL:		9.54MB		300KB		113KB	141KB	101KB

Configuration file: ~/.iftoprc

```
# .iftoprc
# config file for iftop

dns-resolution: no
port-resolution: no
show-bars: yes
promiscuous: yes
port-display: on
hide-source: no
hide-destination: no
use-bytes: yes
line-display: one-line-both
show-totals: yes
log-scale: yes
```

[<http://sickbits.networklabs.org/iftop-finding-traffic-hogs/>]

tcpflow

a tcp/ip session reassembler:

```
$ tcpflow -i eth2 -e -c 'port 25'
```

```
tcpflow[30574]: listening on eth2
074.121.048.024.11013-208.110.191.226.00025: QUIT
208.110.191.226.00025-074.121.048.024.11013: 221 2.0.0 Bye
208.110.191.226.00025-074.121.048.024.19038: 221 2.0.0 Bye
097.107.025.014.36448-208.110.191.226.00025: QUIT
208.110.191.226.00025-097.107.025.014.36448: 221 2.0.0 Bye
208.110.191.226.00025-097.107.025.013.05337: 220 mail2.touhofclass.com ESMTP 220 mail2.touhofclass.com SMTP
097.107.025.013.05337-208.110.191.226.00025: EHLO outbound1.email1-pentonmkt.com
208.110.191.226.00025-097.107.025.013.05337: 250-mail2.touhofclass.com
208.110.191.226.00025-097.107.025.013.05337: 250-8BITMIME
250-ENHANCEDSTATUSCODES
250 SIZE
208.110.191.226.00025-067.138.109.167.56773: 220 mail2.touhofclass.com ESMTP 220 mail2.touhofclass.com SMTP
067.138.109.167.56773-208.110.191.226.00025: EHLO av67d578.avondano.com
208.110.191.226.00025-067.138.109.167.56773: 250-mail2.touhofclass.com
097.107.025.013.05337-208.110.191.226.00025: MAIL FROM:<multichannelmerchant@pentonmkt.com>
```

Blue = Client -> Server
Red = Server -> Client
Green = Undecided

Flow, both directions

[<http://sickbits.networklabs.org/tcpflow-a-tcp-ip-session-reassembler/>]

tcpflow

a tcp/ip session reassembler:

Color (**-e**), stdout (**-c**), snap length (**-b**)

tcpflow -i eth0 -b 96 -e -c port 80

```
root@nms:/home/jon/mycaps/flows# ls
023.060.078.066.00080-208.110.191.250.12848
023.060.079.055.00080-208.110.191.250.33712
023.060.112.069.00080-208.110.191.250.03871
023.060.112.069.00080-208.110.191.250.09582
023.060.112.069.00080-208.110.191.250.40581
023.060.112.069.00080-208.110.191.250.45994
023.060.112.069.00080-208.110.191.250.48362
023.060.126.163.00080-208.110.191.250.41548
023.060.126.163.00080-208.110.191.250.45239
023.060.126.163.00080-208.110.191.250.64556
050.097.092.252.00080-208.110.191.250.55214
063.084.095.026.00080-208.110.191.250.14055
```

\$ file ./*

```
064.208.138.214.00080-208.110.191.250.38338: ASCII text, with CRLF line terminators
064.208.138.214.00080-208.110.191.250.51652: ASCII text, with CRLF line terminators
066.114.053.020.00080-208.110.191.250.08920: ASCII text, with CRLF line terminators
066.114.053.020.00080-208.110.191.250.09424: data
066.114.053.020.00080-208.110.191.250.32213: DBase 3 data file (1901906576 records)
066.114.053.020.00080-208.110.191.250.32411: ASCII text, with CRLF line terminators
066.114.053.020.00080-208.110.191.250.43769: data
066.114.053.020.00080-208.110.191.250.54533: data
```

```
root@nms:/home/jon/mycaps# tcpflow -i eth3 -b 96 -e -c port 80
tcpflow[30648]: listening on eth3
208.110.191.250.09506-063.236.252.137.00080: GET /1999/100671894/r
Host: homedepot.ugc.bazaarvoice.com
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_8; en-u
1
Accept: */*
Referer: http://www.homedepot.com/Flooring-Floor-Tile-Wall-Tile-Ti
gId=10053&langId=-1&storeId=10051
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive

199.047.217.144.00080-208.110.191.250.38489: HTTP/1.1 200 OK
X-DB-Timeout: 120
Pragma: no-cache
Cache-Control: no-cache
Content-Type: text/plain
Date: Thu, 17 May 2012 12:11:19 GMT
Content-Length: 15
```

[<http://sickbits.networklabs.org/tcpflow-a-tcp-ip-session-reassembler/>]

tcpick



*tcp stream sniffer
and connection
tracker*

Read file (**-r**),
color output (**-C**),
display ports/ip/flags (**-h**),
print data to stdout (**-yP**)
, packet count (**-e**), and
set BPF filter

```
# tcpick -r 05-11-2012_12:30 eth3.pcap -C -h -yP -e 15 "port ( 21 or 20 )"
```

```
root@nms:/home/jon/mycaps/test# tcpick -r 05-11-2012_12:30_
Starting tcpick 0.2.1 at 2012-05-17 15:15 EDT
Timeout for connections is 600
when 15 packets will be sniffed, tcpick exits
tcpick: reading from 05-11-2012_12:30_eth3.pcap
setting filter: "port (ftp or ftp-data)"
208.110.191.250:4703 S > 64.14.24.6:ftp (0)
1 SYN-SENT 208.110.191.250:4703 > 64.14.24.6:ftp
64.14.24.6:ftp AS > 208.110.191.250:4703 (0)
1 SYN-RECEIVED 208.110.191.250:4703 > 64.14.24.6:ftp
208.110.191.250:4703 A > 64.14.24.6:ftp (0)
1 ESTABLISHED 208.110.191.250:4703 > 64.14.24.6:ftp
64.14.24.6:ftp AP > 208.110.191.250:4703 (39)
220 Welcome to Kalio, Inc FTP Server.
208.110.191.250:4703 A > 64.14.24.6:ftp (0)
208.110.191.250:4703 AP > 64.14.24.6:ftp (15)
USER [REDACTED]
64.14.24.6:ftp AP > 208.110.191.250:4703 (27)
331 Please send PASS now.
208.110.191.250:4703 A > 64.14.24.6:ftp (0)
208.110.191.250:4703 AP > 64.14.24.6:ftp (14)
PASS [REDACTED]
64.14.24.6:ftp AP > 208.110.191.250:4703 (25)
230-Welcome [REDACTED]
```

[<http://sickbits.networklabs.org/tcpick-tcp-stream-sniffer-and-connection-tracker/>]

tcpick



```
# tcpick -r 05-11-2012_12:30 eth3.pcap -C -h -wR -e 10 "port 25"
```

*tcp stream sniffer
and connection
tracker*

Read file (**-r**),
color output (**-C**),
display ports/ip/flags (**-h**),
write to cwd (**-wR**),
packet count (**-e**), and set
BPF filter

```
root@nms:/home/jon/mycaps/test# tcpick -r 05-11-2012_12:30 eth3.pcap -C -h -wR -e 10 "port 25"
Starting tcpick 0.2.1 at 2012-05-17 15:11 EDT
Timeout for connections is 600
when 10 packets will be sniffed, tcpick exits
tcpick: reading from 05-11-2012_12:30_eth3.pcap
setting filter: "port 25"
173.213.100.10:59440 S > 208.110.191.226:smtp (0)
1 SYN-SENT 173.213.100.10:59440 > 208.110.191.226:smtp
208.110.191.226:smtp AS > 173.213.100.10:59440 (0)
1 SYN-RECEIVED 173.213.100.10:59440 > 208.110.191.226:smtp
173.213.100.10:59440 A > 208.110.191.226:smtp (0)
1 ESTABLISHED 173.213.100.10:59440 > 208.110.191.226:smtp
208.110.191.226:smtp AP > 173.213.100.10:59440 (66)
173.213.100.10:59440 A > 208.110.191.226:smtp (0)
173.213.100.10:59440 AP > 208.110.191.226:smtp (23)
208.110.191.226:smtp A > 173.213.100.10:59440 (0)
208.110.191.226:smtp AP > 173.213.100.10:59440 (28)
173.213.100.10:59440 A > 208.110.191.226:smtp (0)
208.110.191.226:smtp AP > 173.213.100.10:59440 (49)

10 packets captured
1 tcp sessions detected
```

Server and Client flows:

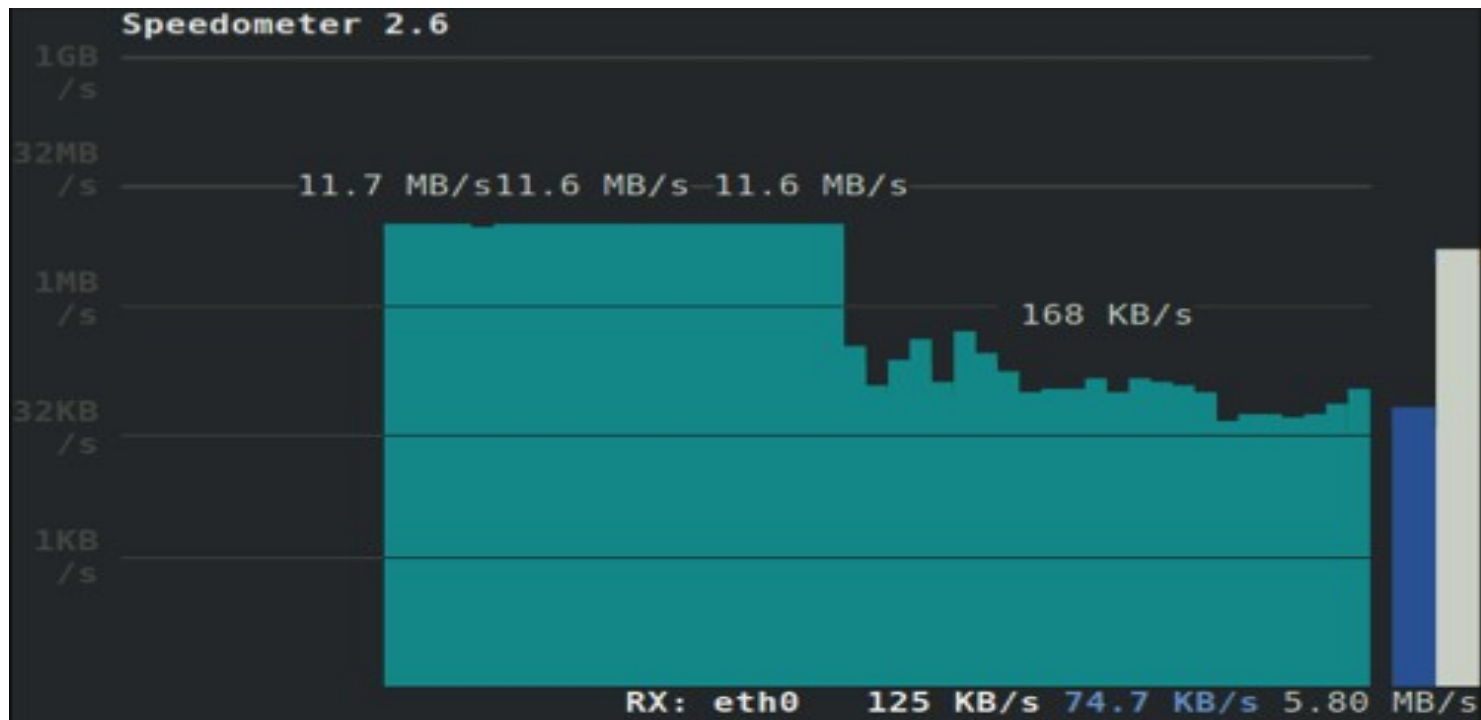
```
root@nms:/home/jon/mycaps/test# ls
05-11-2012_12:30_eth3.pcap          tcpick_173.213.100.10_208.110.191.226_smtp.serv.dat
tcpick_173.213.100.10_208.110.191.226_smtp.clnt.dat  tcpick_173.213.100.10_208.110.191.226_smtp.serv.lck
tcpick_173.213.100.10_208.110.191.226_smtp.clnt.lck
```

[<http://sickbits.networklabs.org/tcpick-tcp-stream-sniffer-and-connection-tracker/>]

speedometer

speedometer is a simple bandwidth utilization sensing tool that displays the current throughput usage in a moving bar graph fashion.

\$ speedometer -rx eth0

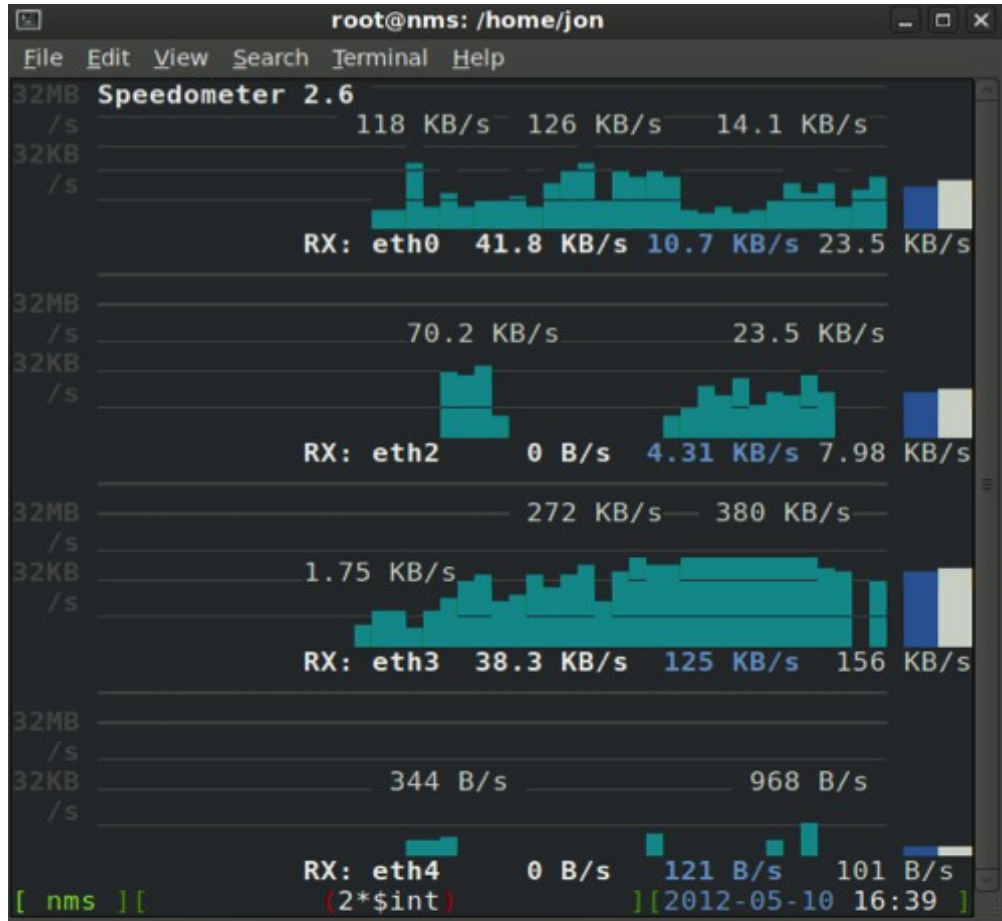


[<http://sickbits.networklabs.org/speedometer-a-graphic-network-throughput-tool/>]

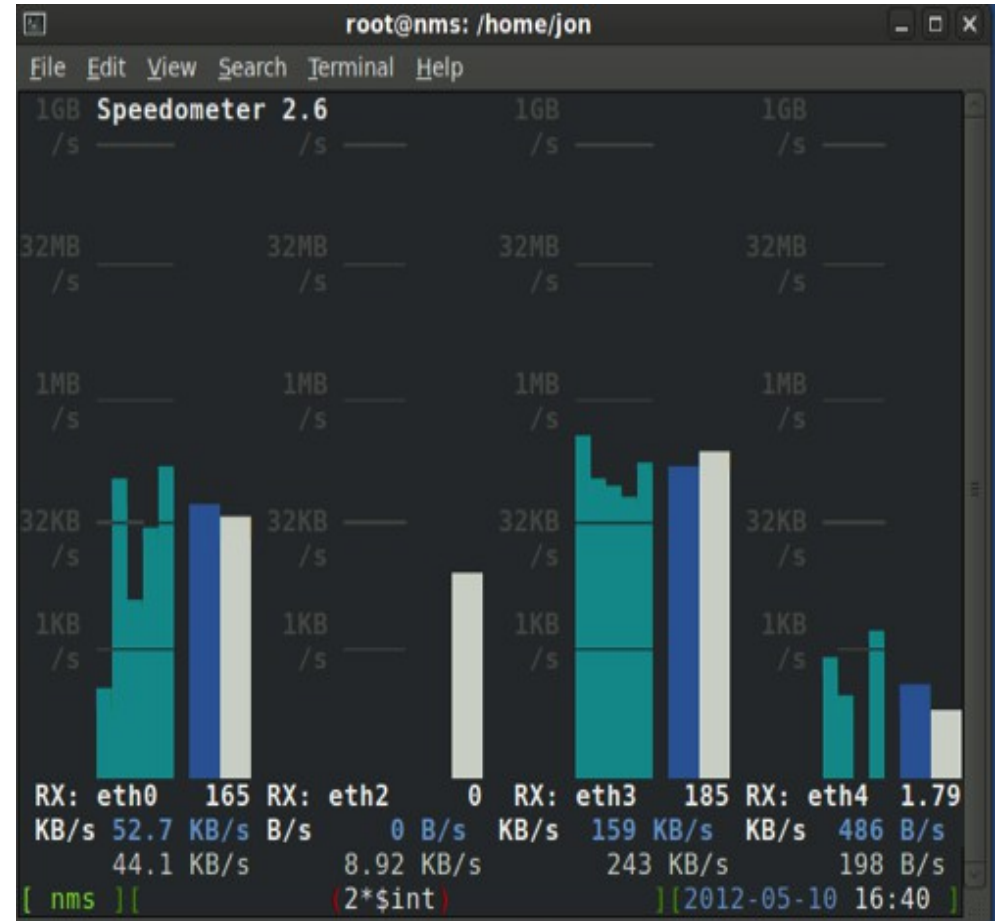
speedometer

Speedometer can handle multiple interfaces at once

```
$ speedometer -rx eth0 -rx eth2 -rx eth3 -rx eth4
```



```
$ speedometer -rx eth0 -c -rx eth2 -c -rx -eth3 -c -rx -eth4
```



[<http://sickbits.networklabs.org/speedometer-a-graphic-network-throughput-tool/>]

ngrep – network grep

```
$ ngrep -S 49 -qI 05-11-2012_12\:30 eth3.pcap "GET *.jpg" 'port 80' -n 3
```

```
root@nms:/home/jon/mycaps# ngrep -S 49 -qI 05-11-2012 12\:30 eth3.pcap "GET *.jpg" 'port 80' -n 3
input: 05-11-2012_12:30_eth3.pcap
filter: (ip) and ( port 80 )
match: GET *.jpg

T 208.110.191.250:4041 -> 64.14.24.79:80 [A]
GET /images/xxs/B864-001Greensw.jpg HTTP/1.1..Hos

T 208.110.191.250:3613 -> 208.111.168.6:80 [AP]
GET /images/art/0511SlidePromoImage1.jpg HTTP/1.1

T 208.110.191.250:20626 -> 208.111.168.6:80 [AP]
GET /images/art/0511SlidePromoImage2.jpg HTTP/1.1
```

Displays the first 49 bytes (**-S**) of packet, quiet mode (**-q**), read input from file (**-I**), grab first 3 packets (**-n**)

Note: (**-S**) is not the same as the snap length (**-s**) which specifies the size to capture.
[me]

ngrep – web traffic log

```
# ngrep -l bad_user.pcap -q -W single -t "GET" ip src 192.168.1.1 | awk  
'{ print $2, $3, $11, $9}' | sed 's/\.\{1,3\}User-Agent//' | grep -v -E '(ad|  
cache|analytics|wxdata|voicefive|imwx|weather.com|counterpath|  
cloudfront|2mdn.net|click|api|acuity|tribal|pixel|touchofclass|flickr|  
yting|pulse|twitter|facebook|graphic|revsci|digi|rss|cdn|brightcove|  
atdmt|btrll|metric|content|trend|serv|content|global|fwmmr|typekit|[a-  
z]*-[a-z]*\.com|pinit|cisco|tumblr)' | sed '/ [\t]*$/d' > url.txt
```

```
root@nms:/home/jon# head -60 ./mypcaps/url.txt | grep www\.*.com  
2012/05/11 14:55:15.675703 www.google.com: /url?sa=t&rct=j&q=812-6  
%2Factiverrain.com%2Fmonicaleitch&ei=gmCtT-DlE4ym8ATw15XdDA&usg=AFQ  
2012/05/11 14:55:16.203936 www.google.com: /cse/brand?form=cse-sea  
2012/05/11 14:55:15.675708 www.google.com: /url?sa=t&rct=j&q=812-6  
%2Factiverrain.com%2Fmonicaleitch&ei=gmCtT-DlE4ym8ATw15XdDA&usg=AFQ  
2012/05/11 14:55:16.203941 www.google.com: /cse/brand?form=cse-sea  
2012/05/11 14:55:22.773408 www.hiconversion.com: /enabling/update.  
2012/05/11 14:55:25.116847 www.procato.com: /rgb/EFEFDF/  
2012/05/11 14:55:36.638038 www.hiconversion.com: /enabling/update.  
2012/05/11 14:55:41.195917 www.hiconversion.com: /enabling/update.  
2012/05/11 14:55:44.782370 www.hiconversion.com: /enabling/update.  
2012/05/11 14:55:52.436037 www.procato.com: /_pub/?format=984x120_  
2012/05/11 14:56:22.080632 www.google.com: /search?client=safari&r  
2012/05/11 14:56:22.570601 www.google.com: /csi?v=3&s=web&action=&
```

[me]

Contact

✦ Questions, suggestions, polite criticism:
jonschipp@gmail.com

✦ More info:

sickbits.net
dclinux.org