HOWTO for the linux packet generator

Date: 041221

Enable CONFIG_NET_PKTGEN to compile and build pktgen.o either in kernel
or as module. Module is preferred. insmod pktgen if needed. Once running
pktgen creates a thread on each CPU where each thread has affinity to its CPU.
Monitoring and controlling is done via /proc. Easiest to select a suitable
a sample script and configure.

On a dual CPU:

```
ps aux | grep pkt
root       129  0.3  0.0     0    0 ?        SW    2003 523:20 [pktgen/0]
root       130  0.3  0.0     0    0 ?        SW    2003 509:50 [pktgen/1]
```

For monitoring and control pktgen creates:
```
        /proc/net/pktgen/pgctrl
        /proc/net/pktgen/kpktgend_X
        /proc/net/pktgen/ethX
```

Viewing threads

```
/proc/net/pktgen/kpktgend_0
Name: kpktgend_0  max_before_softirq: 10000
Running:
Stopped: eth1
Result: OK: max_before_softirq=10000
```

Most important the devices assigned to thread. Note! A device can only belong
to one thread.

Viewing devices

Parm section holds configured info. Current hold running stats.
Result is printed after run or after interruption. Example:

```
/proc/net/pktgen/eth1

Params: count 10000000  min_pkt_size: 60  max_pkt_size: 60
     frags: 0  delay: 0  clone_skb: 1000000  ifname: eth1
     flows: 0 flowlen: 0
     dst_min: 10.10.11.2  dst_max:
     src_min:   src_max:
     src_mac: 00:00:00:00:00:00  dst_mac: 00:04:23:AC:FD:82
     udp_src_min: 9  udp_src_max: 9  udp_dst_min: 9  udp_dst_max: 9
     src_mac_count: 0  dst_mac_count: 0
     Flags:
Current:
```

```
     pkts-sofar: 10000000  errors: 39664
     started: 1103053986245187us  stopped: 1103053999346329us idle: 880401us
     seq_num: 10000011  cur_dst_mac_offset: 0  cur_src_mac_offset: 0
     cur_saddr: 0x10a0a0a  cur_daddr: 0x20b0a0a
     cur_udp_dst: 9  cur_udp_src: 9
     flows: 0
Result: OK: 13101142(c12220741+d880401) usec, 10000000 (60byte,0frags)
  763292pps 390Mb/sec (390805504bps) errors: 39664

Configuring threads and devices
===============================
This is done via the /proc interface easiest done via pgset in the scripts

Examples:

 pgset "clone_skb 1"     sets the number of copies of the same packet
 pgset "clone_skb 0"     use single SKB for all transmits
 pgset "pkt_size 9014"   sets packet size to 9014
 pgset "frags 5"         packet will consist of 5 fragments
 pgset "count 200000"    sets number of packets to send, set to zero
                         for continuous sends until explicitly stopped.

 pgset "delay 5000"      adds delay to hard_start_xmit(). nanoseconds

 pgset "dst 10.0.0.1"    sets IP destination address
                         (BEWARE! This generator is very aggressive!)

 pgset "dst_min 10.0.0.1"            Same as dst
 pgset "dst_max 10.0.0.254"          Set the maximum destination IP.
 pgset "src_min 10.0.0.1"            Set the minimum (or only) source IP.
 pgset "src_max 10.0.0.254"          Set the maximum source IP.
 pgset "dst6 fec0::1"     IPV6 destination address
 pgset "src6 fec0::2"     IPV6 source address
 pgset "dstmac 00:00:00:00:00:00"    sets MAC destination address
 pgset "srcmac 00:00:00:00:00:00"    sets MAC source address

 pgset "queue_map_min 0" Sets the min value of tx queue interval
 pgset "queue_map_max 7" Sets the max value of tx queue interval, for multiqueue
devices
                         To select queue 1 of a given device,
                         use queue_map_min=1 and queue_map_max=1

 pgset "src_mac_count 1" Sets the number of MACs we'll range through.
                         The 'minimum' MAC is what you set with srcmac.

 pgset "dst_mac_count 1" Sets the number of MACs we'll range through.
                         The 'minimum' MAC is what you set with dstmac.

 pgset "flag [name]"     Set a flag to determine behaviour.  Current flags
                         are: IPSRC_RND #IP Source is random (between min/max),
                              IPDST_RND, UDPSRC_RND,
                              UDPDST_RND, MACSRC_RND, MACDST_RND
                              MPLS_RND, VID_RND, SVID_RND
                              QUEUE_MAP_RND # queue map random
                              QUEUE_MAP_CPU # queue map mirrors
smp_processor_id()
```

```
 pgset "udp_src_min 9"   set UDP source port min, If < udp_src_max, then
                         cycle through the port range.

 pgset "udp_src_max 9"   set UDP source port max.
 pgset "udp_dst_min 9"   set UDP destination port min, If < udp_dst_max, then
                         cycle through the port range.
 pgset "udp_dst_max 9"   set UDP destination port max.

 pgset "mpls 0001000a,0002000a,0000000a" set MPLS labels (in this example
                                         outer label=16,middle label=32,
                                         inner label=0 (IPv4 NULL)) Note that
                                         there must be no spaces between the
                                         arguments. Leading zeros are required.
                                         Do not set the bottom of stack bit,
                                         that's done automatically. If you do
                                         set the bottom of stack bit, that
                                         indicates that you want to randomly
                                         generate that address and the flag
                                         MPLS_RND will be turned on. You
                                         can have any mix of random and fixed
                                         labels in the label stack.

 pgset "mpls 0"          turn off mpls (or any invalid argument works too!)

 pgset "vlan_id 77"      set VLAN ID 0-4095
 pgset "vlan_p 3"        set priority bit 0-7 (default 0)
 pgset "vlan_cfi 0"      set canonical format identifier 0-1 (default 0)

 pgset "svlan_id 22"     set SVLAN ID 0-4095
 pgset "svlan_p 3"       set priority bit 0-7 (default 0)
 pgset "svlan_cfi 0"     set canonical format identifier 0-1 (default 0)

 pgset "vlan_id 9999"    > 4095 remove vlan and svlan tags
 pgset "svlan 9999"      > 4095 remove svlan tag


 pgset "tos XX"          set former IPv4 TOS field (e.g. "tos 28" for AF11 no
ECN, default 00)
 pgset "traffic_class XX" set former IPv6 TRAFFIC CLASS (e.g. "traffic_class B8"
for EF no ECN, default 00)

 pgset stop              aborts injection. Also, ^C aborts generator.


Example scripts
===============

A collection of small tutorial scripts for pktgen is in examples dir.

pktgen.conf-1-1                 # 1 CPU 1 dev
pktgen.conf-1-2                 # 1 CPU 2 dev
pktgen.conf-2-1                 # 2 CPU's 1 dev
pktgen.conf-2-2                 # 2 CPU's 2 dev
pktgen.conf-1-1-rdos            # 1 CPU 1 dev w. route DoS
```

```
pktgen.conf-1-1-ip6              # 1 CPU 1 dev ipv6
pktgen.conf-1-1-ip6-rdos         # 1 CPU 1 dev ipv6  w. route DoS
pktgen.conf-1-1-flows            # 1 CPU 1 dev multiple flows.
```

Run in shell: ./pktgen.conf-X-Y It does all the setup including sending.

Interrupt affinity
==================
Note when adding devices to a specific CPU there good idea to also assign
/proc/irq/XX/smp_affinity so the TX-interrupts gets bound to the same CPU.
as this reduces cache bouncing when freeing skb's.

Current commands and configuration options
==

** Pgcontrol commands:

start
stop

** Thread commands:

add_device
rem_device_all
max_before_softirq

** Device commands:

count
clone_skb
debug

frags
delay

src_mac_count
dst_mac_count

pkt_size
min_pkt_size
max_pkt_size

mpls

udp_src_min
udp_src_max

udp_dst_min
udp_dst_max

flag
  IPSRC_RND
  TXSIZE_RND

  IPDST_RND
  UDPSRC_RND
  UDPDST_RND
  MACSRC_RND
  MACDST_RND

dst_min
dst_max

src_min
src_max

dst_mac
src_mac

clear_counters

dst6
src6

flows
flowlen

References:
ftp://robur.slu.se/pub/Linux/net-development/pktgen-testing/
ftp://robur.slu.se/pub/Linux/net-development/pktgen-testing/examples/

Paper from Linux-Kongress in Erlangen 2004.
ftp://robur.slu.se/pub/Linux/net-development/pktgen-testing/pktgen_paper.pdf

Thanks to:
Grant Grundler for testing on IA-64 and parisc, Harald Welte, Lennert Buytenhek
Stephen Hemminger, Andi Kleen, Dave Miller and many others.

Good luck with the linux net-development.