

# **PktGen**

Versione: 0.1  
Data: 25/09/06  
Riservatezza: N/A  
Numero di pagine: 11

AUTORE: Cicala Gabriele  
REVISIONE: N/A  
APPROVAZIONE: N/A

## **Executive summary**

Il presente documento costituisce la versione 0.1 di una breve descrizione che illustra il funzionamento dell'utility pktgen che permette la generazione di pacchetti attraverso il kernel Linux.

## Sommario

|  |                    |
|--|--------------------|
| <a href="#">PktGen.....</a>                        | <a href="#">1</a>  |
| <a href="#">Sommario.....</a>                      | <a href="#">2</a>  |
| <a href="#">Introduzione.....</a>                  | <a href="#">3</a>  |
| <a href="#">Compilazione del kernel Linux.....</a> | <a href="#">3</a>  |
| <a href="#">Performance.....</a>                   | <a href="#">3</a>  |
| <a href="#">Comandi disponibili.....</a>           | <a href="#">4</a>  |
| <a href="#">start.....</a>                         | <a href="#">4</a>  |
| <a href="#">stop.....</a>                          | <a href="#">4</a>  |
| <a href="#">add_device.....</a>                    | <a href="#">4</a>  |
| <a href="#">rem_device_all.....</a>                | <a href="#">4</a>  |
| <a href="#">max_before_softirq.....</a>            | <a href="#">4</a>  |
| <a href="#">debug.....</a>                         | <a href="#">4</a>  |
| <a href="#">clone_skb.....</a>                     | <a href="#">4</a>  |
| <a href="#">clear_counters.....</a>                | <a href="#">4</a>  |
| <a href="#">pkt_size.....</a>                      | <a href="#">5</a>  |
| <a href="#">min_pkt_size.....</a>                  | <a href="#">5</a>  |
| <a href="#">max_pkt_size.....</a>                  | <a href="#">5</a>  |
| <a href="#">frags.....</a>                         | <a href="#">5</a>  |
| <a href="#">count.....</a>                         | <a href="#">5</a>  |
| <a href="#">ipg.....</a>                           | <a href="#">5</a>  |
| <a href="#">dst.....</a>                           | <a href="#">5</a>  |
| <a href="#">dst_min.....</a>                       | <a href="#">5</a>  |
| <a href="#">dst_max.....</a>                       | <a href="#">5</a>  |
| <a href="#">src_min.....</a>                       | <a href="#">5</a>  |
| <a href="#">src_max.....</a>                       | <a href="#">6</a>  |
| <a href="#">dst6.....</a>                          | <a href="#">6</a>  |
| <a href="#">src6.....</a>                          | <a href="#">6</a>  |
| <a href="#">dstmac.....</a>                        | <a href="#">6</a>  |
| <a href="#">srcmac.....</a>                        | <a href="#">6</a>  |
| <a href="#">src_mac_count.....</a>                 | <a href="#">6</a>  |
| <a href="#">dst_mac_count.....</a>                 | <a href="#">6</a>  |
| <a href="#">flag [name].....</a>                   | <a href="#">6</a>  |
| <a href="#">udp_src_min.....</a>                   | <a href="#">7</a>  |
| <a href="#">udp_src_max.....</a>                   | <a href="#">7</a>  |
| <a href="#">udp_dst_min.....</a>                   | <a href="#">7</a>  |
| <a href="#">udp_dst_max.....</a>                   | <a href="#">7</a>  |
| <a href="#">flows.....</a>                         | <a href="#">7</a>  |
| <a href="#">flowlen.....</a>                       | <a href="#">7</a>  |
| <a href="#">Script d'uso.....</a>                  | <a href="#">7</a>  |
| <a href="#">Utilizzo.....</a>                      | <a href="#">9</a>  |
| <a href="#">Conclusioni.....</a>                   | <a href="#">10</a> |
| <a href="#">Bibliografia.....</a>                  | <a href="#">11</a> |

## **Introduzione**

Questo modulo permette la injection di pacchetti standardizzati, con un rate configurabile attraverso una specifica interfaccia. Viene utilizzato ai fini di stress testing e analisi delle performance di interfacce di rete.

É da ricordare che questo tool permette la generazione di pacchetti. Allo stato attuale non si riesce a inviare un throughput costante in termini di bps. Le variabili su cui si può agire sono la grandezza dei pacchetti e la loro quantità.

## **Compilazione del kernel Linux**

Il modulo in oggetto viene incluso nel kernel Linux ma non è immediatamente disponibile con le release che generalmente vengono installate con la distribuzione scelta. Per attivarlo si rende necessario ricompilare il kernel.

Al seguente link vi è una comoda guida per la compilazione del kernel per sistemi Debian:  
[http://www.falkotimme.com/howtos/debian\\_kernel2.6\\_compile/](http://www.falkotimme.com/howtos/debian_kernel2.6_compile/)

In genere si consiglia di impostare l'opzione CONFIG\_NET\_PKTGEN a M (è consigliato usare il modulo piuttosto che renderlo nativo nel kernel compilato).

Una volta entrati nella schermata principale (ovvero dopo aver lanciato make menuconfig) per l'impostazione del kernel il percorso da seguire è il seguente:  
Networking -> Networking Options -> Network Testing -> Packet Generator.

## **Performance**

Su di un laptop ACER ASPIRE 5672WLMi così configurato:

PROCESSORE: Intel Core Duo processor T2300 (1.66 Ghz, 667 Mhz FSB, 2MB L2 cache)

HDD: 100 GB 5400rpm SATA

RAM: 1GB DDR2

NETWORK CARD: Broadcom Corporation NetLink BCM5789 Gigabit Ethernet PCI Express (rev 21)

KERNEL LINUX: 2.6.17.11.

si hanno performance del seguente tipo:

con pacchetti di 64 byte: 522927pps 267Mb/sec (267738624bps)

con pacchetti di 1500 byte: 82008pps 984Mb/sec (984096000bps)

Ovviamente le performance cambiano in dipendenza dell'hardware utilizzato.

## **Comandi disponibili**

Di seguito l'elenco dei comandi disponibili:

### **start**

Comando col quale si comincia la generazione dei pacchetti.

### **stop**

Ferma la injection dei pacchetti. Oltre a questo comando è possibile fermare la generazione dei pacchetti semplicemente con la combinazione dei tasti Ctrl-C.

In genere, quando si hanno più interfacce di rete, è consigliabile non limitare il count dei pacchetti (ovvero impostarlo a 0) e utilizzare la combinazione Ctrl-C per arrestare la injection. In questo modo non si rischia di inficiare il test a causa di differenti tempi di stop sulle interfacce.

### **add\_device**

Aggiunge un device al processo di generazione. Generalmente eth0, eth1, ecc.

### **rem\_device\_all**

Rimuove tutti i device precedentemente configurati attraverso lo script di configurazione.

### **max\_before\_softirq**

Permette la do\_softirq() dopo aver spedito un numero di pacchetti.

### **debug**

### **clone\_skb**

Numero identicamente uguale di copie dello stesso pacchetto.

### **clear\_counters**

Cancellazione dei contatori.

## **pkt\_size**

Grandezza del pacchetto a livello data link escluso il CRC (4 byte).

## **min\_pkt\_size**

Attraverso questa e la successiva direttiva è possibile impostare un range di valori per il campo `pkt_size`.

## **max\_pkt\_size**

## **frags**

Numero di frammenti per un pacchetto.

## **count**

Numero di pacchetti da spedire. In caso il valore fosse uguale a 0 si avrebbe un flusso continuo.

## **ipg**

Inter-packet gap. Indica un valore artificiale di gap che è possibile inserire tra i pacchetti indicato in nanosecondi.

## **dst**

Indirizzo IP di destinazione.

## **dst\_min**

Come per il `pkt_size` è possibile stabilire, tramite la `dst_max`, un range di valori di IP.

## **dst\_max**

## **src\_min**

Idem come prima per il sorgente.

## **src\_max**

## **dst6**

Indirizzo di destinazione in formato IPv6.

## **src6**

Indirizzo sorgente in formato IPv6.

## **dstmac**

MAC di destinazione.

## **srcmac**

MAC sorgente.

## **src\_mac\_count**

Anche per i valori di MAC sorgente è possibile stabilire un range di valori. Il valore da cui partire è quello impostato tramite la direttiva srcmac.

## **dst\_mac\_count**

Idem come prima per quanto riguarda di MAC destinazione.

## **flag [name]**

Flag utilizzato per modificare il comportamento del packet generator. I valori possibili sono:

IPSRC\_RND: IP sorgente viene generato in modalità random.

IPDST\_RND: idem come prima per il valore di IP di destinazione.

TXSIZE\_RND:

UDPSRC\_RND:

UDPDEST\_RND:

MACSRC\_RND:

MACDST\_RND:

## **udp\_src\_min**

Anche per le porte UDP sorgente è possibile stabilire un range di valori. Con questo parametro si definisce il valore minimo.

## **udp\_src\_max**

Con questo parametro si definisce il valore massimo.

## **udp\_dst\_min**

Stesso comportamento per le porte di destinazione.

## **udp\_dst\_max**

## **flows**

Numero di flussi concorrenti.

## **flowlen**

Lunghezza del flusso.

## ***Script d'uso***

Il tool di pktgen è configurabile esclusivamente attraverso il filesystem `/proc`; la via più veloce è utilizzando uno script.

Come si vedrà dagli esempi disponibili ai link inseriti in bibliografia gli script disponibili sono vari; generalmente dipendono dal numero di processori e dal numero di schede di rete disponibili. Vengono distinti al seguente modo: `pktgen.conf-(num_processori-num_schede_rete)`.

Nel nostro caso abbiamo 2 processori ed una scheda di rete per cui lo script utile è il `pktgen.conf-2-1`.

Per le nostre prove è stato utilizzato il seguente:

```
#!/bin/sh

#modprobe pktgen

function pgset() {
    local result

    echo $1 > $PGDEV

    result=`cat $PGDEV | fgrep "Result: OK:"`
}
```

```

    if [ "$result" = "" ]; then
        cat $PGDEV | fgrep Result:
    fi
}

function pg() {
    echo inject > $PGDEV
    cat $PGDEV
}
if [ -z $2 ]; then
    echo "Usage: $0 {packet_size count}"
    exit 1
fi

# Config Start Here -----

# thread config
# Each CPU has own thread. Two CPU example. We add eth0, eth1 respectively.

PGDEV=/proc/net/pktgen/kpktgend_0
    echo "Removing all devices"
    pgset "rem_device_all"
    echo "Adding eth0"
    pgset "add_device eth0"
    echo "Setting max_before_softirq 10000"
    pgset "max_before_softirq 10000"

# We need to remove old config since we dont use this thread. We can only
# one NIC on one CPU due to affinity reasons.

PGDEV=/proc/net/pktgen/kpktgend_1
    echo "Removing all devices"
    pgset "rem_device_all"

# device config
# delay 0 means maximum speed.

CLONE_SKB="clone_skb 10000000"
# NIC adds 4 bytes CRC
PKT_SIZE="pkt_size "$1

# COUNT 0 means forever
COUNT="count "$2
DELAY="delay 0"
FLOWS="flows 0"
FLOWLEN="flowlen 0"
DST="dst 192.168.10.1"
DST_MAC="dst_mac 00:14:22:75:AB:42"

PGDEV=/proc/net/pktgen/eth0
    echo "Configuring $PGDEV"
    pgset "$COUNT"
    pgset "$CLONE_SKB"
    pgset "$PKT_SIZE"

```

```

pgset "$DELAY"
pgset "$FLOWS"
pgset "$FLOWLEN"
pgset "$DST"
pgset "$DST_MAC"

# Time to run
PGDEV=/proc/net/pktgen/pgctrl

echo "Running... ctrl^C to stop"
pgset "start"
echo "Done"
echo "";echo "Destination is: "
echo "$DST"
echo "$DST_MAC"
echo ""

# Result can be viewed in /proc/net/pktgen/eth0
echo "Result can be viewed in /proc/net/pktgen/eth0"
echo " "
sleep 2
echo
"*****
* "
cat /proc/net/pktgen/eth0
echo
"*****
* "

```

## **Utilizzo**

Dato che in fase di impostazione e compilazione del kernel si è deciso di utilizzare l'opzione "modulo" per quanto riguarda pktgen, la prima operazione da fare è il suo caricamento nell'insieme dei moduli in uso.

Se la sua installazione è stata corretta basta utilizzare il comando modprobe:

```
# modprobe -a pktgen
```

Successivamente lanciare lo script precedentemente editato per il nostro caso:

```
./pktgen.conf-2-1 64 10000
```

Il primo valore numerico dopo lo script indica la variabile `pkt_size` e il successivo la variabile `count`.

Una volta terminato l'invio dei pacchetti, si avrà in output la seguente schermata:

```
# ./pktgen.conf-2-1 64 100000
Removing all devices
Adding eth0
```

```
Setting max_before_softirq 10000
Removing all devices
Configuring /proc/net/pktgen/eth0
Running... ctrl^C to stop
Done
```

```
Destination is:
dst 192.168.10.1
dst_mac 00:14:22:75:AB:42
```

Result can be viewed in /proc/net/pktgen/eth0

```
*****
Params: count 100000  min_pkt_size: 64  max_pkt_size: 64
       frags: 0  delay: 0  clone_skb: 1000000  ifname: eth0
       flows: 0  flowlen: 0
       dst_min: 192.168.10.1  dst_max:
       src_min:  src_max:
       src_mac: 00:16:36:38:80:AF  dst_mac: 00:14:22:75:AB:42
       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: 100000  errors: 0
       started: 1158232049436117us  stopped: 1158232049627774us  idle: 2043us
       seq_num: 100002  cur_dst_mac_offset: 0  cur_src_mac_offset: 0
       cur_saddr: 0x20aa8c0  cur_daddr: 0x10aa8c0
       cur_udp_dst: 9  cur_udp_src: 9
       flows: 0
Result: OK: 191657(c189614+d2043) usec, 100000 (64byte,0frags)
       521765pps 267Mb/sec (267143680bps) errors: 0


---


```

## ***Conclusioni***

Il tool è stato utilizzato per il testing della scheda di acquisizione **Endace Dag 4.3 GE**. In questa prima fase di testing è stato utilizzato un laptop ma la giusta collocazione del tool è

su di una macchina desktop/server molto più preformante che permetta l'alloggiamento di più processori e parallelamente più schede di rete. La motivazione è data dalla possibilità di pilotare una scheda di rete per ogni processore. In questo modo si avrebbe un throuput maggiore.

Dal punto di vista schematico si avrebbero due o più punti sorgente di pacchetti (attraverso le due interfacce). Entrambi potrebbero essere convogliati sullo stesso switch per ottenere così sulla porta di trunking un traffico vicino o maggiore al Gigabit.

Nota dolente di questo tool è il fatto che allo stato attuale non è possibile generare un traffico con un throughput ben specifico.

## ***Bibliografia***

- [1] <ftp://robur.slu.se/pub/Linux/net-development/pktgen-testing/>
- [2] <ftp://robur.slu.se/pub/Linux/net-development/pktgen-testing/examples/>
- [3] <ftp://gsyprf10.external.hp.com/pub/pktgen-testing/>
- [4] <http://linux-net.osdl.org/index.php/Pktgen>
- [5] <http://www.tnt.dist.unige.it/np/index.php/PktGen>
- [6] <http://www.tnt.dist.unige.it/np/index.php/Files>
- [7] [http://iou.parisc-linux.org/ols\\_2004/documents/pktgen-testing/](http://iou.parisc-linux.org/ols_2004/documents/pktgen-testing/)