ipgen - Interactive Packet Generator

https://github.com/iij/ipgen/

ryo@iij.ad.jp

ipgen とは？

- ipgen とは、パケットジェネレータ、ベンチマーク、パフォーマンス測定ツールです。

動機

- 私たちは日々ルータの開発をしています
- ルータを作る上で、IPフォワーディングのチューニングやテストをするために、何度もパフォーマンステストを行わなければなりません
- その際、いろんなツールや測定器を使います

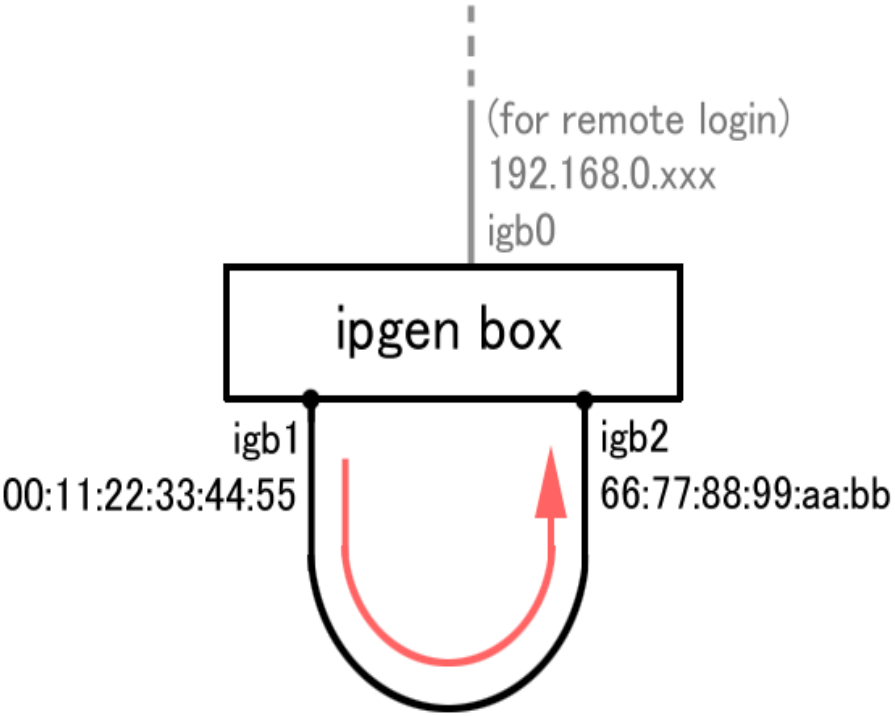
- ftp, ping –f, iperf, ttcp, nuttcp, etc...
  - シンプルでお手軽 :-)
  - 測定装置(PC)自体のOSやIPスタックのパフォーマンスに依存する → ワイヤーレート出せないことが多い。:-(
  - おおよその性能は測定できるが、詳細な値はわからない :-(
- 商用のテスト装置（SPIRENT communications, Ixia, Artiza Networks, etc...）
  - 信頼性の高いベンチマークテストを行うことができる :-)
  - 値段が高い！ :-(
  - いつも誰かが使っているので、測定機器が空いてない。(EBUSY) :-(

- 自席で手軽に実行できるものが欲しい！
  - NetBSD MP network stack project の開発効率を上げたい
- ベンチマーク結果は、商用製品のそれとは言わないまでも、それなりの精度は欲しい

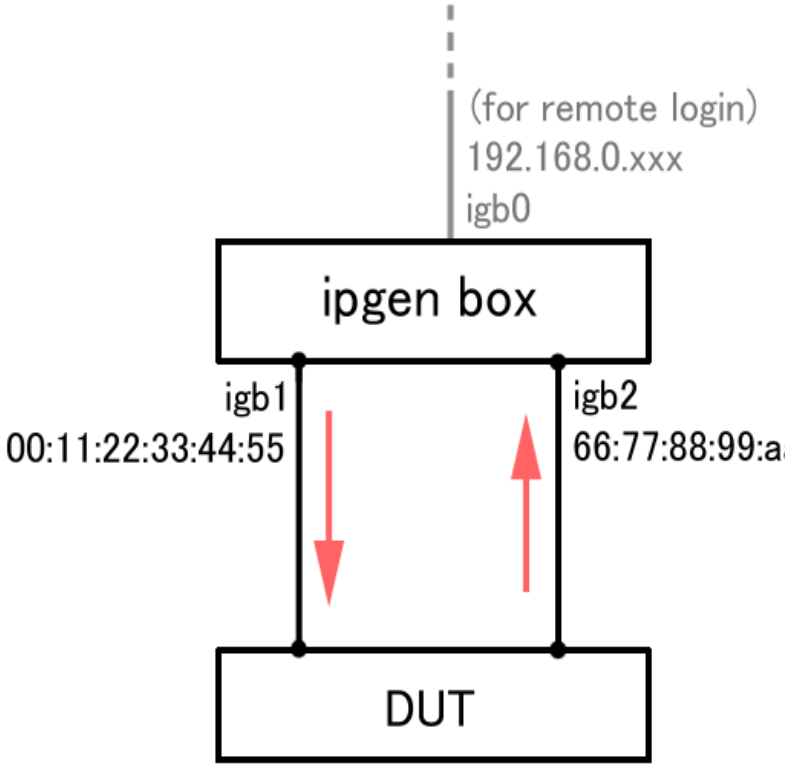
というわけで作りました！

構成例その1

loopback test



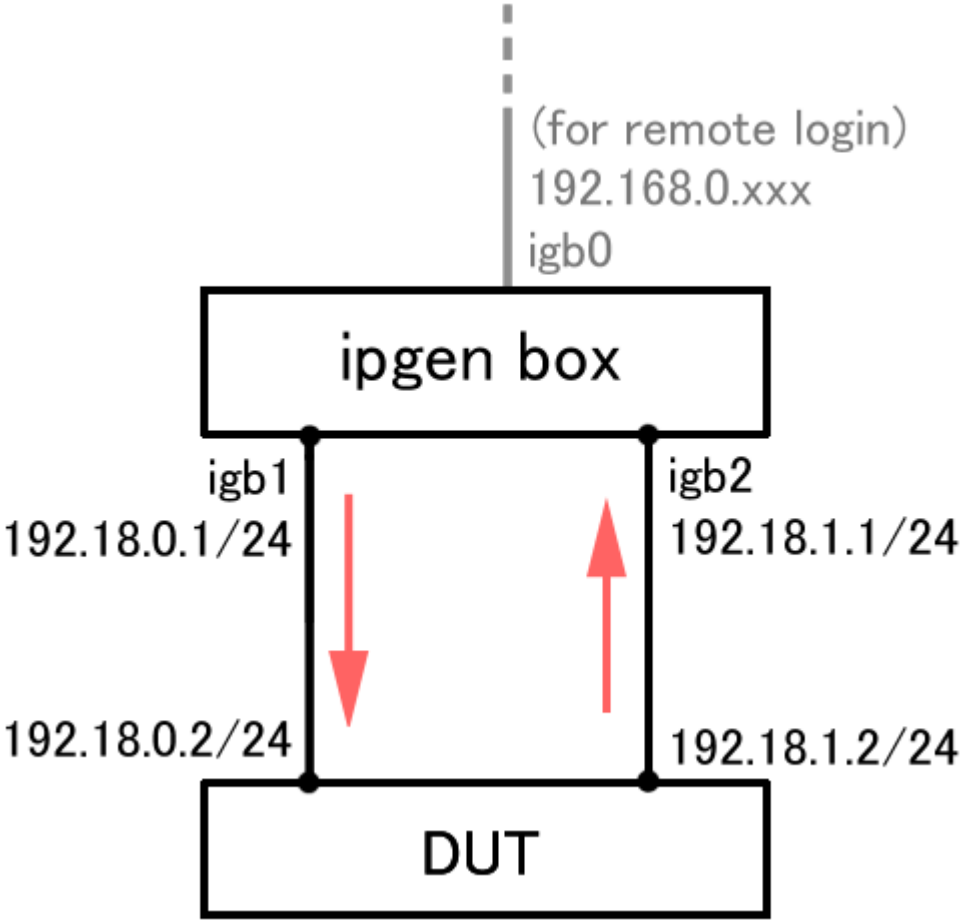
bridge test (L2 forwarding)




```
# ipgen -T igb1,66:77:88:99:aa:bb -R igb2,00:11:22:33:44:55
```

構成例その２

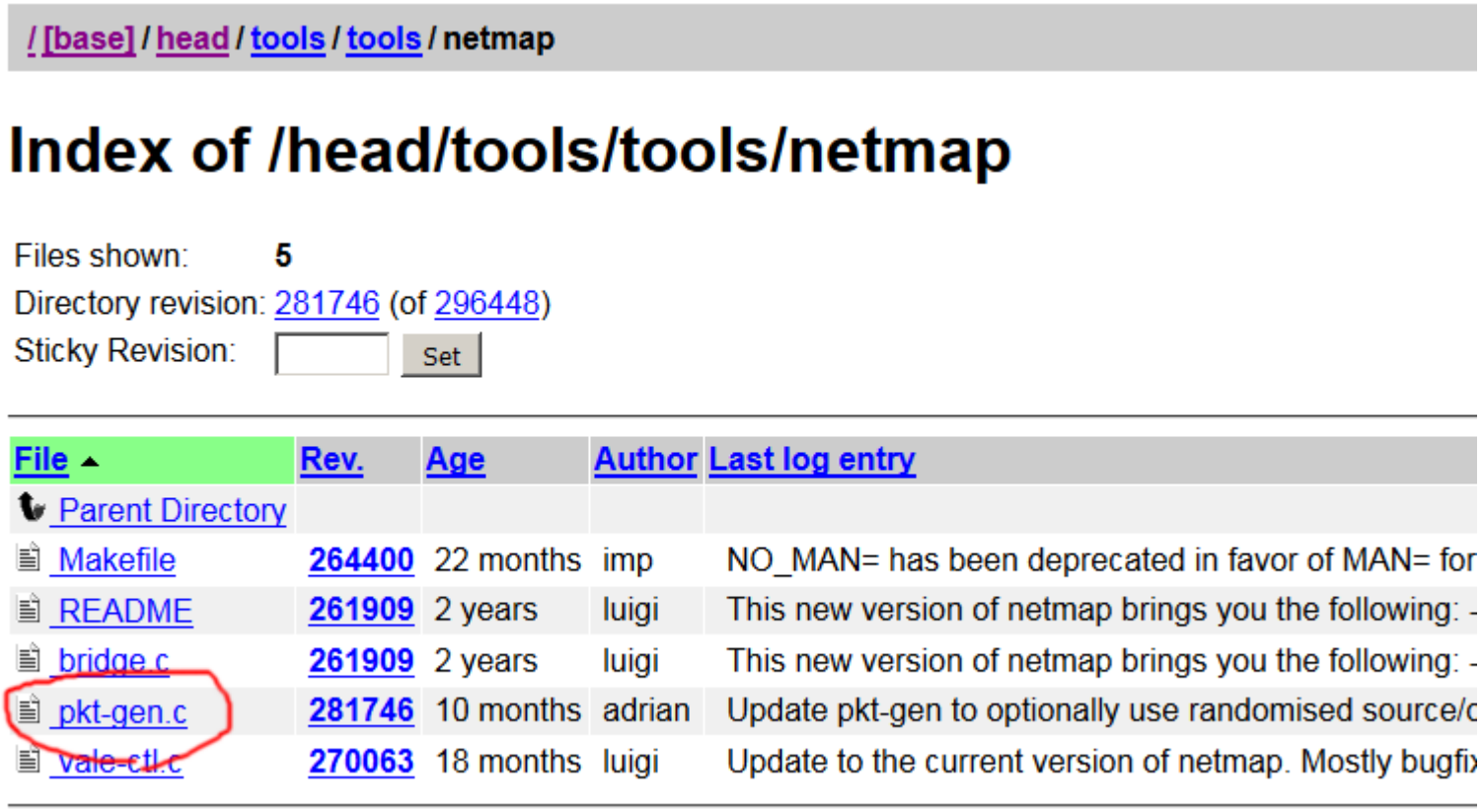
L3 forwarding test




```
# ipgen  -T igb1,198.18.0.2,198.18.0.1/24   -R igb2,192.18.1.2,192.18.1.1/24
```

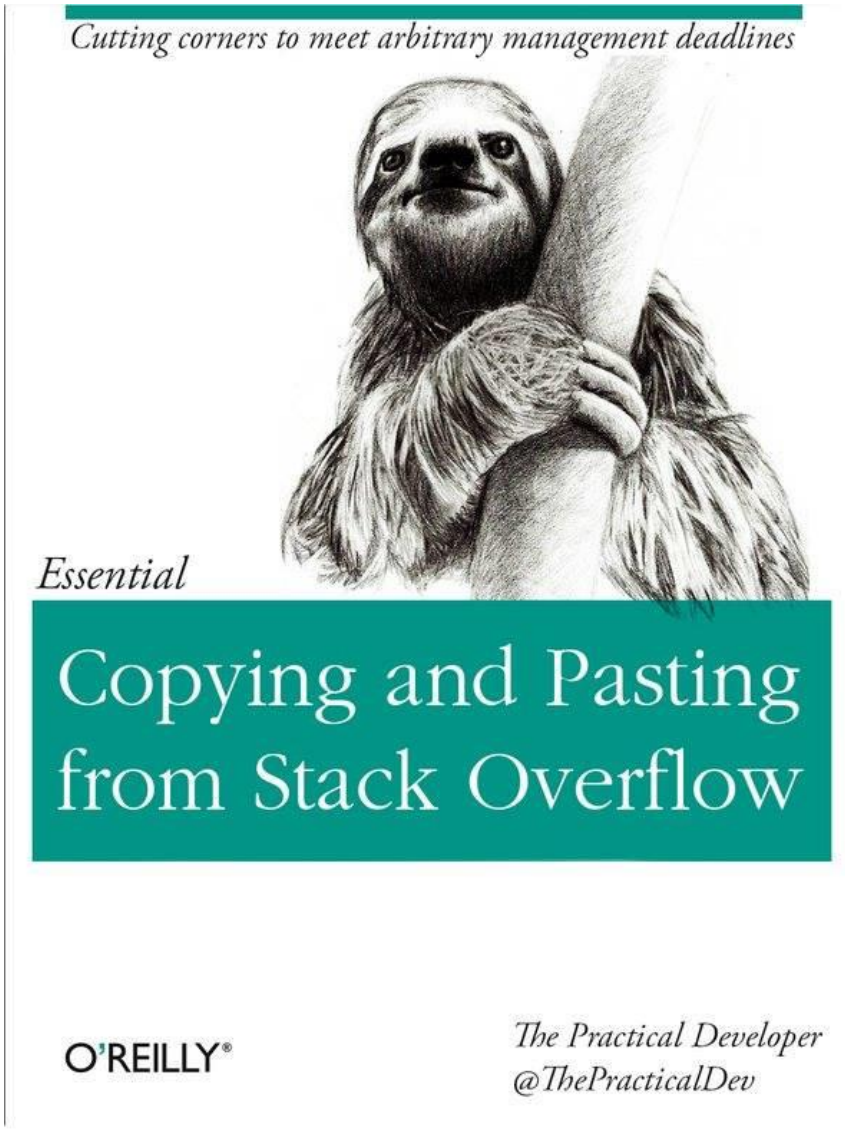
FreeBSDのnetmapを使用

FreeBSDのソースリポジトリには、netmap作者Luigiさんが作った、シンプルなパケットジェネレータプログラムのサンプルがすでにあった。
FreeBSD:tools/tools/netmap



!!

netmap API関連部分等、これを参考にして作成



コピペプログラミングのすゝめ

Features

- インタラクティブなユーザインターフェイス
- Drop/Duplicate/Reorder カウンタ
- 複数フロー(セッション)のサポート
- Inter Packet Gapの調整機能
- RFC2544テスト機能
- IPv6対応

drop/duplicate/reorder counter

それぞれのパケットにはシーケンス番号が埋め込まれています。
ipgenは内部にbitmapの受信フラグを持っています。

```
 #seq  received bitmap
  ---  --------------------------------------------------
    0: 11111111111111111111111111111111111111111111111
   50: 11111111111111111111111111111111111111111111111
  100: 11111111111111111110000000000000000000000000000
  150: 00000000000000000000000000000000000000000000000
  200: 00000000000000000000000000000000000000000000000
  250: 00000000000000000000000000000000000000000000000
       :
       :
```

#105→#106→#108→#109→#110→

100: 1111111011111000000000000000000000000000000000
^

#107 が落ちた?(drop?)

#105→#106→#108→#109→#110→#107→#111…
~~~~

100: 1111111111110000000000000000000000000000000000
^

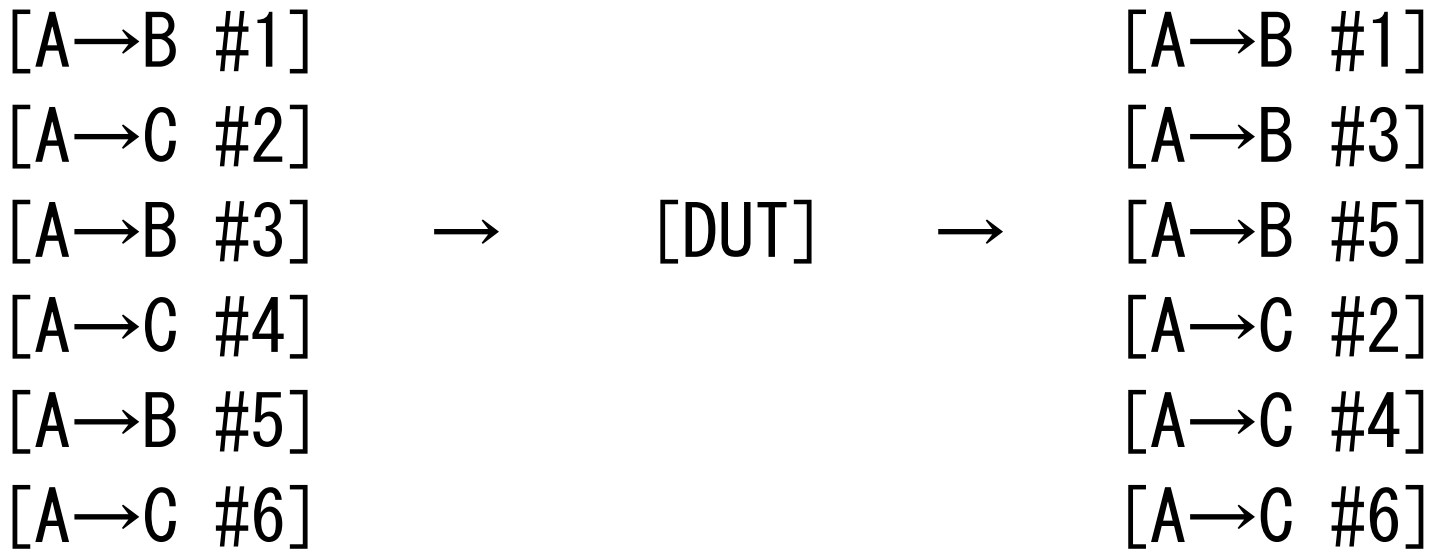
#107 は落ちていなかった。順番が入れ替わっていた(reorder)

#105→#106→#108→#109→#110→#107→#111→#108→
~~~~ ~~~~

100: 1111111111110000000000000000000000000000000000
^

#108 を重複して受信!(dup)

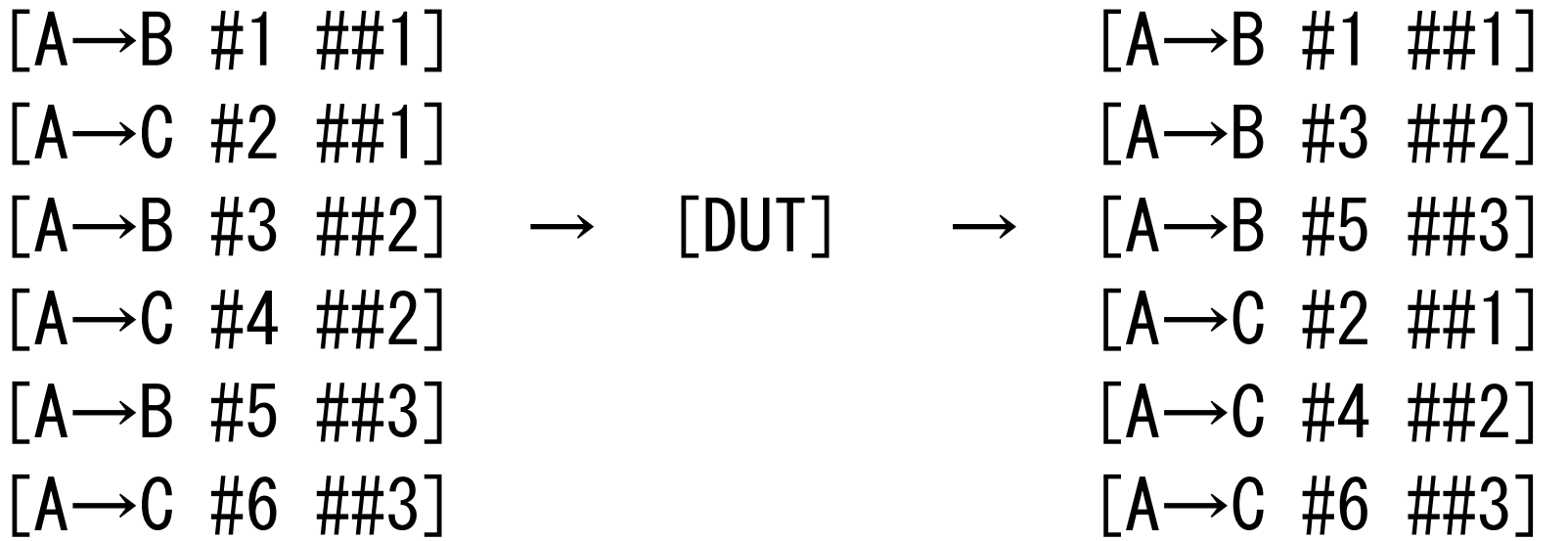
ipgen のフロー別リオーダチェック



#1→#3→#5→#2→#4→#6 . . . リオーダしてる?

全体で見るとリオーダしているが、フロー毎に考えるとリオーダは発生していない（許容範囲）

ipgen のフロー別リオーダチェック



##1→##2→##3→##1→##2→##3

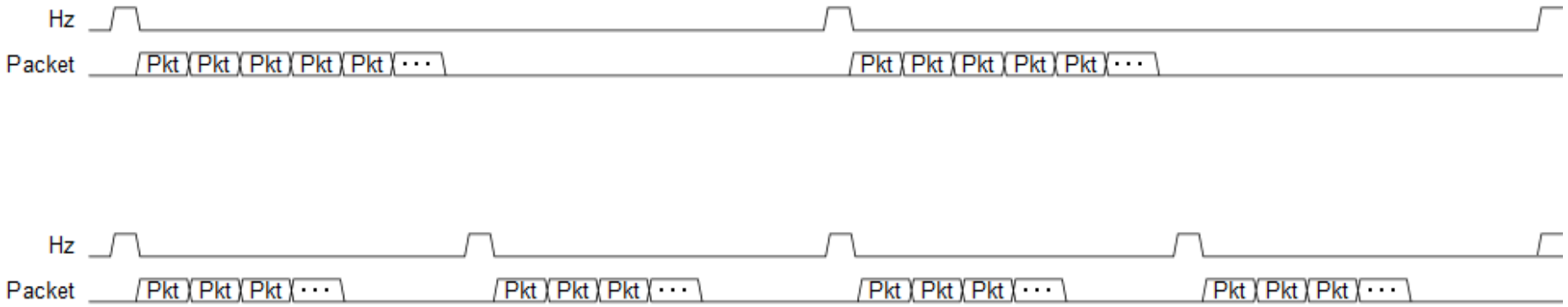
このようにフロー毎にもシーケンス番号を持たせることにより、チェックしている。

Burst送信問題

- 1000ppsでパケットを送信する時のことを考える。
- １秒単位で送信タイミングを制御している場合、最初に1,000パケット分をワイヤーレートで送信してしまい、残り時間はなにもしない状態になってしまう。

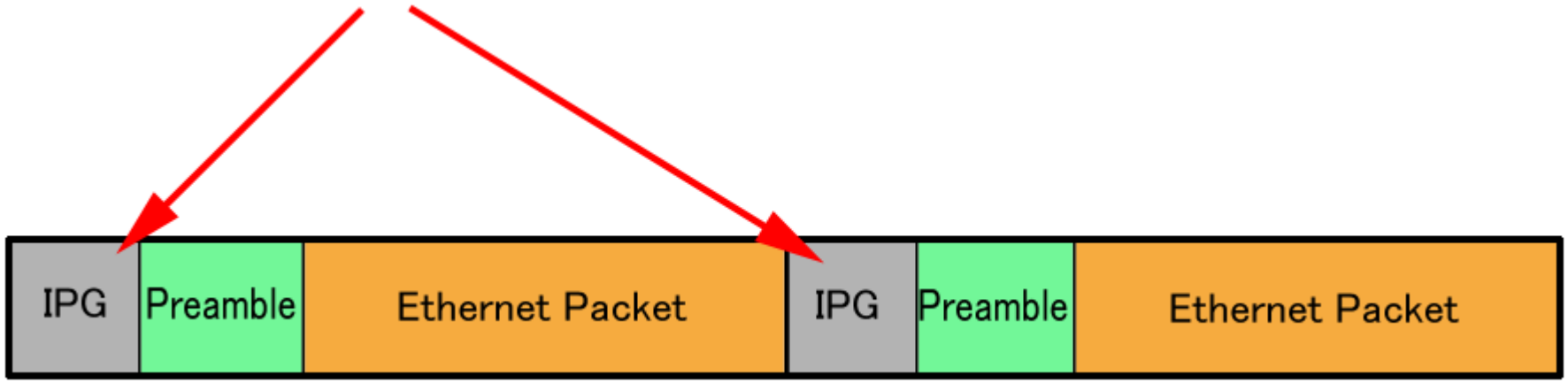
- では、制御間隔をもっと短くすれば問題は解決するのか？
- 解決しない。0.001秒単位で制御したとしても、1000pps程度であれば1パケット/0.001秒綺麗に送信できるが、100,000ppsではやはり同じことが起こる。

- もっともっと制御間隔を短くするしかない？



Inter Packet Gap
(Inter Frame Gap)

- IPGとは?



フレーム間ギャップ

各フレーム同士の間には96ビット時間以上の間、信号の無いフレーム間ギャップを確保する。

このMACフレームをレイヤー1、つまり物理層に渡して伝送路の空きで送出する。(from wikipedia)

IPGの間隔は、大抵のイーサネットコントローラで制御可能。IntelのGbEでも当然制御可能！

6.12.3 Transmit IPG Register—TIPG (0x0410; R/W)

This register controls the Inter Packet Gap (IPG) timer.

Field	Bit(s)	Initial Value	Description
IPGT	9:0	0x08	IPG Back to Back Specifies the IPG length for back to back transmissions in both full and half duplex. Measured in increments of the MAC clock: 8 ns MAC clock when operating @ 1 Gb/s. 80 ns MAC clock when operating @ 100 Mb/s. 800 ns MAC clock when operating @ 10 Mb/s. IPGT specifies the IPG length for back-to-back transmissions in both full duplex and half duplex. Note that an offset of 4 byte times is added to the programmed value to determine the total IPG. As a result, a value of 8 is recommended to achieve a 12 byte time IPG.
IPGR1	19:10	0x04	IPG Part 1 Specifies the portion of the IPG in which the transmitter defers to receive events. IPGR1 should be set to 2/3 of the total effective IPG (8). Measured in increments of the MAC clock: 8 ns MAC clock when operating @ 1 Gb/s. 80 ns MAC clock when operating @ 100 Mb/s 800 ns MAC clock when operating @ 10 Mb/s.
IPGR	29:20	0x06	IPG After Deferral Specifies the total IPG time for non back-to-back transmissions (transmission following deferral) in half duplex. Measured in increments of the MAC clock: 8 ns MAC clock when operating @ 1 Gb/s. 80 ns MAC clock when operating @ 100 Mb/s 800 ns MAC clock when operating @ 10 Mb/s. An offset of 5-byte times must be added to the programmed value to determine the total IPG after a defer event. A value of 7 is recommended to achieve a 12-byte effective IPG. Note that the IPGR must never be set to a value greater than IPGT. If IPGR is set to a value equal to or larger that IPGT, it overrides the IPGT IPG setting in half duplex resulting in inter-packet gaps that are larger then intended by IPGT. In this case, full duplex is unaffected and always relies on IPGT.
Reserved	31:30	00b	Reserved Write 0, ignore on read.

ただし、ユーザランドからIPGを制御するAPIは当然存在しない。（本来デバイスドライバ内で行うもの）
というわけでパッチ書きました。

```
Index: if_igb.c
===================================================================
--- if_igb.c    (revision 292398)
+++ if_igb.c    (working copy)
@@ -547,6 +548,12 @@
                }
        }

+       SYSCTL_ADD_PROC(device_get_sysctl_ctx(dev),
+           SYSCTL_CHILDREN(device_get_sysctl_tree(dev)),
+           OID_AUTO, "tipg", CTLTYPE_INT|CTLFLAG_RW,
+           adapter, 0, igb_sysctl_tipg, "I",
+           "Transmit IPG register");
+
        /*
        ** Start from a known state, this is
        ** important in reading the nvm and
@@ -6377,3 +6384,23 @@
        IGB_CORE_UNLOCK(adapter);
        return (0);
 }
+
+static int
+igb_sysctl_tipg(SYSCTL_HANDLER_ARGS)
+{
+       struct adapter *adapter = (struct adapter *)arg1;
+       int error, value;
+       u32 reg;
+
+       value = E1000_READ_REG(&adapter->hw, E1000_TIPG) & E1000_TIPG_IPGT_MASK;
+       error = sysctl_handle_int(oidp, &value, 0, req);
+       if (error || req->newptr == NULL)
+               return (error);
+
+       reg = E1000_READ_REG(&adapter->hw, E1000_TIPG);
+       reg &= ~E1000_TIPG_IPGT_MASK;
+       reg |= value & E1000_TIPG_IPGT_MASK;
+       E1000_WRITE_REG(&adapter->hw, E1000_TIPG, reg);
+
+       return (0);
+}
```

このパッチにより、sysctl(8)でIPGを設定できるようになります。

```
# sysctl dev.igb | grep tipg
dev.igb.5.tipg: 8
dev.igb.4.tipg: 8
dev.igb.3.tipg: 8
dev.igb.2.tipg: 8
dev.igb.1.tipg: 8
dev.igb.0.tipg: 8
```

ipgenは、igb(4)を使う場合は、自動的にIPGの設定を行います（このパッチがkernelにあたっているかどうか=sysctlできるかどうかは自動検出）

IPGを制御することにより、安定したトラフィックをかけることが可能に

no IPG adjustment (burst traffic mode)

with IPG adjustment (steady traffic mode)

RFC2544 test

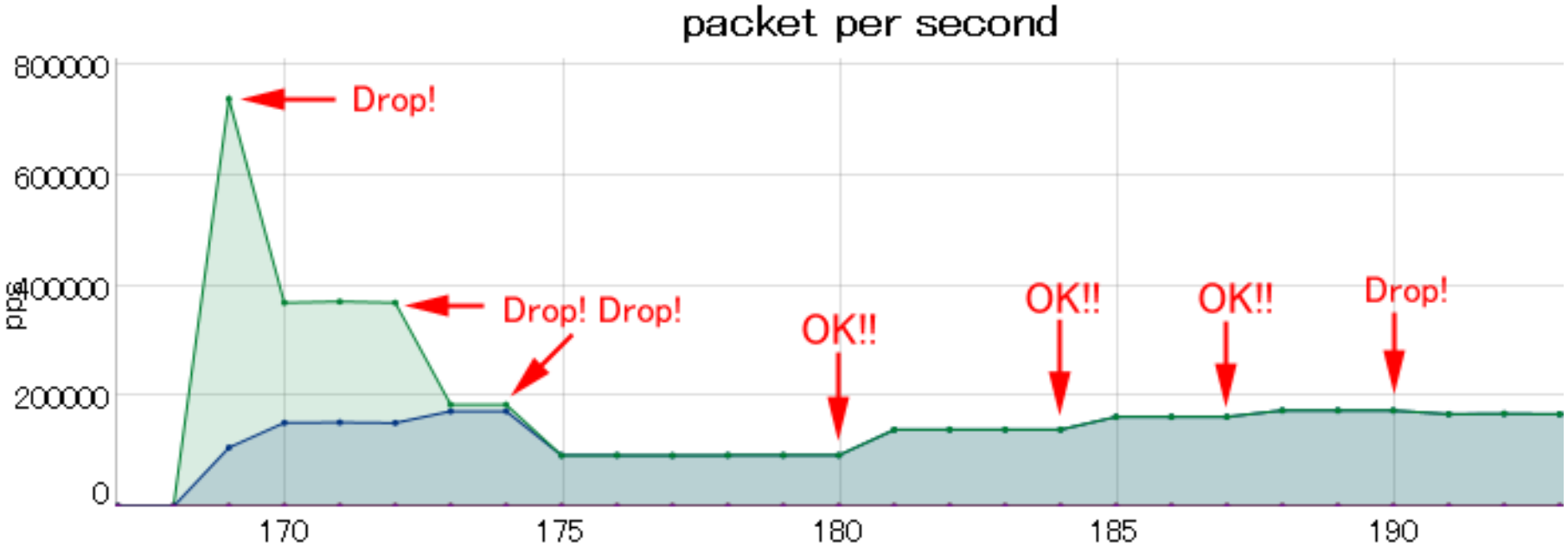
Q. RFC2544って何？

A. RFC2544読みましょう :)

> *テストの目的はＤＵＴがフレーム損失なしで処理することができる最小バースト間隔を決定することです。*
> *(RFC2544日本語訳: Ishida So)*

ipgen は RFC2544 テストモードをサポートしています

すべてのppsにおいてフレーム損失率を調べると膨大な時間がかかるため、バイナリ探索を行います。



バイナリ探索の例。パケットを落とすとppsを下げ、パケットを一度も落とさないで規定時間が過ぎたらppsを上げます。ppsの増減幅を狭めながらこれを繰り返します。

ipgen の RFC2544 テストモードの結果

ipgen ***--rfc2544*** -T igb2,00:60:e0:5c:4e:e7 ＼
-R igb4,00:60:e0:5c:4e:e5

```
# ipgen -T igb2,00:60:e0:5c:4e:e7 -R igb4,00:60:e0:5c:4e:e5 --rfc2544 --rfc2544-trial-duration 0
ipgen v1.21
igb(4) TIPG feature supported
HZ=1000
igb2 -> igb4, IP pktsize 46, 1488095 pps, 999 Mbps (999999840 bps)
igb2: waiting link up.......OK
igb4: waiting link up.OK
igb4(00:60:e0:5c:4e:e7) -> 00:60:e0:5c:4e:e5
igb2(00:60:e0:5c:4e:e5) -> 00:60:e0:5c:4e:e7
Exiting...


framesize|0M  100M 200M 300M 400M 500M 600M 700M 800M 900M 1Gbps
---------+----+----+----+----+----+----+----+----+----+----+
      64 |#######                                              105.71Mbps,  157310/1488095pps
     128 |######                                                93.13Mbps,   78654/ 844594pps
     256 |################                                    276.96Mbps,  125433/ 452898pps
     512 |####################################                650.38Mbps,  152814/ 234962pps
    1024 |#####################################################  999.99Mbps,  119731/ 119731pps
    1280 |#####################################################  999.99Mbps,   96153/  96153pps
    1408 |###################################################### 1000.00Mbps,   87535/  87535pps
    1518 |###################################################### 1000.00Mbps,   81274/  81274pps

framesize|0   |100k|200k|300k|400k|500k|600k|700k|800k|900k|1.0m|1.1m|1.2m|1.3m|1.4m|1.5m pps
---------+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
      64 |########                                                                  157310/1488095pps,  10.57%
     128 |####                                                                       78654/ 844594pps,   9.31%
     256 |#######                                                                  125433/ 452898pps,  27.70%
     512 |########                                                                 152814/ 234962pps,  65.04%
    1024 |######                                                                   119731/ 119731pps, 100.00%
    1280 |#####                                                                    96153/  96153pps, 100.00%
    1408 |#####                                                                    87535/  87535pps, 100.00%
    1518 |#####                                                                    81274/  81274pps, 100.00%
```

まとめ

- 手軽に使えるパケットジェネレータ、計測ツールを作りました。
- RFC2544テストモードもサポートしました
- netmapはすばらしい！
  誰かNetBSDにもnetmapを移植してください