Open Source Traffic Analyzer

Daniel Turull

June 2010

Outline

Outline

Introduction

- Traffic analysis is crucial for developlment of network systems
- New features in modern systems
  - SMP
  - Multiples queues in network cards
- Pktgen: packet **generator** at high rates inside Linux Kernel

Goals

- Investigate current solutions for traffic analyses
- Understand how Pktgen works
- Design and implement a network analyser inside the Linux Kernel, taking advantage of the new features in the modern systems.
- Integrate the results in the current Pktgen module
- Evaluate and calibrate the behaviour of the module implemented

Outline

Network analysis

Diferent aproaches

- **Dedicated hardware:** *Ixia IxNetwork, Spirent SmartBits*
- **Software based:**
  - Libraries: *Pcap, Ncap, DashCap*
  - User space: *Iperf, Netperf, NetPIPE, LMBench, Ttcp, nuttcp, Mausezahn, D-ITG, Harpoon, RUDE, BRUTE*
  - Kernel space: *Pktgen, Kute*
- **Network processors:** *Caldera Technologies - LANforge-FIRE, TNT Pktgen, BRUNO*

Network Analysis. Metrics

IETF

- Definitions in RFC 1242, RFC 2285
- Methodologies: RFC 2544, RFC 2889
  - Throughput
  - Latency
  - Frame loss rate
  - Back-to-back frame

Others:

- Inter-arrival times
- Jitter RFC 4689

Linux Network (I)

Basic elements in network subsystem:

- Socket buffer (skb)
- Net device

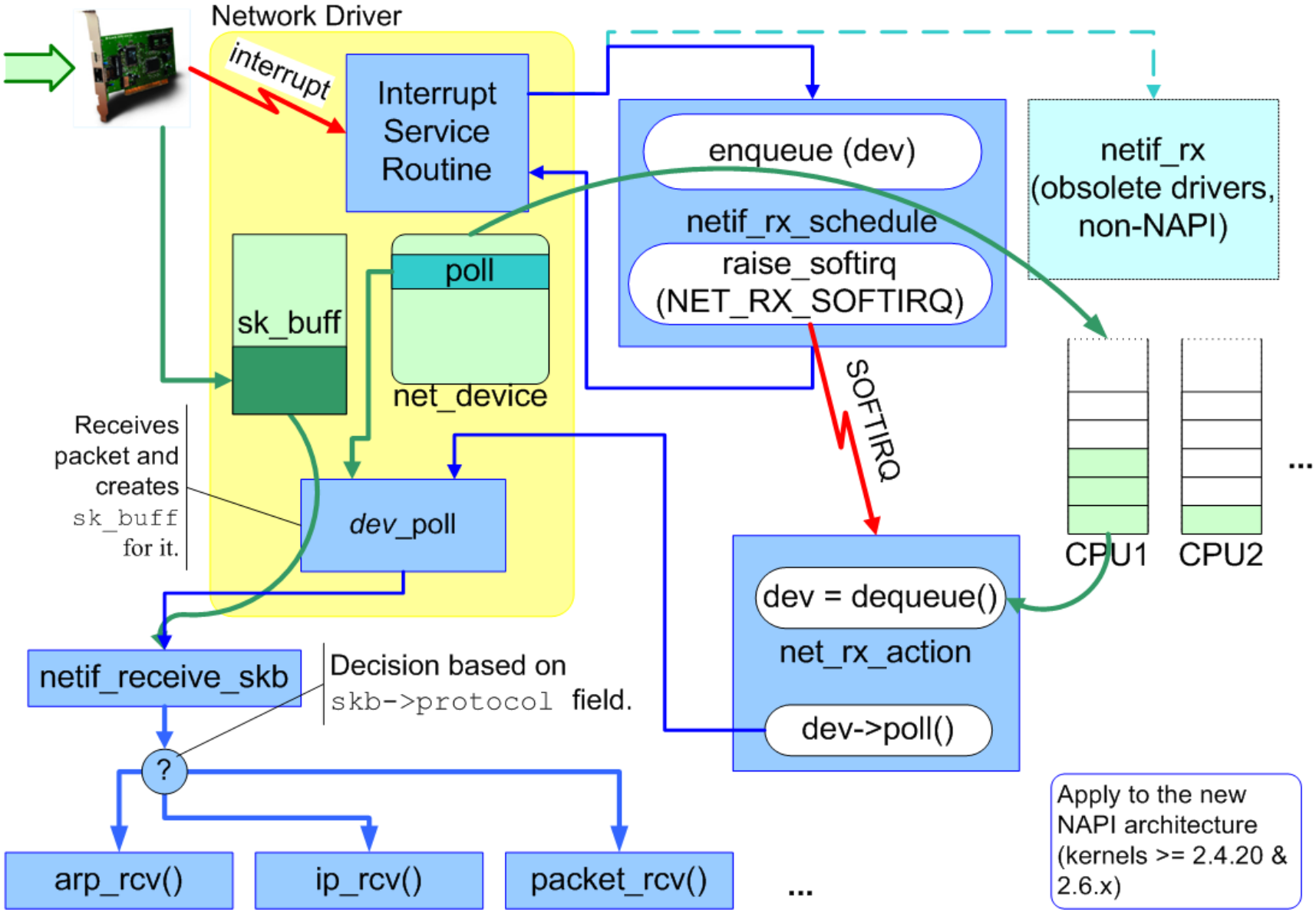
Packet reception

- Interrupt driven
- Polling

NAPI. Advantages of both of them

- Low load (interrupt)
- High load (polling)
- Moreover: Direct access to device memory and no queues

Linux Network (II)



Source: http://www.invisiblethings.org/papers/ITUnderground2004 Linux kernel backdoors.ppt

Pktgen Features

- MPLS, VLAN, IPSEC
- IPv4 and IPv6
- Customized packets with multiples addresses
- Clone packets to improve performance
- Multi queue
- Proc file systems as user interface
- Control the delay between packets
- UDP to send its headers

Outline

Requirements

- Num of packets / Bytes received from the transmitter
- Num of packets / Bytes lost in the network or element under test
- Percentage of packets / Bytes lost
- Throughput received from the link. Also the output throughput of pktgen will be adjustable by the user
- Inter-arrival time
- Jitter
- Latency between transmitter and receiver

Architecture

- Each CPU has its counters and variables for the different flows or different NICS
- Load balancing (configured via SMP affinity)

Optimal architecture:

- Counters, Throughput, jitter, inter-arrival: in different machines
- Latency: same machine

Receiver metrics (I)

- Packets / Byte Received
  - Counters
- Packet / Byte loss
  - Offline. Subtract (Data extracted from initialization)
- Throughput
  - Time first packet arrive, Time last packet arrive

$$\text{Throughput} = \frac{\text{packets received}}{\text{end time} - \text{start time}} \text{ (pps)}$$

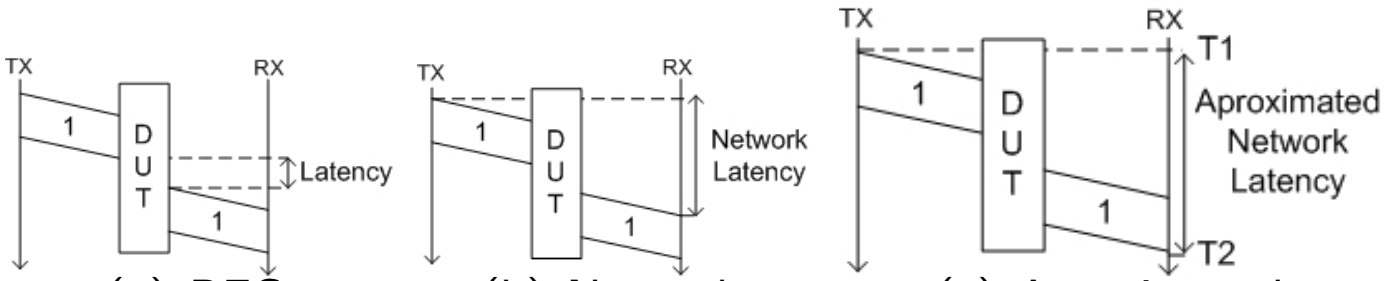
$$\text{Throughput} = \frac{\text{bytes received} \times 8}{\text{end time} - \text{start time}} \text{ (bps)}$$

Receiver metrics (II)

- Inter-arrival time: avg, var, max, min

$$\text{Inter arrival time} = T_{current} - T_{last\ arrival}$$

- Jitter: avg, var, max, min
  - Necessary constant rate
  - Method used: Inter-arrival
  - Subtract of two consecutive inter-arrival times
- Latency: avg, var, max, min



(a) RFC (b) Network (c) Aproximated

Operation

Layer 3

- Advantages
  - No device dependent and more generic
  - Transparent to other communications
  - The reception is made by the kernel
- Drawbacks
  - Less performance (Theoretically)

Auto-configuration

- New pktgen header
- Configure packet at the beginning (pkts to send, bytes to send)
- Reset counters

Aplication intereface

Control:

- Transmission:
  - rate [rate in Mbps]
  - ratep [rate in pps]
  - config [0 or 1]
- Reception:

  new proc file: /proc/net/pktgen/pgrx
  - rx [device]
  - rx_reset
  - rx_disable
  - display [human or script]
  - statistics [counter, basic or time]

Display: per CPU and Global

Outline

Some details of implementation

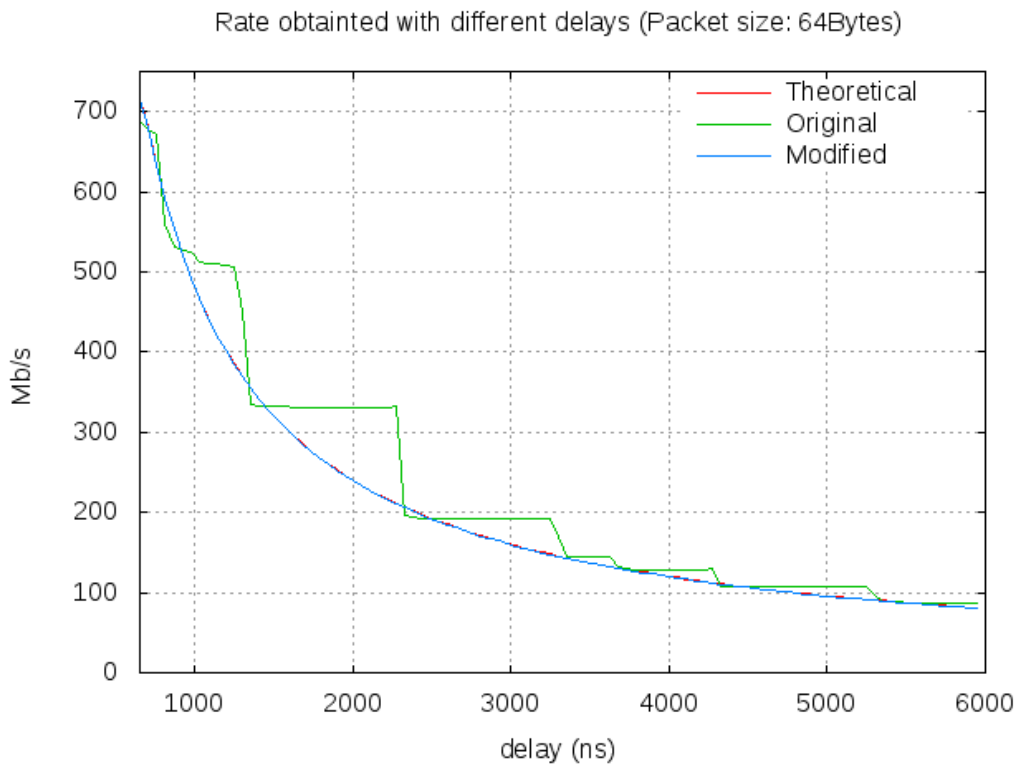
- Receiving packets *dev_add_pack()*

```
static struct packet_type pktgen_packet_type __read_mostly = {
        .type = __constant_htons(ETH_P_IP),
        .func = pktgen_rcv_basic,
        .dev  = NULL,
};
```

- Multiple CPUs
- Autoconfiguration: possible to receive multiples packets
- Improvment of transmission rate
- Hook

Transmission rate

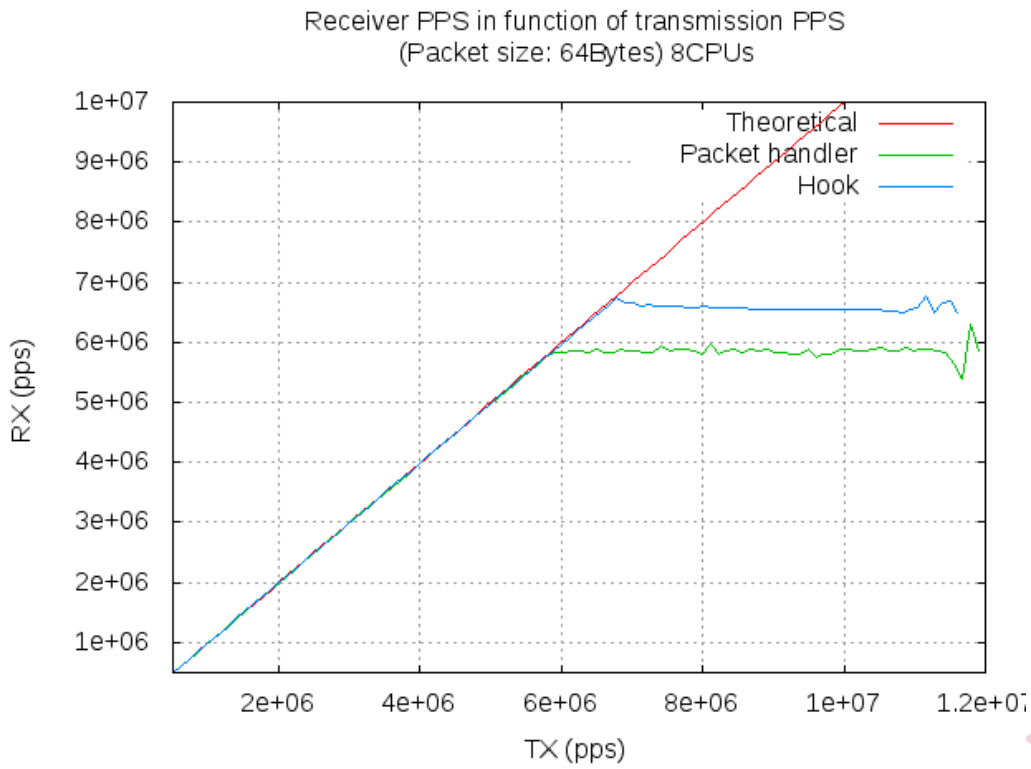
- Changed resolution from microseconds to nanoseconds
- New commands for a direct control
- Accepted in the Linux Kernel (11 June 2010)



Hook

- Avoid IP process
- Modification of the network core (dev.c)

1. Check if pktgen packet
2. Process packet with pktgen (if pktgen, packet drop)
3. Otherwise, packet continues its path to other protocols



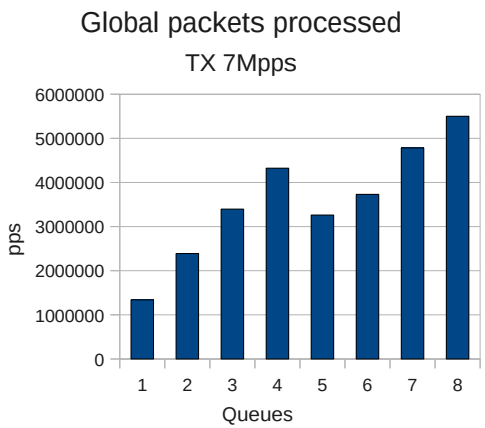
Outline

Scenario

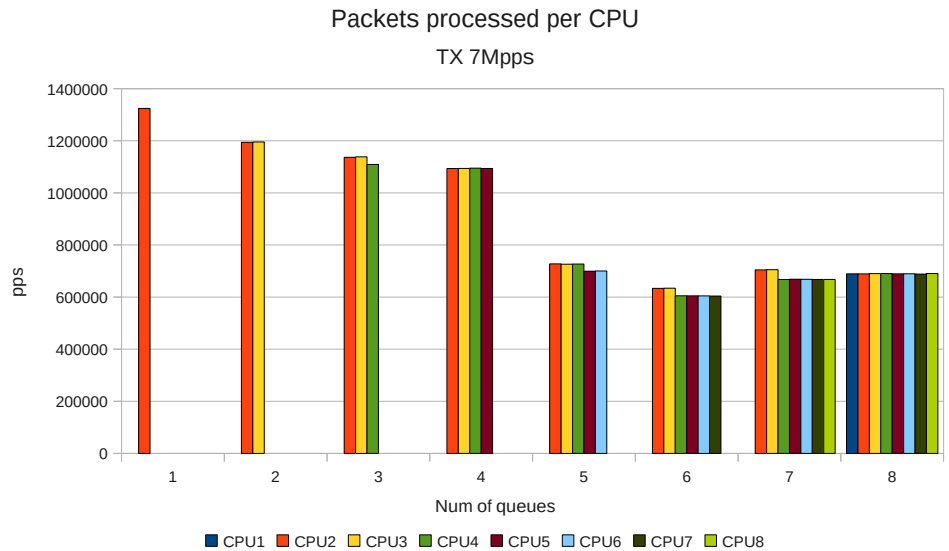
- Intel(R) Xeon(R) CPU E5520 at 2.27GHz (Quad-Core Hyperthreading)
- 3 GB of RAM (DDR3 1333MHz)
- 4 Intel 82576 Gigabit Network (2 x Dual Copper Port)
- 2 Intel 82599EB 10-Gigabit Network (1 x Dual Fibre Port)
- Bifrost Distribution.
  Kernel: net-next-2.6 (2.6.34-rc2) (April 2010)

Throughput

RX: depens on num CPUs and type



(d) Receiver with different number of RSS queues



(e) Packets processed per CPU

Inter-arrival Time and Jitter

- Current Results
- Timer Frequency
- Spin Time
- Old version

Inter-arrival Time and Jitter

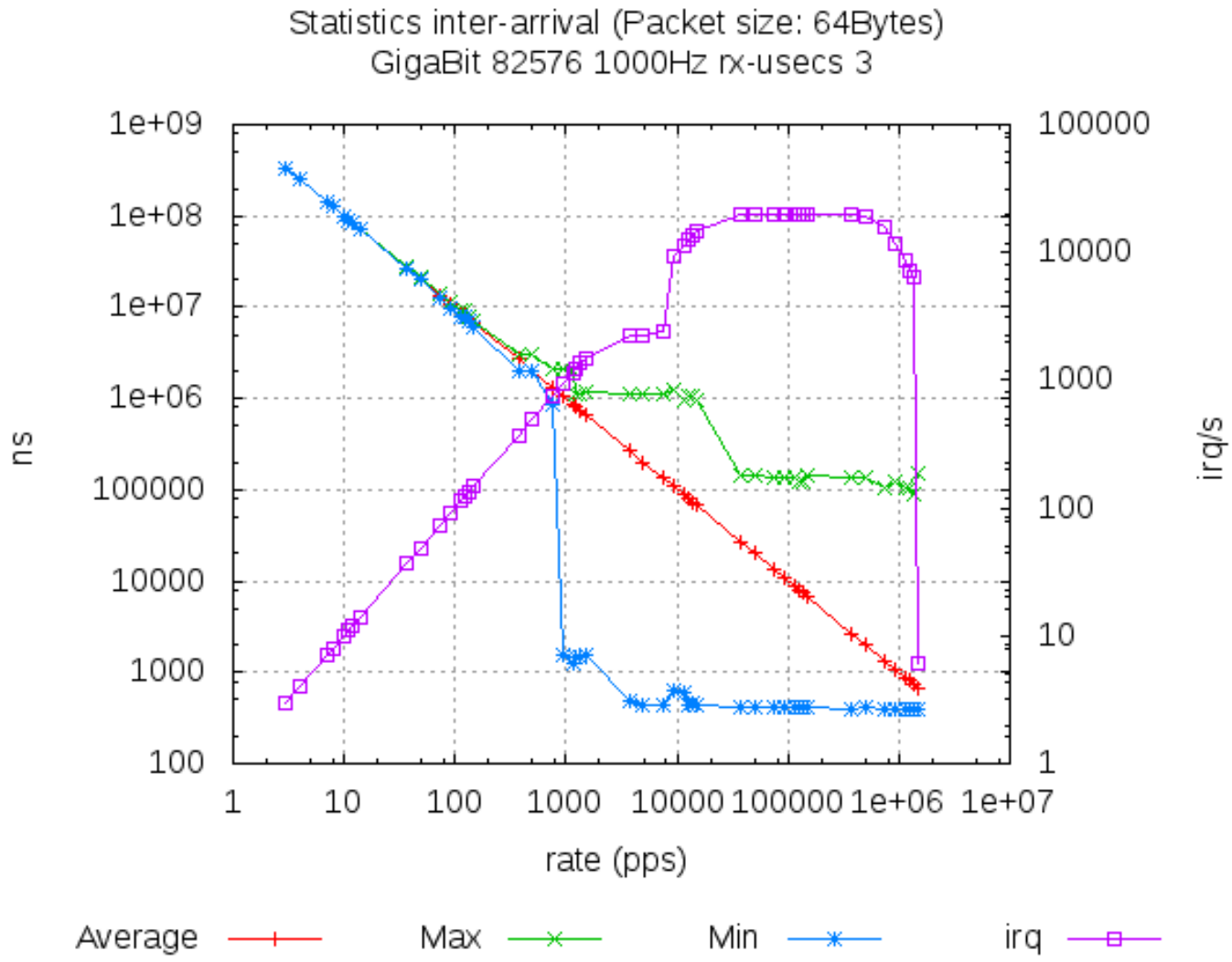


Figure: Inter-arrival. Current

Inter-arrival Time and Jitter

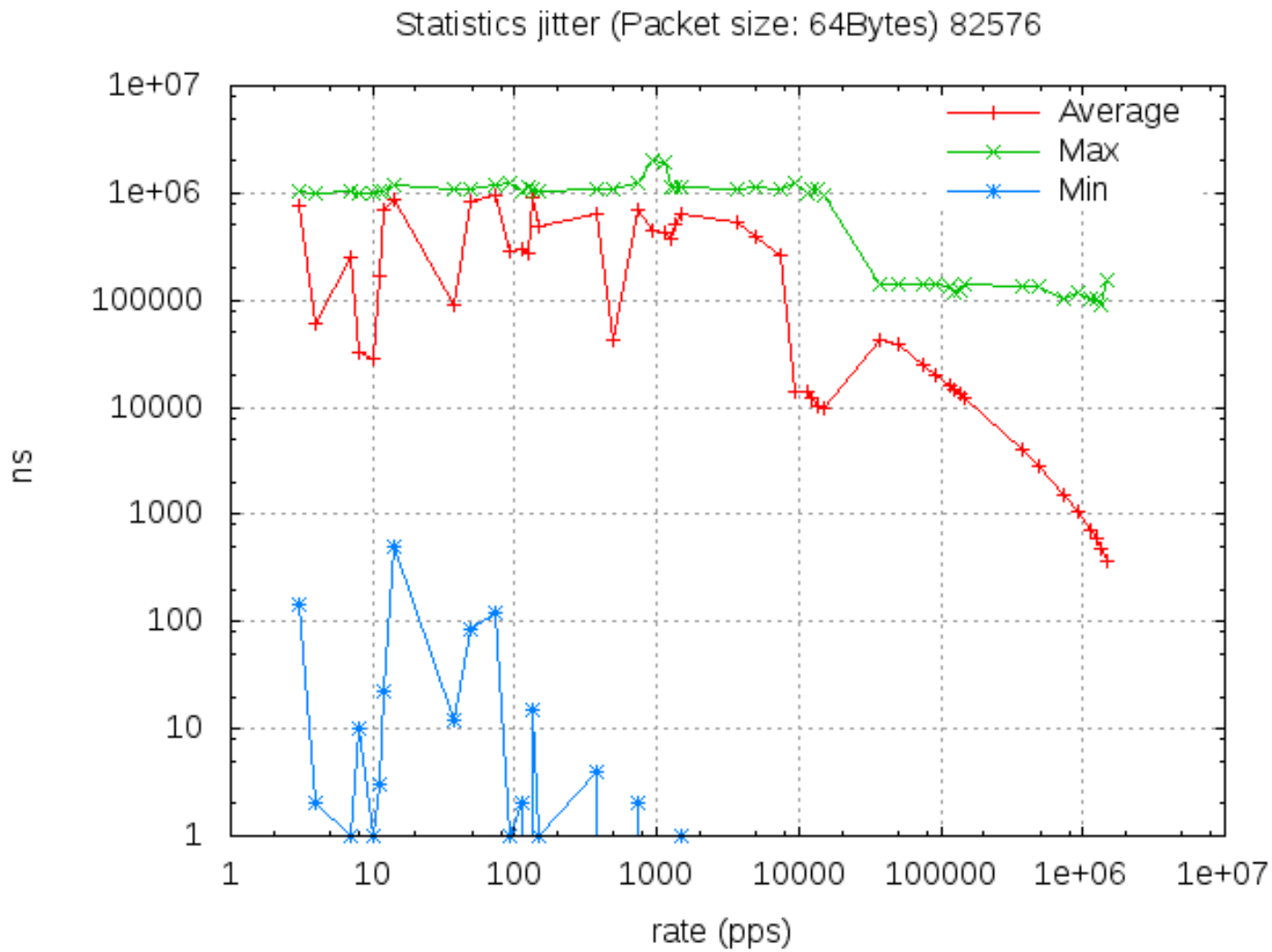


Figure: Jitter. Current

Inter-arrival Time and Jitter

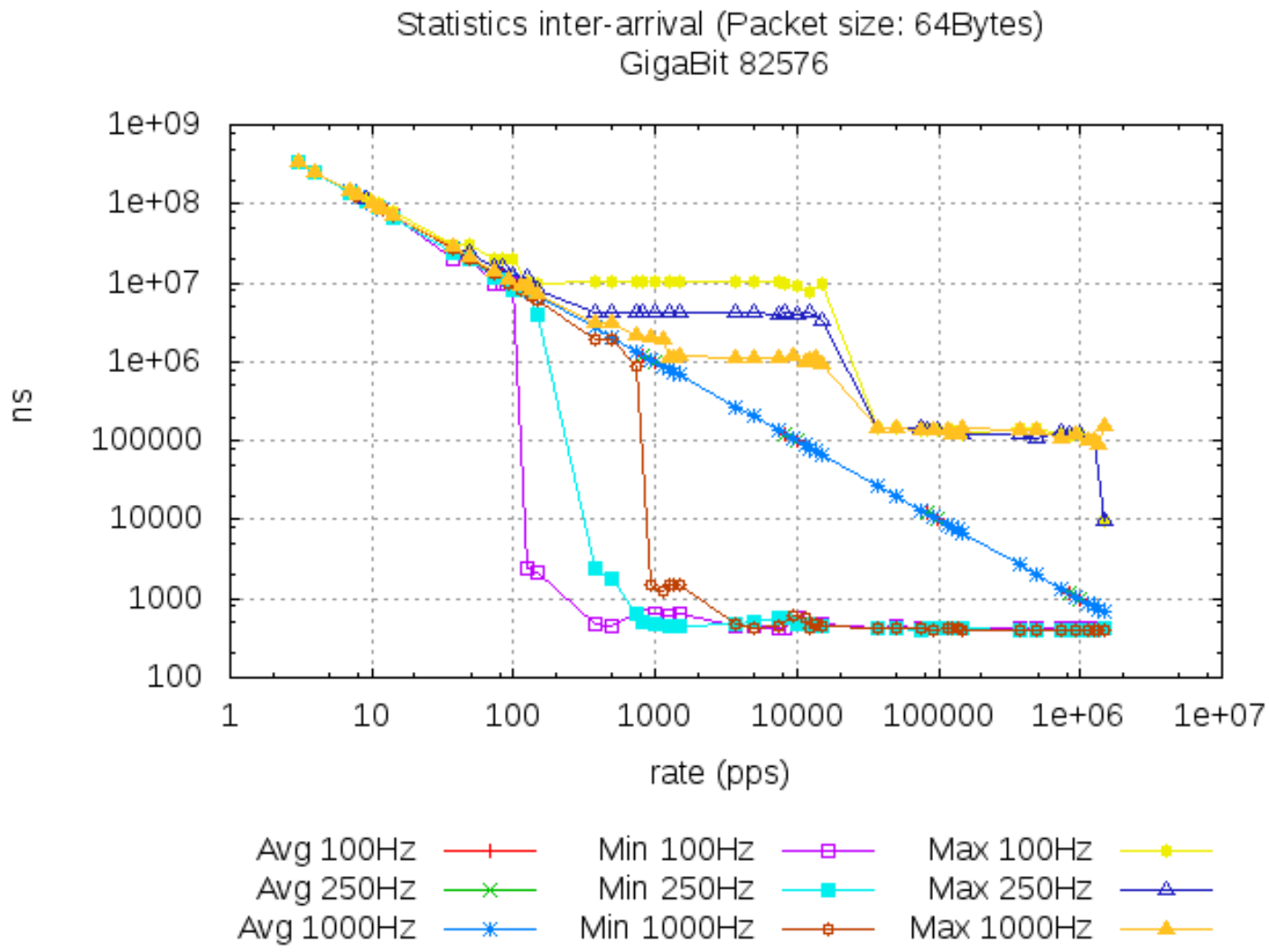


Figure: Inter-arrival. Frequency

Inter-arrival Time and Jitter

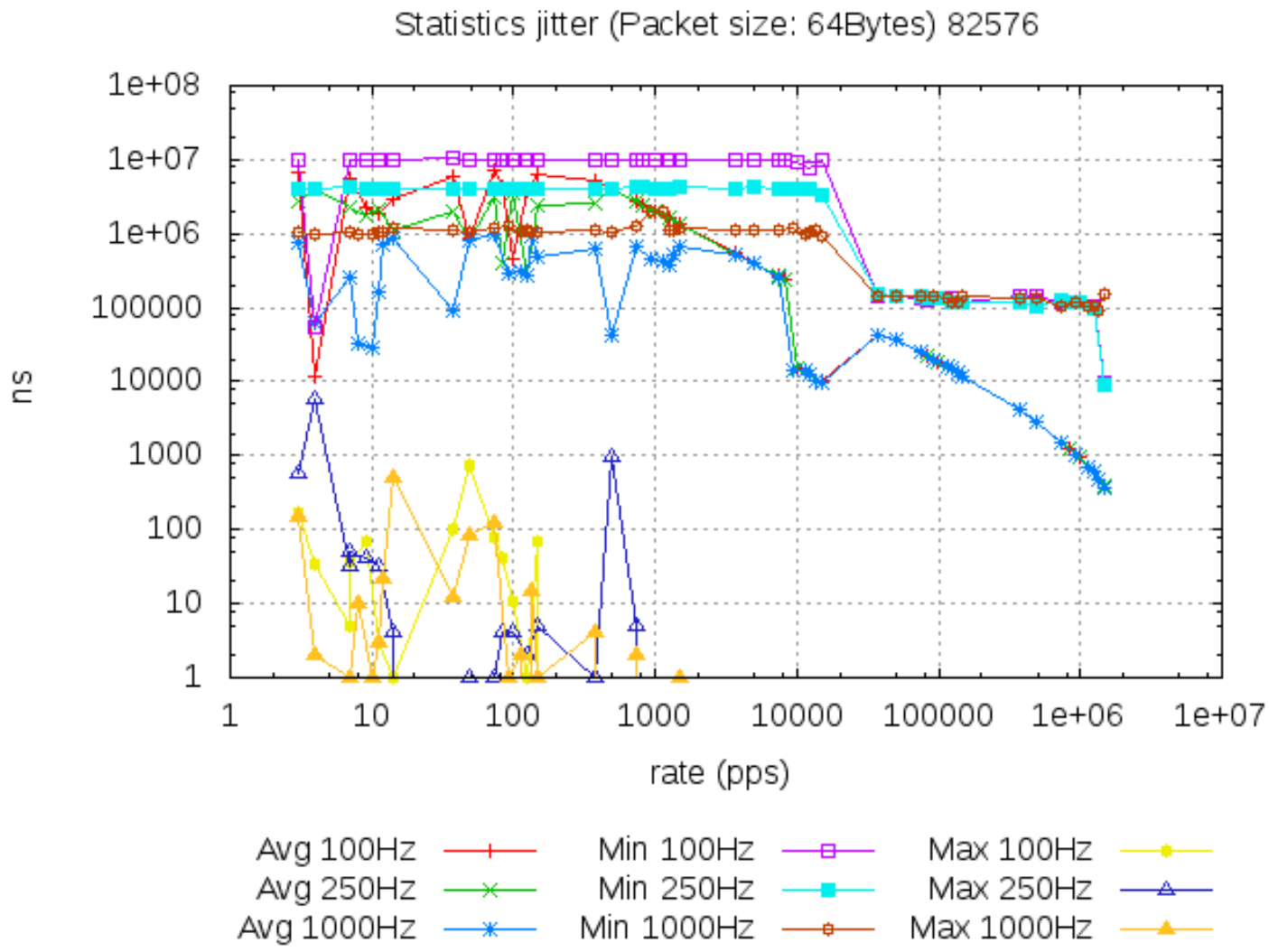


Figure: Jitter. Frequency

Inter-arrival Time and Jitter

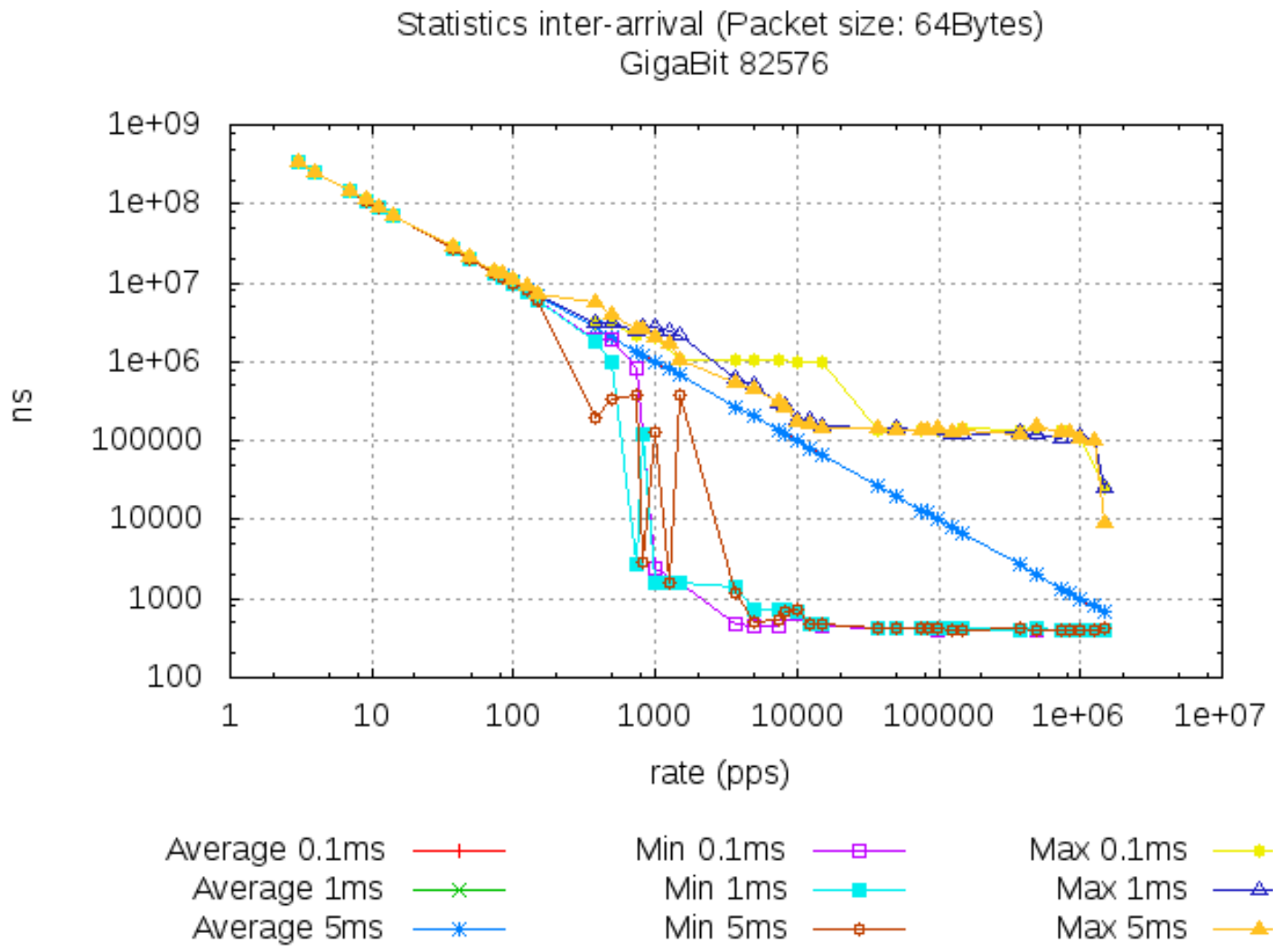


Figure: Inter-arrival. Spin

Inter-arrival Time and Jitter

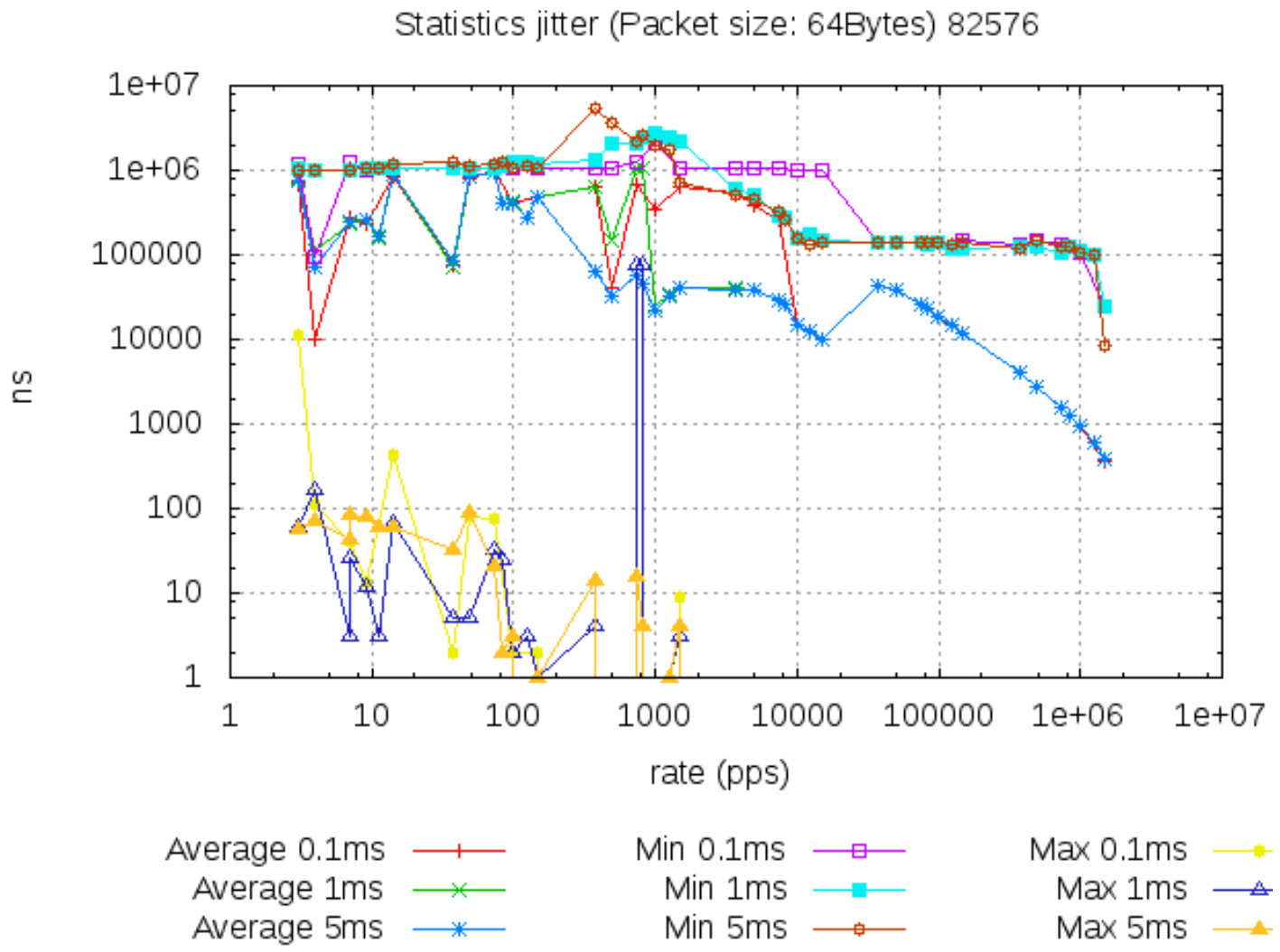


Figure: Jitter. Spin

Inter-arrival Time and Jitter

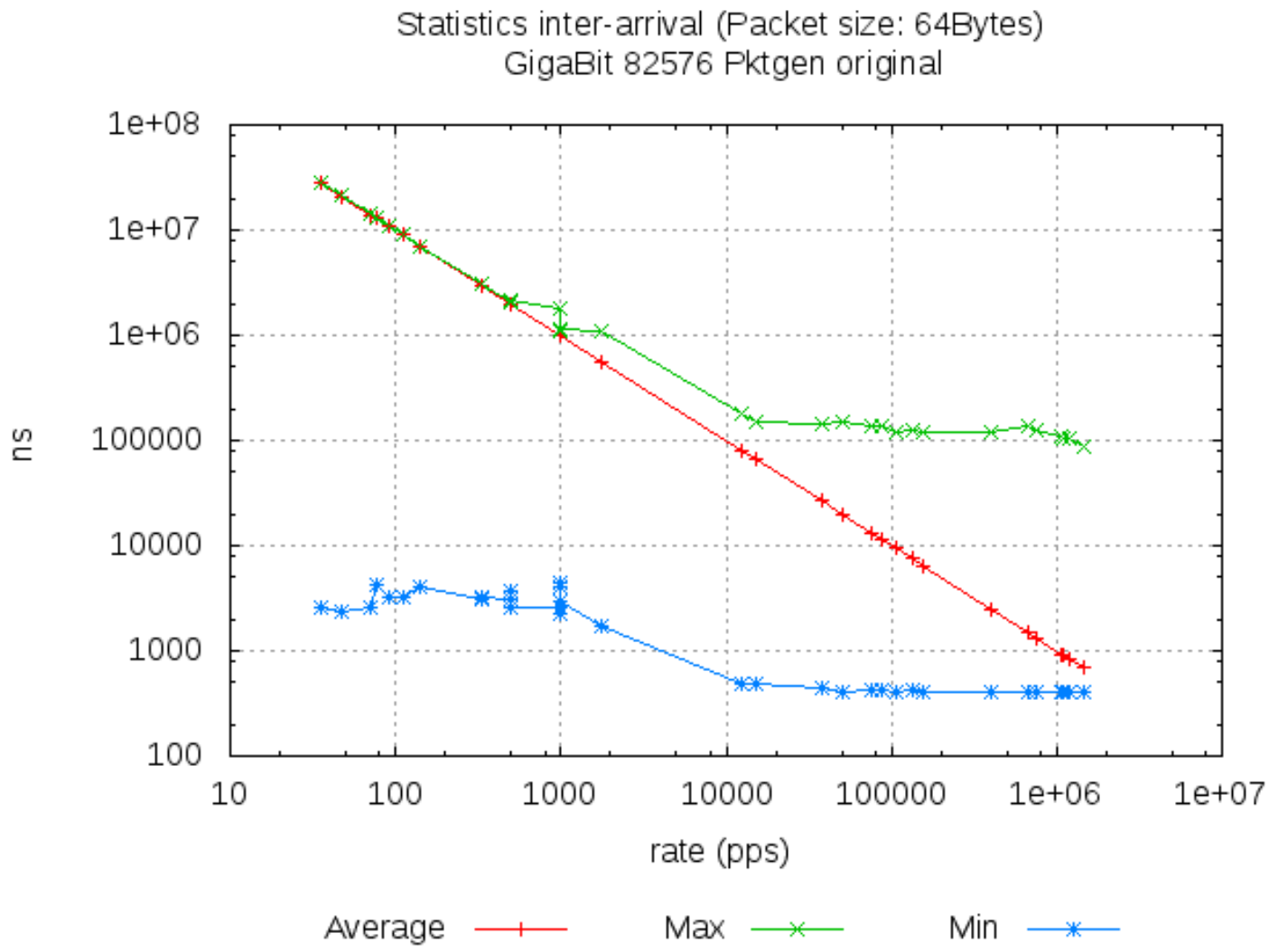


Figure: Inter-arrival. Original

Inter-arrival Time and Jitter

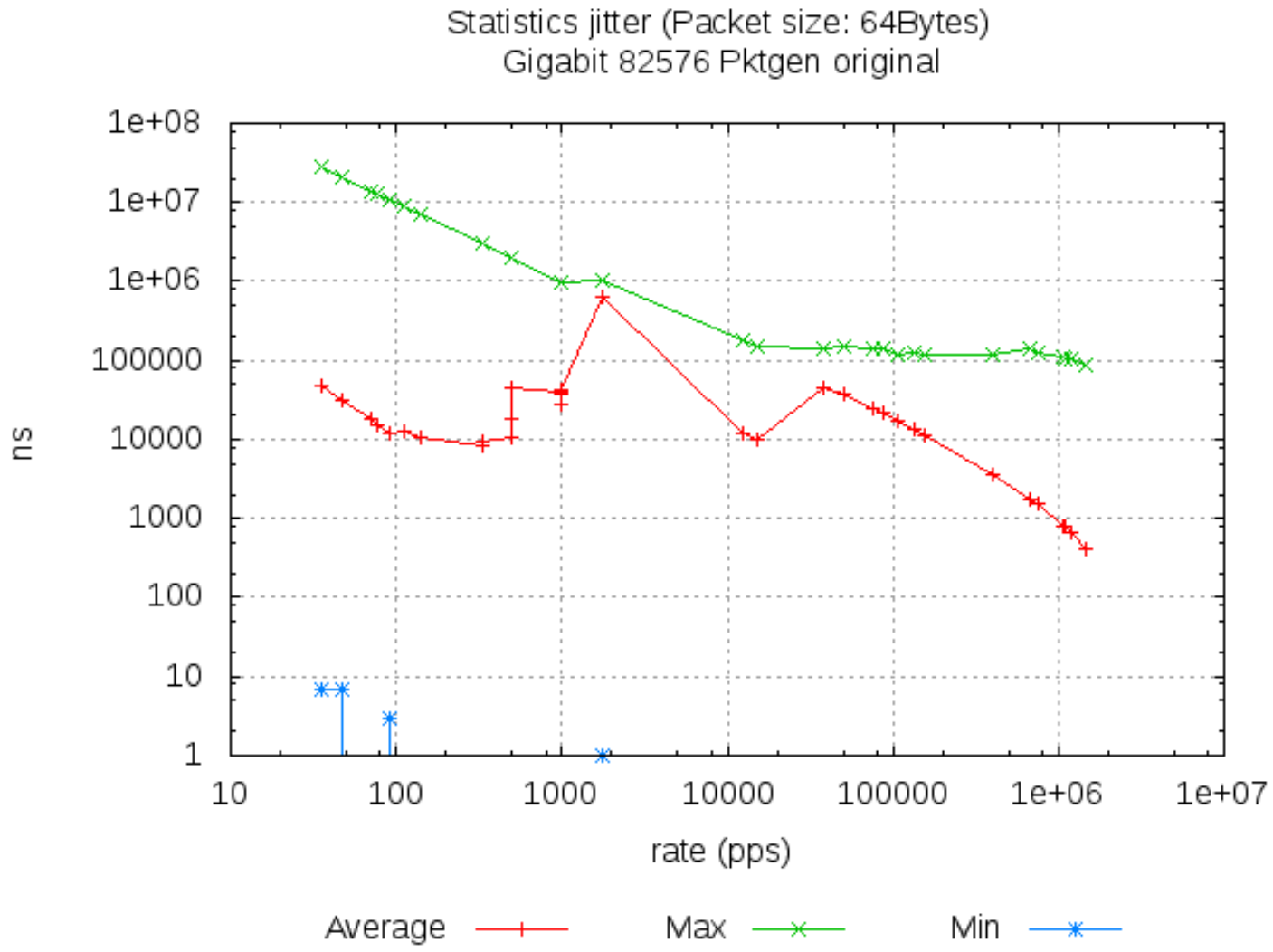
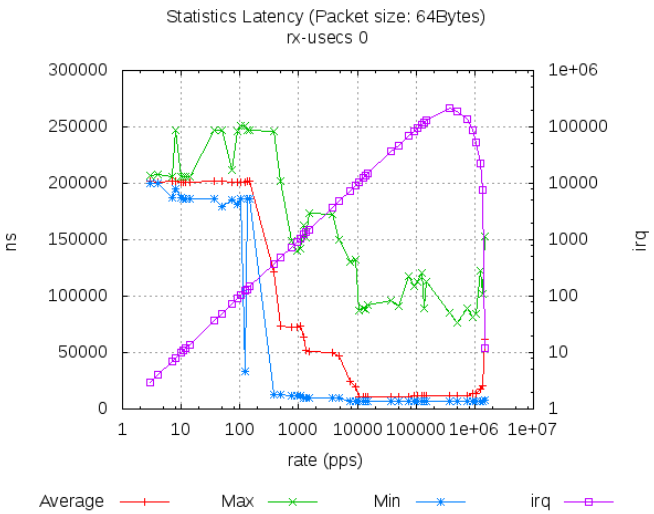


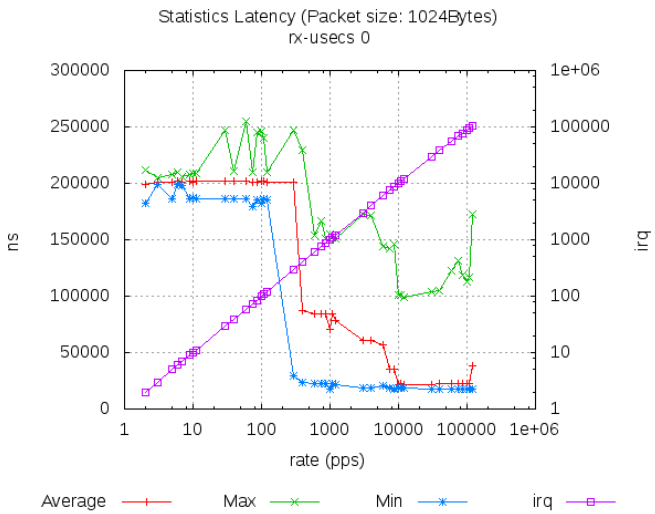
Figure: Jitter. Original

Latency

- TX and RX same machine
- Unexpected behaviour: high latency at low rates



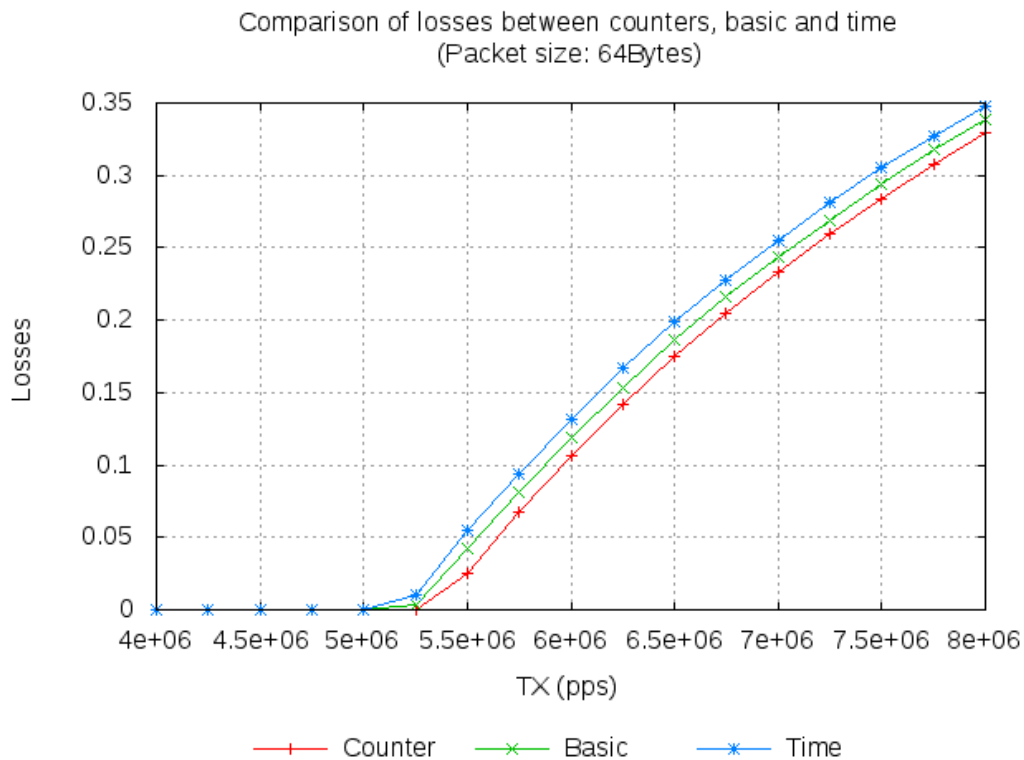
(a) Packet size 64 Bytes



(b) Packet size 1024 Bytes

Comparison between methods of collecting statistics

- Counters
- Basic
- Time



Header split

- Header + Data in different memory regions
- IP stack is were most of the perfomance is drop

Test	Received Rate (No hook)	Received Rate (Hook)
Split headers	5.8 Mpps	6.64 Mpps
NO Split	6.5 Mpps	6.74 Mpps

Outline

Conclusions

- New features of modern network cards and SMP systems
- Improvment of granularity and usability of pktgen's transmission

Conclusions

- New features of modern network cards and SMP systems
- Improvment of granularity and usability of pktgen's transmission
- Receiver side statistics for different scenarios
  - Counters, basic, time
- Receiver is a powerful tool to understand how the Linux kernel behave
  - Receiving packets in SMP
  - Inter-arrival time and jitter
  - Latency in function of the rate

Conclusions

- New features of modern network cards and SMP systems
- Improvment of granularity and usability of pktgen's transmission
- Receiver side statistics for different scenarios
  - Counters, basic, time
- Receiver is a powerful tool to understand how the Linux kernel behave
  - Receiving packets in SMP
  - Inter-arrival time and jitter
  - Latency in function of the rate
- Displaying results in human and script readable
- Integrated in current version of pktgen

Conclusions

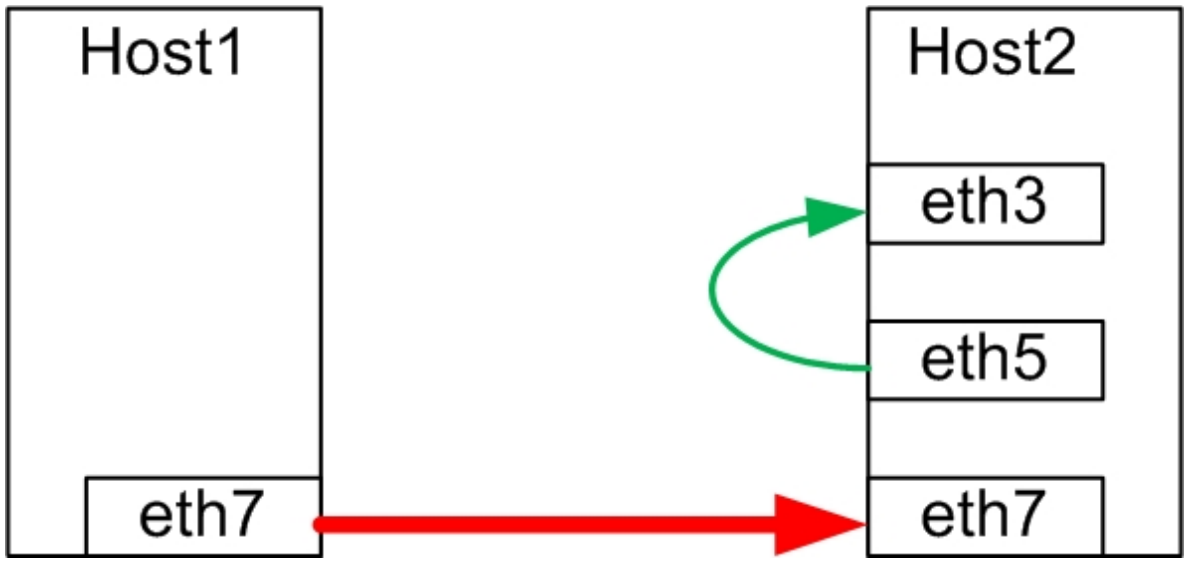
- New features of modern network cards and SMP systems
- Improvment of granularity and usability of pktgen's transmission
- Receiver side statistics for different scenarios
  - Counters, basic, time
- Receiver is a powerful tool to understand how the Linux kernel behave
  - Receiving packets in SMP
  - Inter-arrival time and jitter
  - Latency in function of the rate
- Displaying results in human and script readable
- Integrated in current version of pktgen
- Tested in different research works
- In process of been integrated in the main Linux Kernel

Future work

- New applications and new studies
- Study of the influence of how the the inter-arrival time is affected of the delay strategy in Pktgen sender
  - CBR traffic
- Implementing a latency test with some synchronization

Outline

Demonstration



1. Enable Receiver and selecting statistics
2. Throughput Test
3. Different Displays
4. Time test (Same Machine)

Thank you for your attention

Any question?