# *Lightweight enhanced monitoring for high-speed networks*

Rosa Vilardi, Dr. Luigi Alfredo Grieco, Prof. Gennaro Boggia
Electrical and Information Engineering Department (DEI)
Politecnico di Bari
Italy

Dr. Chadi Barakat
INRIA Sophia Antipolis
Sophia Antipolis
France

**5th NMRG workshop, 87th IETF meeting**

**Berlin, July 30th 2013**

# Agenda

- Problem definition, applications, and challenges
- LEMON algorithm
  - ✓ Main contribution
  - ✓ Key assumptions & features
  - ✓ Theoretical model
  - ✓ Practical integration in IPFIX
- Performance evaluation and experimental results
- Conclusions & Future Work

# Traffic monitoring today…

## ❑ Problems

- ➢ Processing of large amount of data for their classification and characterization
- ➢ Saving of precious resources (cpu, memory, bandwidth)

## ❑ Solutions

- ✓ Packet sampling techniques
  - reduce monitoring overhead
  - introduce estimation errors
- ✓ Flow-based monitoring systems (NetFlow/IPFIX)
  - inspect the traffic composition

## ❑ Ok that's good, but…

- ➢ The exporting process is triggered by timers **STATICALLY** established and set in the order of some minutes
- ➢ Traffic characteristics are estimated with a **COARSE** and **FIXED** time resolution
- ➢ Management tools could recognize an anomalous event long after it occurs, not while it is in progress

# Traffic monitoring tomorrow(?)…

## LEMON

Lightweight Enhanced MOnitoring for high-speed Networks

- ✓ Real-time traffic monitoring at router interface
- ✓ Compliant with IPFIX exporting protocol
- ✓ Low impact on existing technologies
- ✓ Low processing and communication overhead

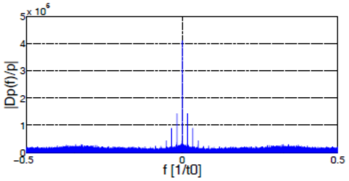
❑ **Main contributions**

- ➢ **ACCURATE** flow measurements in a **CUSTOMIZED** and **DYNAMIC** way
- ➢ **DYNAMIC EXPORTING TIMING** to the management applications for prompt detection of network anomalies

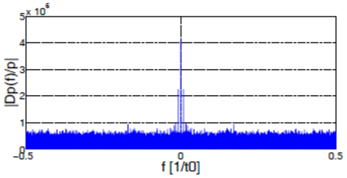
# *Why LEMON?*

## Motivations [*]

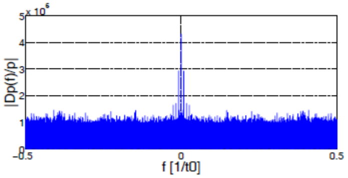
- Traffic anomalies correspond to rapid and often short term shift of the data traffic
- High frequency changes in the bitrate spectrum are hard to detect in the time domain
- Bitrate estimation error (due to **PACKET SAMPLING**) is modeled by aliasing effects on the reconstructed signal spectrum



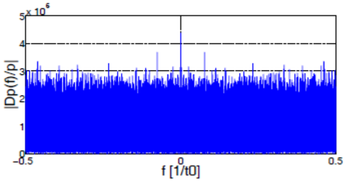
p=1



p=0.1



p=0.003



p=0.005

- Dynamic tuning of the temporal observation window (***time bin***) can lead to respect a target performance

[*] L. A. Grieco, C. Barakat, and M. Marzulli,**" Spectral Models for Bitrate Measurement from Packet Sampled Traffic",** *IEEE Trans. on Network and Service Management*, vol. 8, no. 2, Jun., 2011.

# Why LEMON?

## ❑ Key assumptions [*]

- ➢ **FLOW BITRATE ESTIMATION**; the accuracy is evaluated looking at its SNR value
- ➢ The **SNR** is linked to:
  - packet sampling probability, **p**
  - Monitoring time bin (exporting timer), **T**

## ❑ Variable packet size (VPS) model for SNR [*]

$$SNR = \frac{p\left(T \cdot C \cdot \overline{D}^2 + 0.89M\right)}{0.89M\left(1 - p\right)} = \frac{p}{1 - p}\left[\frac{T \cdot C \cdot \overline{D}^2}{0.89M} + 1\right]$$

**Average packet transmission rate** $C$

**First and second order moment of the packet size** $\overline{D}^2$ **and** $M$

(The monitoring time bin T is modeled as a low-pass filter with a frequency band that is 0.89/T wide)

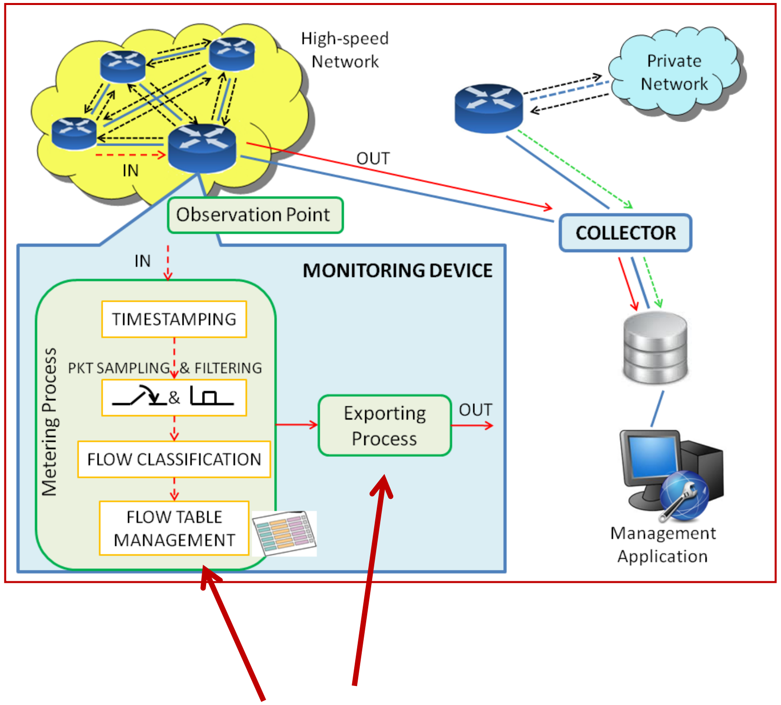
# Why LEMON?

- **Key assumptions [**]**
  - **UNLIKE** the other systems, a **TARGET SNR** is set as a system requirement
  - **p** is kept fix, **Ti** is tuned accordingly (both in time and depending of each flow), to ensure the required target **SNRth**
  - **Di**, **Mi**, and **Ci** are based on the past history of the i-th flow, using an EWMA filter

$$T_i(k) = \left[ \frac{1-p}{p} \cdot SNR_{th} - 1 \right] \cdot \frac{0.89 \cdot M_i}{C_i \cdot \overline{D}_i^2}$$

[**] R. Vilardi, L.A. Grieco, C. Barakat, and G. Boggia**," Lightweight enhanced monitoring for high speed networks",** *ETT, Transactions on Emerging Telecommunications Technologies, Wiley*, 2013, DOI: 10.1002/ett.2637.

# How to integrate LEMON in IPFIX?



**LEMON**

## Before…

Measures exported **AFTER** flow expiration

- **flowIdleTimeout** (300s dafault)
- **flowActiveTimeout** (1800s default)

## Now…

Measures exported **ALSO WHILE** flow is still active

***flowBinTimeout*** (compliant to IPFIX RFC5102)

- Dynamic in time
- Customized to each flow

**The algorithm:** Three main processing operations:

- Working parameter setting
- Per-flow bin counters management
- Data exporting

# Dataset

- MAWI Project: traffic @ Asian Transpacific Links
  Three distinct traces 15 min long
  **Flow key**: SourceIP first 8 bits (aggregate flows)

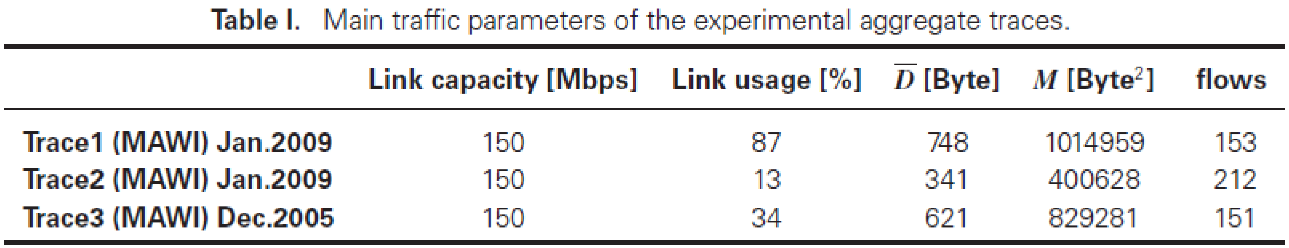


**Table I.** Main traffic parameters of the experimental aggregate traces.

| | Link capacity [Mbps] | Link usage [%] | $\overline{D}$ [Byte] | $M$ [Byte$^2$] | flows |
|---|---|---|---|---|---|
| Trace1 (MAWI) Jan.2009 | 150 | 87 | 748 | 1014959 | 153 |
| Trace2 (MAWI) Jan.2009 | 150 | 13 | 341 | 400628 | 212 |
| Trace3 (MAWI) Dec.2005 | 150 | 34 | 621 | 829281 | 151 |

- European ISP: traffic @ xDSL router (~1000 customers) attached to a DSLAM
  Single trace 3 hours long
  **Flow key**: SourceIP+Prot_type

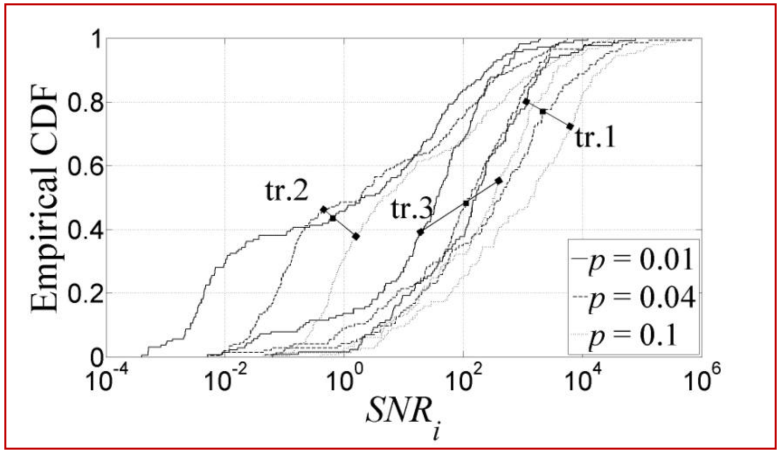
  Data bit rate 12.74 Mbps
  Average pkt size: 455.60 bytes
  Average pkt rate: 3664.90 pkt/s

# Model validation

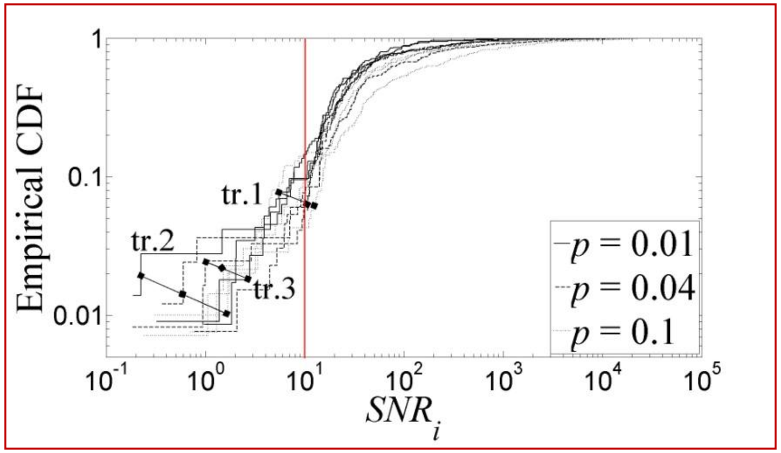
## System requirement: target SNR

- ✓ Fixed-scale monitoring systems don’t guarantee the min target SNR
- ✓ LEMON captures packet sampling effects and targets SNR larger than the threshold constraint

**MAWI traces**



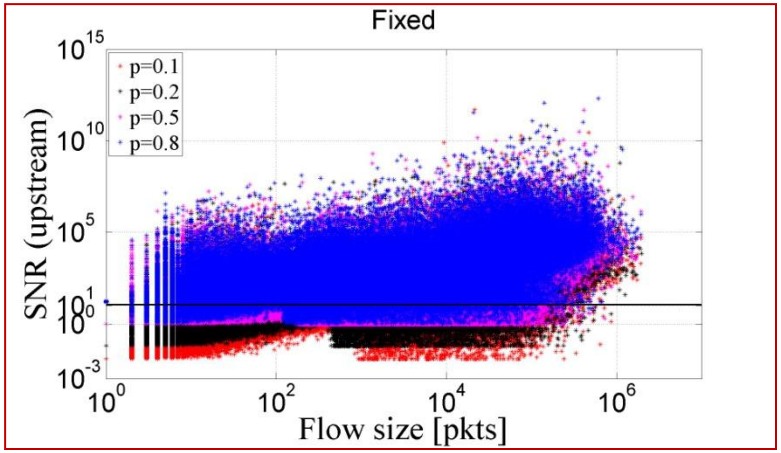
**Fixed time bin (T=240s)**

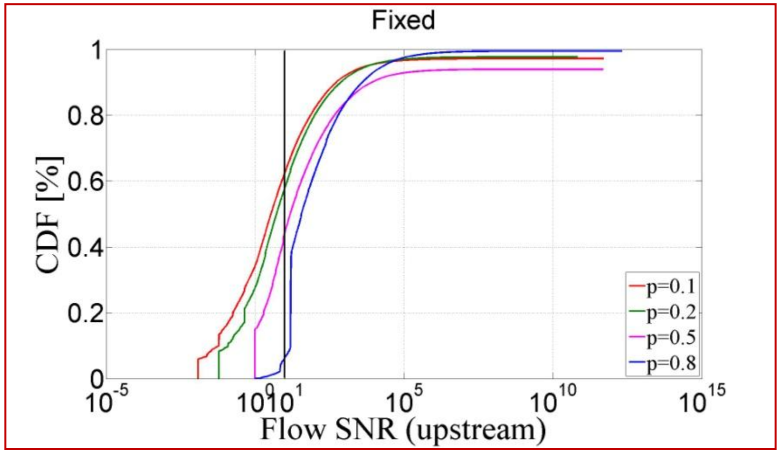


**LEMON VPS model (SNRth=10)**

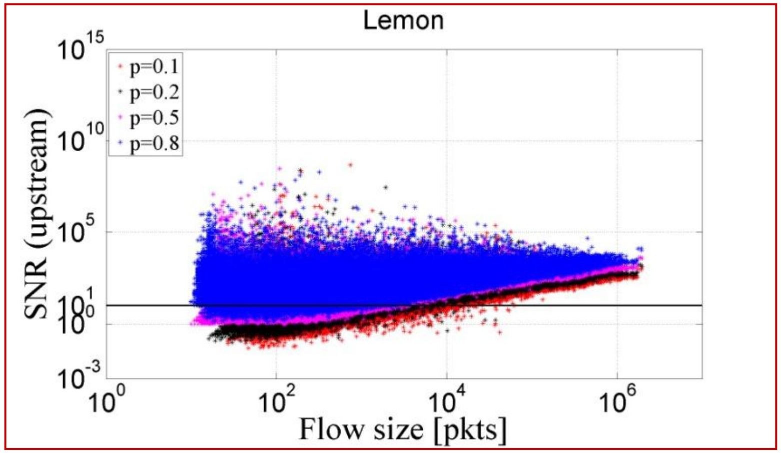
# Model validation

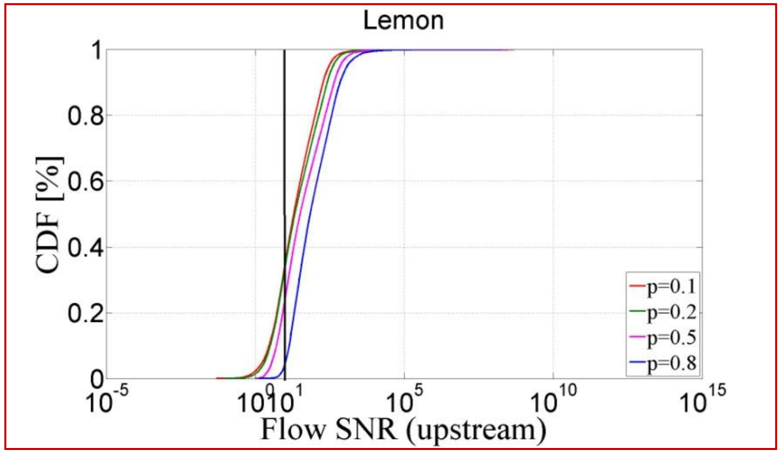
## ISP trace





**Fixed time bin (T=60s)**



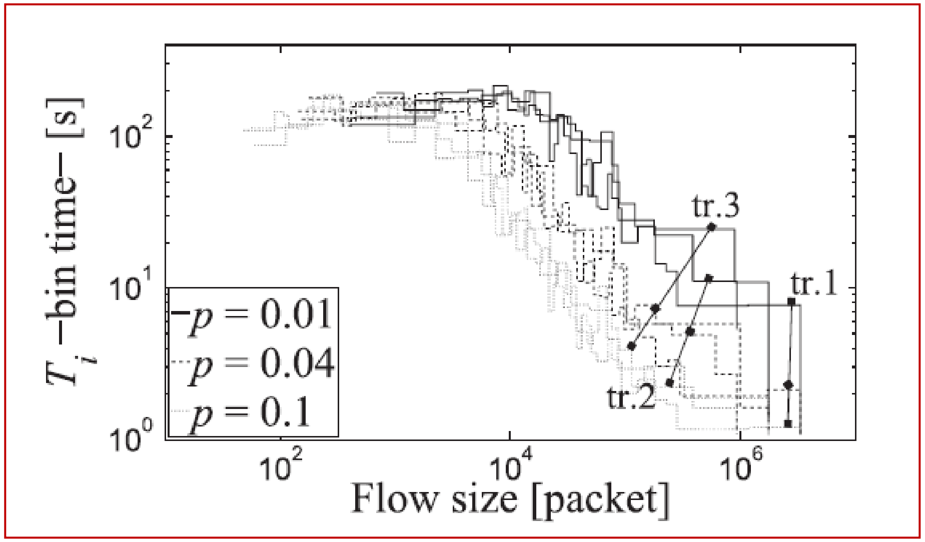


**LEMON VPS model (SNRth=10)**

# Model validation

- **Time resolution**
  - ✓ smaller values of the time bin windows for larger flows: finer time resolution

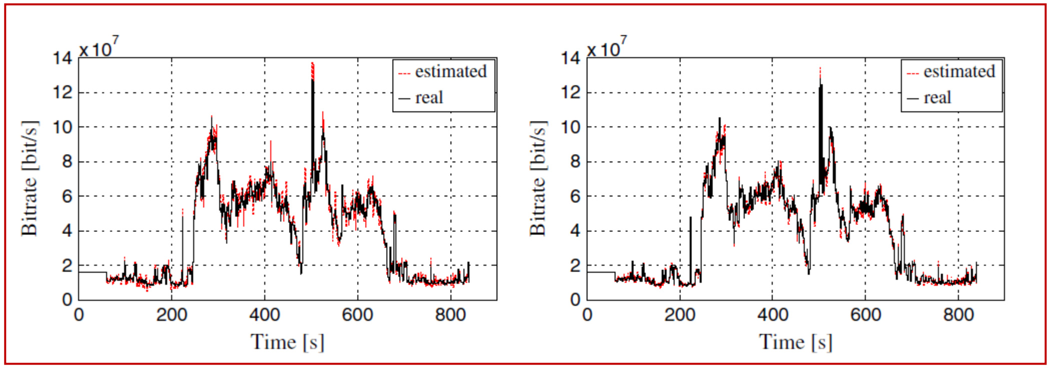
**MAWI traces**

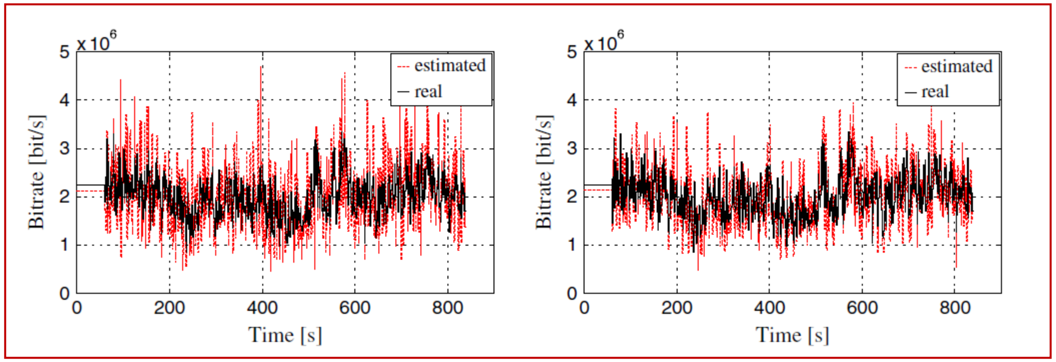


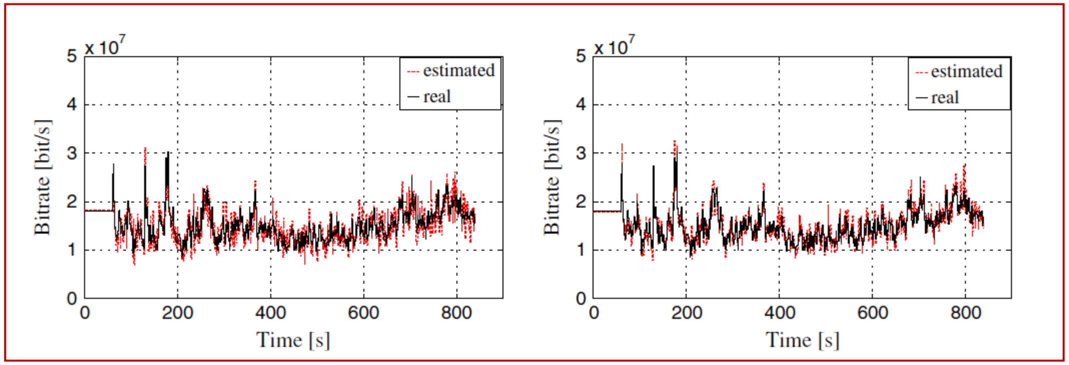
**LEMON VPS model (SNRth=10)**

# Bitrate estimation accuracy

MAWI traces







# Communication overhead

## ❑ MAWI traces

Due to both the flow records attributes (***information element* data records**), and the control messages (***control information* records**)

- ➢ **policy=0** a single IPFIX message is sent at each flowBinTimeout expiration for a single flow
- ➢ **policy=1** an aggregate IPFIX message is sent at the expiration of 10 flowBinTimeout
- ➢ **policy=2** an aggregate IPFIX message is sent at the end of a timeout lasting 5 s, for each expired flowBinTimeout

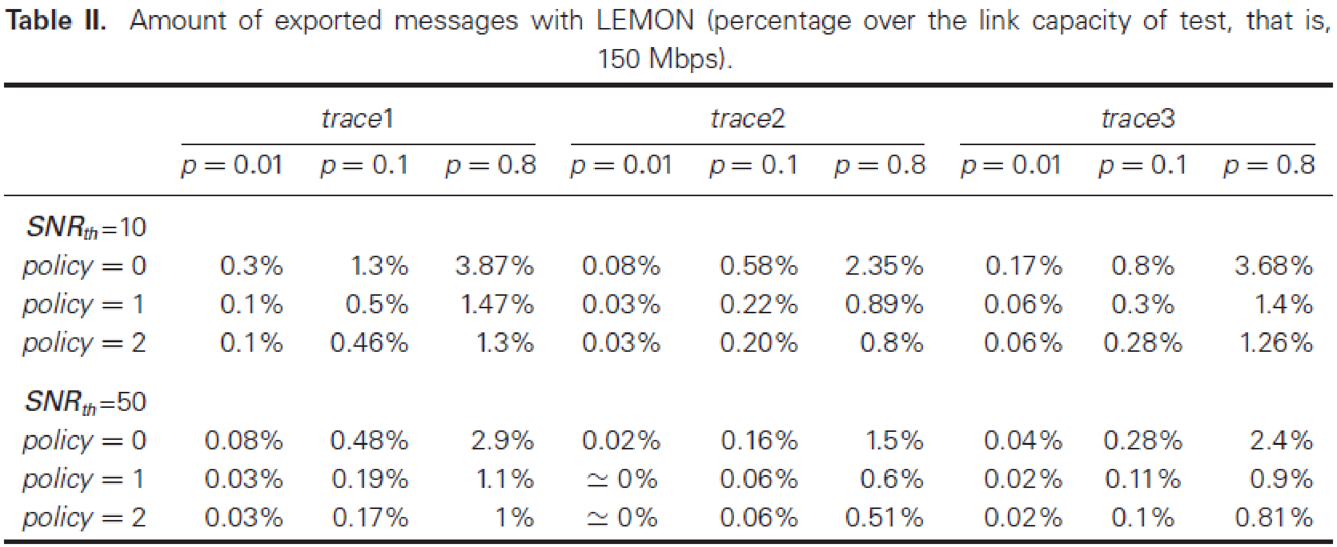


**Table II.** Amount of exported messages with LEMON (percentage over the link capacity of test, that is, 150 Mbps).

| | *trace1* | | | *trace2* | | | *trace3* | | |
|---|---|---|---|---|---|---|---|---|---|
| | $p = 0.01$ | $p = 0.1$ | $p = 0.8$ | $p = 0.01$ | $p = 0.1$ | $p = 0.8$ | $p = 0.01$ | $p = 0.1$ | $p = 0.8$ |
| $SNR_{th}=10$ | | | | | | | | | |
| $policy = 0$ | 0.3% | 1.3% | 3.87% | 0.08% | 0.58% | 2.35% | 0.17% | 0.8% | 3.68% |
| $policy = 1$ | 0.1% | 0.5% | 1.47% | 0.03% | 0.22% | 0.89% | 0.06% | 0.3% | 1.4% |
| $policy = 2$ | 0.1% | 0.46% | 1.3% | 0.03% | 0.20% | 0.8% | 0.06% | 0.28% | 1.26% |
| $SNR_{th}=50$ | | | | | | | | | |
| $policy = 0$ | 0.08% | 0.48% | 2.9% | 0.02% | 0.16% | 1.5% | 0.04% | 0.28% | 2.4% |
| $policy = 1$ | 0.03% | 0.19% | 1.1% | $\simeq 0\%$ | 0.06% | 0.6% | 0.02% | 0.11% | 0.9% |
| $policy = 2$ | 0.03% | 0.17% | 1% | $\simeq 0\%$ | 0.06% | 0.51% | 0.02% | 0.1% | 0.81% |

# Memory consumption

## ❑ ISP trace

engine 3 line card (256 MB of memory and 16 network interfaces) embedded in Cisco 12000 routers

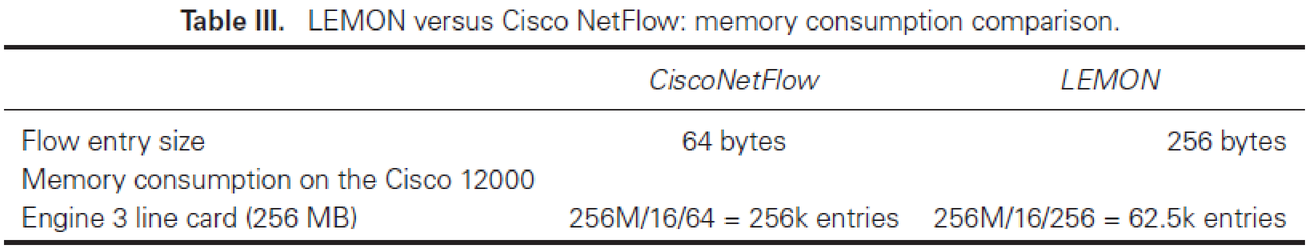
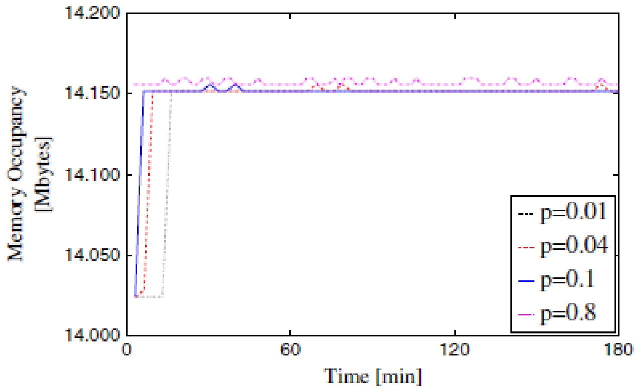
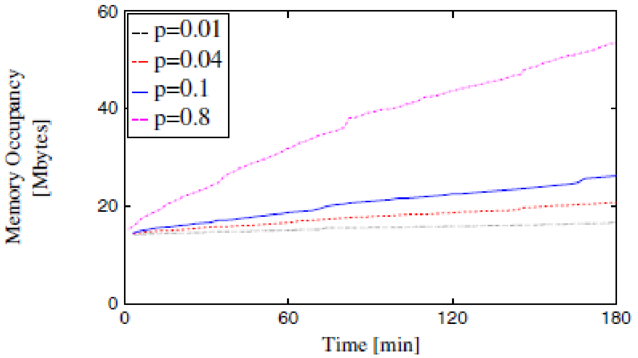


**Table III.** LEMON versus Cisco NetFlow: memory consumption comparison.

| | *CiscoNetFlow* | *LEMON* |
|---|---|---|
| Flow entry size | 64 bytes | 256 bytes |
| Memory consumption on the Cisco 12000<br>Engine 3 line card (256 MB) | 256M/16/64 = 256k entries | 256M/16/256 = 62.5k entries |



**Flow key**: SourceIP first 8 bits

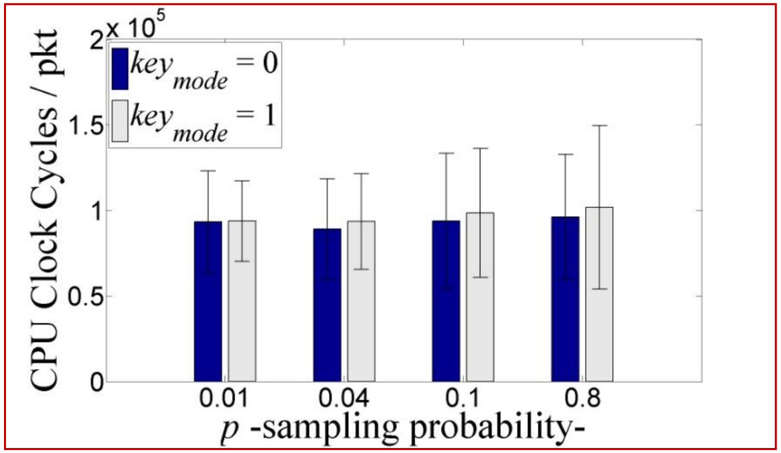


**Flow key**: SourceIP+Prot_type

# Processing overhead

## ON BOARD…

- Intel Core 2 Duo P7450 (2.13 GHz, 3 MB L2 cache, 800 MHz DDR2), 6 GB of RAM and Ubuntu 10.04.4 on board
- CPU cycle number measured by the *ReaD Time Stamp Counter (RDTSC) CPU instruction*



**Per- packet processing overhead**

## ON Cisco 12000…

- Main processor: 667 MHz
- Processing time: 150 us (per-pkt clock cycles / CPU clock)
- pkts processed per sec: 1 / (150 * 10^6) = 6670 packet/s
- Max traffic rate: (6670 * 800 * 8) / 0.01 = 4.27 Gbps (p=0.01, mean pkt size 800 bytes)

# *Conclusion*

## *Is LEMON IPFIX-friendly?*

- ✓ **ADAPTIVE** traffic monitoring (**DYNAMIC** in time, **CUSTOMIZED** to each flow) **=> high granularity** for the measures
- ✓ **ACCURATE** flow bitrate estimation compliant with prior **target accuracy requirements**
- ✓ **LOW** communication overhead in IPFIX message exporting operations
- ✓ **LOW** processing overhead, easily integrated and supported by current routers

# ?????   ||   /* … */

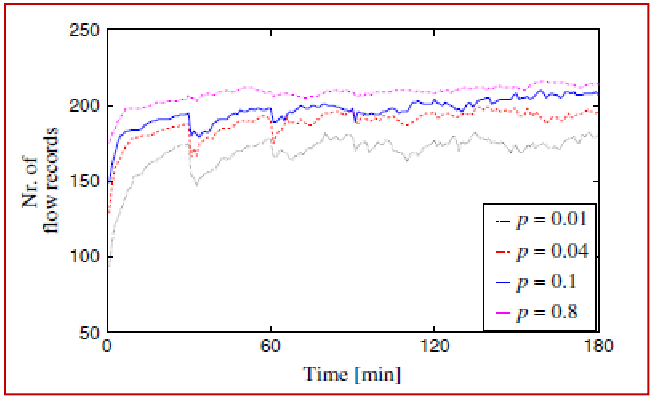
*ROSA VILARDI*

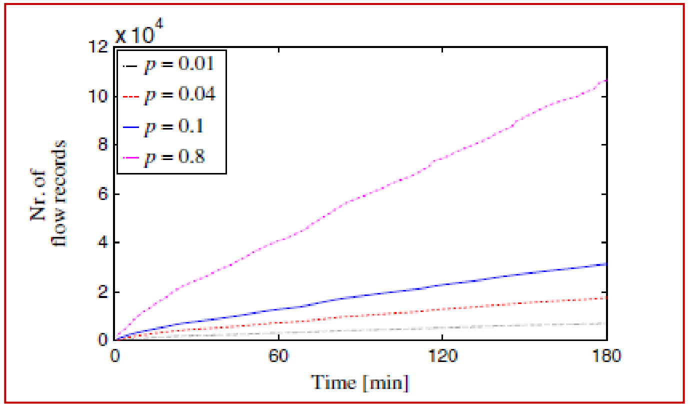
- **Web page:** http://telematics.poliba.it/vilardi/
- **Skype contact:** rosa.vilardi
- **E-mail:** r.vilardi@poliba.it

# *Something more…*

# Flow Table consumption

## ISP trace

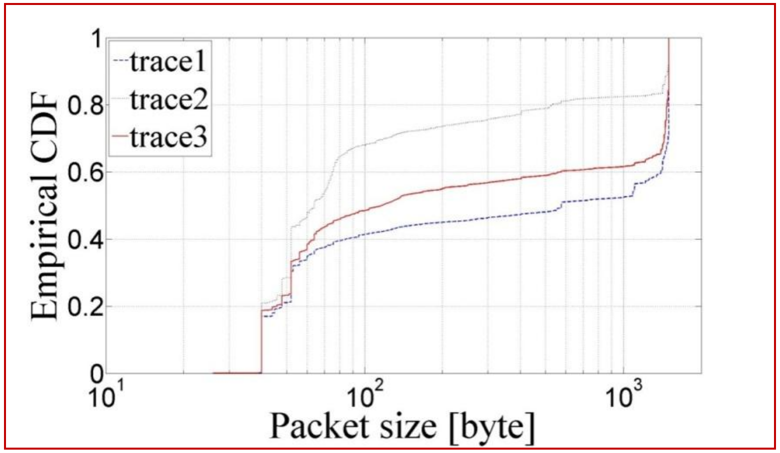




**Flow key**: SourceIP first 8 bits

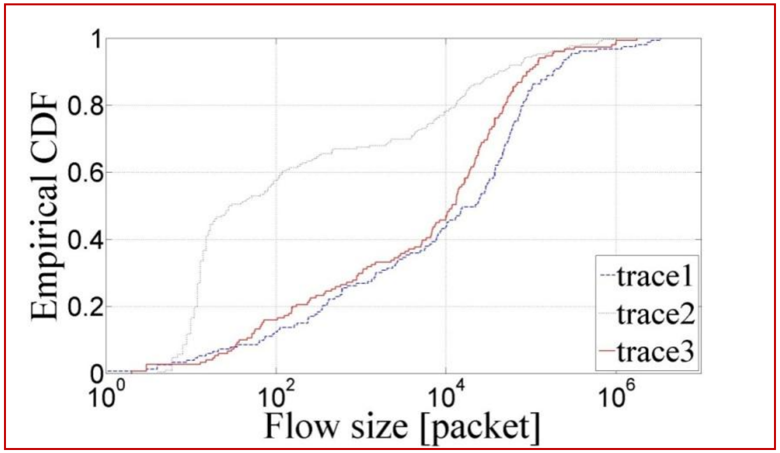
**Flow key**: SourceIP+Prot_type

# Characteristics of the MAWI traces



Pkt size CDF

Flow size CDF


Empirical CDF
10^0
10^2
10^4
10^6
Flow size [packet]
trace1
trace2
trace3