# SF-TAP: Scalable and Flexible Traffic Analysis Platform Running on Commodity Hardware

Yuuki Takano, Ryosuke Miura, Shingo Yasuda
Kunio Akashi, Tomoya Inoue

NICT, JAIST
(Japan)

in USENIX LISA 2015

# Table of Contents

# Motivation (1)

- Programmable application level traffic analyzer
  - We want …
    - to write traffic analyzers in any languages such as Python, Ruby, C++, for many purposes (IDS/IPS, forensic, machine learning).
    - **not** to write codes handling TCP stream reconstruction (quite complex).
    - modularity for many application protocols.

# Motivation (2)

- High speed application level traffic analyzer
  - We want …
    - to handle high bandwidth traffic.
    - to handle high connections per second.
    - horizontal and CPU core scalable analyzer.

# Motivation (3)

- Running on Commodity Hardware
- We want …
  - open source software.
  - not to use expensive appliances.

# Related Work

SF-TAP

+ modularity and scalability

GASPP [USENIX ATC 2014]

SCAP [IMC 2012]

libnids

(flow oriented analyzer)

l7-filter

nDPI

libprotoident

(application traffic detector)

(low level traffic capture)

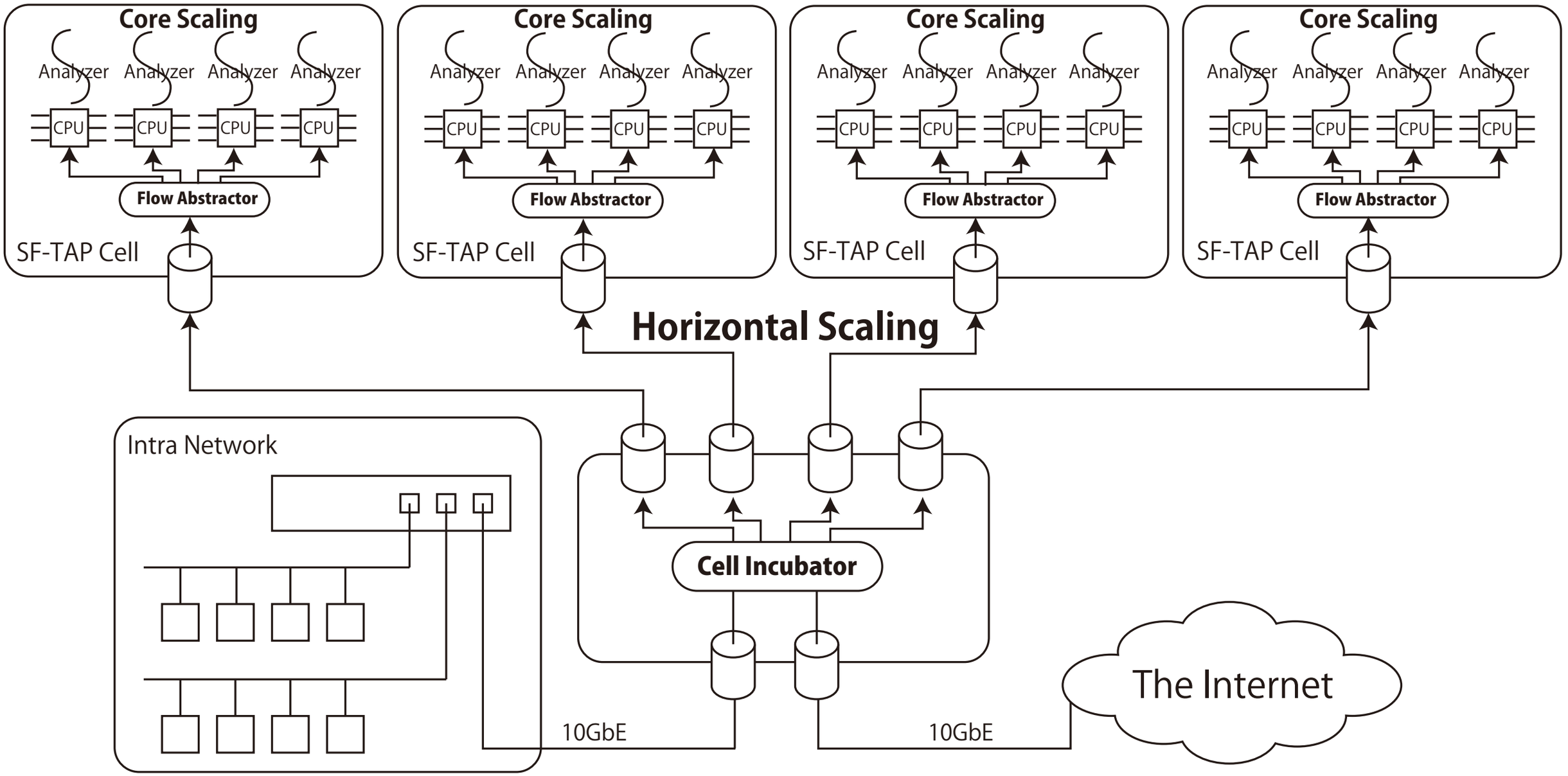
DPDK

pcap

netmap [USENIX ATC 2012]

BPF [USENIX ATC 1993]

# High-level Architecture of SF-TAP



# Design Principle (1)

- Flow Abstraction
  - abstract flows by application level protocols
  - provide flow abstraction interfaces like /dev, /proc or BPF
  - for multiple programming languages
- Modular Architecture
  - separate analyzing and capturing logic
  - easily replace analyzing logic

# Design Principle (2)

- Horizontal Scalable
  - analyzing logic tends to require many computer resources
    - volume effect should solve the problem
- CPU Core Scalable
  - both analyzing and capturing logic should be core scalable for efficiency

# Design of SF-TAP (1)

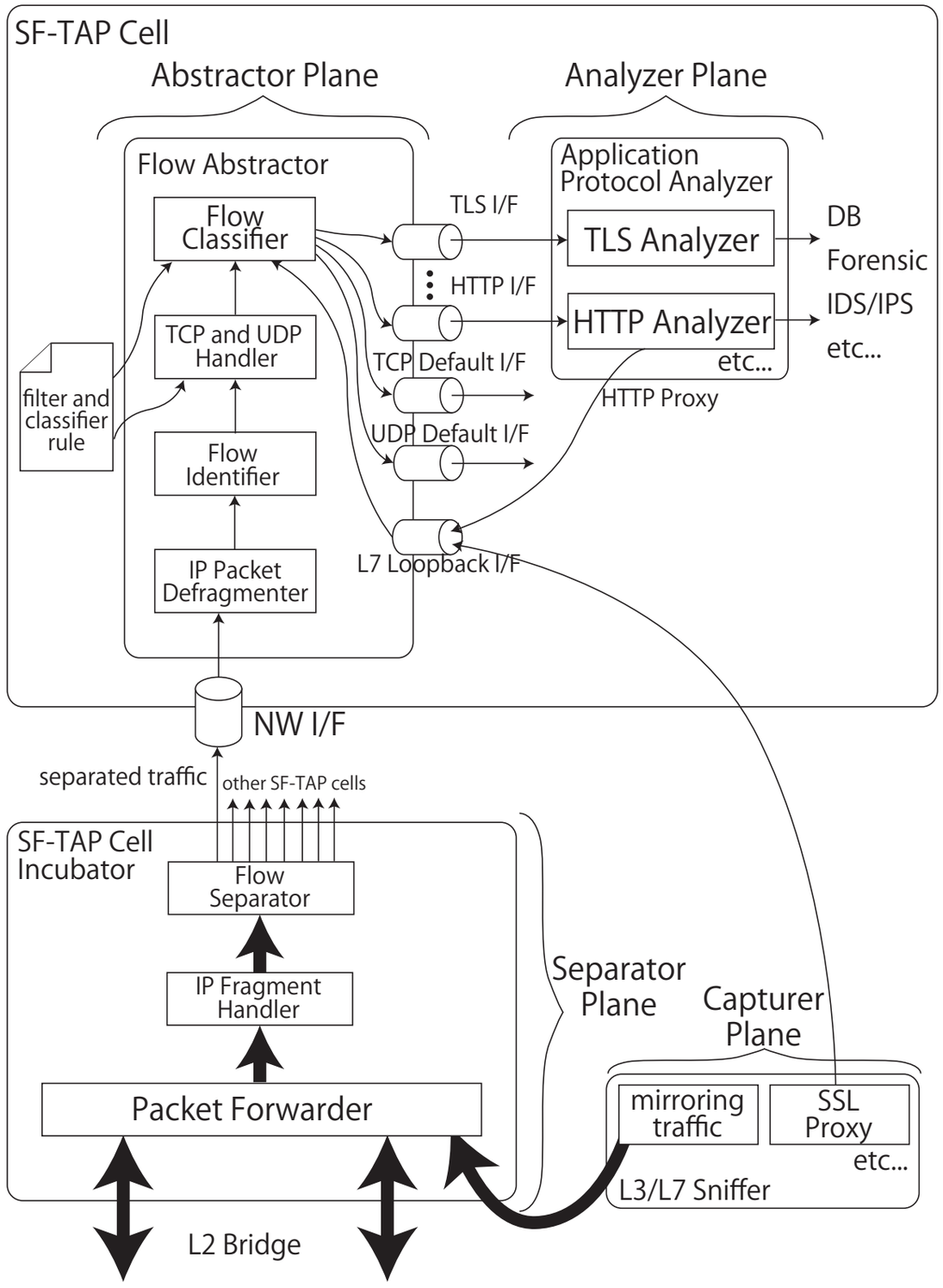
## defined 4 planes

### Analyzer Plane
application level analyzers
Forensic, IDS/IPS, etc…
(users of SF-TAP implements here)

### Abstractor Plane
flow abstraction
(we implemented)

### Separator Plane
flow separation
(we implemented)

### Capturer Plane
traffic capturing
(ordinary tech.)



# Design of SF-TAP (2)
# SF-TAP Cell Incubator

## Flow Separator

separate flows to multiple Ifs

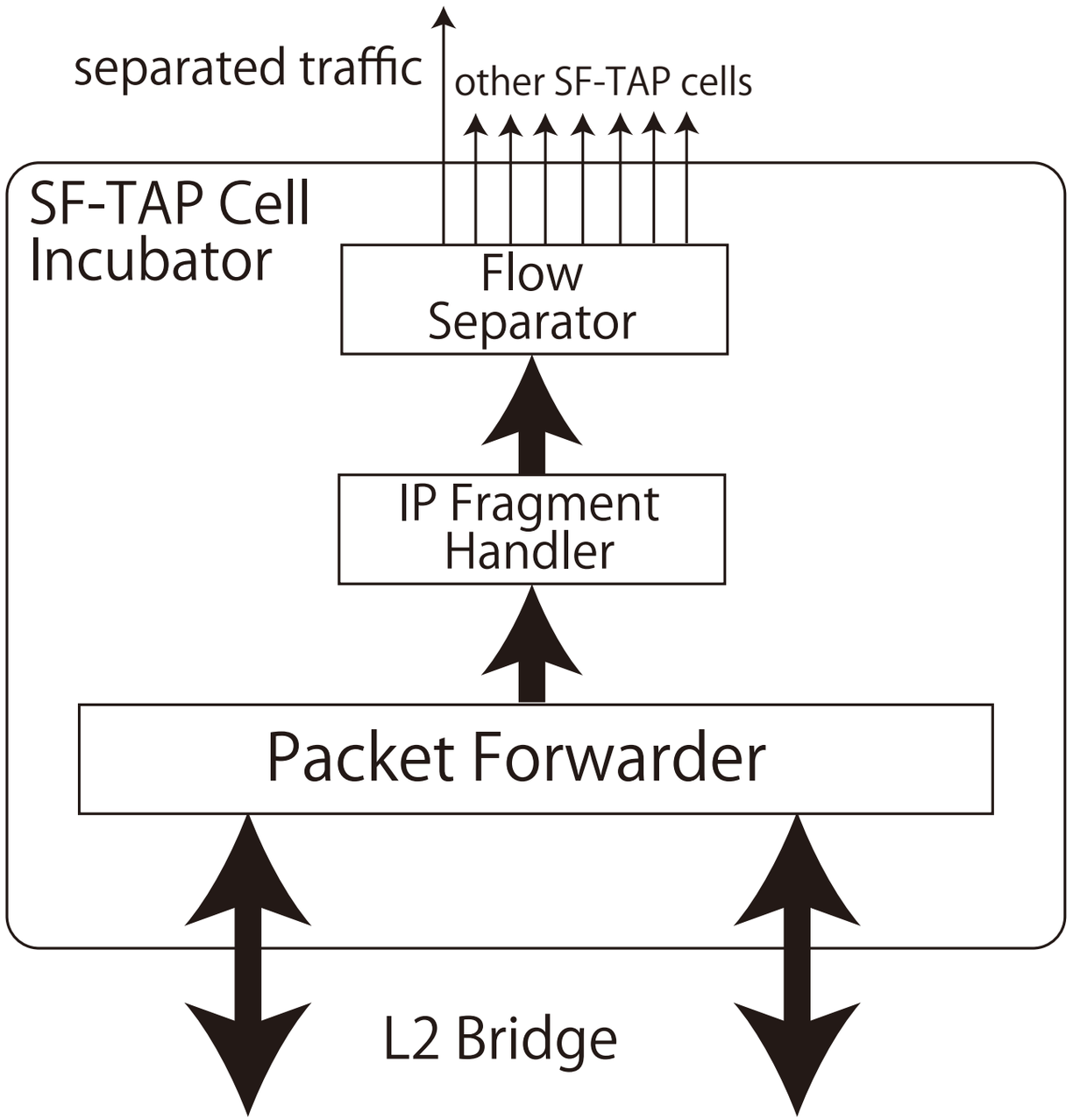
## IP Fragment Handler

handle fragmented packets

## Packet Forwarder

layer 2 bridge

layer 2 frame capture



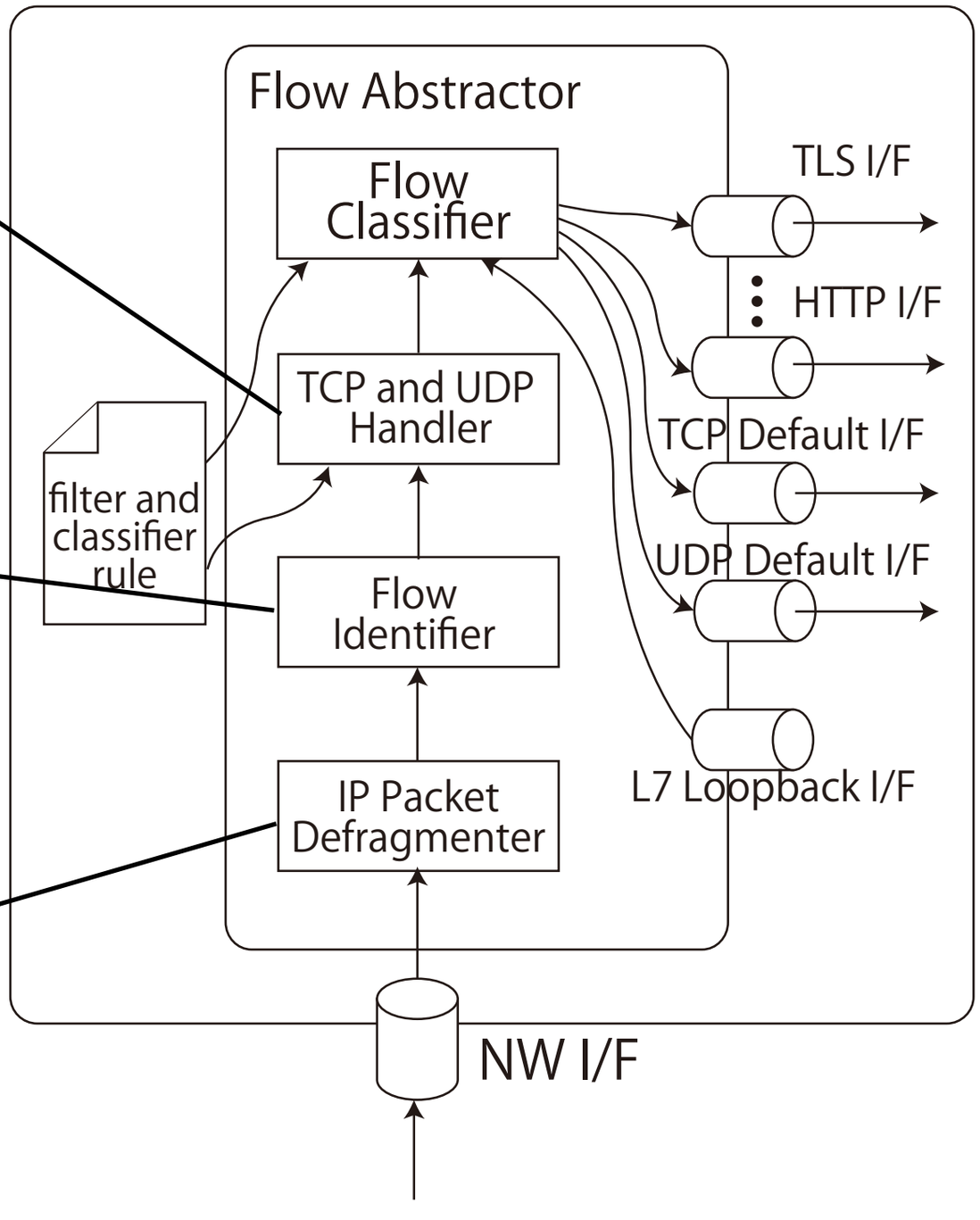
# Design of SF-TAP (3)
# SF-TAP Flow Abstractor

## TCP and UDP Handler
reconstruct TCP flows

nothing to do for UDP

## Flow Identifier
identify flows by IP and port

## IP Packet Defragmenter
defragment IP packets if needed

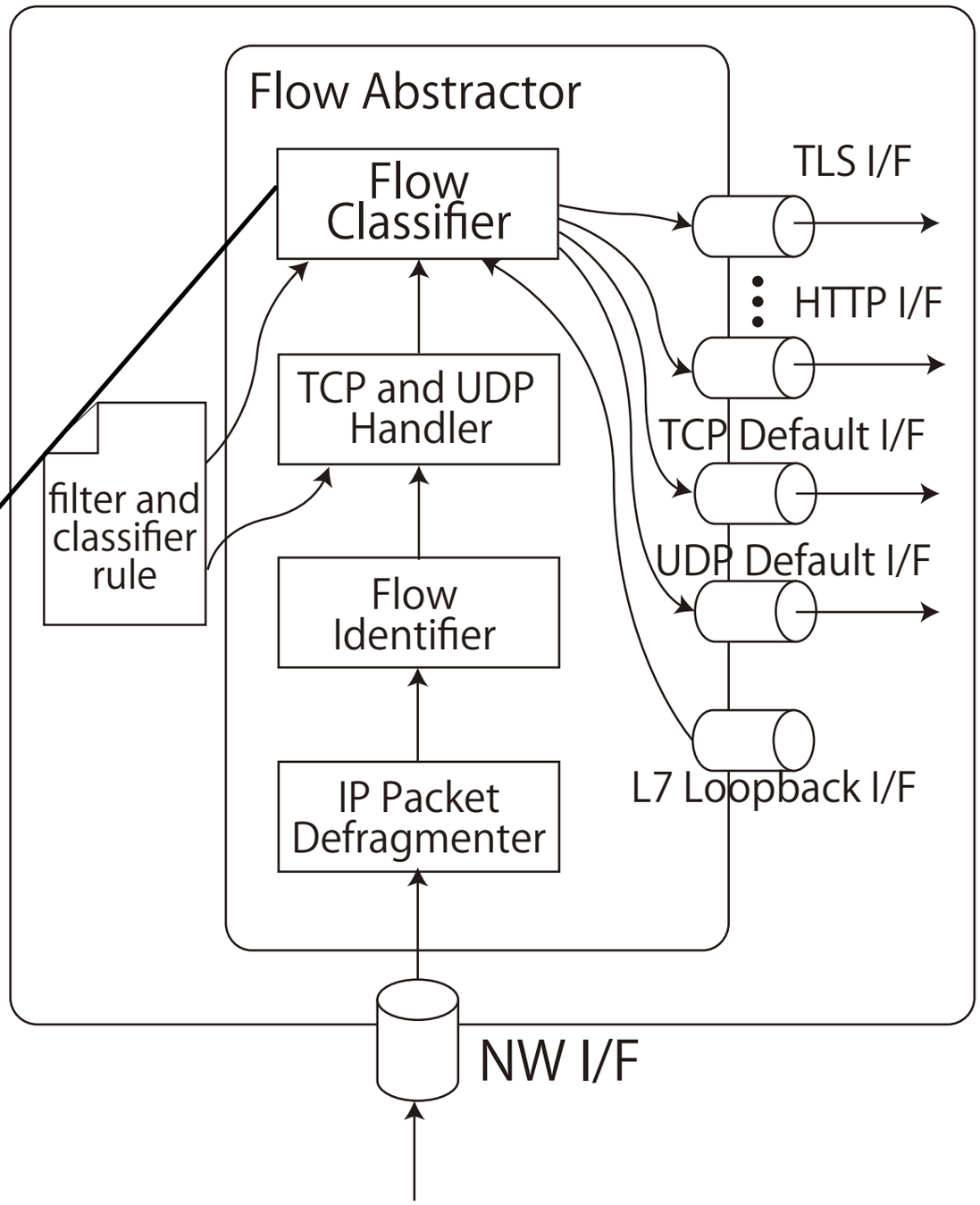


# Design of SF-TAP (4)
# SF-TAP Flow Abstractor

## Flow Classifier

classify flows by
regular expressions

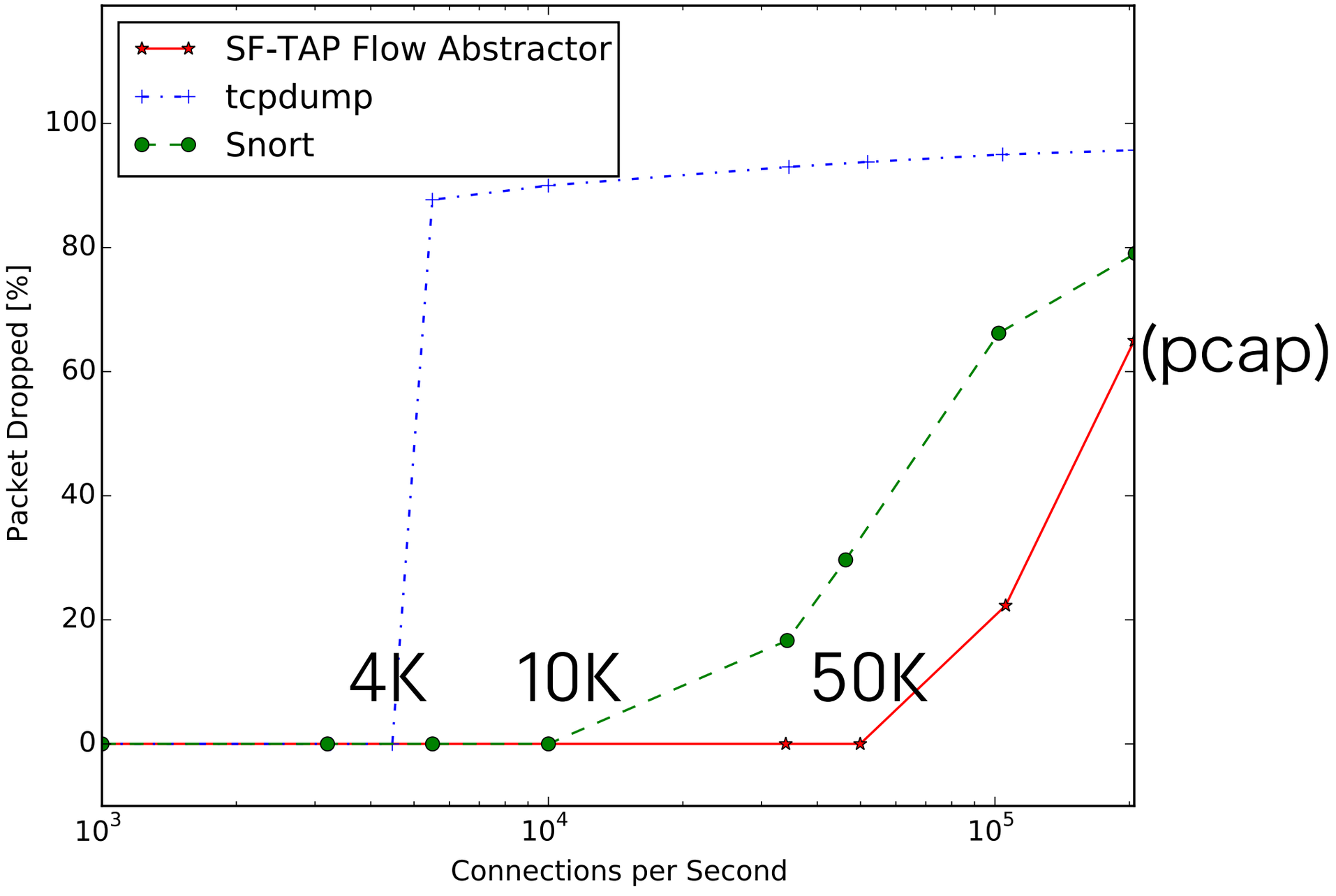
output to abstraction IFs



# Implementation

- SF-TAP cell incubator
  - C++11
  - it uses netmap, available on FreeBSD
- SF-TAP flow abstractor
  - C++11
  - it uses pcap or netmap (updated from the paper)
  - available on Linux, *BSD, and MacOS
- Source Code
  - https://github.com/SF-TAP
- License
  - 3-clauses BSD

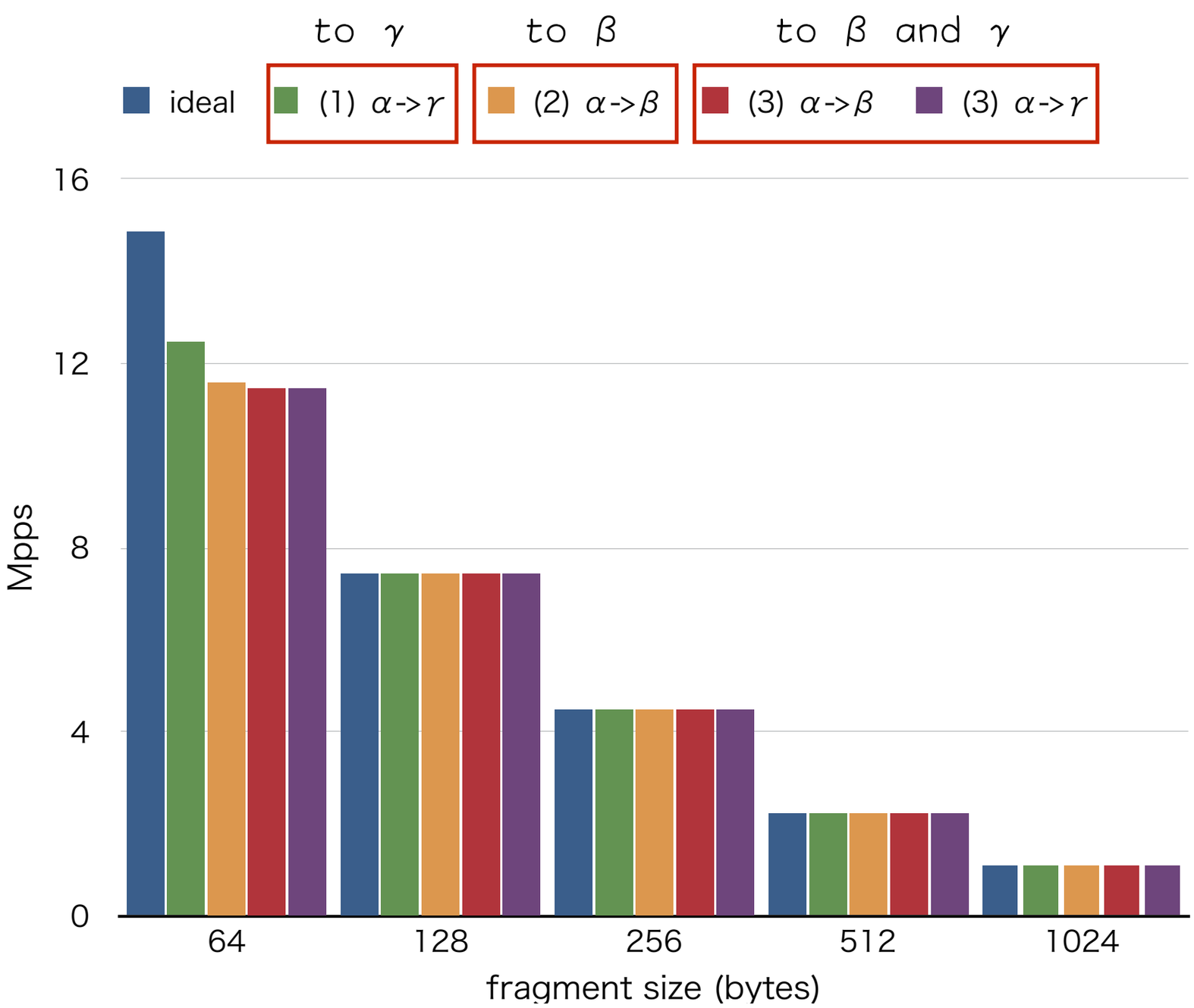
# Performance Evaluation (1)

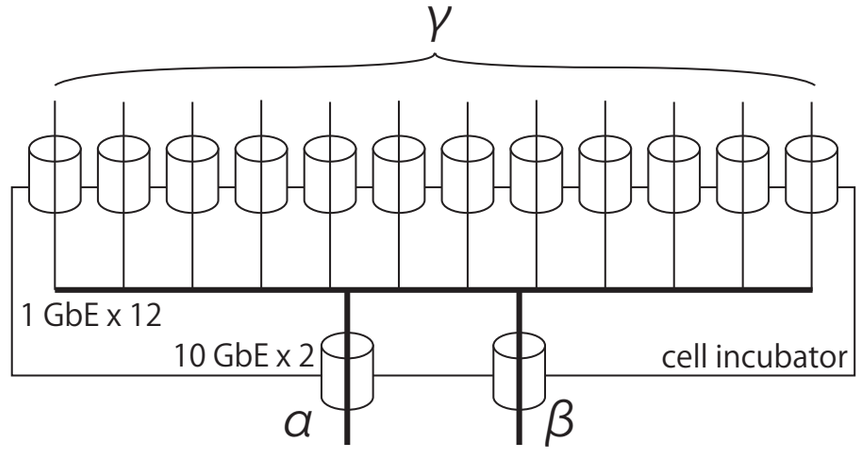


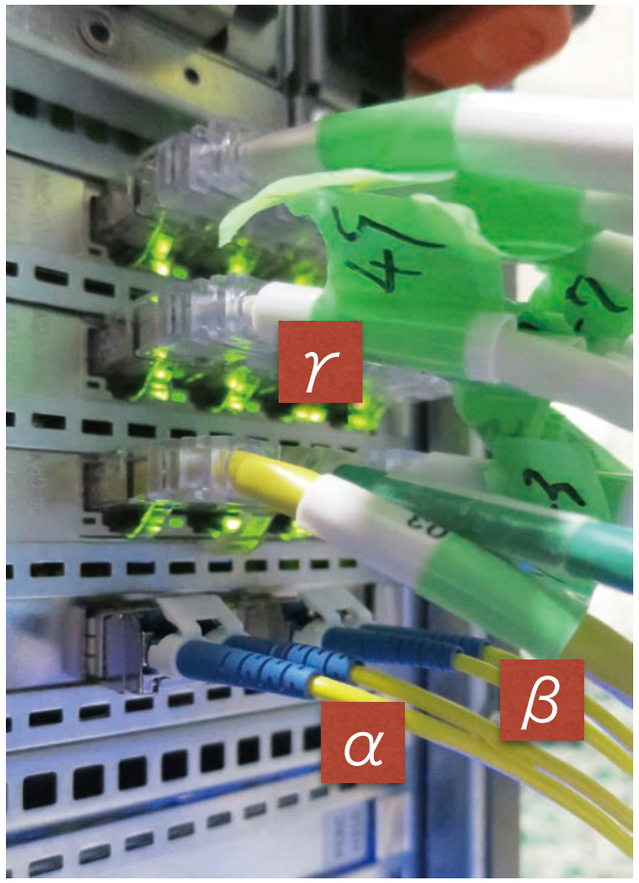

(pcap)

packet drop against connections per second

# Performance Evaluation (1)




forwarding performance of SF-TAP cell incubator

# Other Features

- L7 Loopback interface for encapsulated flows
- Load balancing mechanism for application protocol analysers
- Separating and mirroring modes of SF-TAP cell incubator
- See more detains in our paper

# Conclusion

- We proposed SF-TAP for application level traffic analysis.
- SF-TAP has following features.
  - flow abstraction
  - running on commodity hardware
  - modularity
  - scalability
- We showed SF-TAP has achieved high performance in our experiments.