Improving network packet capture by using Linux kernel's huge pages

Master's thesis

Computer and Information Engineering

Laura Trivelloni

Advisor: Prof. Marco Cesati

Co-Advisors: Ing. Emiliano Betti
Ing. Pierpaolo Santucci

April 8, 2019

1. Introduction
2. Problem analysis
3. Proposed solutions
4. Results
5. Conclusions

Introduction

Scenario

Internet of Things (IoT) phenomenon:

- multitude of type of connected devices
- sometimes sensitive data flow through the network

Network Traffic Analysis

Network Traffic Analysis is a common activity aimed at:

- network monitoring
- recognition of traffic characteristics
- information gathering for fighting criminal or terroristic activities

Problem analysis

Socket buffer

The OS's socket buffer has a key role:

- to store temporarily incoming packets before to be analysed
- to mitigate:
  - peaks of traffic
  - large per-packet processing time

Performance issue

Improving the performance of the buffer management to face with:

- increasing data rate supported by the network cards
  - 10 Gigabit Ethernet
- packet loss due to processing delay on packet
  - more or less frequent delays based on the type of analysis

Linux huge pages

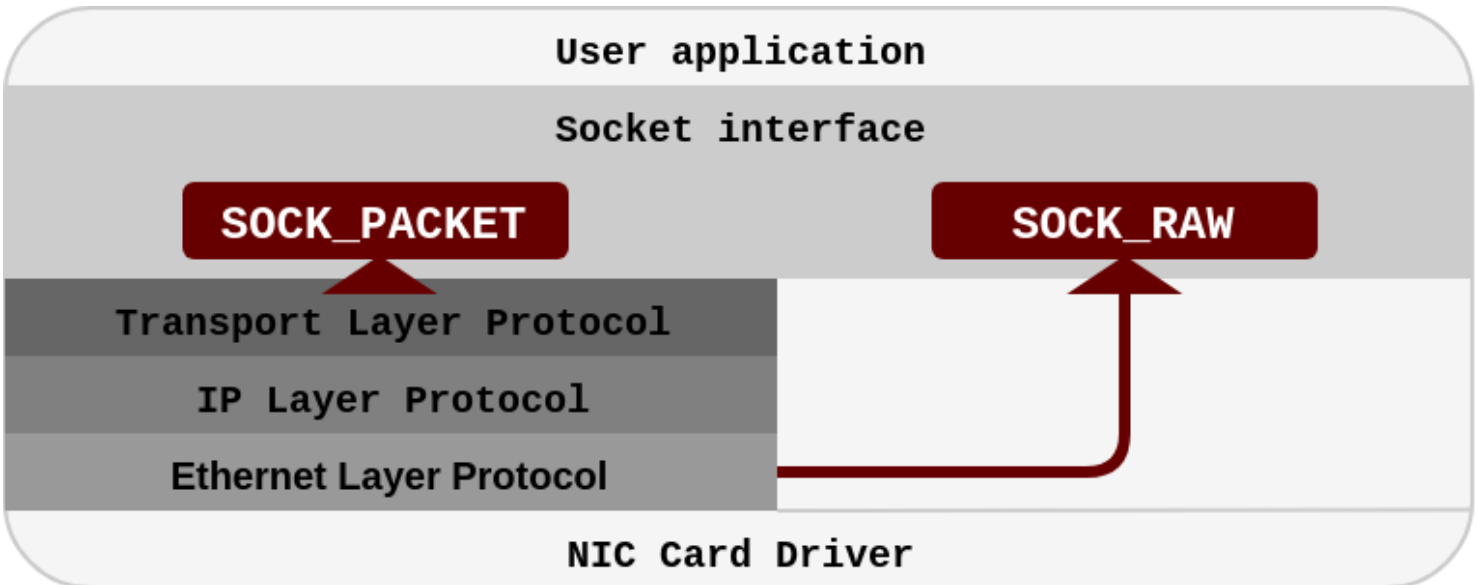
Low-level performance improvement using larger than normal memory pages → huge pages

- wider address range for each TLB entry
- the same number of TLB entry covers more addresses
- more likely TLB hits events

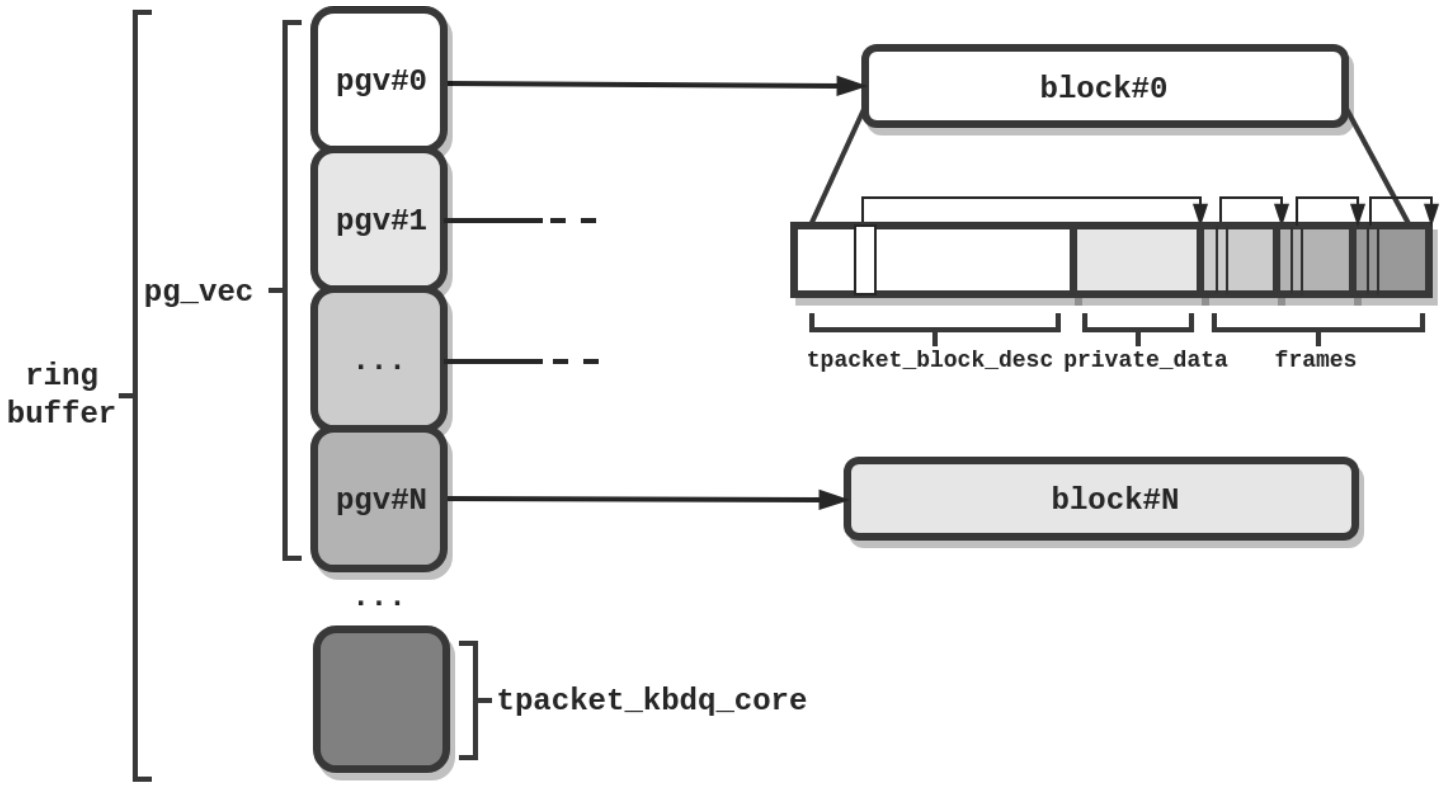
AF_PACKET kernel module

The `AF_PACKET` module implements the networking domain of the raw sockets:

- direct polling on the Ethernet Layer
- to avoid TCP/IP stack analysis overhead



Buffer implementation



Current buffer allocation policy

Attempts of block allocation in order of preference:

1. physically contiguous non-blocking memory reclaim
2. only virtually contiguous memory request
3. physically contiguous repeated memory reclaim

Proposed solutions

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*
   - HugeTLB filesystem

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*
   - HugeTLB filesystem
2. memory backed with *on-demand huge pages*

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*
   - HugeTLB filesystem
2. memory backed with *on-demand huge pages*
   - Transparent Huge Pages

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*
   - HugeTLB filesystem
2. memory backed with *on-demand huge pages*
   - Transparent Huge Pages
3. physically contiguous non-blocking memory reclaim

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*
   - HugeTLB filesystem
2. memory backed with *on-demand huge pages*
   - Transparent Huge Pages
3. physically contiguous non-blocking memory reclaim
4. only virtually contiguous memory request

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*
   - HugeTLB filesystem
2. memory backed with *on-demand huge pages*
   - Transparent Huge Pages
3. physically contiguous non-blocking memory reclaim
4. only virtually contiguous memory request
5. physically contiguous repeated memory reclaim

New block allocation policies

One allocation policy among the following can be forced:

1. memory backed with *pre-allocated huge pages*
   - HugeTLB filesystem
2. memory backed with *on-demand huge pages*
   - Transparent Huge Pages
3. physically contiguous non-blocking memory reclaim
4. only virtually contiguous memory request
5. physically contiguous repeated memory reclaim

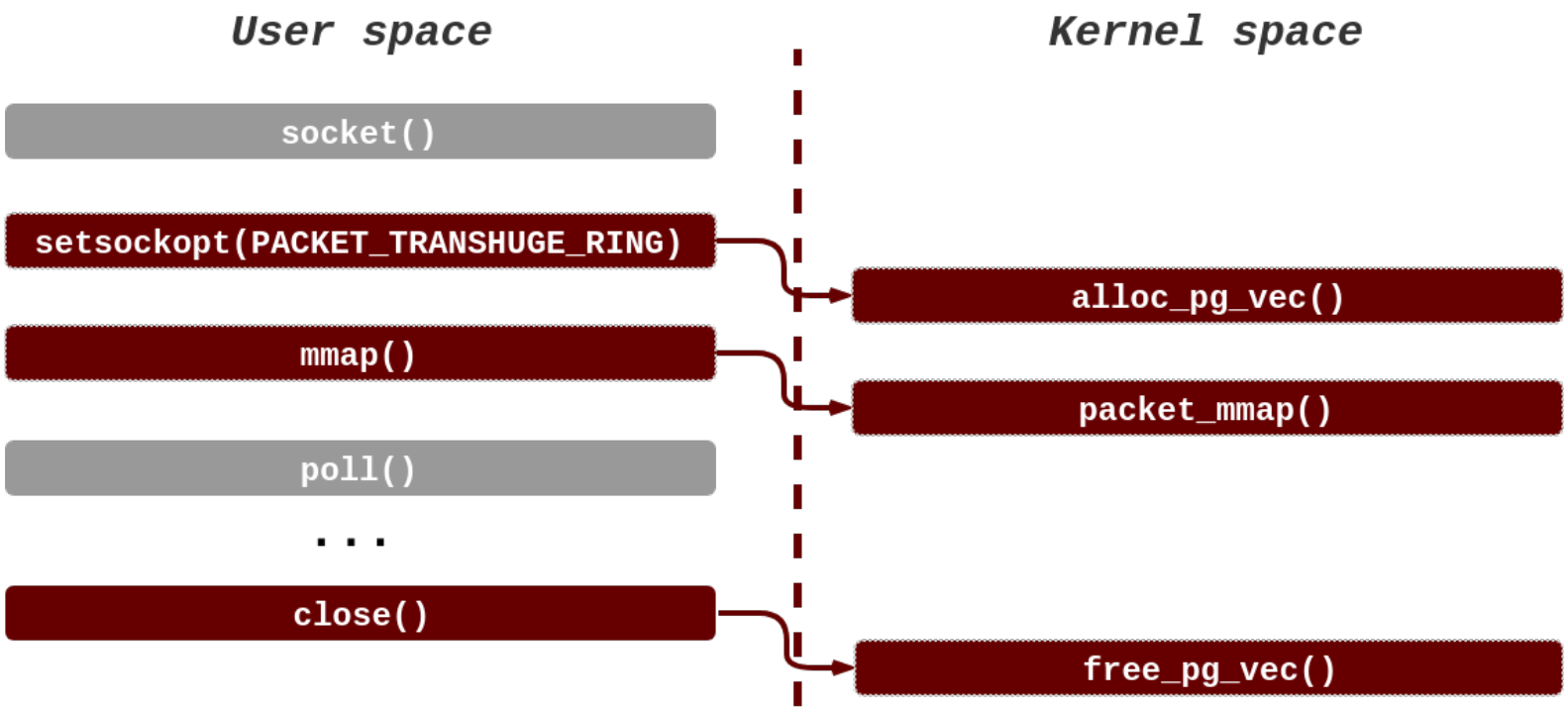
If not specified, the original allocation policy is performed.

Interface

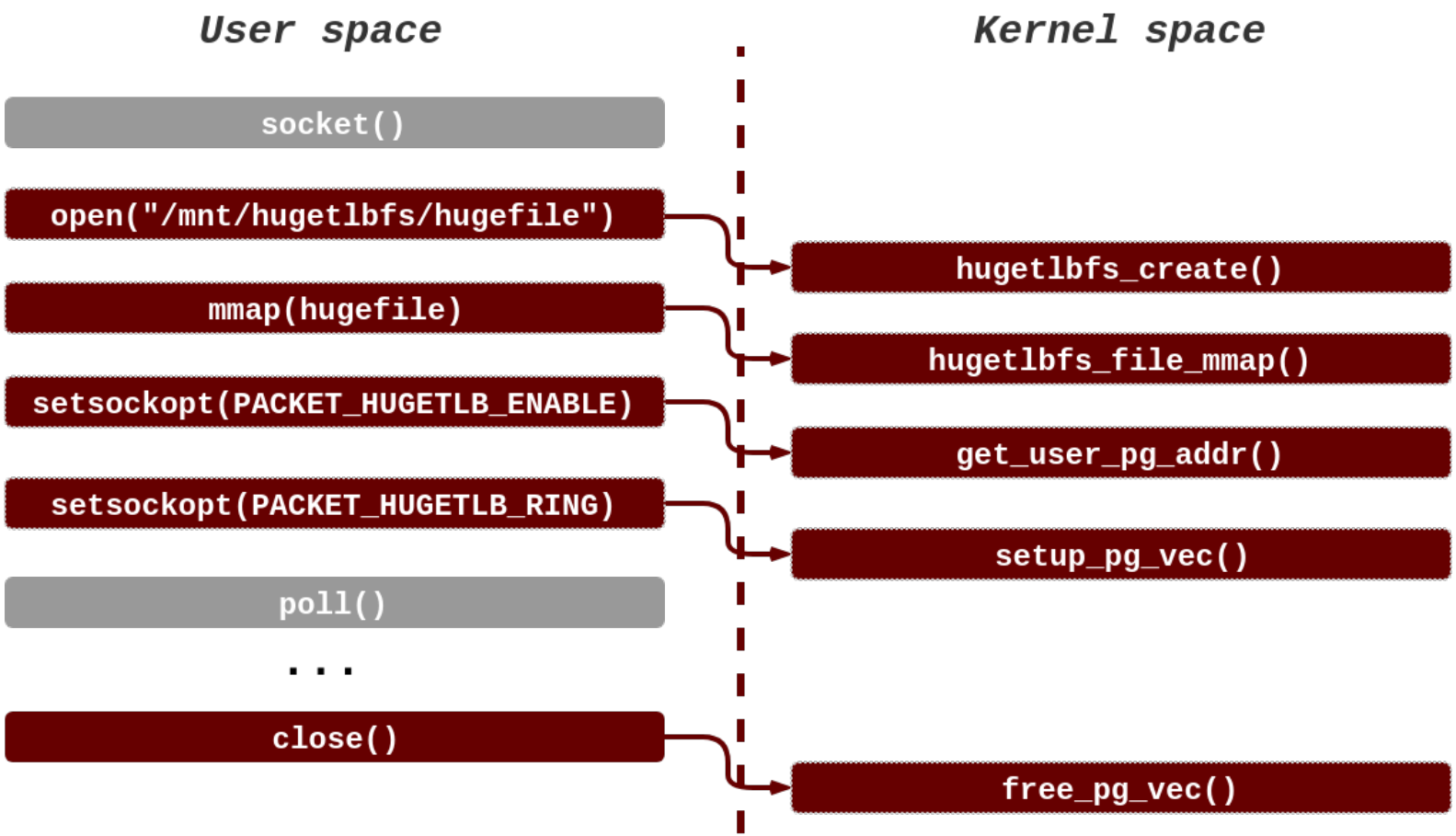
The solution is exported in user space as a socket configuration option.
Application awareness implies:

- easy testing activities
- to reach compatibility with existing implementations

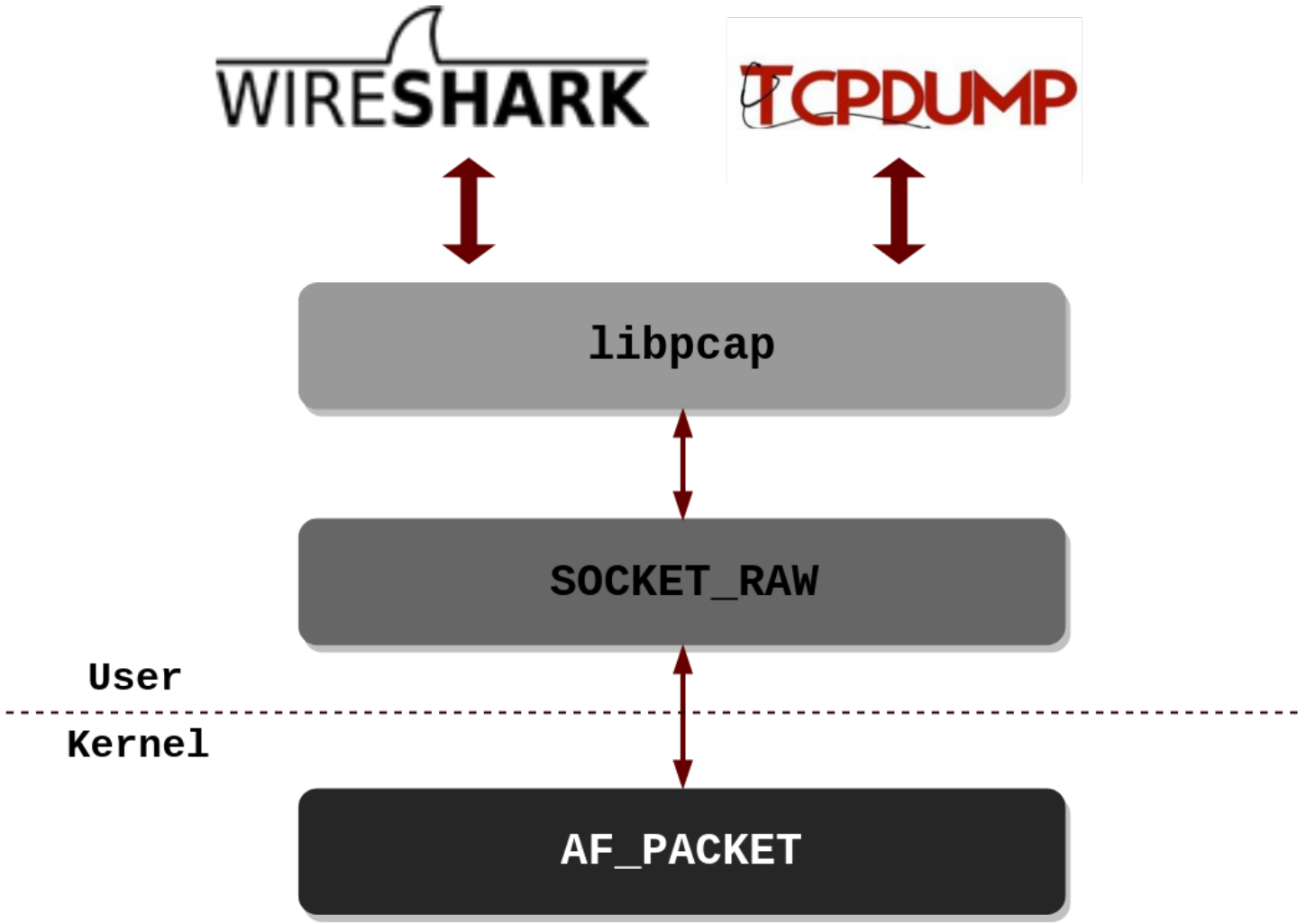
THP solution



HugeTLB filesystem solution



Integration with common applications



Integration with the `pcap` library

Making cascaded modifications to a library is a costly operation. → `LD_PRELOAD` trick

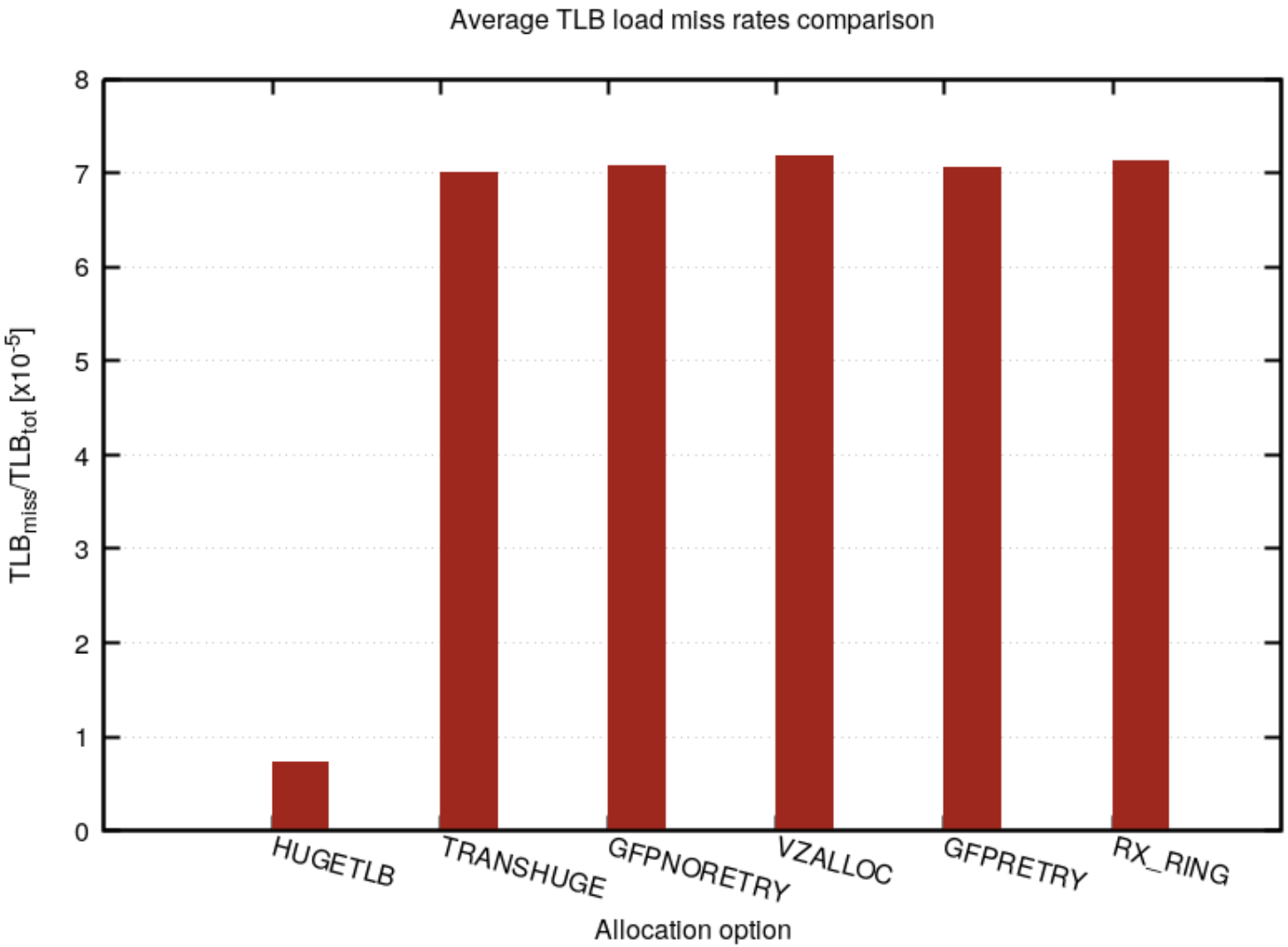
- exploiting library dynamic linking
- implementing alternative routines corresponding to the orginal symbols
- using environment variables to transfer not supported information

Results

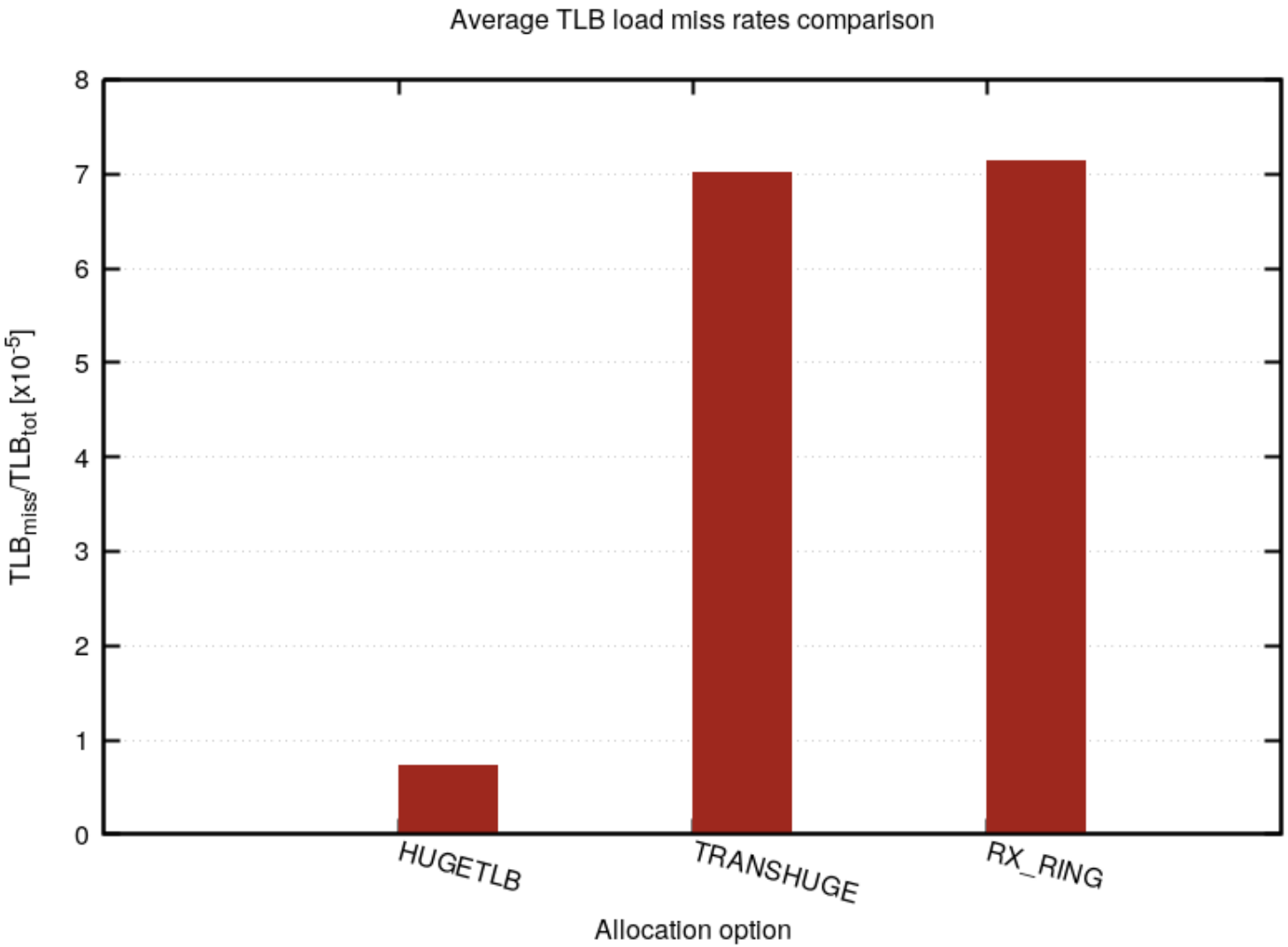
Test configuration

- generated synthethic randomized traffic
  - pktgen kernel tool
- traffic saved as a `pcap` trace
  - tcpdump tool
- `pcap` trace sent using tcpreplay tool
  - more instances to reach the desired rate
- rate at 10 Gb/s
  - nload tool

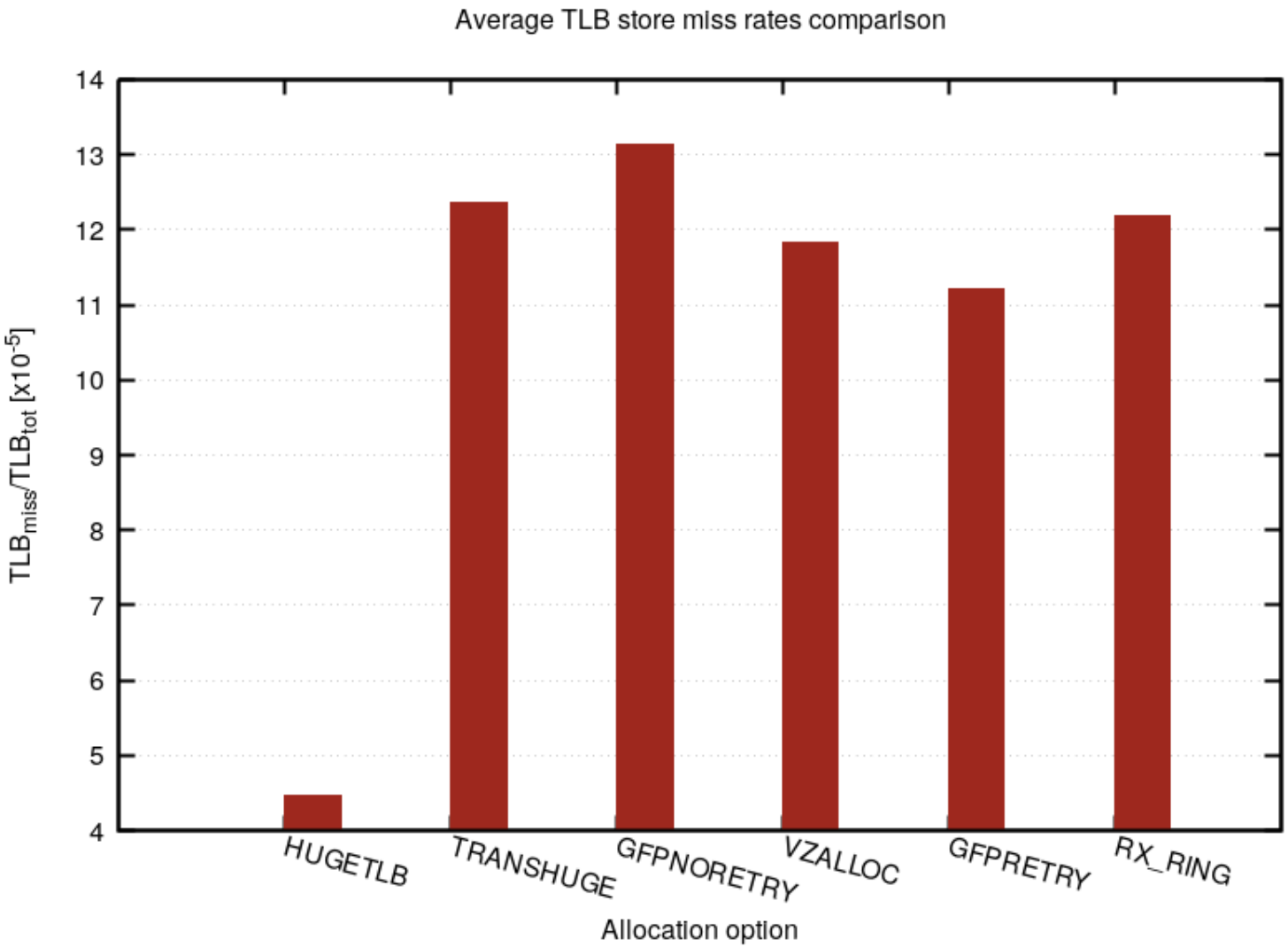
TLB load miss rates



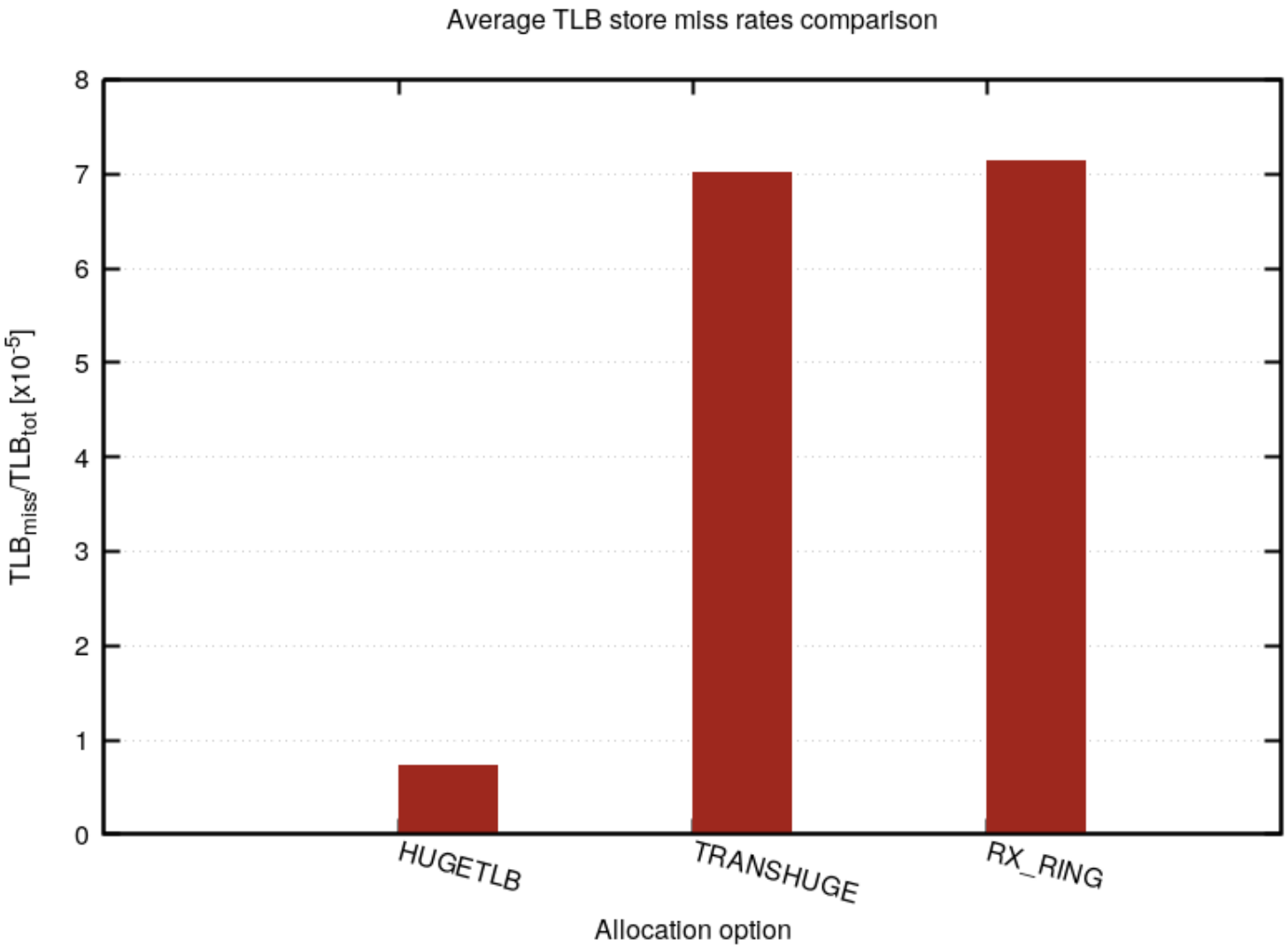
TLB load miss rates - main policies



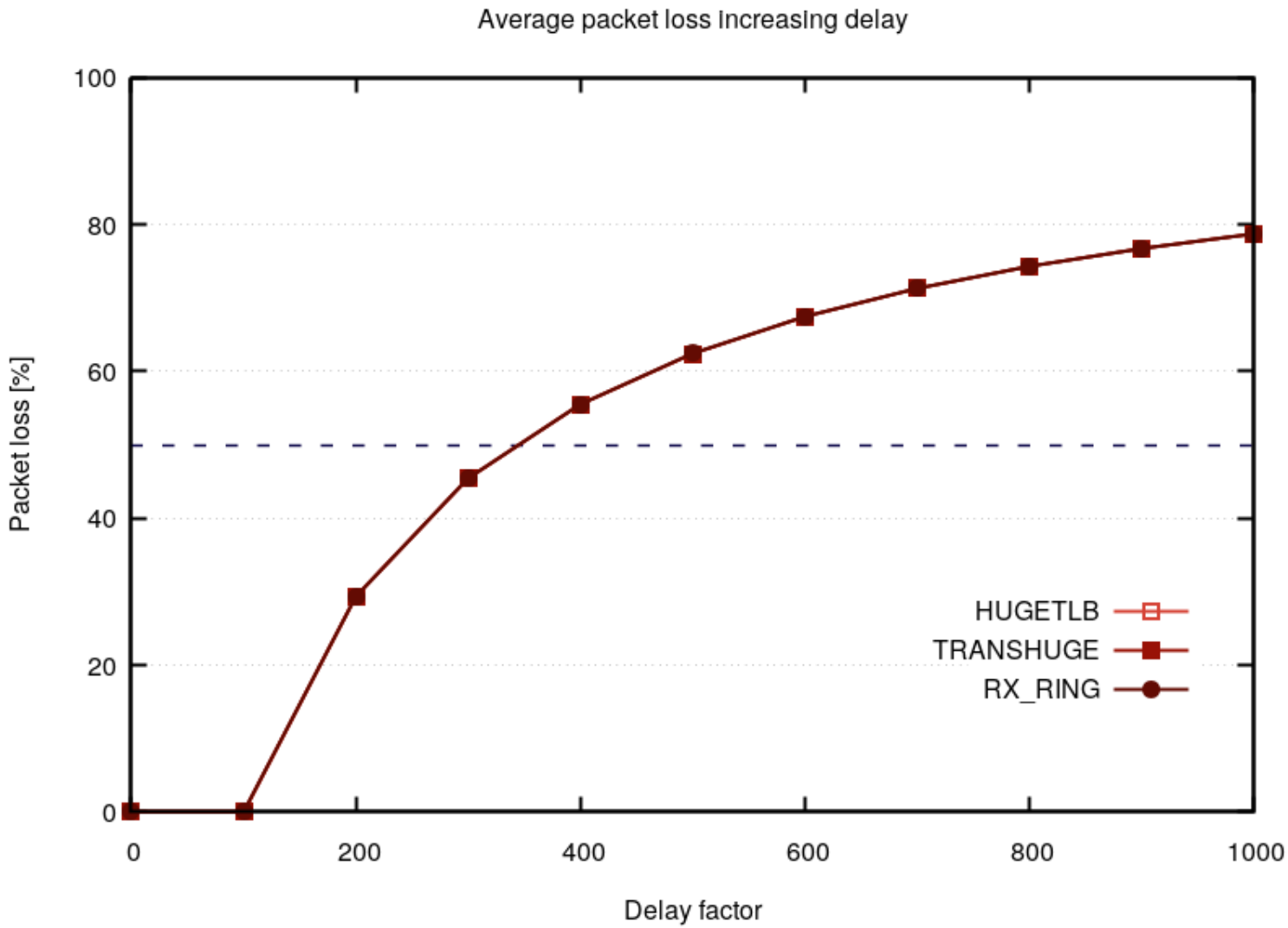
TLB store miss rates



TLB load miss rates - main policies



Packet loss



Conclusions

Conclusions

In conclusion, ...

Ideas for future works

- grouping all block allocation modes in only one option
- making allocation policies completely transparent to the user
  - all the allocation modes in order of preference (as original implementation)

Thank You for listening!

Laura Trivelloni