Collecting Packet Traces at High Speed

Universidad Pública de Navarra

Date: 28 September 2006

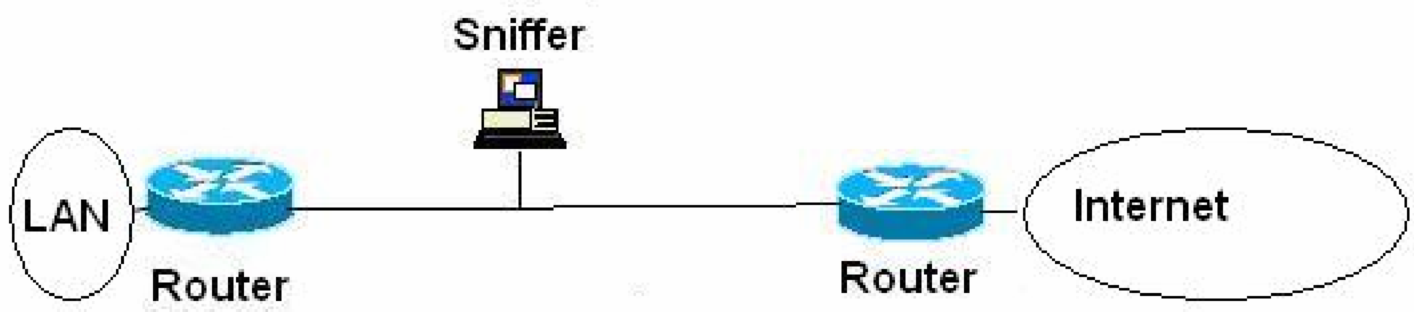
Gorka Aguirre Cascallana
Eduardo Magaña Lizarrondo

INDEX

- INTRODUCTION
- TECHNICS
- TESTBEDS
  - INTERRUPT COALESCENCE
  - NAPI
  - SHARED MEMORY (PF_RING)
- CONCLUSIONS
- FUTURE WORK

INTRODUCTION

- The purpose of this work is capturing packet traces at Gigabit Speed
- Low featured CPU
- Linux system
- 2 Gigabit Ethernet NIC test



PROBLEMS

- Operating Systems are not designed for such a High Speed traffic:
  - Operating System is usually interrupt based
  - System locks and inestability due to excessive interrupts handling
  - Packet loss
  - Packet Transmission Malfunction

TECHNICS

- OPERATING SYSTEM LEVEL:
  - Interrupt Mitigation
  - Napi
  - Shared Memory
- HARDWARE LEVEL:
  - Scatter and Gather
  - Checksum Offload
  - Data Alignment
  - Packet fragmentation
  - Jumbo Frames

OPERATING SYSTEM LEVEL

- Interrupt Mitigation:
  - Reduces the number of interrupts generating a single interrupt for a cluster of packets
  - NIC's driver parameter tune Interrupt Mitigation behaviour
  - Interrupt Coalescence is an interesting parameter, which changes automatically the number of interrupts per packets according to traffic workload

OPERATING SYSTEM LEVEL (II)

- NAPI (Polling)
  - NAPI is the new network system for Linux
  - Both Interrupt Mitigation and Polling are used
  - NAPI begins using Interrupt Mitigation and when receive livelock is detected Polling is activated
  - Interrupt Mitigation can be modified by NIC’s driver parameters

OPERATING SYSTEM LEVEL (III)

- Shared Memory
  - Types of Memory:
    - **Kernel**
    - **User**
  - Whenever a packet is received:
    - The packet is copied to kernel memory
    - It is processed by the protocol stack and sent to socket struct
    - The processed packet is copied to user memory so an application can handle it
  - For each packet 2 copies are made
  - A shared memory would allow working with 1 copy

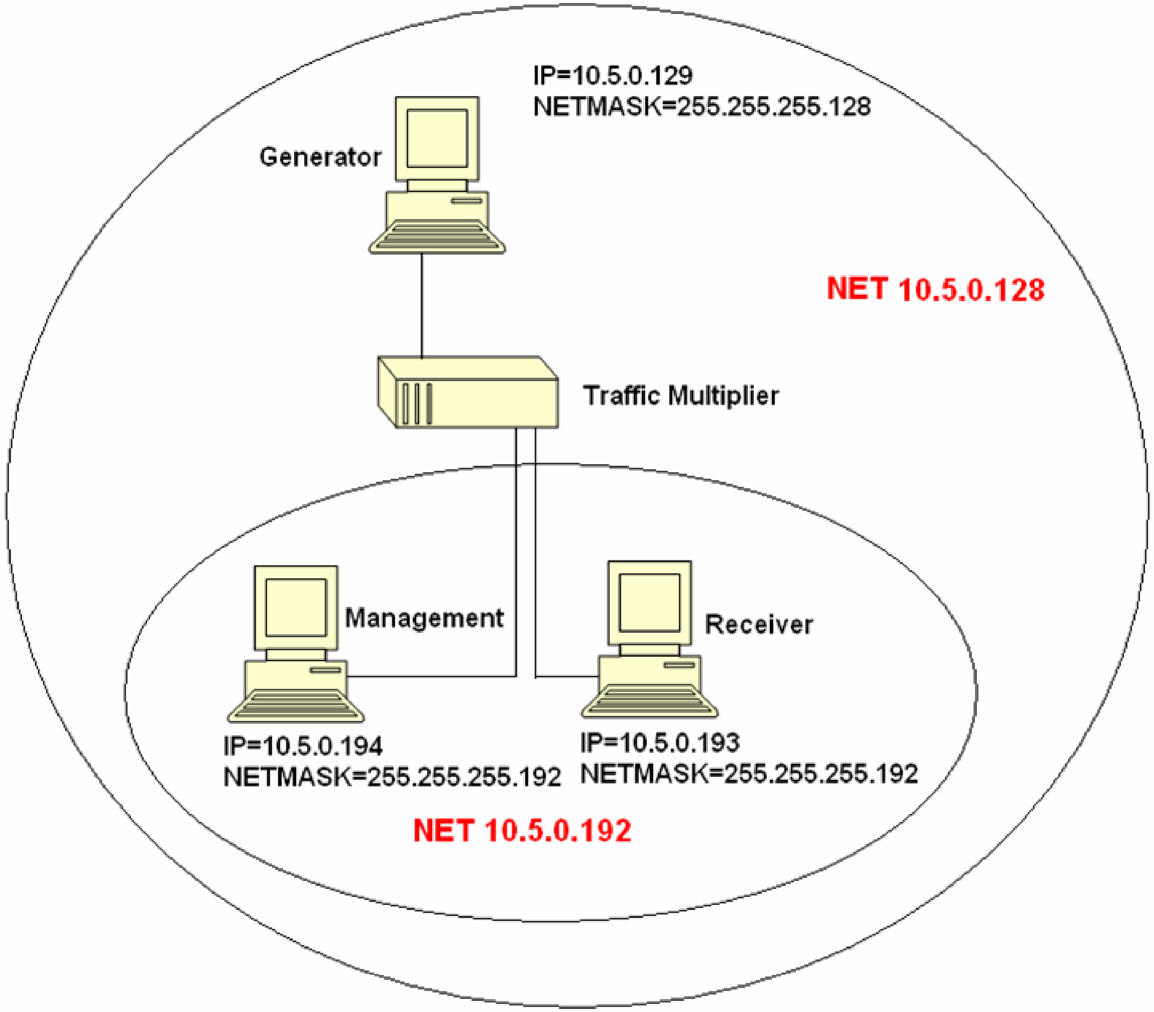
HARDWARE LEVEL

- **Scatter and Gather: W**rite and read from non related (non contiguous) memory addresses
- **Checksum Offload:** TCP / UDP / IP protocol Checksums at NIC hardware level
- **Data alignment:**
- **Packet Fragmentation:** This functionality is done at NIC Hardware level
- **Jumbo Frames:** Packet size > 1500 bytes

TESTBED

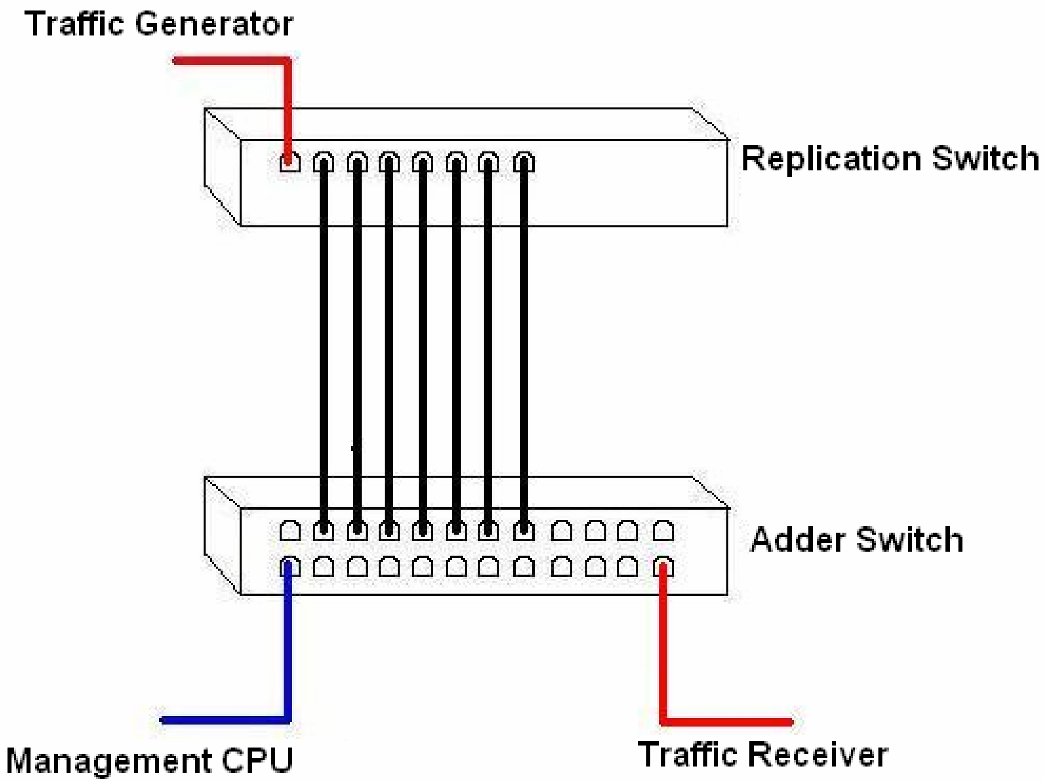
- Composed by 2 main part:
  - **Traffic Generator:** Flood the reception system
  - **Receiver:** High featured CPU with the NIC that is going to be tested

TESTBED



- 2 network:
  - 1 Isolated from the generator
  - 1 which can handle all CPUs
- Traffic Multiplier

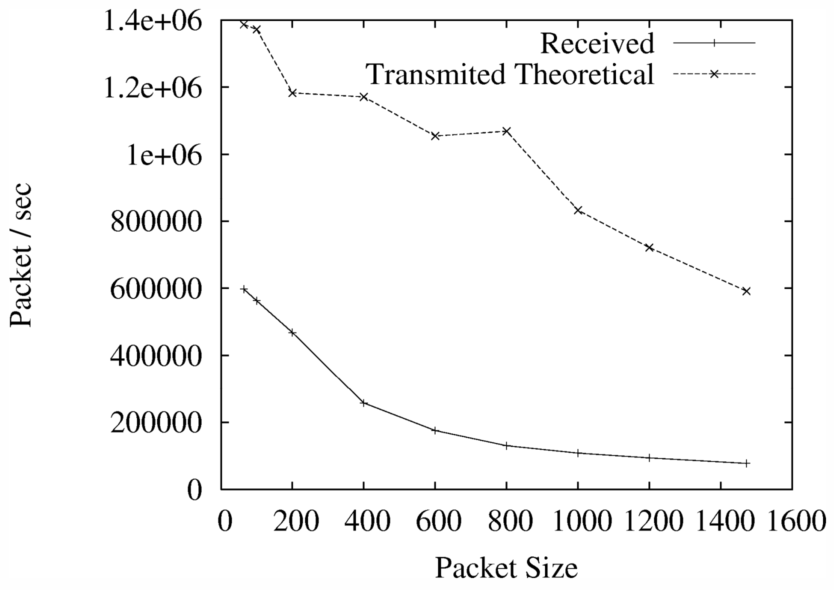
TRAFFIC MULTIPLIER

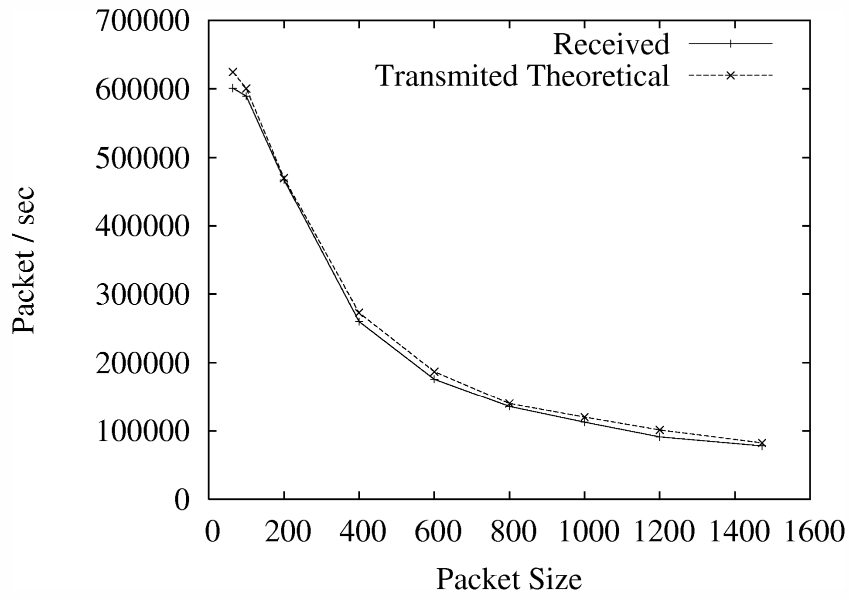


- Replication Switch :
  - Traffic from generator is sent to all ports
  - None of the ports has a MAC Address Stored
- Adder Switch :
  - Reception port needs to fix the MAC Address to add the received traffic
  - Periodical pings between Managment and Reception CPU
  - Feedback traffic to transmition network is forbidden

TRAFFIC MULTIPLIER

- Generated traffic = N x Transmission traffic
- 1 Gigabit per sec is the Teotherical Maximum traffic that a switch can afford
- Flow Control Parameter





COMBINATION PARAMETERS

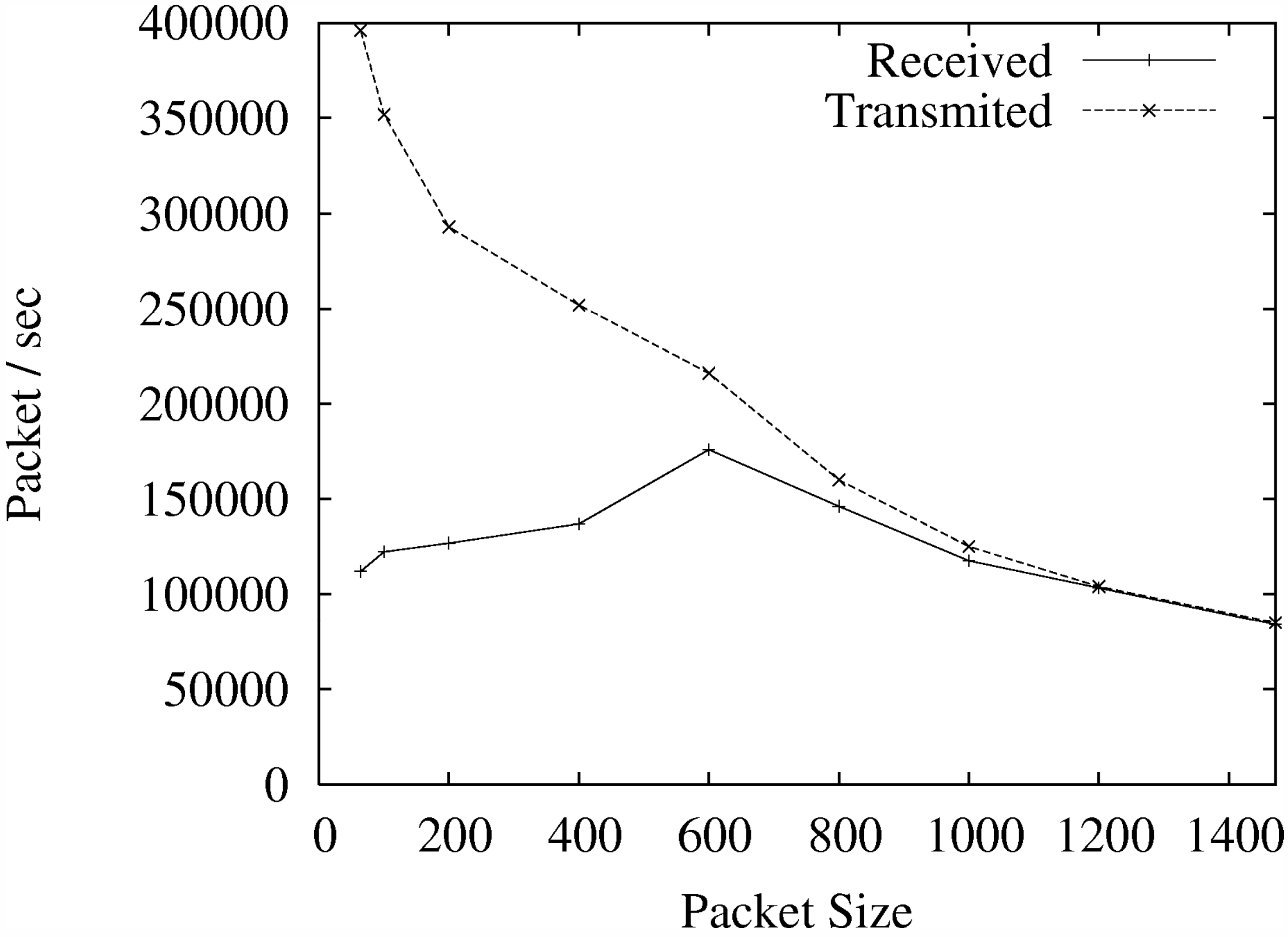
- BCM5700 parameters to change Interrupt Mitigation behaviour are:
  - **rx_std_desc_cnt**: Configures the number of receive descriptors on the kernel memory for frames up to 1528 bytes
    rx_max_coalesce_frames: Configures the number of received frames before the NIC generates receive interrupt
  - **rx_coalesce_ticks**: Configures the number of 1 usec ticks before the NIC generates receive interrupt after receiving a frame
  - **adaptive_coalesce**: Makes adaptive adjustments to the various interrupt coalescing parameters
  - **auto_flow_control**: Enables or disables autonegotiation of flow control

TEST I
(INTERRUPT MITIGATION)

- First Combination
  - Best choice for this system:
    - rx_max_coalesce_frames=0
    - rx_coalesce_ticks=10
    - rx_std_desc_cnt=500
  - Adaptive coalescing gets less received packets, but it reduces the number of packer losses at kernel level (losses take place at NIC level).
  - If rx_max_coalesce_frames > 0, then in a flood mode the system is unstable

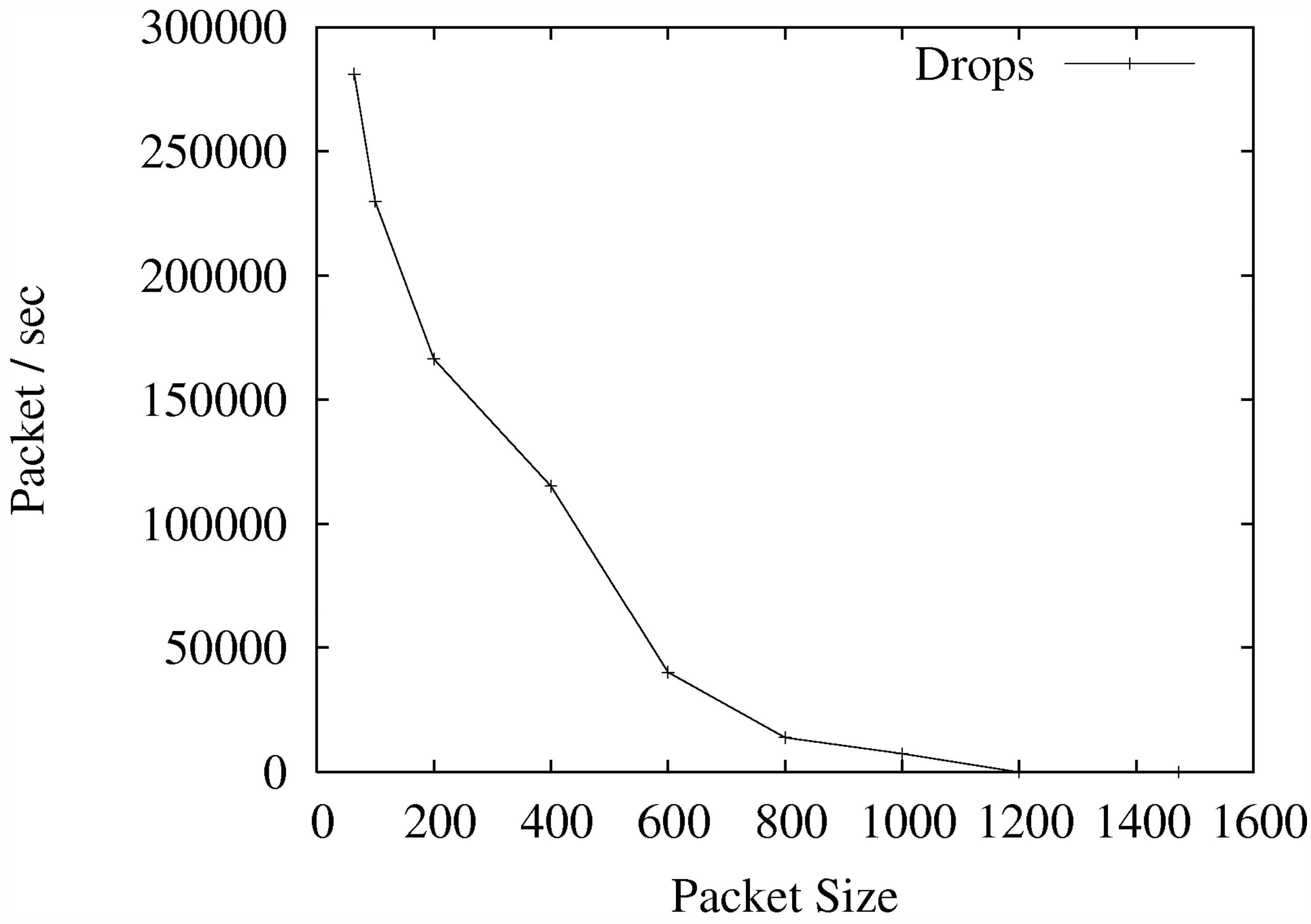
TEST I (II)
(INTERRUPT MITIGATION)

- Number of packets received per sec with the parameters above



TEST I (III) (INTERRUPT MITIGATION)

- Dropped Packets per sec

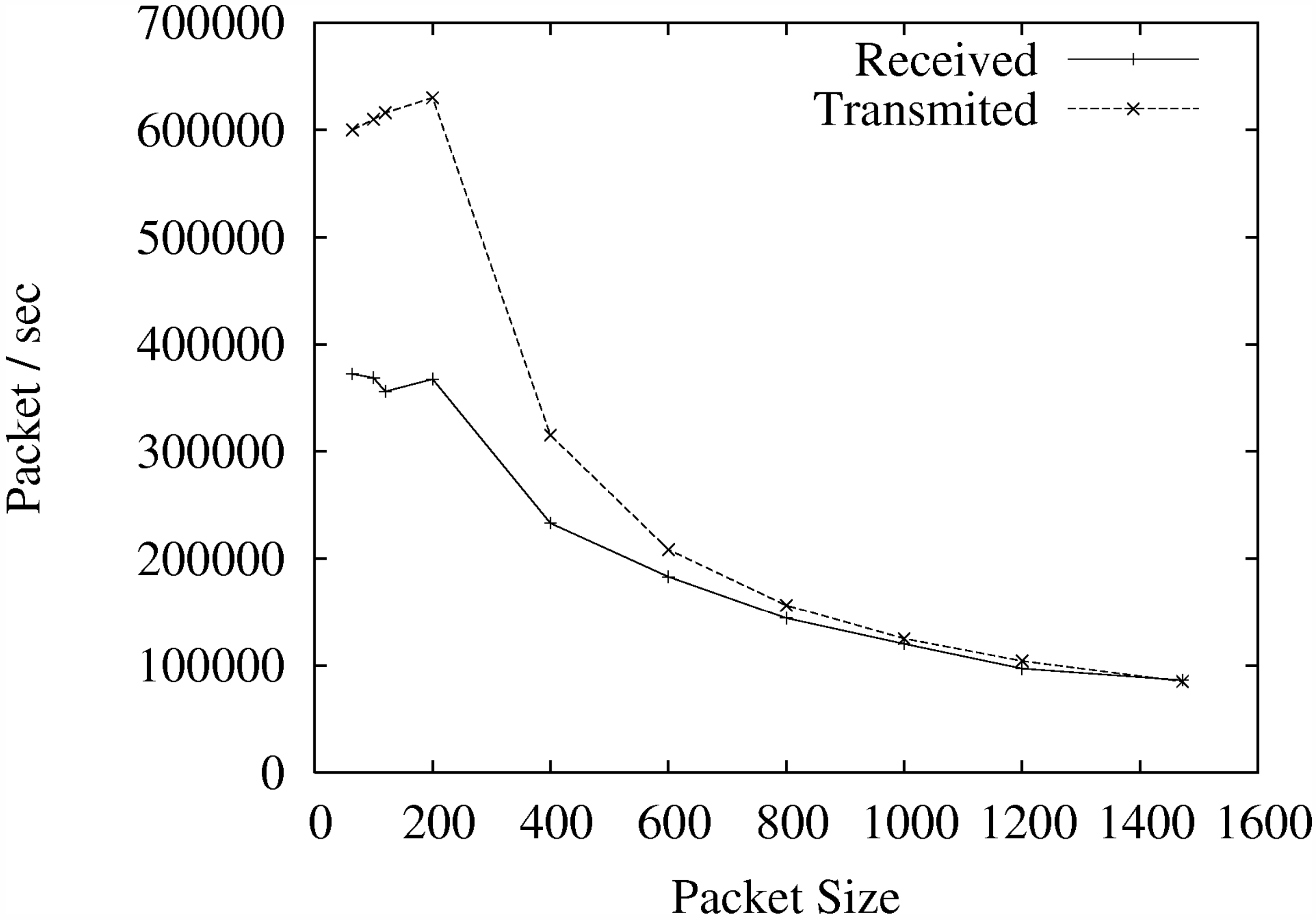


TEST II (NAPI)

- Best choice for NAPI system:
  - The usage of rx_coalesce_ticks parameter obtains a flat response
    - rx_max_coalesce_frames=0
    - rx_coalesce_ticks=50
    - rx_std_desc_cnt=200
  - NAPI system has no drops into kernel memory but has drops into NIC hardware memory

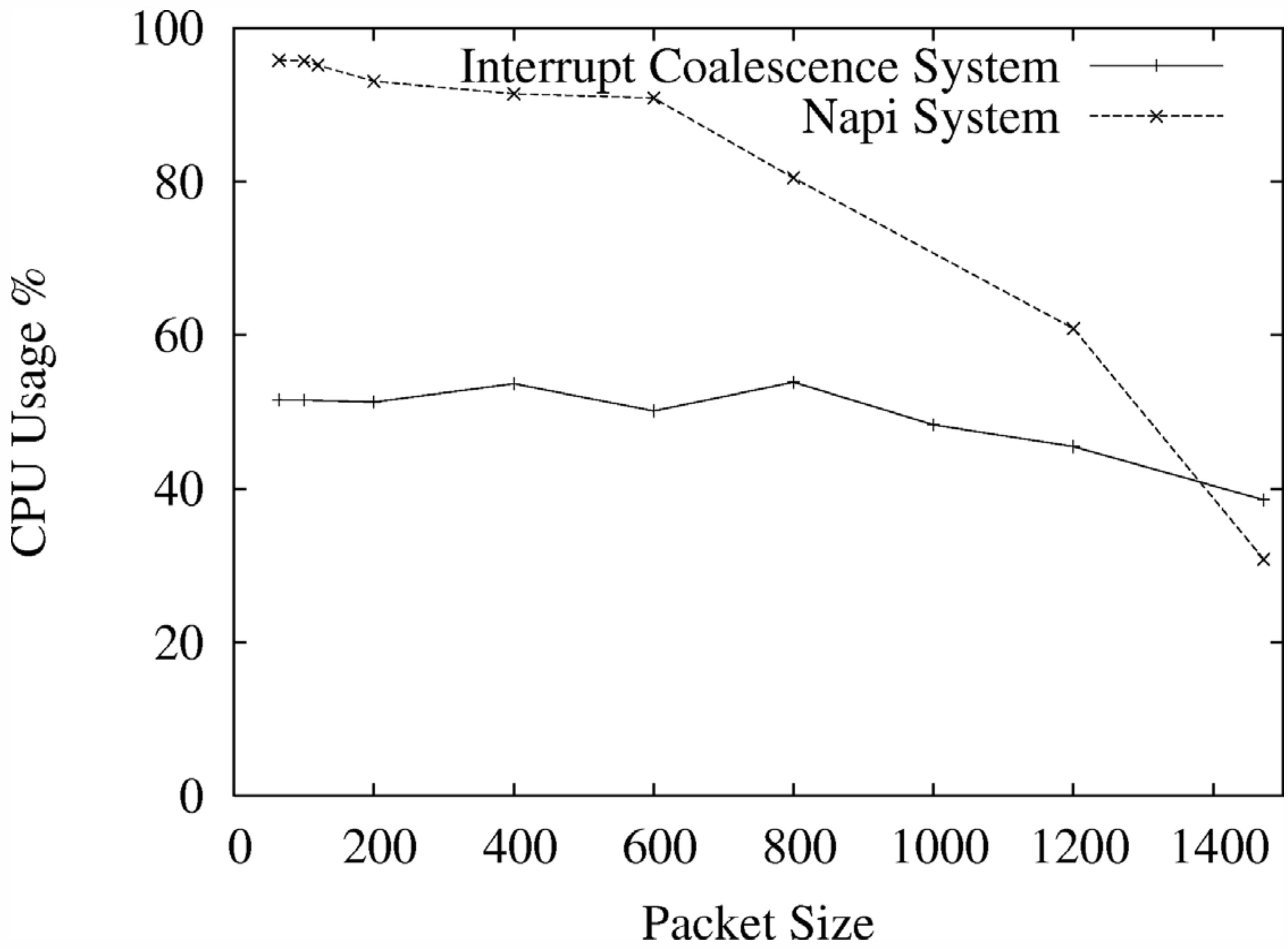
TEST II (NAPI) (II)

- Number of packets received per sec with the parameters above



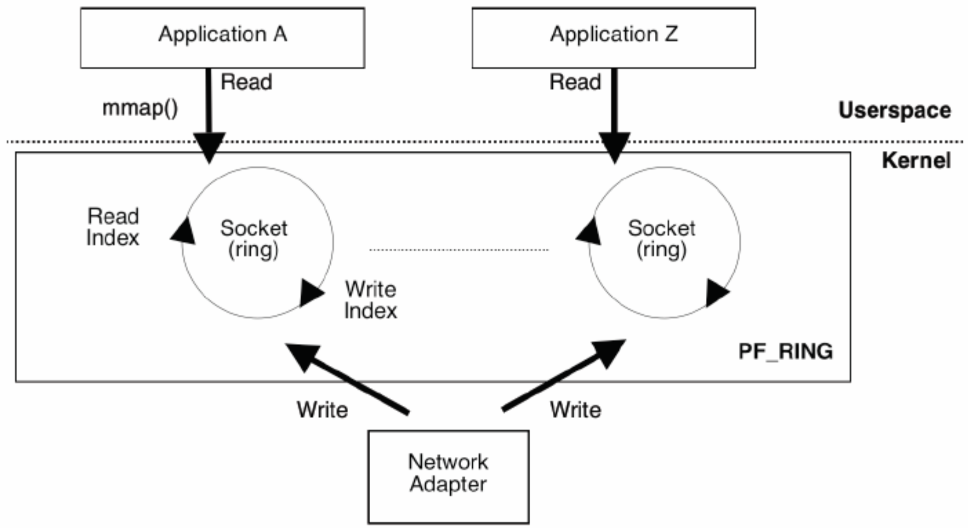
CPU USAGE COMPARISON

- Napi system CPU Usage is better because the polling strategy



TEST III (SHARED MEMORY)

- To carry out this test a module called PF_RING was used:
  - New socket allocates a shared buffer memory
  - Protocol Stack is avoided -> Interrupt Mitigation
  - Works on Libpcap based applications

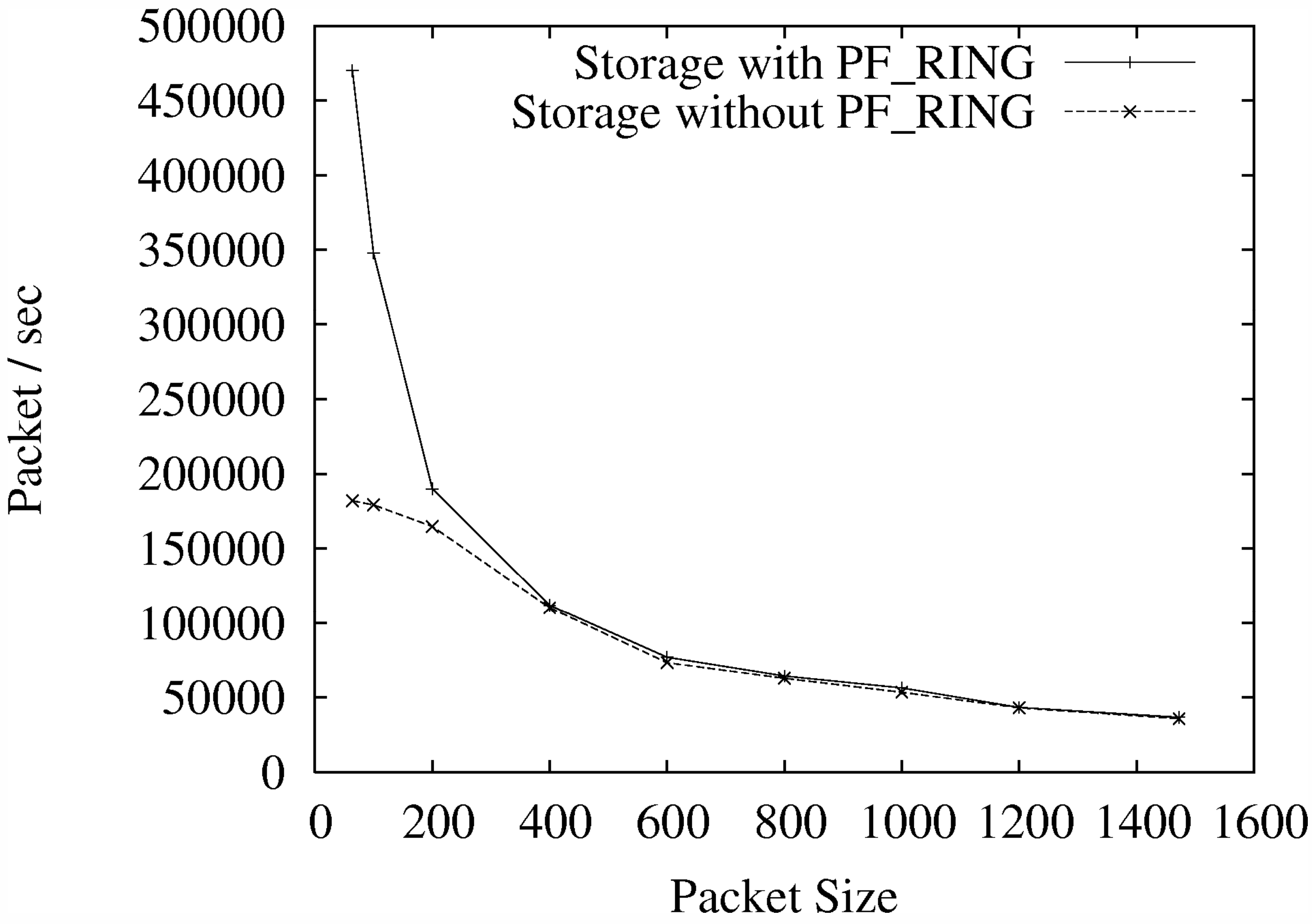


TEST III (SHARED MEMORY) II

- PF_RING parameters:
  - **Bucket_len**: Specifies the slot size of the buffer
  - **Num_slots**: Number of slots the buffer consists of
  - **Sample_rate**: Sample capabilites on received packets
  - **Transparent_mode**: Received packets are processed by protocol stack
- Data storage on the Hard Disk:
  - Tcpdump (designed with libpcap library) is our choice to store packets on the hard disk for a later processing
  - 2 sorts of storage:
    - Entire Packet:
      - Hard Disk transfer rate is a bottleneck
    - Partial Storage:
      - First 60 bytes of each packet will be stored for later statistics

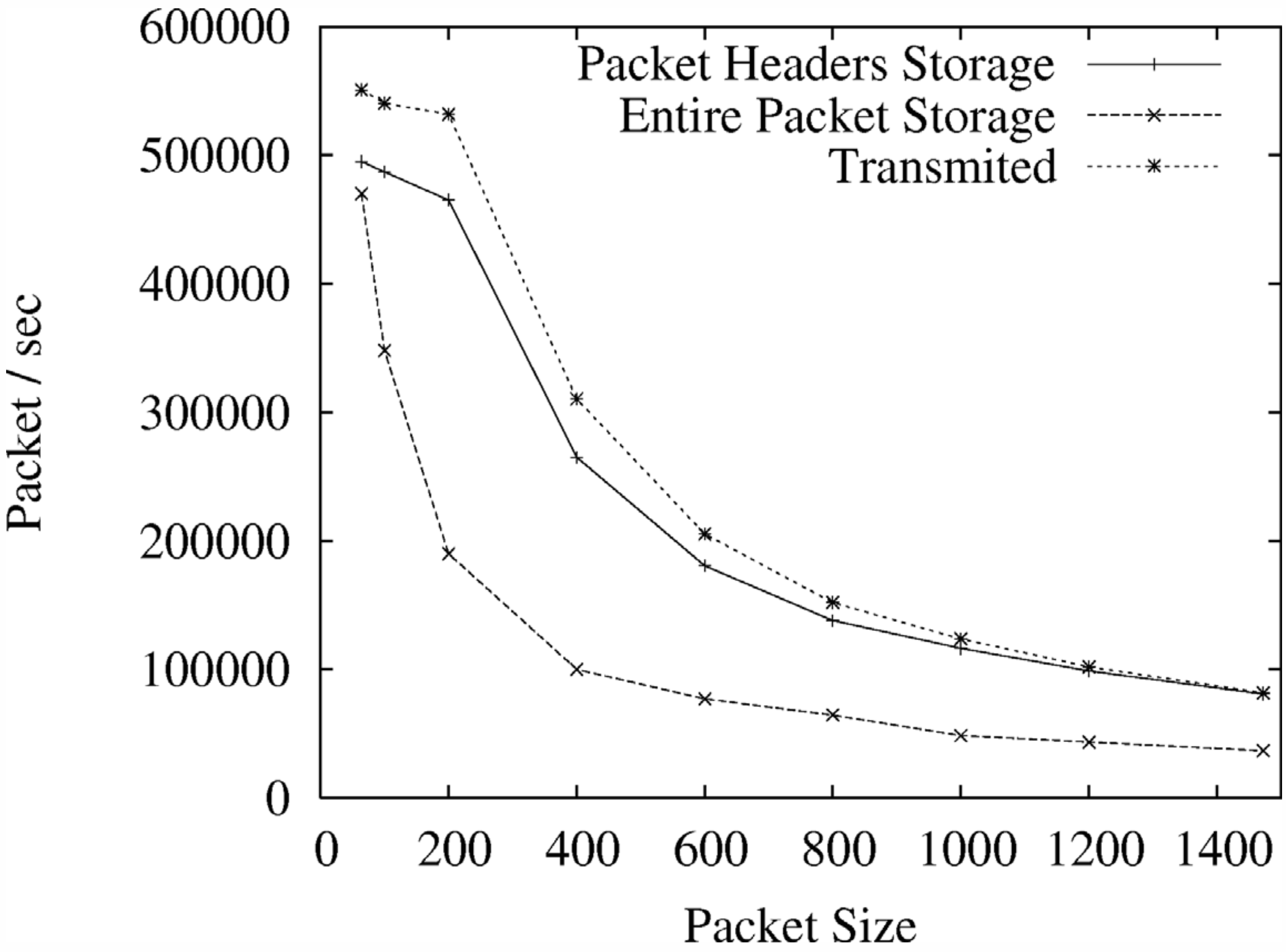
TEST III (PF_RING) IV

- Entire packet storage in the Hard Disk both with and without PF_RING module



TEST III (PF_RING) V

- Partial and Entire packet storage on the hard disk



CONCLUSIONS

- Interrupt mitigation obtains a high outcome of received packets but it drops most of them.

- NAPI system obtains a higher outcome. Packets are dropped on the NIC memory not in the kernel memory. When Polling is activated the number of interrupts drop off to 0.

- Each packet needs 2 copies for a further processing. The PF_RING module is used to test a memory shared strategy which needs just 1 copy for each packet to obtain a packet into the user memory. This strategy is necessary for packet storage via an application.

- Hard Disk data transfer rate is a bottleneck for entire packet data storage. However, partial packet storage obtains an excelent outcome without data loss on kernel – user memory.

FUTURE WORK

- PF_RING modification in order to a direct packet storage in the hard disk. No libpcap library would be necessary
- Further tests with Phil Wood Libpcap, which uses Shared Memory
- Packet data compression at memory

Collecting Packet Traces at High Speed

Universidad Pública de Navarra

Date: 28 September 2006

Gorka Aguirre Cascallana
Eduardo Magaña Lizarrondo