# WireCAP: a Novel Packet Capture Engine for Commodity NICs in High-speed Networks

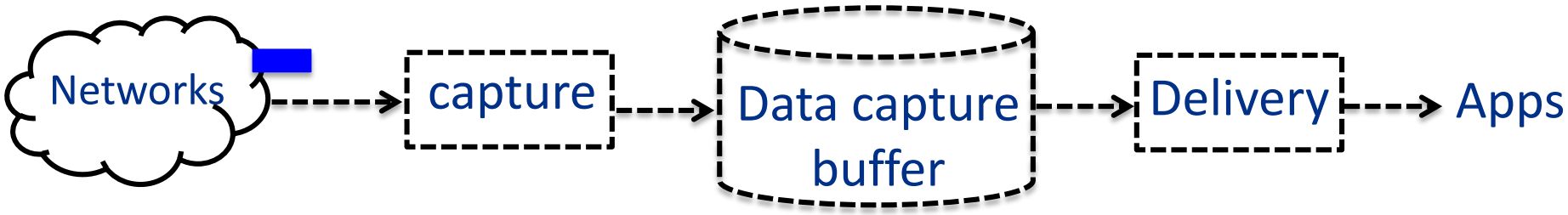
**Wenji Wu, Phil DeMar**
**Fermilab Network Research Group**
wenji@fnal.gov, demar@fnal.gov

**ACM IMC 2014**
**November 5-7, 2014 Vancouver, BC, Canada**

# Packet Capture is an essential function for many network applications

- Security analysis (e.g., IDS/IPS)
  - Snort, www.snort.org
  - Suricata, http://suricata-ids.org
- Network and application performance analysis
  - Riverbed SteelCentral NetProfiler
  - Netscout Sniffer Analysis
  - Wildpackets OmniPeek Network Analyzer
- Traffic characterization studies
  - Benson, IMC'10
  - And more

# A general packet capture process

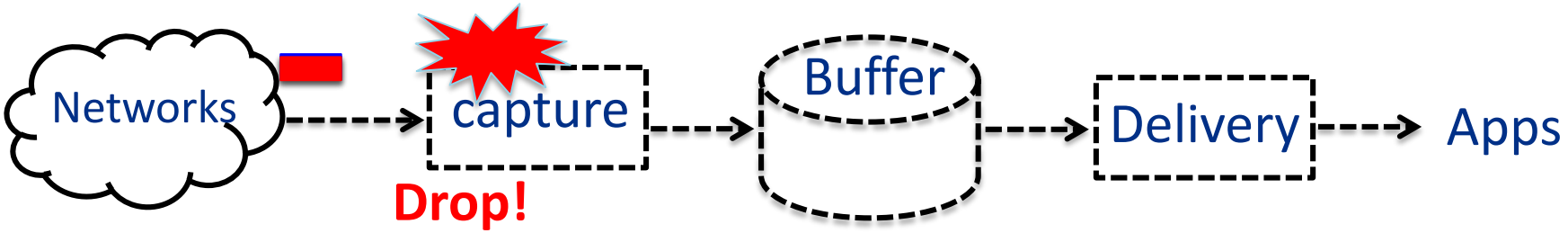



**Typically computationally and I/O throughput intensive**

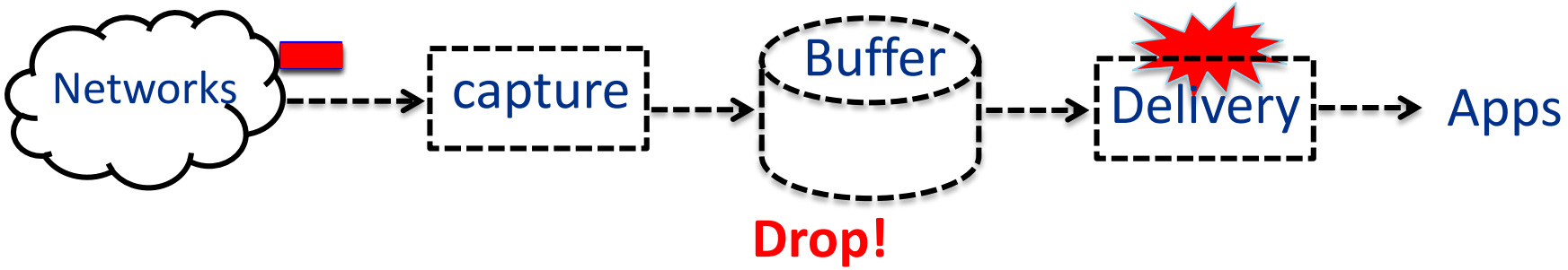
**Significant performance challenges in high-speed networks**

# Packet drop is a major problem with packet capture in high-speed networks

**Type I packet drop: packet capture drops**




**Type II packet drop: packet delivery drops**




**Packet drops degrade the accuracy & integrity of network monitoring applications!**

**Fundamental design goal in packet capture tools: Avoid packet drops!**

# Packet capture approaches

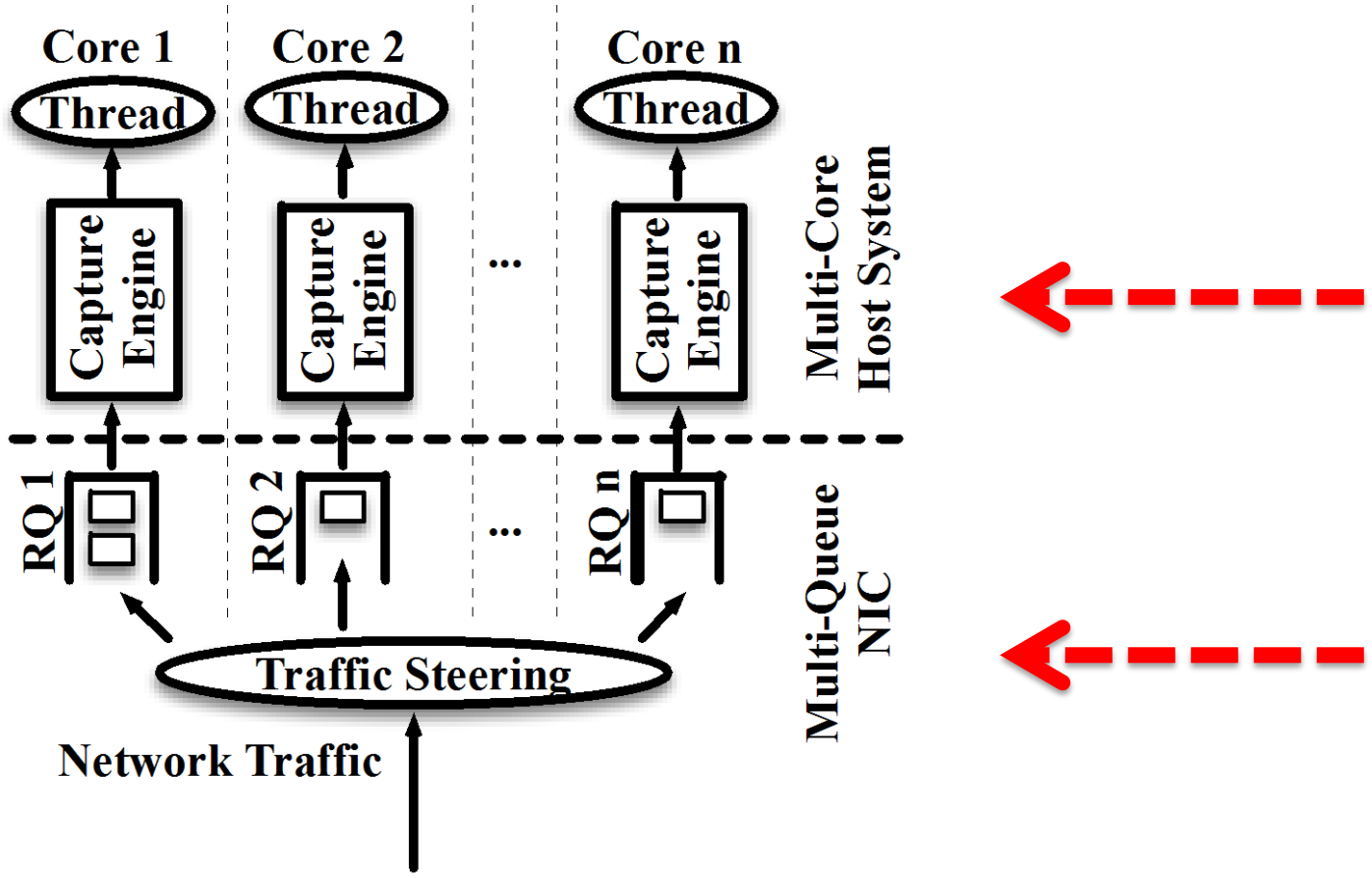
1. Use dedicated packet capture card
   - Pros
     - The least amount of CPU intervention
     - Lossless packet capture and delivery
   - Cons: costly, relatively inflexible, and not very scalable

2. Use a commodity system with a commodity NIC
   - A commodity NIC in promiscuous mode to intercept pkts
   - A capture engine provides capture and delivery services
   - Pros: flexible and cost effective
   - Cons: significant system CPU and memory resources

**The 2nd approach becomes more appealing with recent advances in multicore systems and multi-queue NICs:**

- **A new paradigm in packet capturing and processing.**
- **This is our research focus.**

# A new packet capturing and processing paradigm



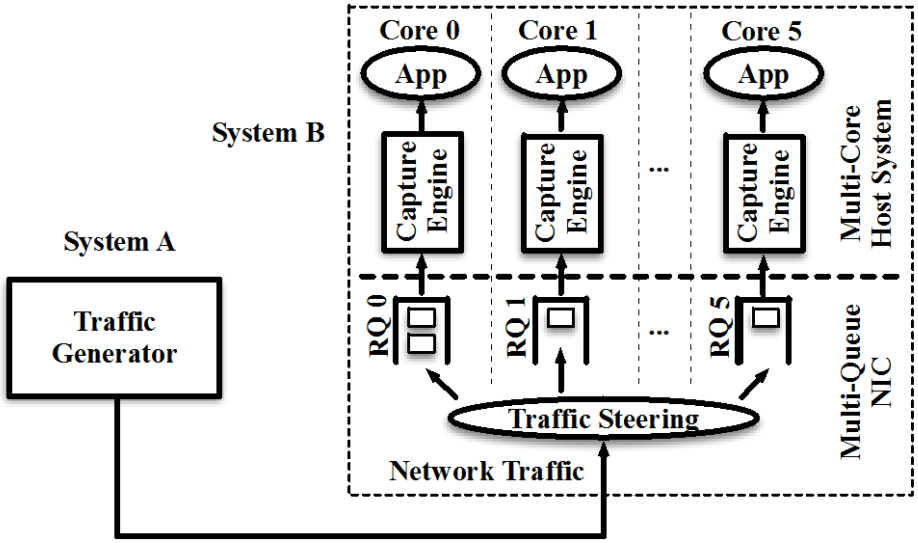

**Assumption: the hardware-based traffic-steering mechanism is capable of evenly distributing the incoming traffic among cores.**

## Question 1

**Can NIC hardware-based traffic-steering mechanisms evenly distribute the incoming traffic among cores?**

## Question 2

**Can existing packet capture engines support this new packet capture and processing paradigm?**



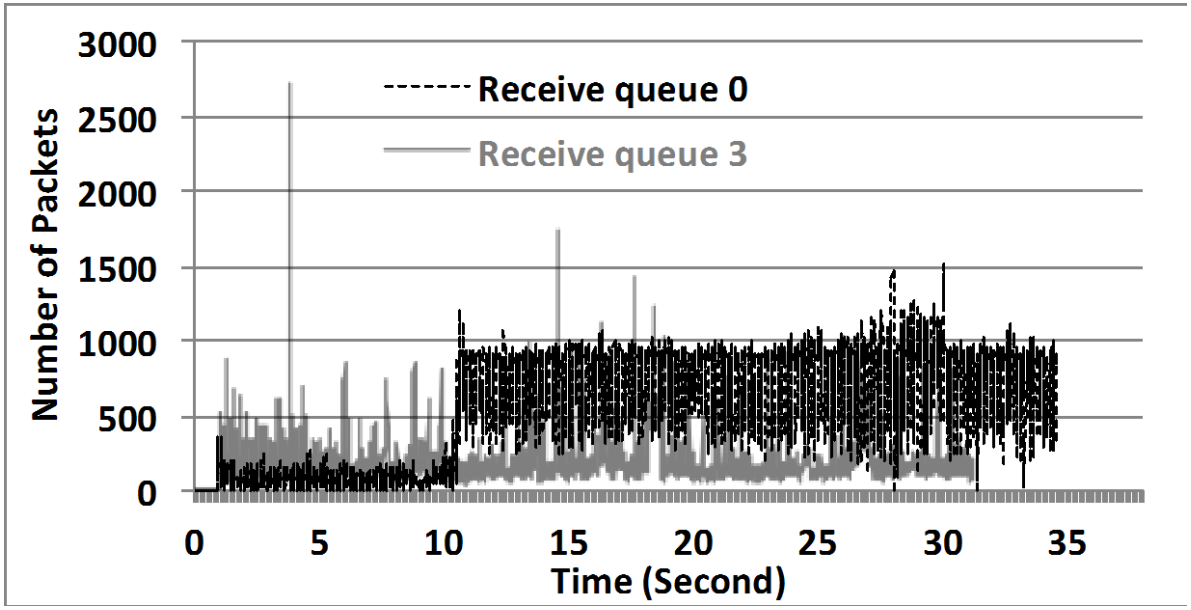

# Experiment Proof

**Intel 82599 10GE NIC**

- Traffic captured at FNAL border router used as experiment data
- System A runs as a traffic generator to replicate captured data
  - both content & spacing…
- On System B, two experiments are run:
  - Exp1. Profiling traffic from each receive queue
  - Exp2: A single-threaded application is run at each core to capture & process pkts from its associated receive queue
  - We evaluate existing packet capture engines.

    -- DNA , PF_RING (Luca et al) -- NETMAP (Luigi ATC'12)

# Observation 1: load imbalance occurs frequently in a multicore system




Two types of load imbalance

- Short-term load imbalance (short-term burst of packets)
- Long-term load imbalance (queue 0 receives much more traffic than queue 3)

**Existing NICs distribute traffic on a per-flow basis!**

# Observation 2: existing packet capture engines suffer significant packet drops

| | NETMAP | DNA | PF_RING |
|---|---|---|---|
| **Receive Queue 0** | | | |
| **Packet Capture Drops** | 46.5% | 50.1% | 0% |
| **Packet Delivery Drops** | 0% | 0% | 56.8% |
| **Receive Queue 3** | | | |
| **Packet Capture Drops** | 33.4% | 9.3% | 0.8% |
| **Packet Delivery Drops** | 0% | 0% | 0% |

**Packet drop rates with a heavy packet-processing load**

**Existing packet capturing engines suffer significant packet drops with load imbalance of either type!**

# Problems with existing packet capture engines

| Engines | Deficiencies | Can it handle Short-term load imbalance ? | Can it handle Long-term load imbalance? |
|---|---|---|---|
| PF_RING | Copying in packet capture<br>Receive livelock problem<br>No offloading mechanism | ✗ | ✗ |
| DNA | Limited buffering capability<br>No offloading mechanism | ✗ | ✗ |
| NETMAP | Limited buffering capability<br>No offloading mechanism | ✗ | ✗ |

# How to avoid packet drops that are caused by load imbalance?

1. To apply a round-robin traffic steering mechanism at NIC level for traffic distribution
   - Does not preserve application logic
   - Results in poor system performance

2. To use existing packet capture engines (e.g. DNA) and to address load imbalance in application layer
   - An application has little knowledge of low-layer conditions
   - Must involve copying
   - Making the application complex and difficult to design

3. **Our solution is to design a new packet capture engine to address load imbalance at packet-capture level**

# Our Solution

## WireCAP: a Novel Packet Capture Engine for Commodity NICs in High-speed Networks

# WireCAP Design Goals

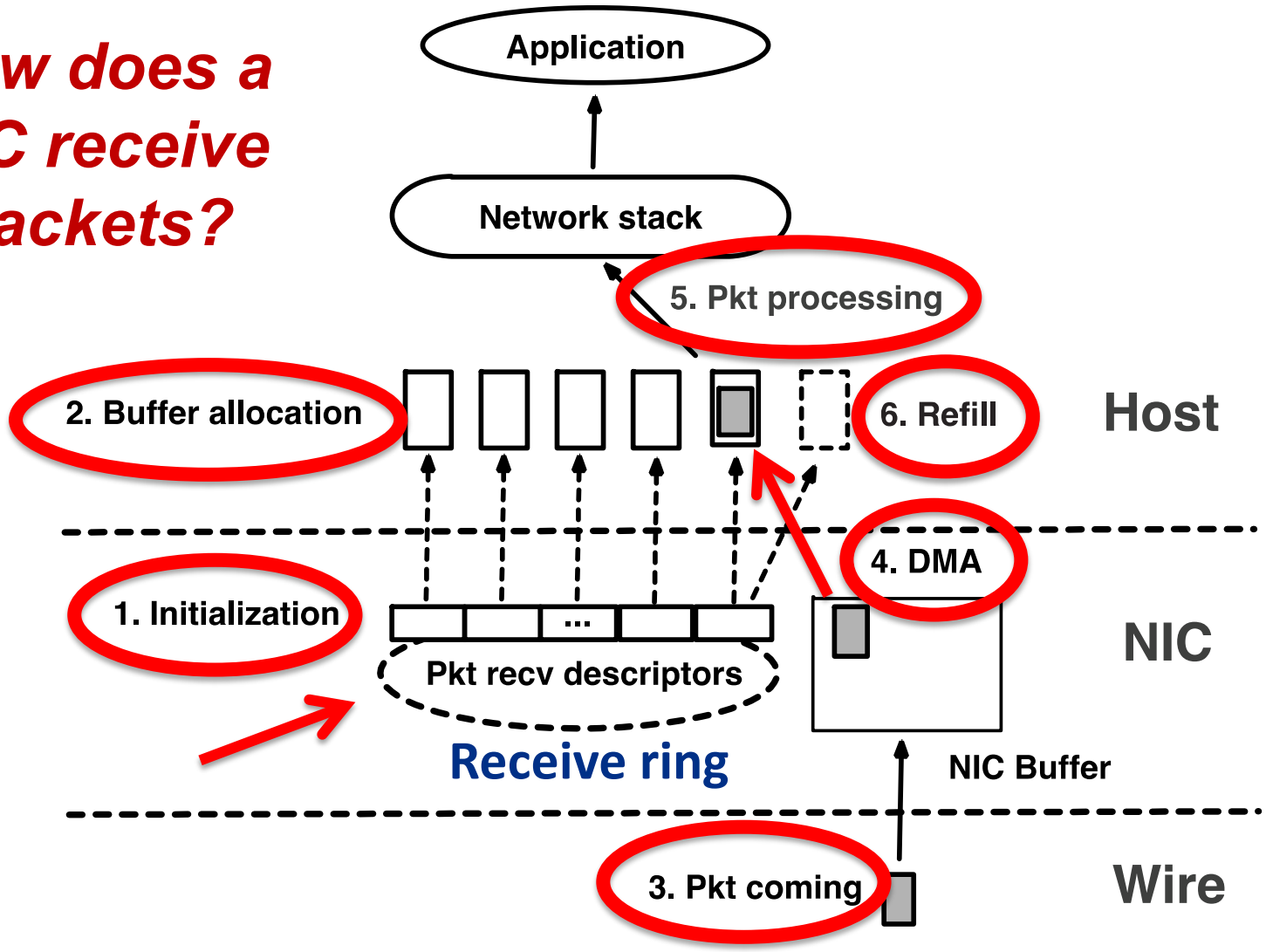
- Provide lossless packet-capture services in high-speed networks
- Provide efficient packet delivery
- Be efficient
- Facilitate design & operation of packet-processing applications in user space
- Wide applicability
- Support middlebox-type applications
  - Needs to implement a transmit function

# WireCAP Key Techniques

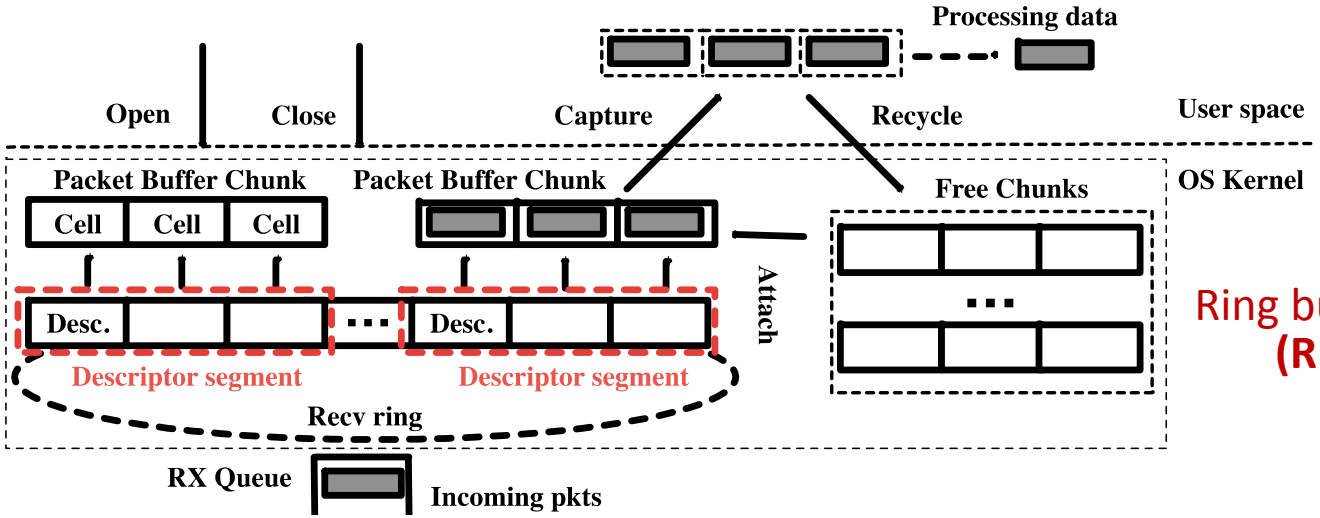
- The ring-buffer-pool mechanism
  - To handle short-term load imbalance
- The buddy-group-based offloading mechanism
  - To handle long-term load imbalance
- Optimization techniques
  - Pre-allocated large packet buffers
  - Zero-copy
  - Packet-level batching processing

# The ring-buffer pool mechanism:

***How does a NIC receive packets?***



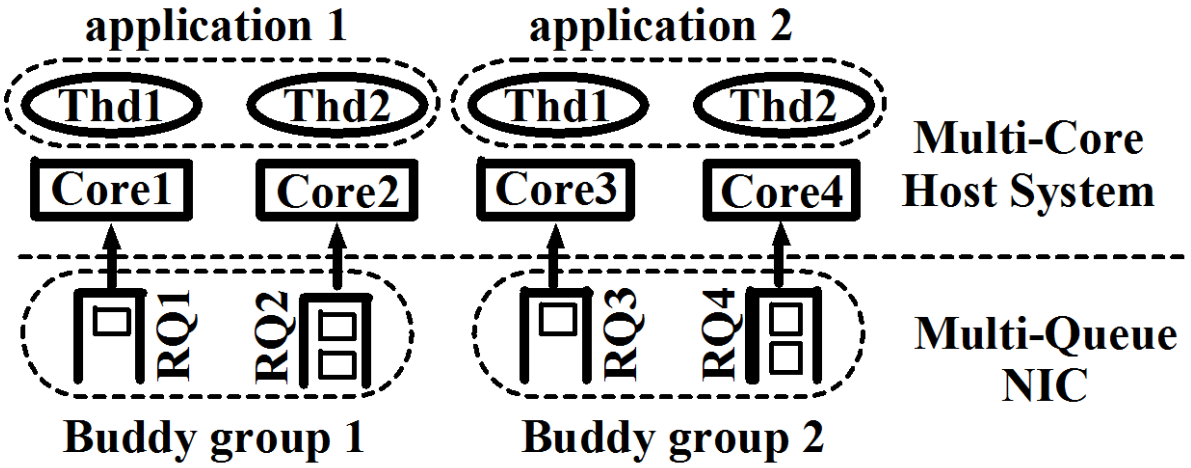
# The ring-buffer pool mechanism




- A receive ring is divided into descriptor segments
- Each segment consists of M receive pkt descriptors
- Ring is preallocated with R pkt buffer chunks (ie. ring buffer pool)
- A pkt buffer chunk consists of M packet buffers
- Each descriptor segment must be attached with an empty pkt buffer chunk to receive pkts
- Supports four operations through ioctl interface

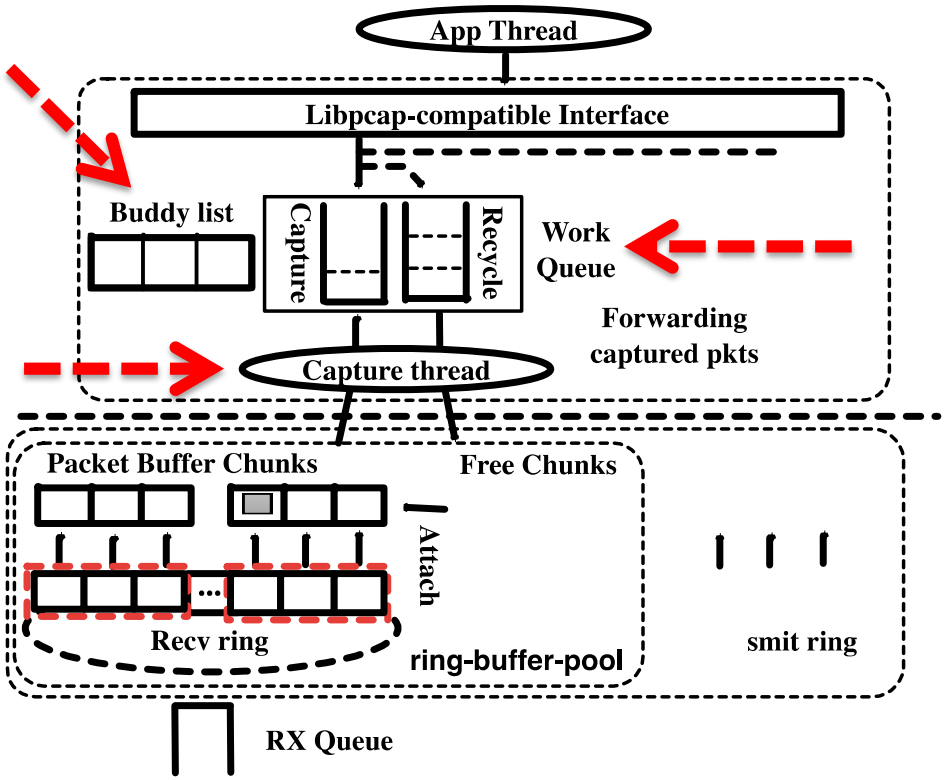
  -- Open  -- Capture  -- Recycle  -- Close

# The buddy-group mechanism




- Executes traffic offloading in an application-aware manner
  - Receive queues accessed by a single application can form a buddy group
  - Traffic offloading is only permitted within a buddy group

# WireCAP Architecture

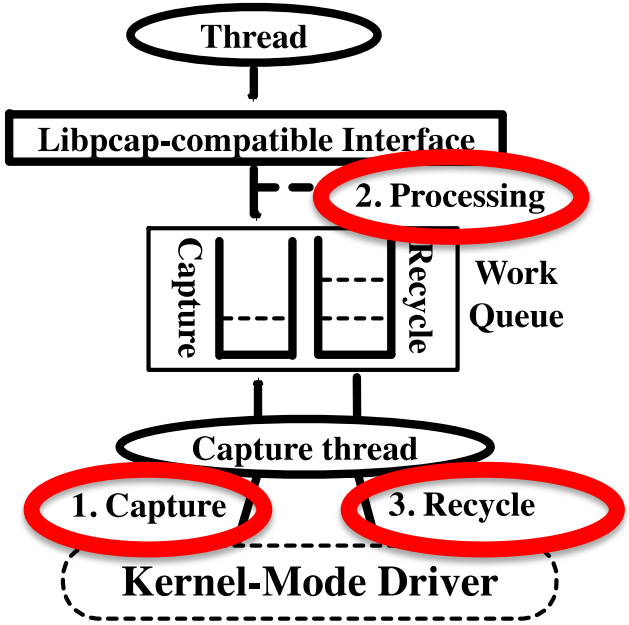



- A Libpcap-compatible interface for low-level network access
- The buddy-group-based offloading mechanism

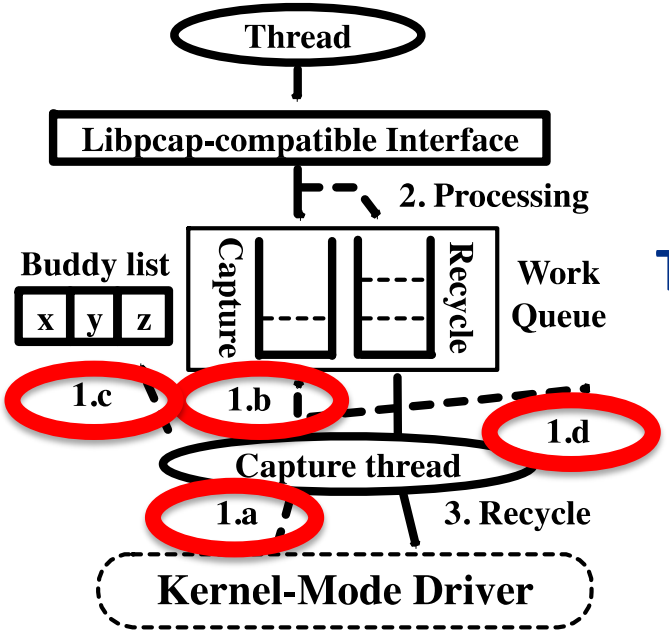
- Low-level packet capture and transmit service
- The ring-buffer-pool mechanism

# WireCAP Operations

- Lossless zero-copy packet capture and delivery




a. basic mode




b. advanced mode

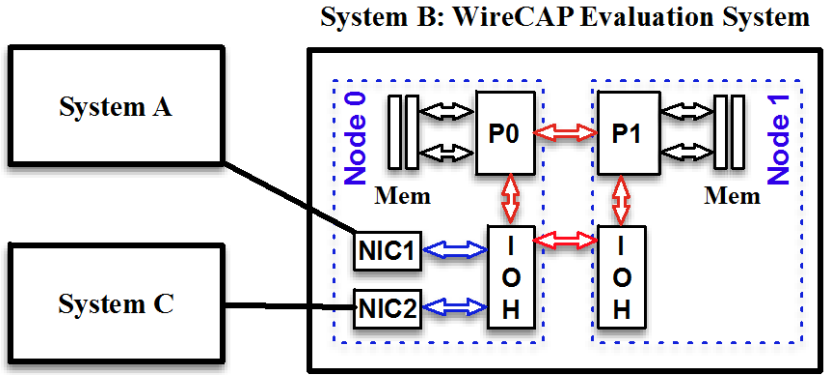
**Threshold: T**

- Zero-copy packet forwarding
  - Only involving metadata operations

# WireCAP Implementation

- The current implementation consists of a kernel-mode driver and a user-mode library
- OSes Supported
  - Linux kernel 3.16
  - We are working to support other OSes
- Commodity NICs supported
  - Intel 82599-based 10GigE NIC
  - Working on support for 40 GigE NICs

# WireCAP Evaluation

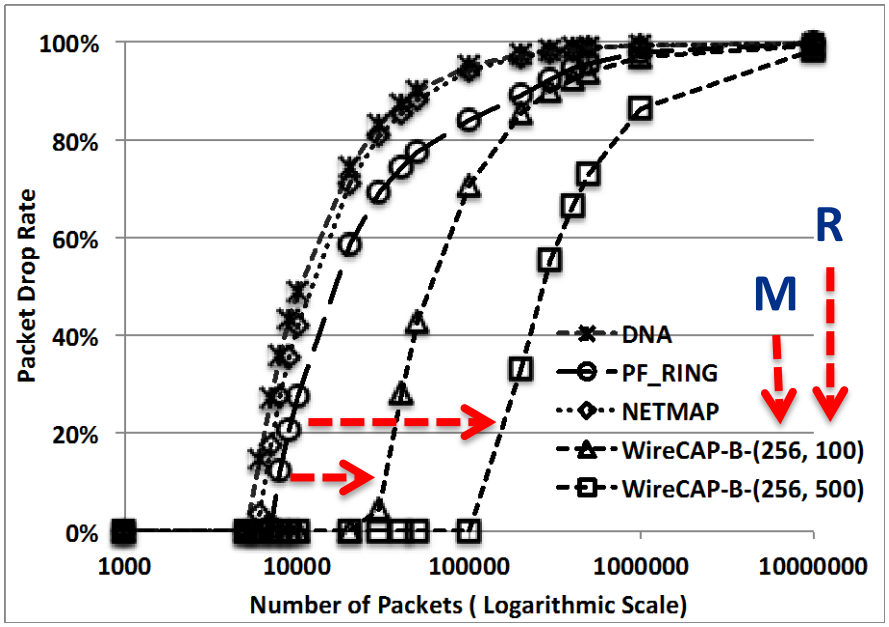



- We use traffic captured from Fermilab border router as experiment data
- System A runs as a traffic generator:

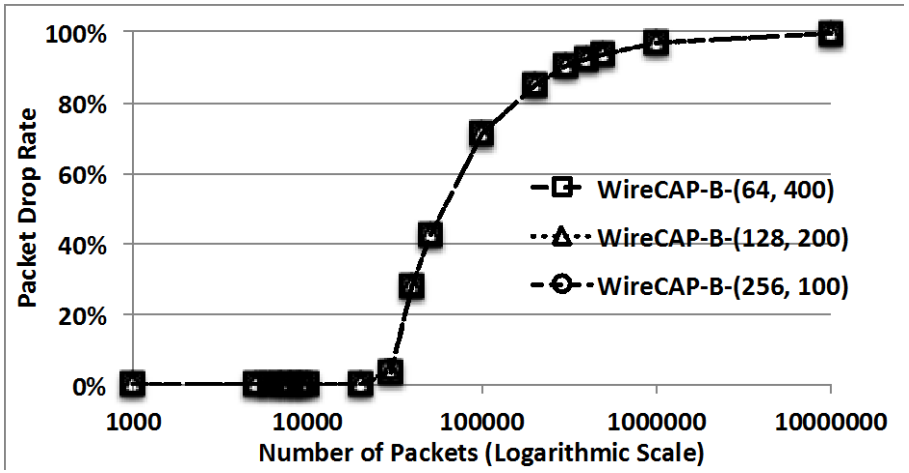
(or)
- generates 64B packets at wire speed
- replays captured data as recorded

- Exp1 - Packet capture in the basic mode
  - System A generates *P* 64B packets at wire speed (14.88 million p/s)
  - NIC1 is configured with one RQ
  - A single-threaded application captures and processes pkts at System B
- Exp2 - Packet capture in the advanced mode
  - System A replays the captured data
  - NIC1 is configured with multiple RQs, all RQs form a buddy group
  - A multiple-threaded application captures and processes pkts at System B
- Exp3 - Packet forwarding
  - System A replays the captured data
  - NIC1 is configured with multiple RQs, all RQs from a buddy group
  - A multi-threaded application captures and processes pkts at System B, processed pkts are forwarded to System C via NIC2

# Packet Capture in the basic mode



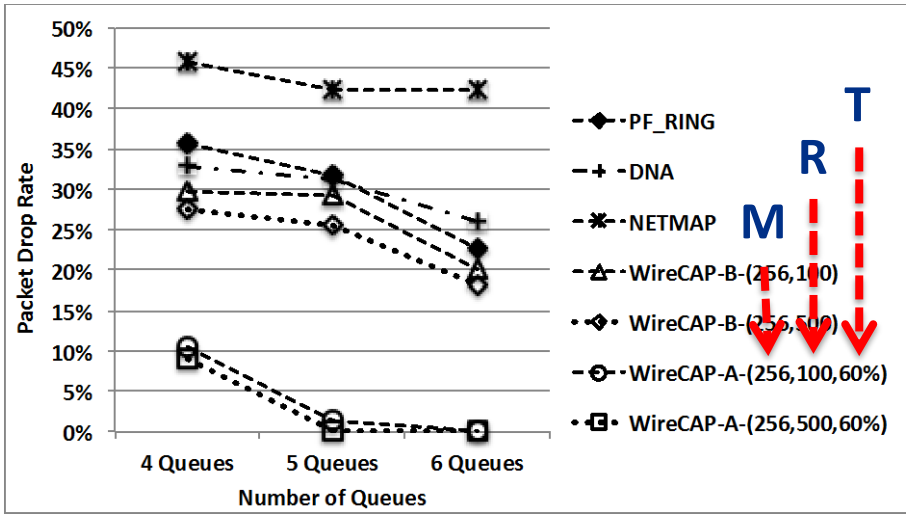

WireCAP vs. existing packet capture engines



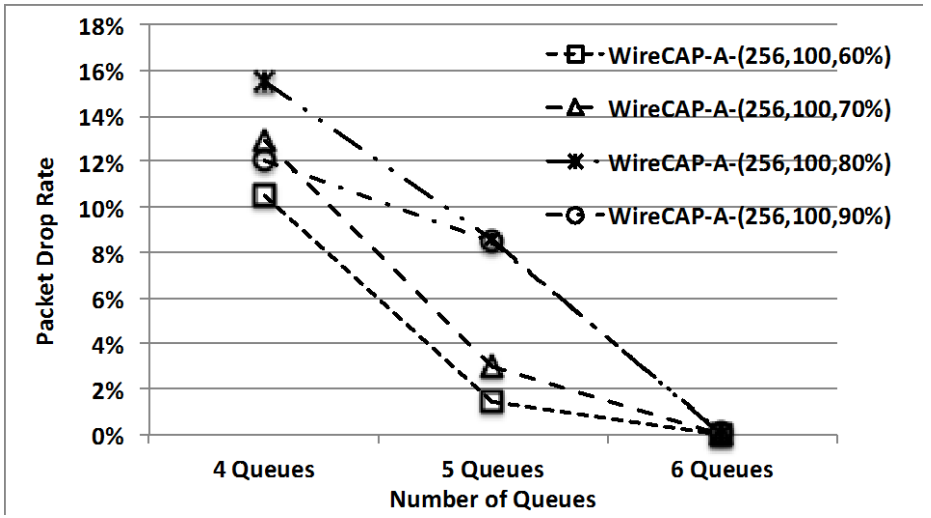

R and M are varied, R*M is fixed

- WireCAP demonstrates superior buffering capabilities for short-term bursts of packets
- WireCAP's buffering capability is proportional to the overall ring buffer capacity R*M

# Packet Capture in the advanced mode



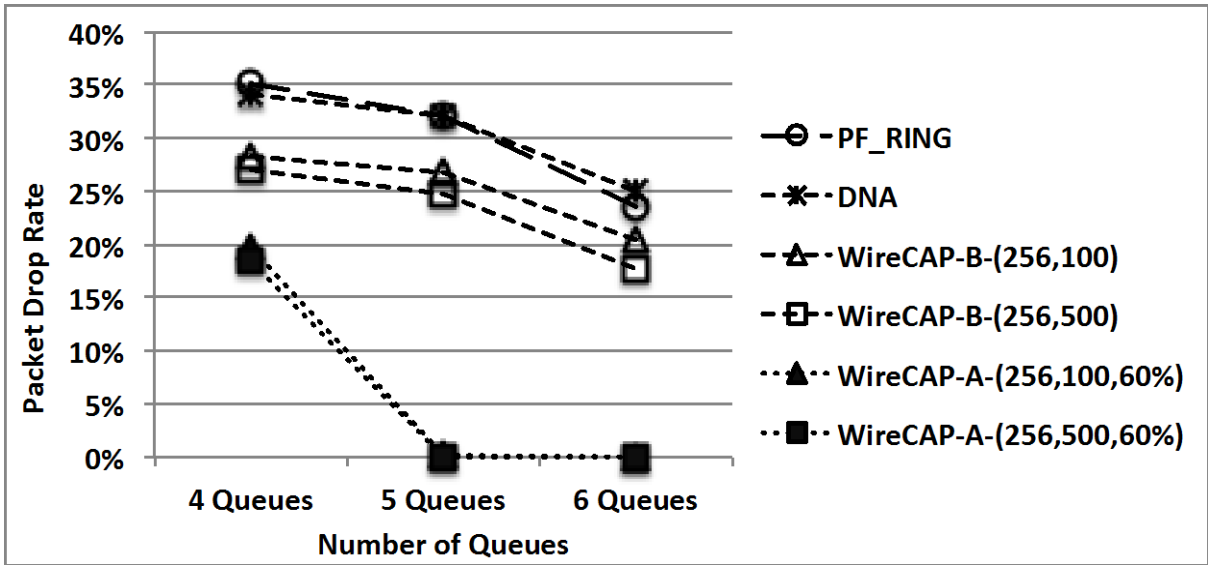

WireCAP vs. existing packet capture engines




R and M are fixed, T is varied

- The buddy-group-based offloading mechanism achieved significant improved performance
- In general, WireCAP performs better when T is set to a relatively lower value

# Packet Forwarding




**WireCAP vs. existing packet capture engines**

- **WireCAP's packet forwarding function is capable of supporting middlebox applications**
- **The *buddy-group-based offloading* mechanism can achieve a significant improved performance**

# Summary

- **WireCAP provides zero-copy lossless packet capture and delivery services by exploiting multi-queue NICs and multicore**

- **WireCAP provides a new packet I/O framework for commodity NICs in high-speed networks**

# Future Work

- **Compare WireCAP with DPDK**
- **Adapt WireCAP for 40GE NICs**

# Questions?

## WireCAP website:

## http://wirecap.fnal.gov

**Source code is available:**

**Please contact Fermilab's Office of Partnerships and Technology Transfer (OPTT) <optt@fnal.gov> to obtain a copy of WireCAP**