Fast and Flexible: Parallel Packet Processing with GPUs and Click

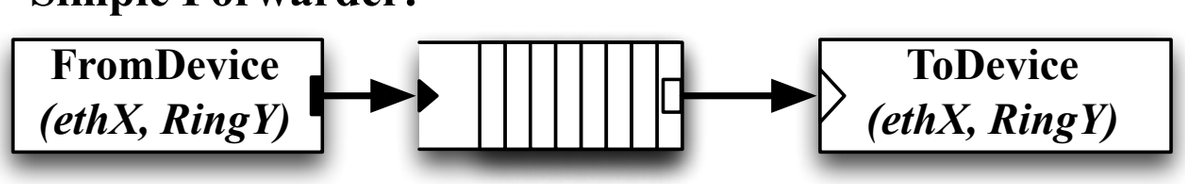
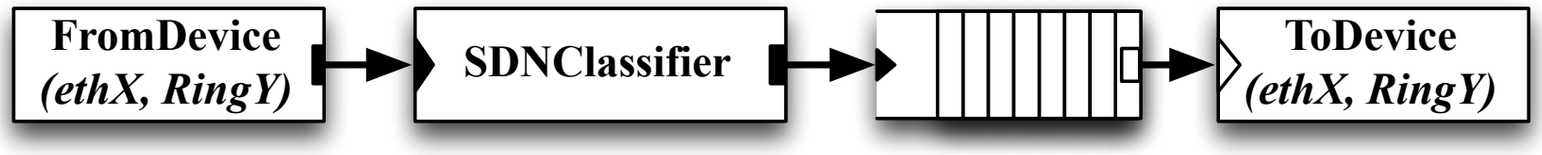
Weibin Sun, Robert Ricci
Flux Research Group, University of Utah

Snap = GPU + Click

- GPU: Parallel Processing
- Click: Modular and Flexible
- Snap provides:
  - A framework that allows people to do GPU-accelerated network research
  - Batched processing pipeline
  - Applications to demonstrate the efficiency

Why Snap?

An Example

Configurations	Throughput	Slow down
Simple Forwarder: 	30.97 Gbps	-
Simple SDN Forwarder: 	17.7 Gbps	42.7%

With four10Gb NICs, totally 40Gbps throughput.

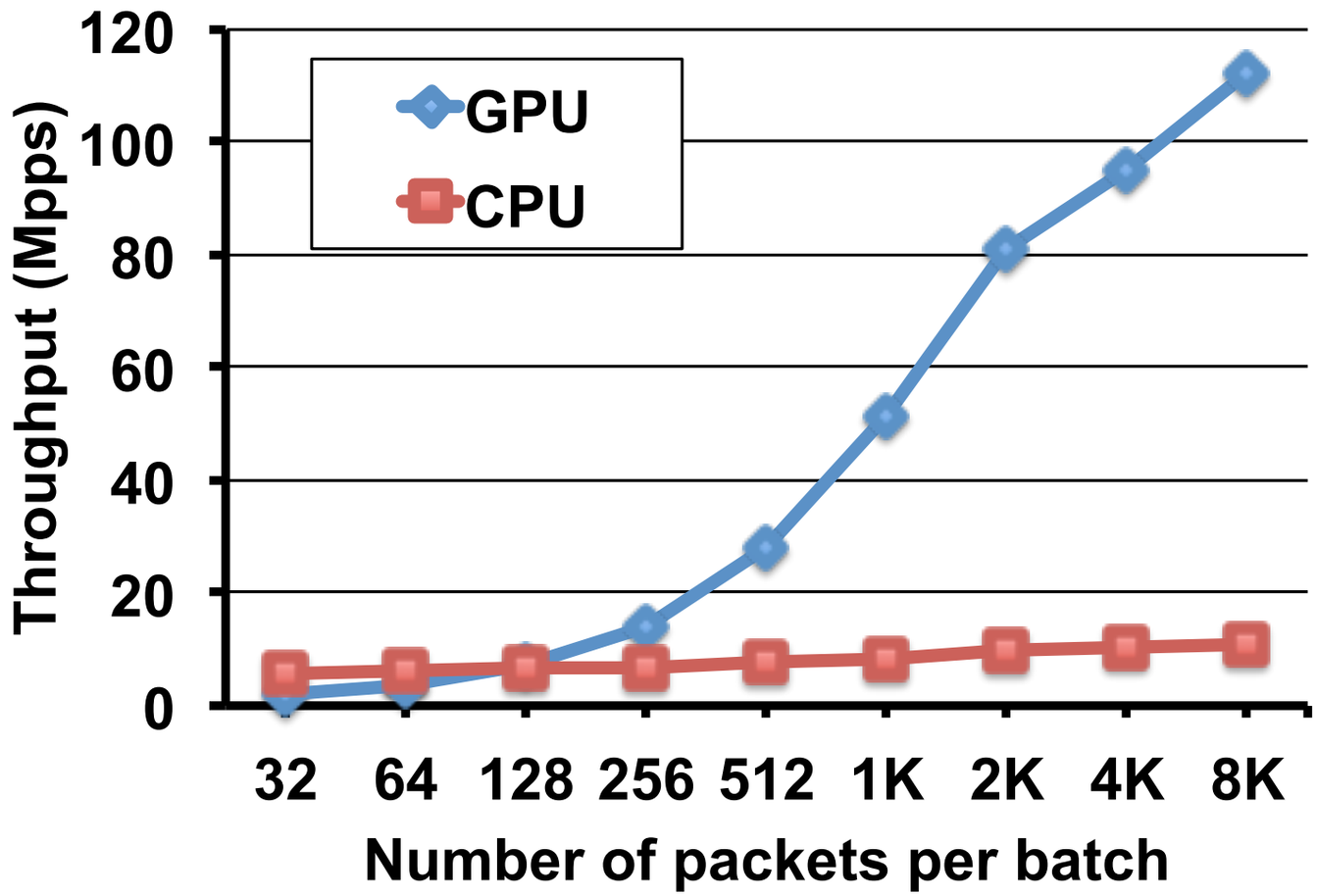
More Computing Power!

Parallel GPU

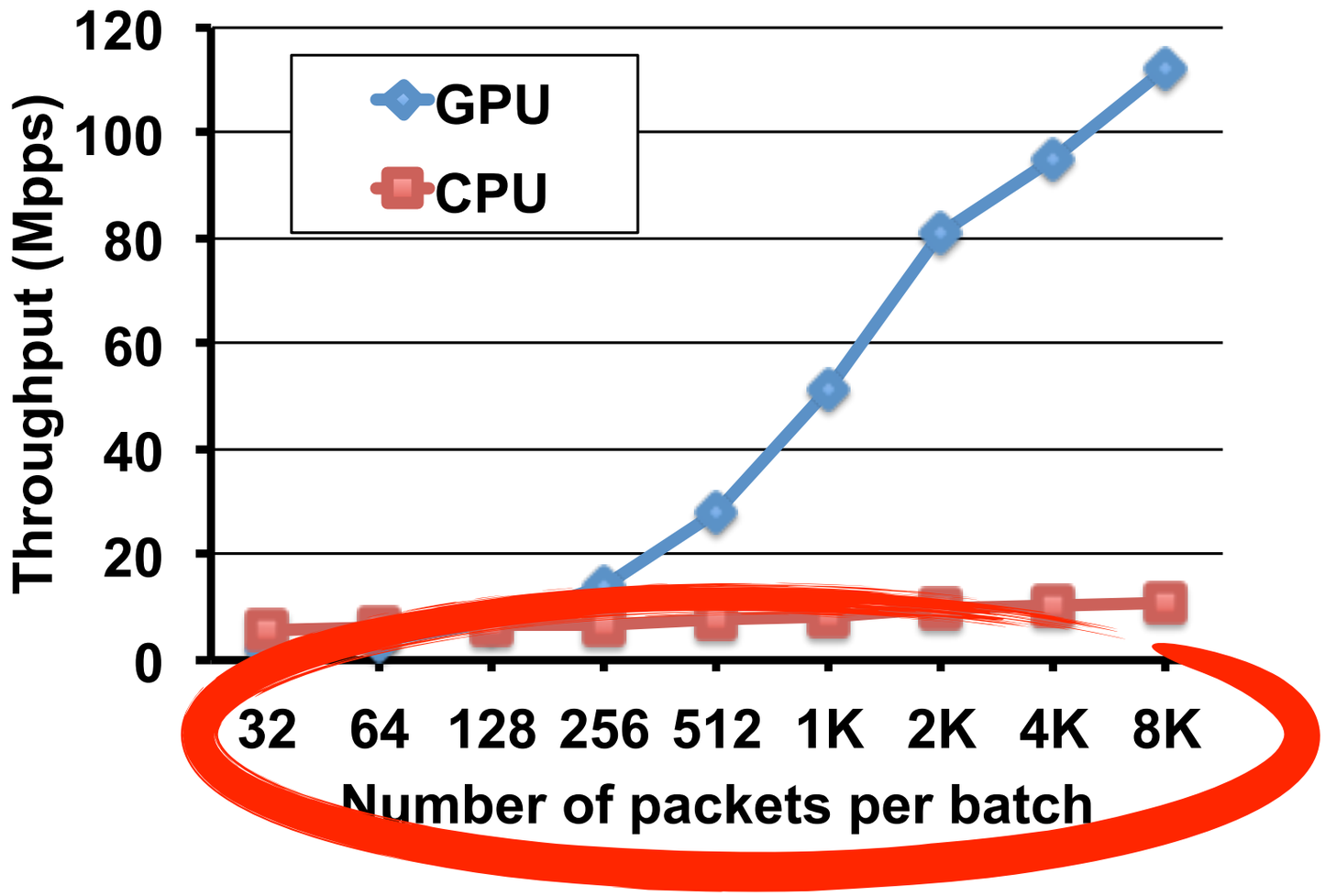
- Thousands of cores
- Most of time: more data = more speedup
- Up to 6GB memory
- ~16GB/s PCIe 3.0 x16 bandwidth

Image: nvidia.com

GPU vs. CPU: Classifier



GPU vs. CPU: Classifier



But, there are already...

- PacketShader [SIGCOMM'10], SSLShader [NSDI'11]
- Gnort [RAID'08], Kargus[CCS'12]
- MIDeA [CCS'11]
- ...

Snap is based on the modular Click

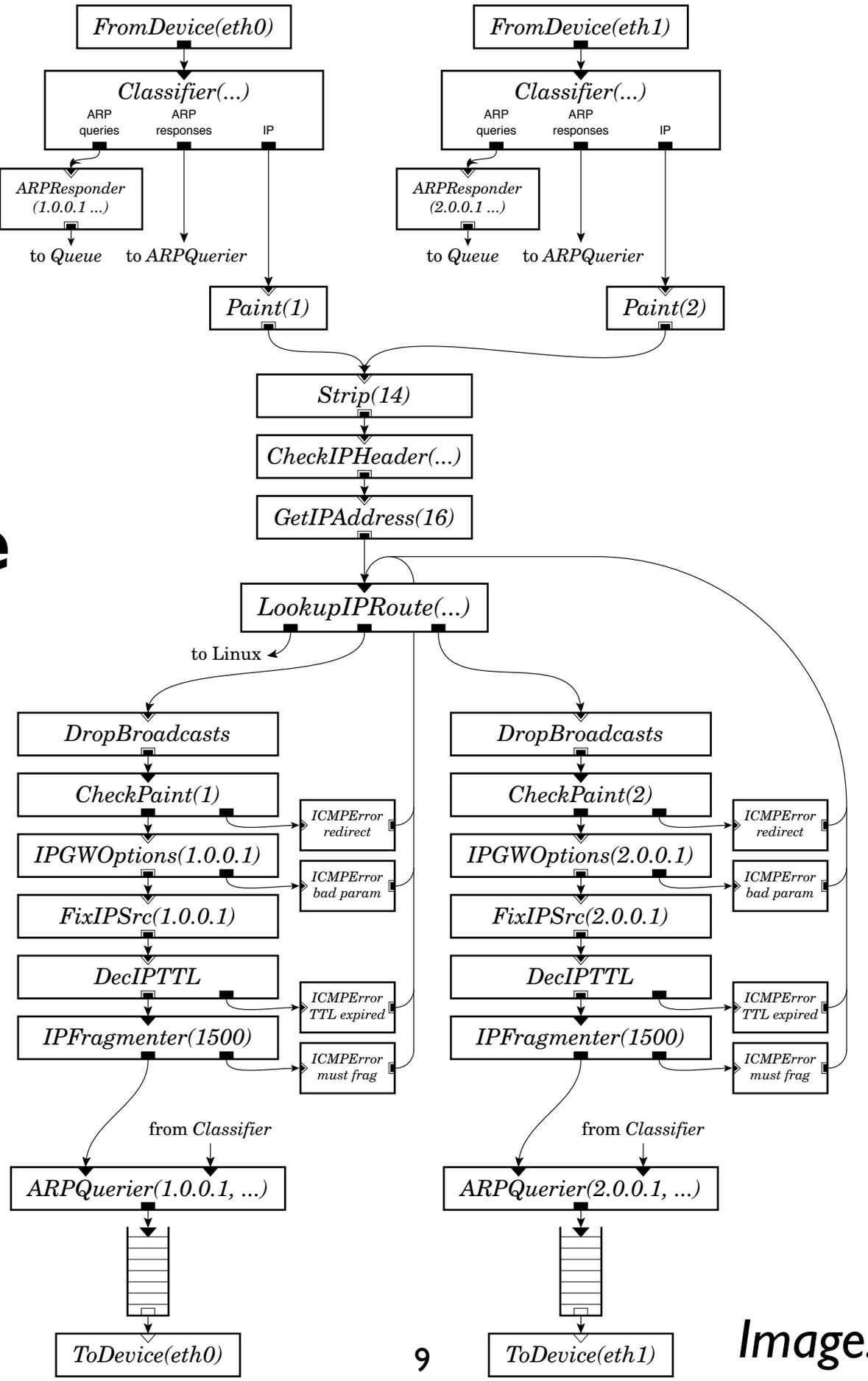


Image: [Click:TOCS'00] Fig.8

More Reasons

- Easy to extend with CUDA for GPU computing
- Fast user-space packet processing with Netmap
- Many existing elements
  - No need to write everything from scratch
  - All layers, including wireless!

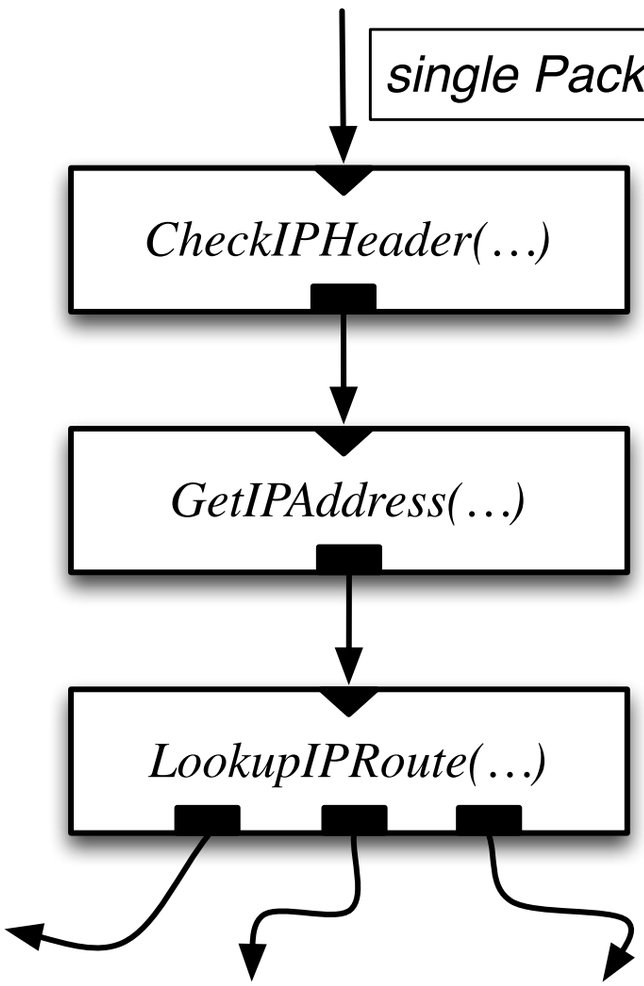
Snap: Key Challenges

- Batching: more data, more speedup
- Optimizing data placement for memory access and copy
- Packet divergence
- Reducing unnecessary memory copy

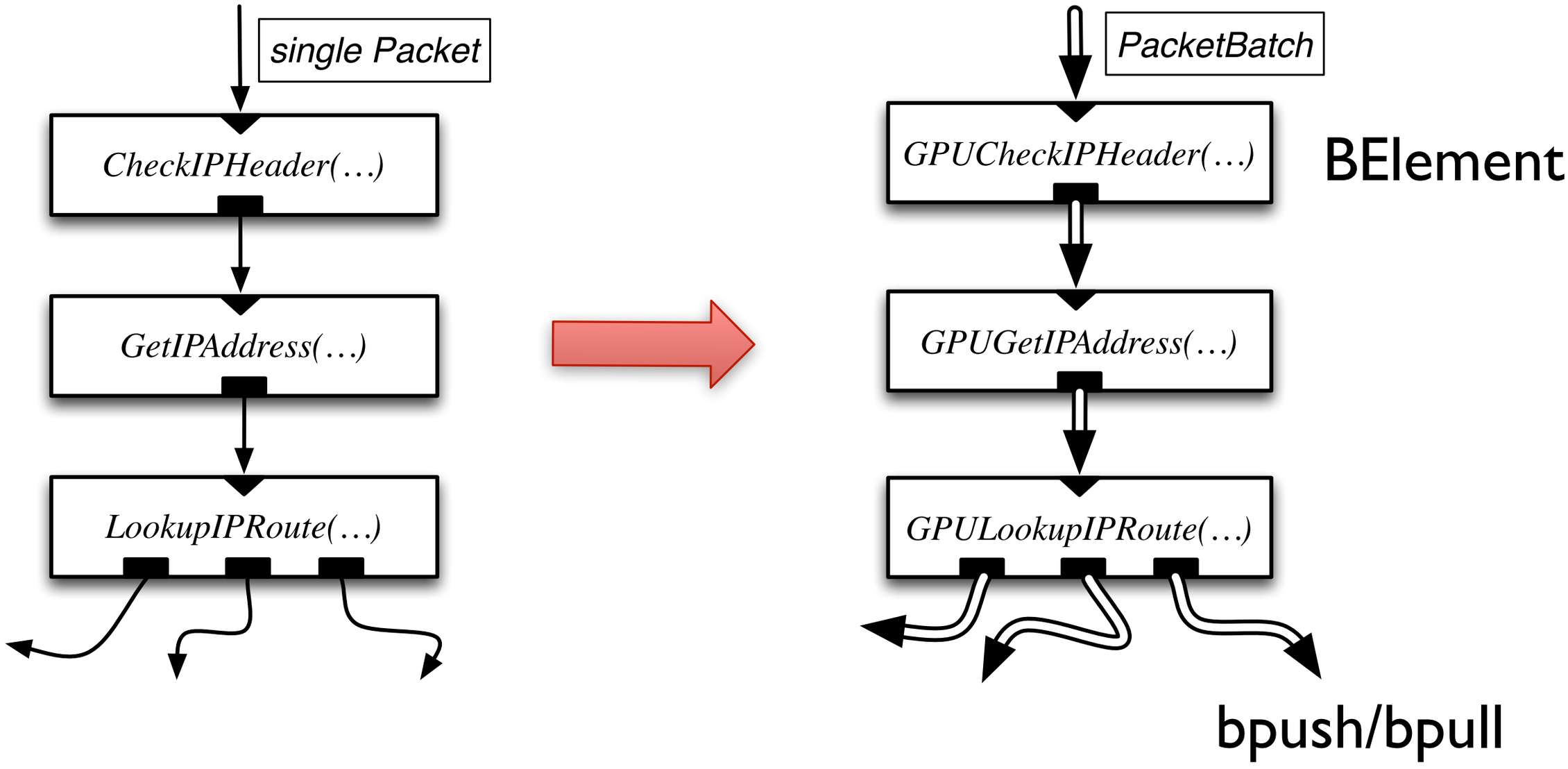
Snap = GPU + Click

- Wider packet processing pipeline in Click
- Processing a batch of packets
- Offloading processing onto GPU
- Packet slicing to reduce memory copy
- Handling packet divergence

Architectural Change of Click

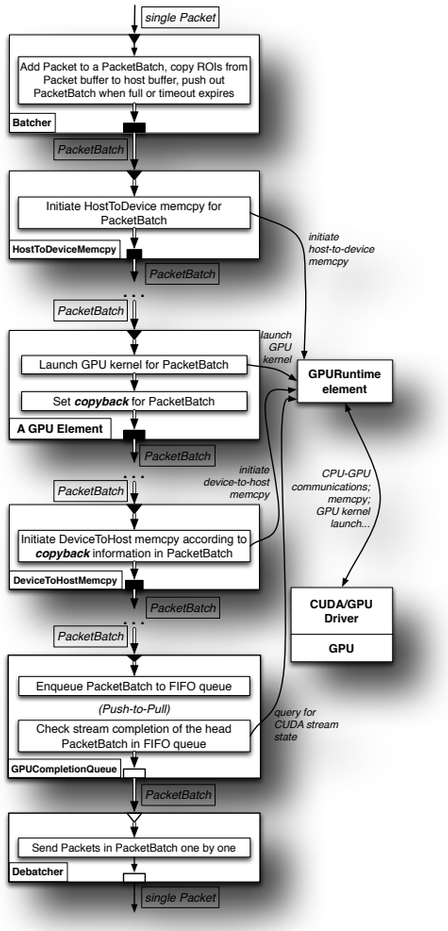


Architectural Change of Click

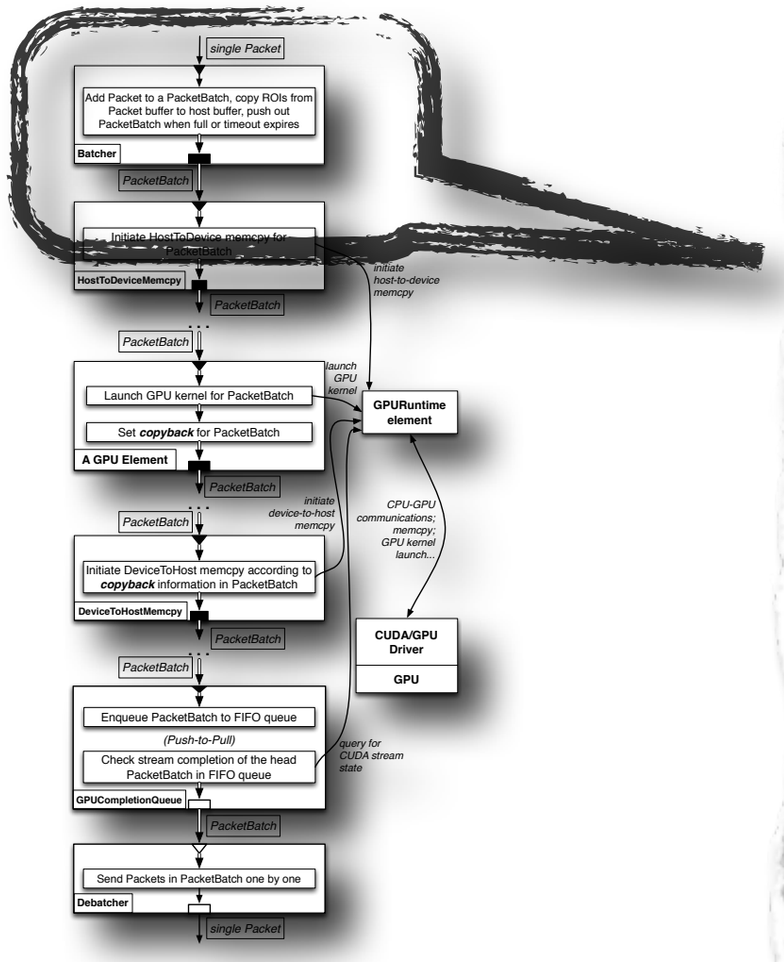


Batched Processing

Snap Batched Processing

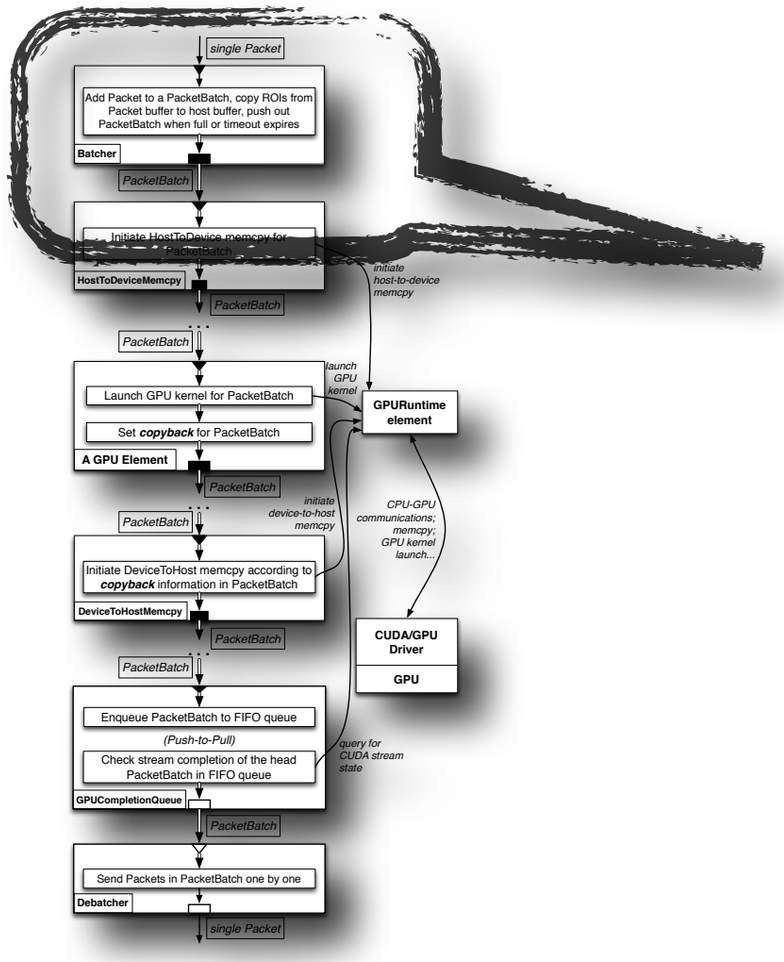


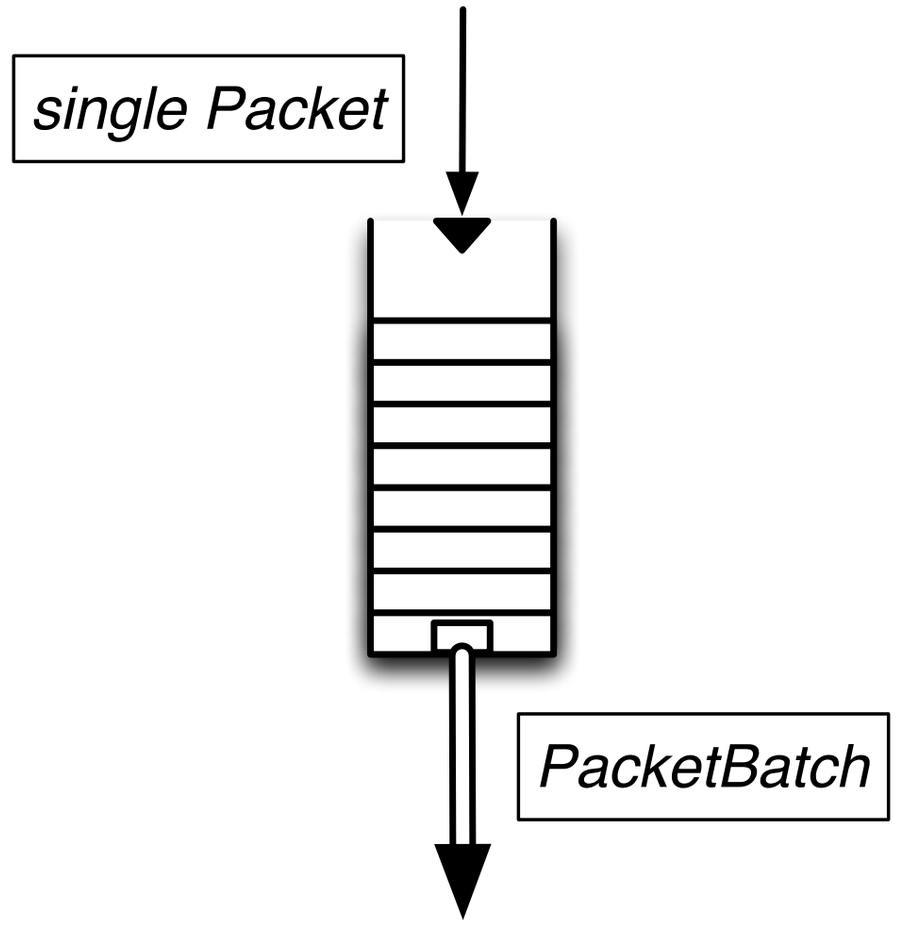
Snap Batched Processing



single Packet
Add Packet to a PacketBatch, copy ROIs from Packet buffer to host buffer, push out PacketBatch when full or timeout expires
Batcher
PacketBatch

Snap Batched Processing



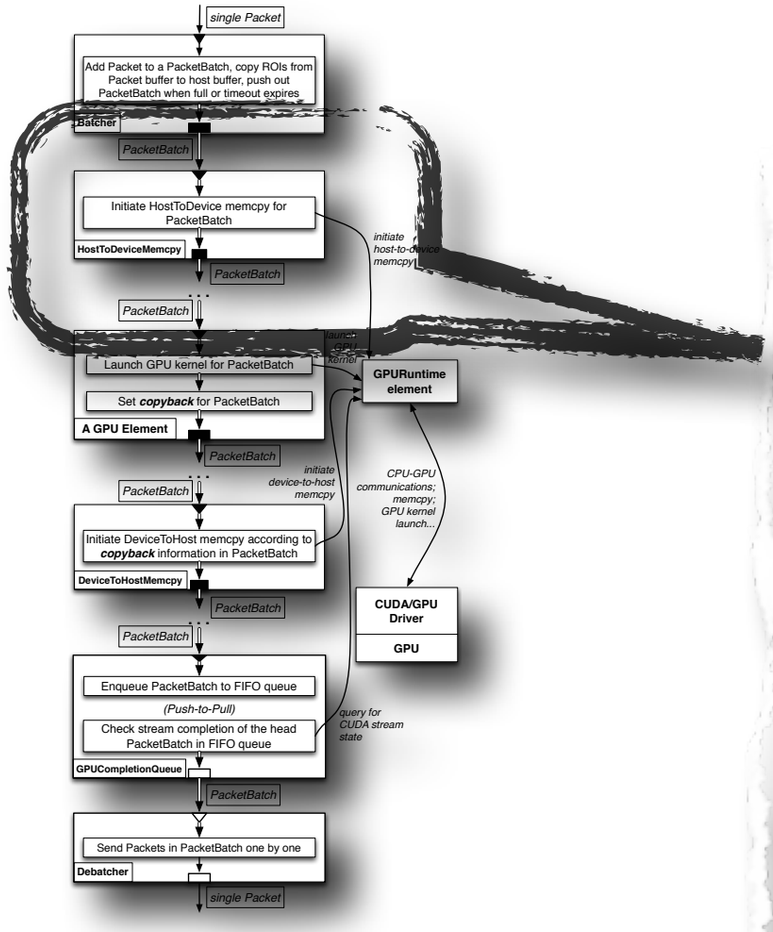


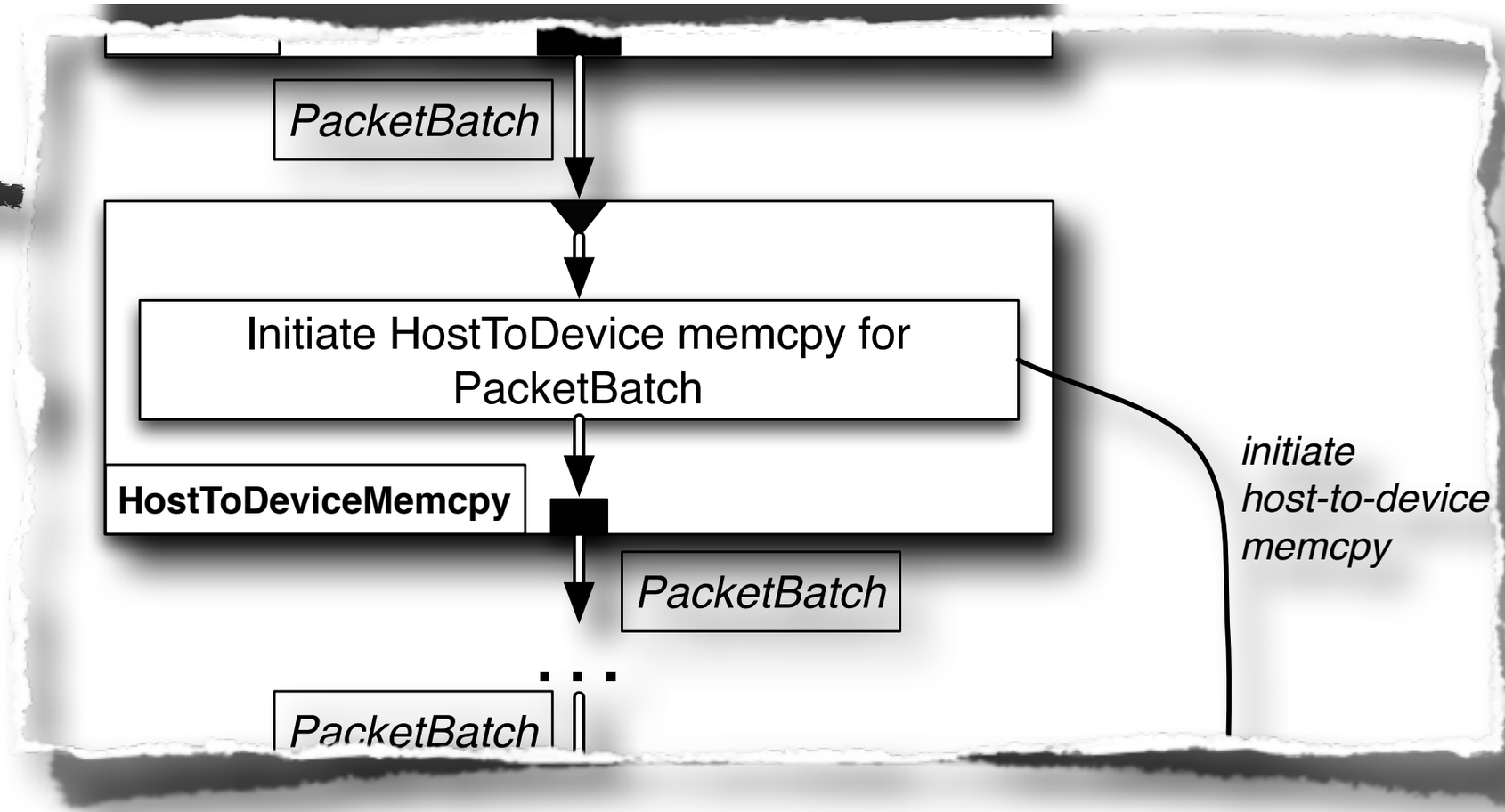
GPU Offloading

GPU Packet Processing

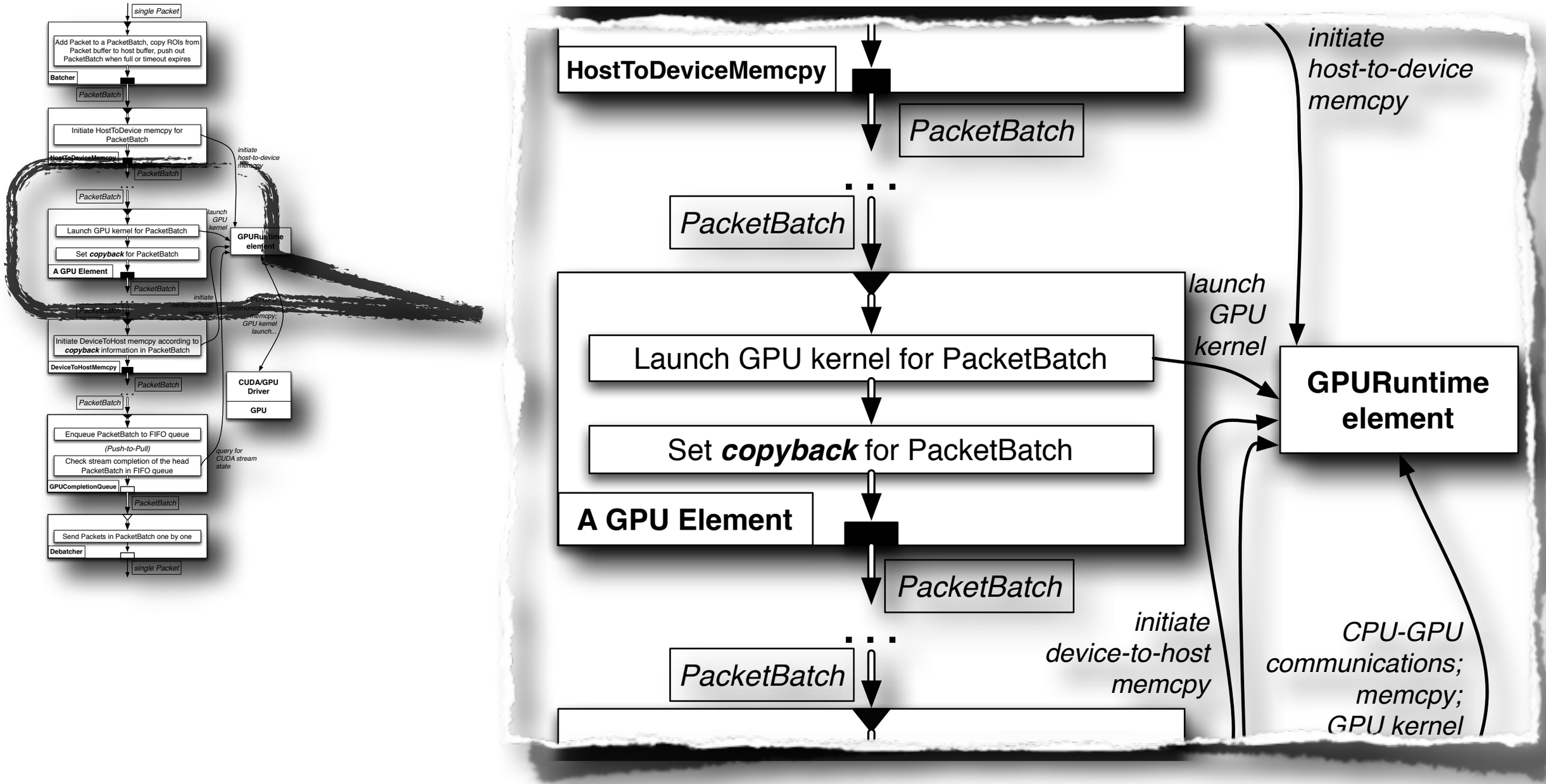
- Mapping processing algorithms to parallel GPU cores: One packet per GPU thread:
  - One IP lookup per GPU thread
  - Classifying one packet per GPU thread
  - Matching one packet against a DFA per GPU thread for pattern matching
  - ...

Copy Packets to the GPU

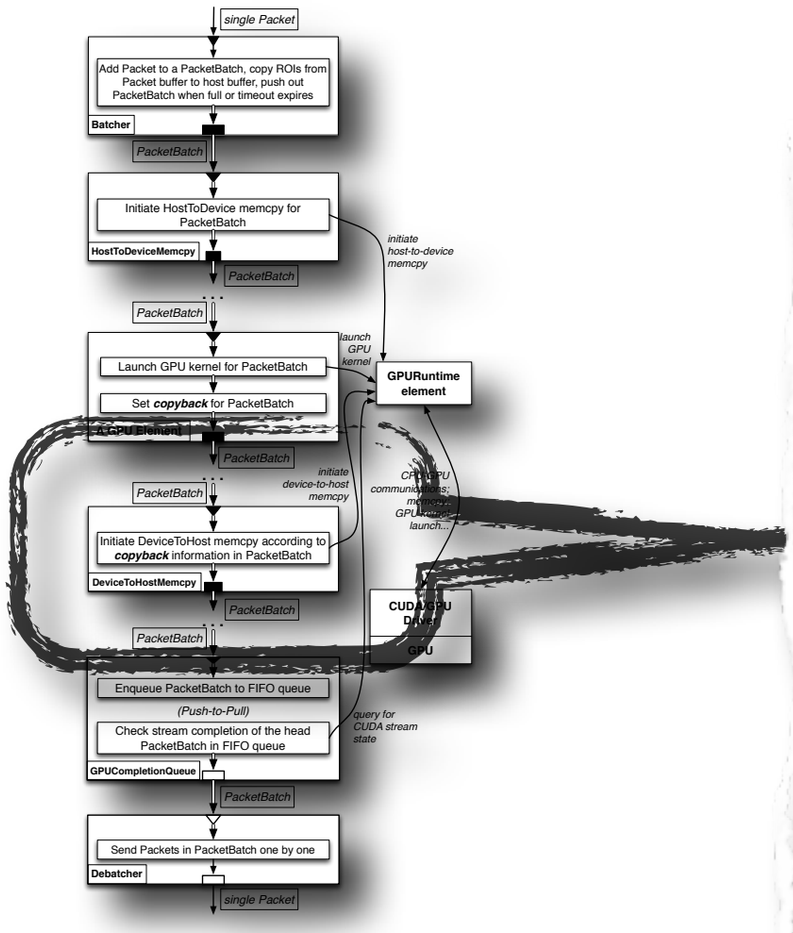


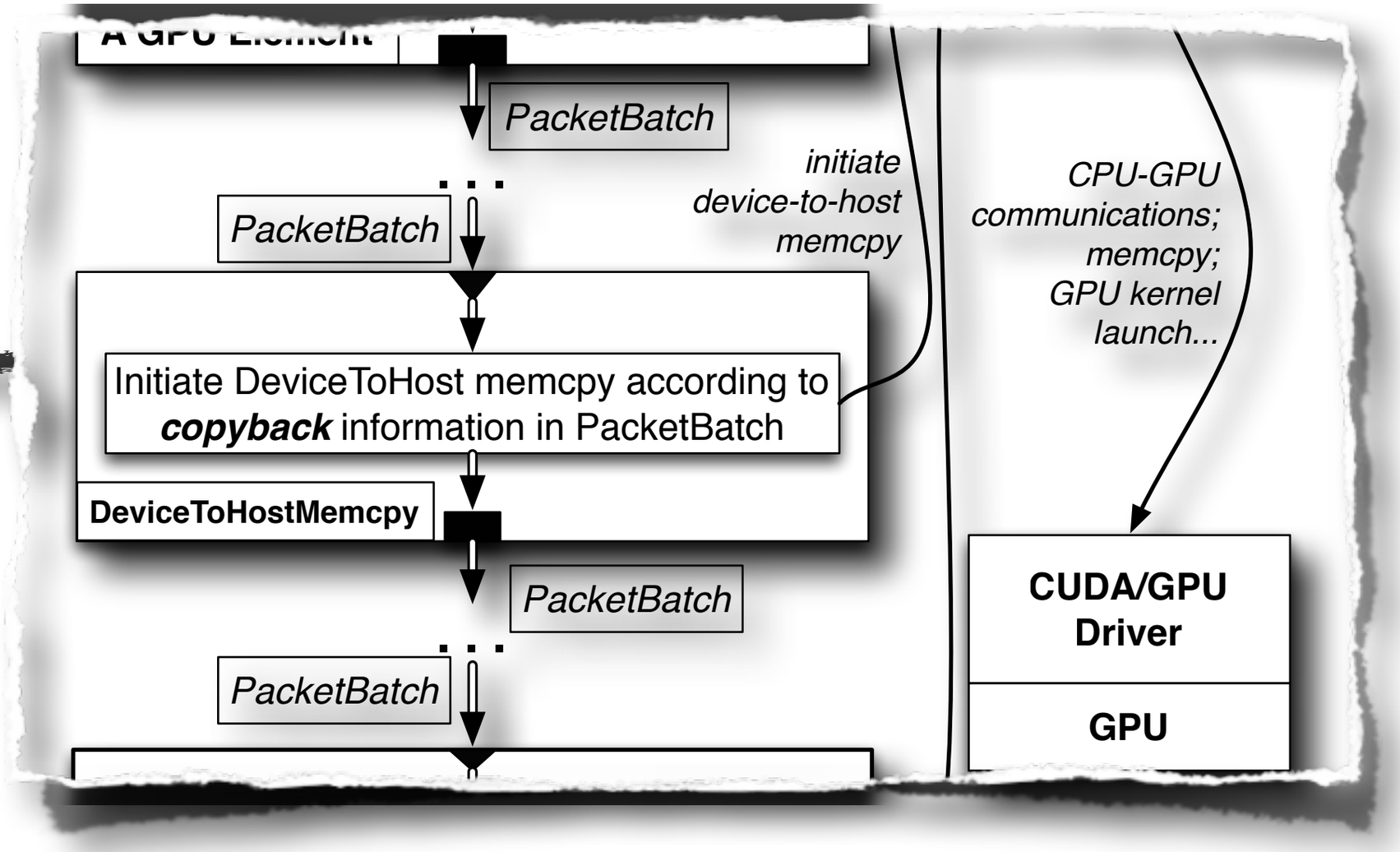


Process Packets on the GPU

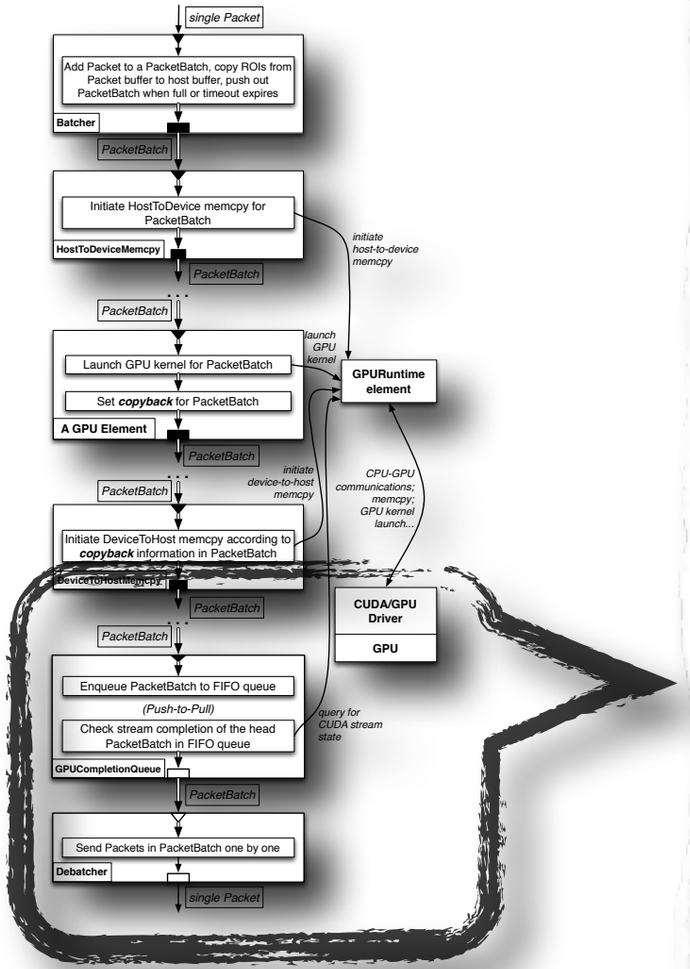


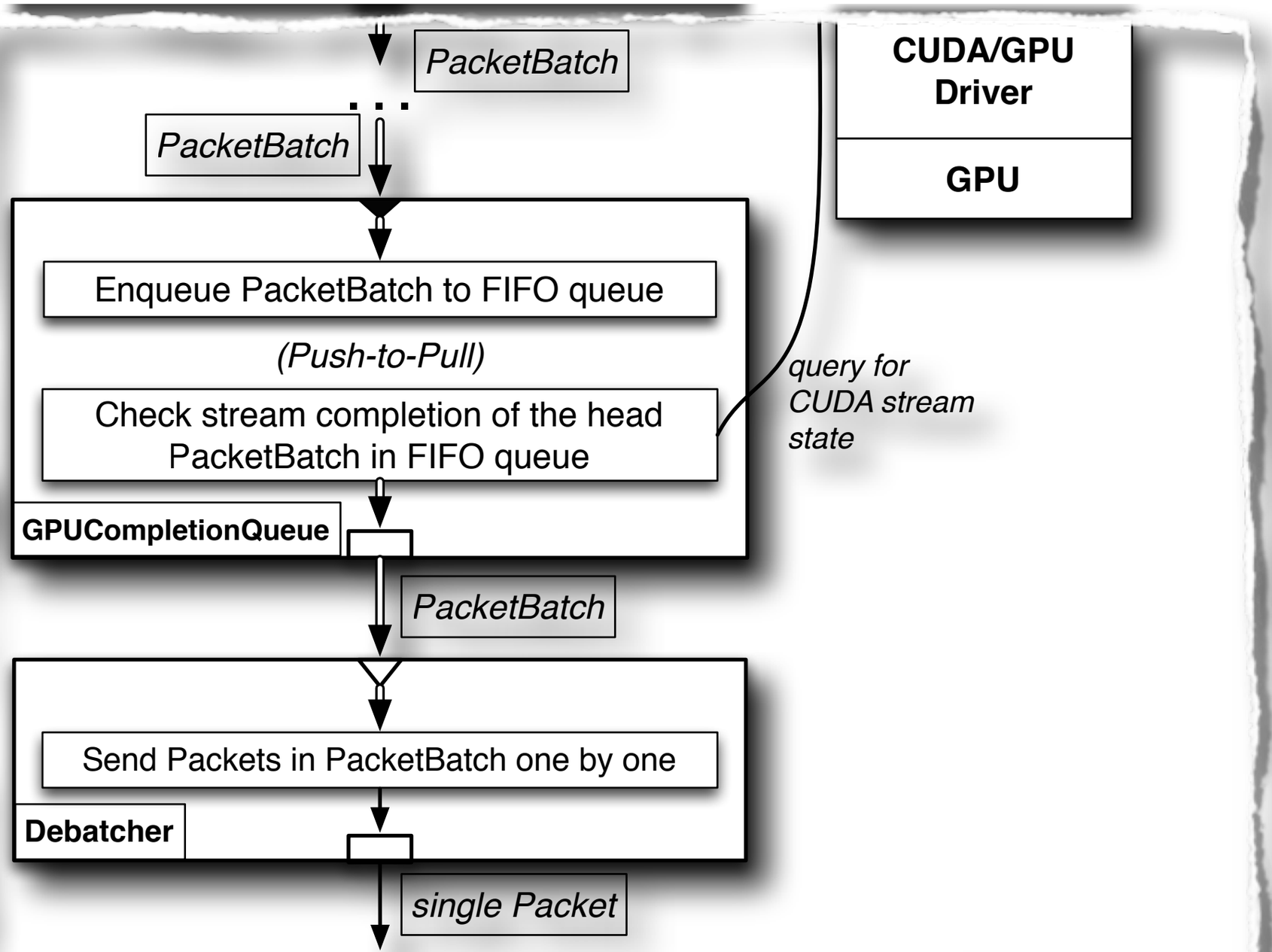
Copy the Result Back





Leave Batched Processing

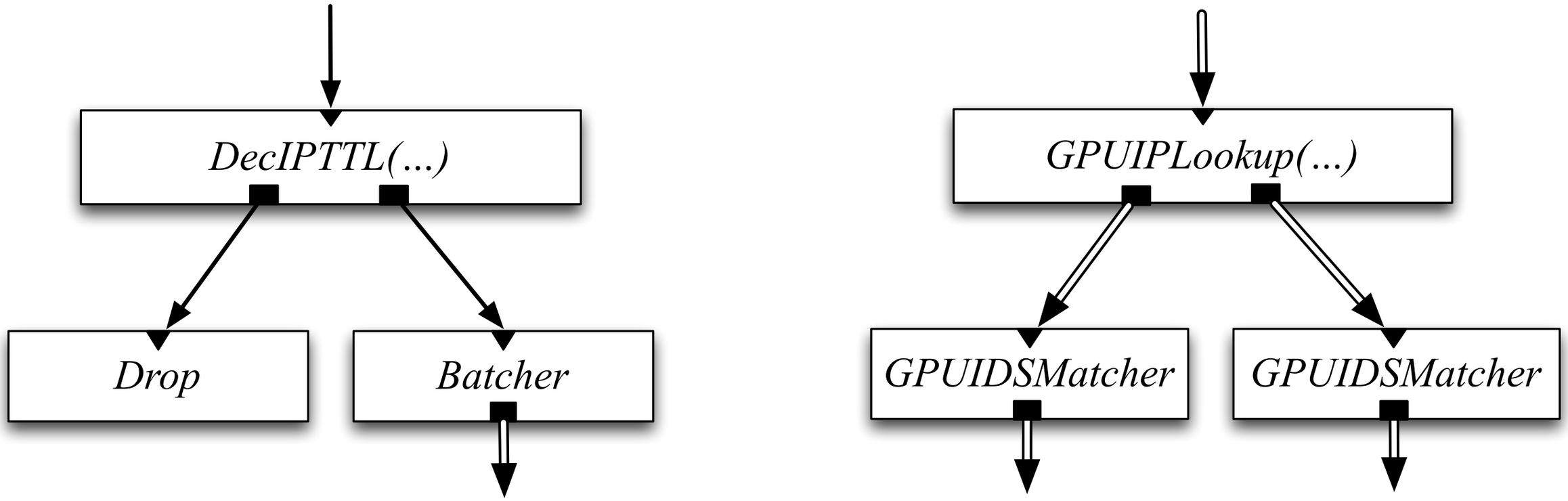




Problems in a Snap GPU Pipeline

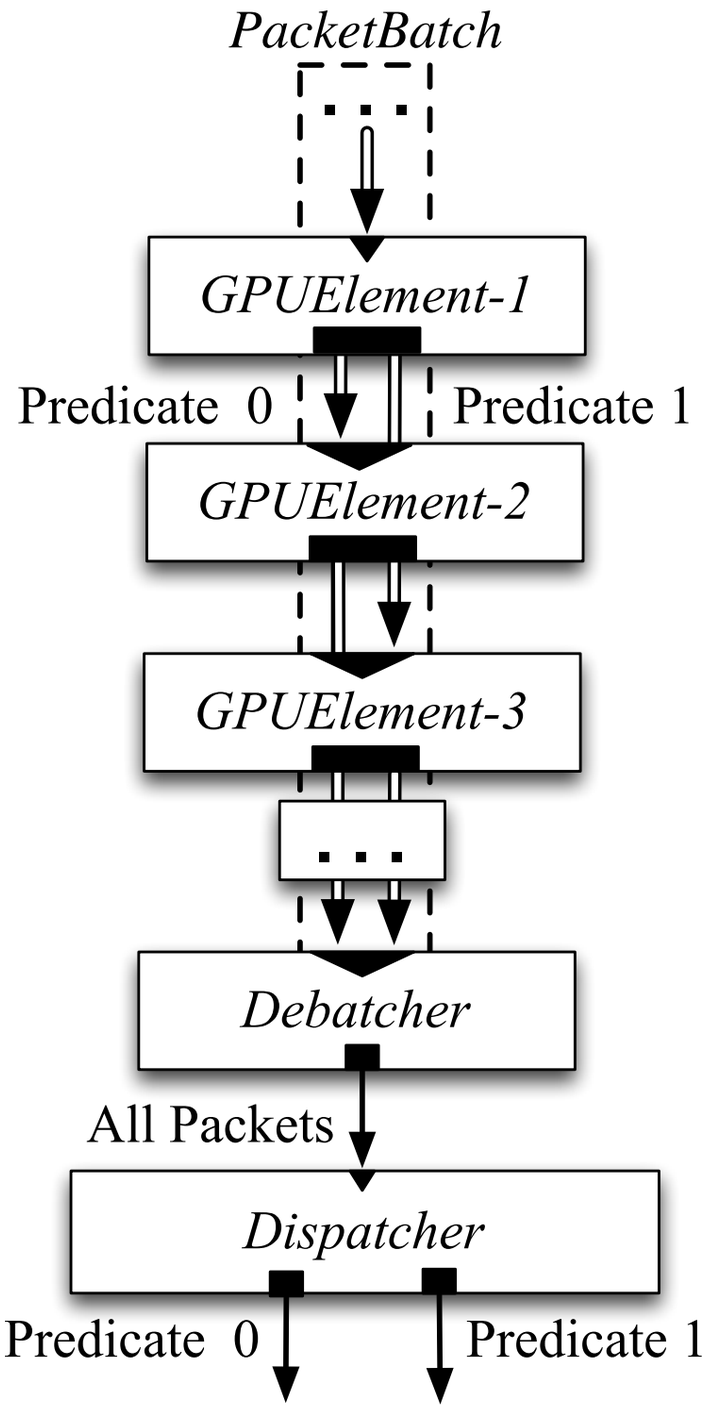
- Packet divergence
- Unnecessary memory copy

Packet Divergence



Handling divergence

- Predicate bits based solution
  - Produced by elements causing divergence
  - Consumed by following elements
  - Imagine them as path code
  - Pre-allocated bits



Problems in a Snap GPU Pipeline

- ~~Packet divergence~~
- Unnecessary memory copy

What does classification need?

Packet

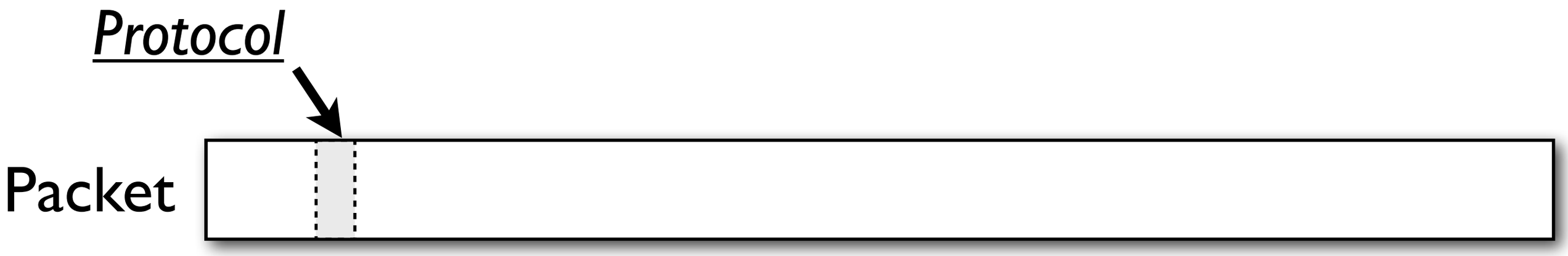
What does classification need?

Five tuples:

Packet

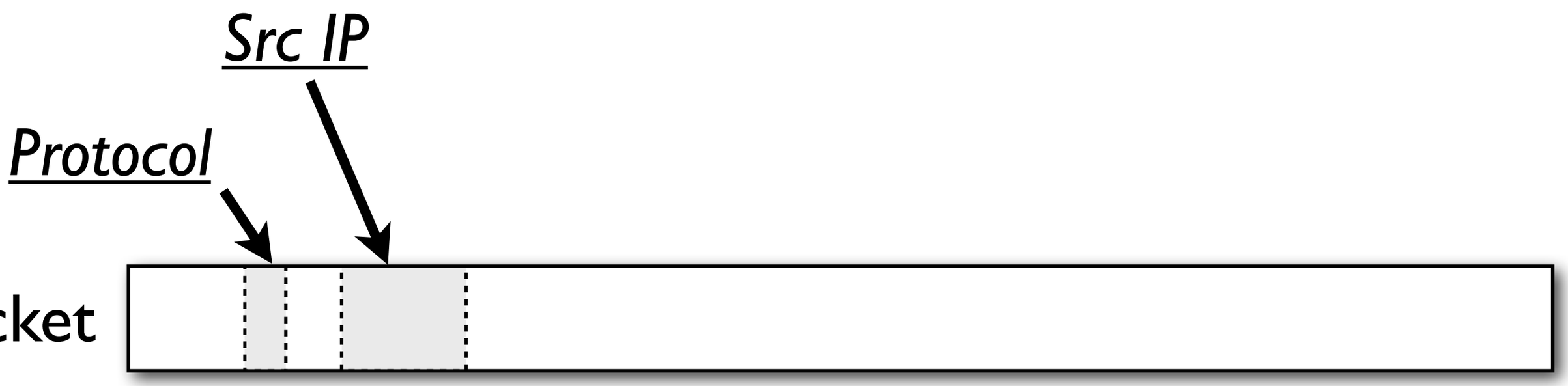
What does classification need?

Five tuples:



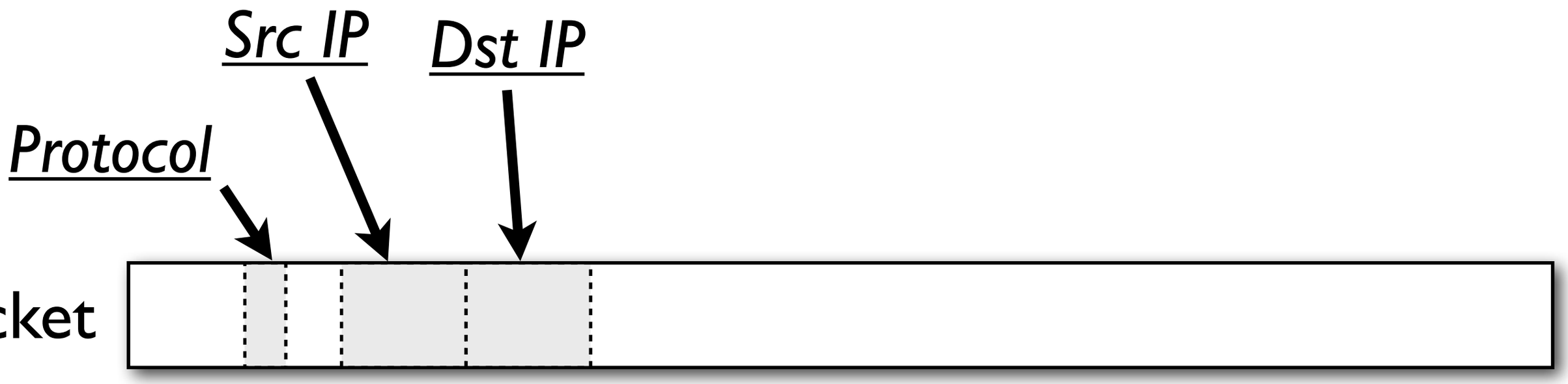
What does classification need?

Five tuples:



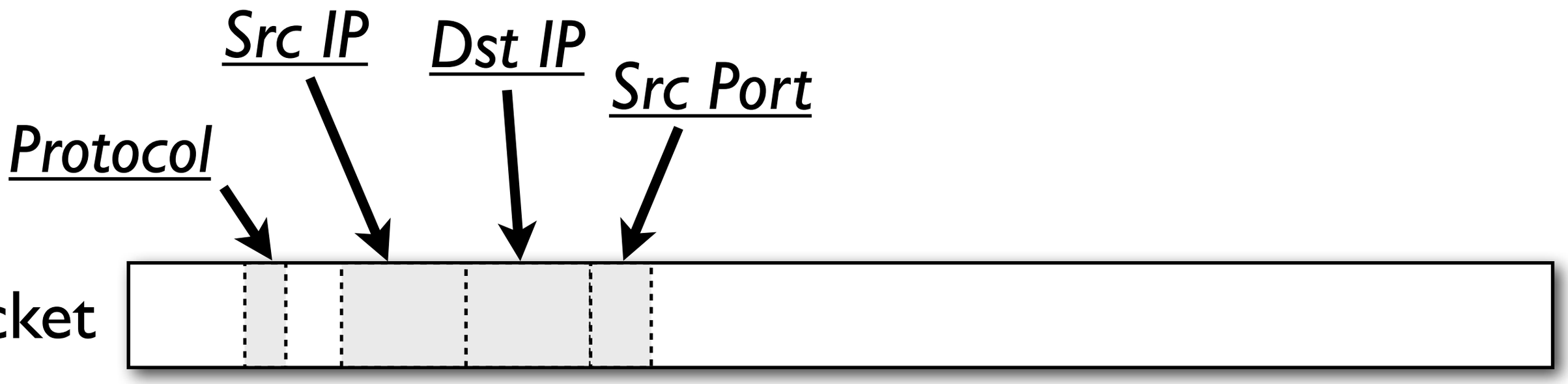
What does classification need?

Five tuples:



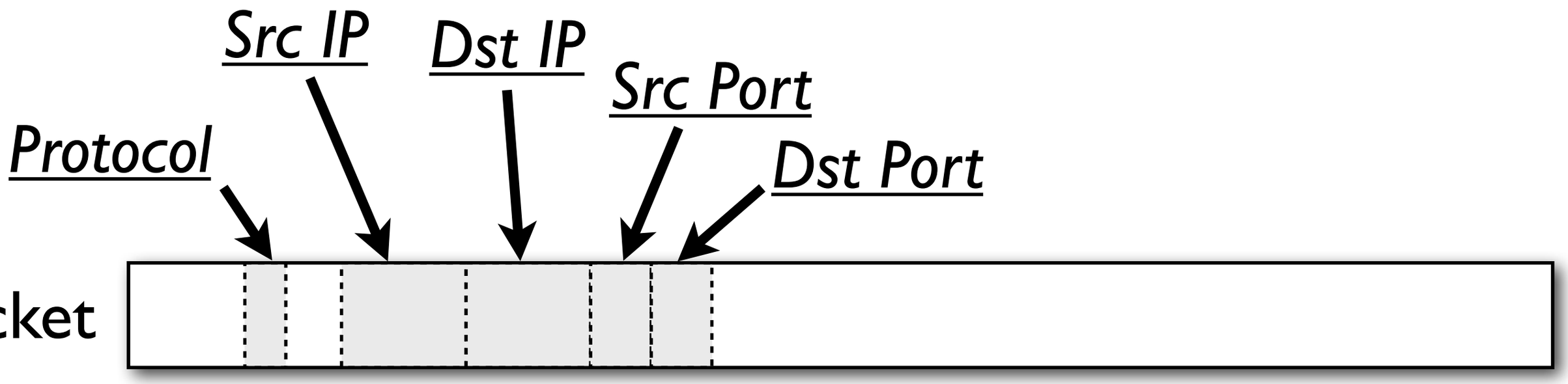
What does classification need?

Five tuples:



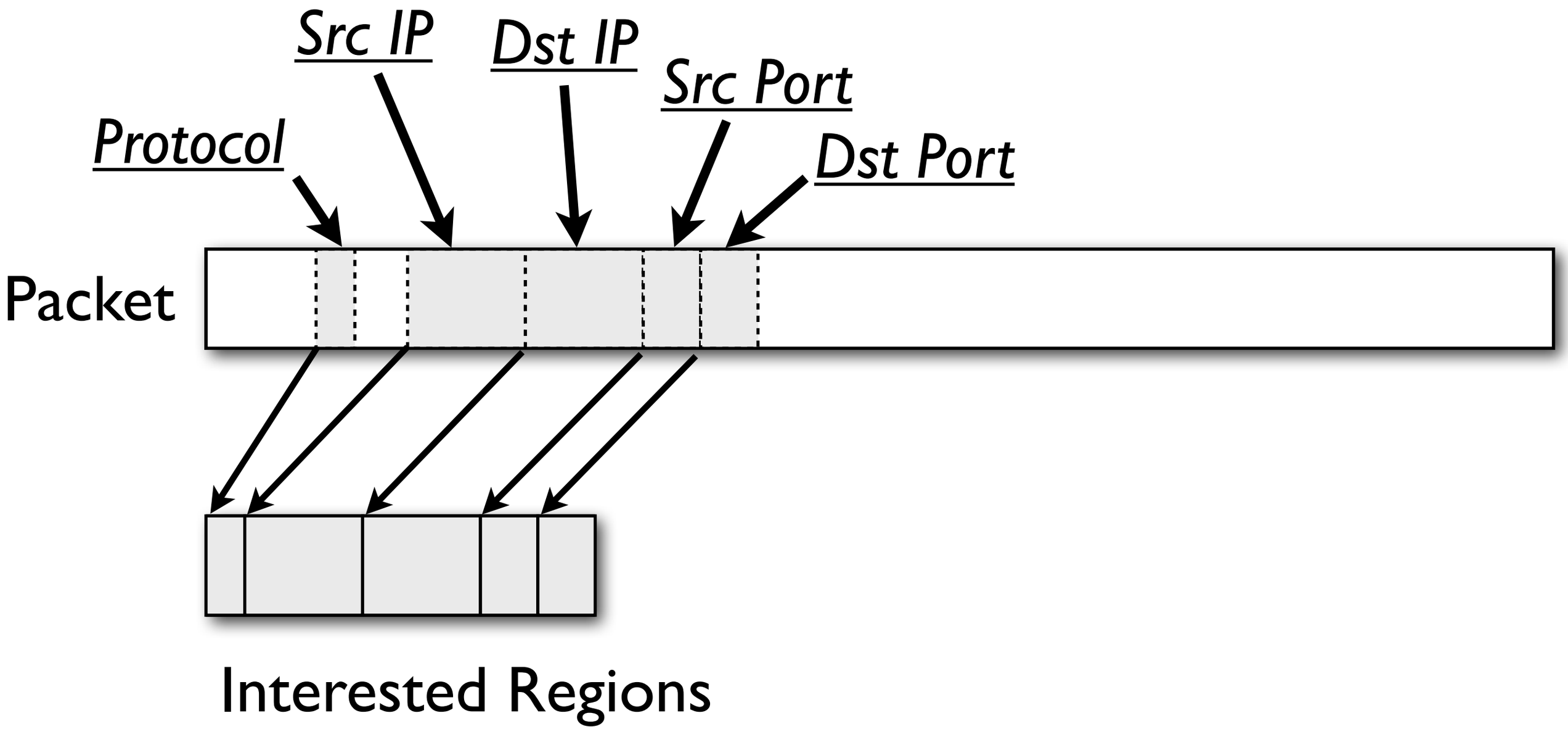
What does classification need?

Five tuples:

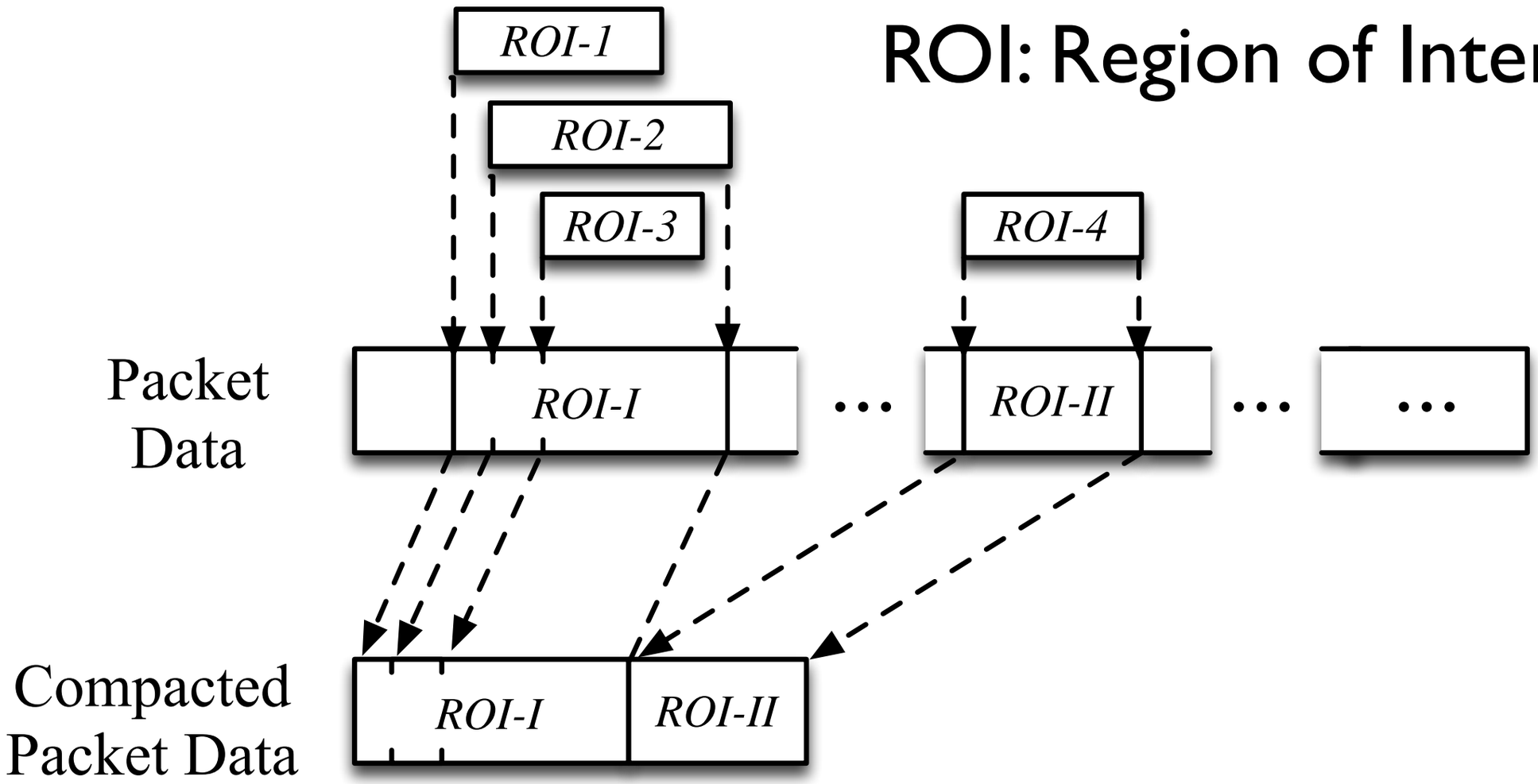


What does classification need?

Five tuples:



Packet Slicing



ROI: Region of Interest

ROI Constants:

- ETH_BEGIN
- ETH_END
- IP_BEGIN
- IP_END
- IP_SRC
- IP_DST
- UDP_BEGIN
- UDP_END
- UDP_SRC_PORT
- ...

Details in the Paper

- Batcher
- BElement, interfaces
- PacketBatch data structure, Packet Slicing
- Divergence handling
- BElement Scheduling
- Packet I/O
- ...

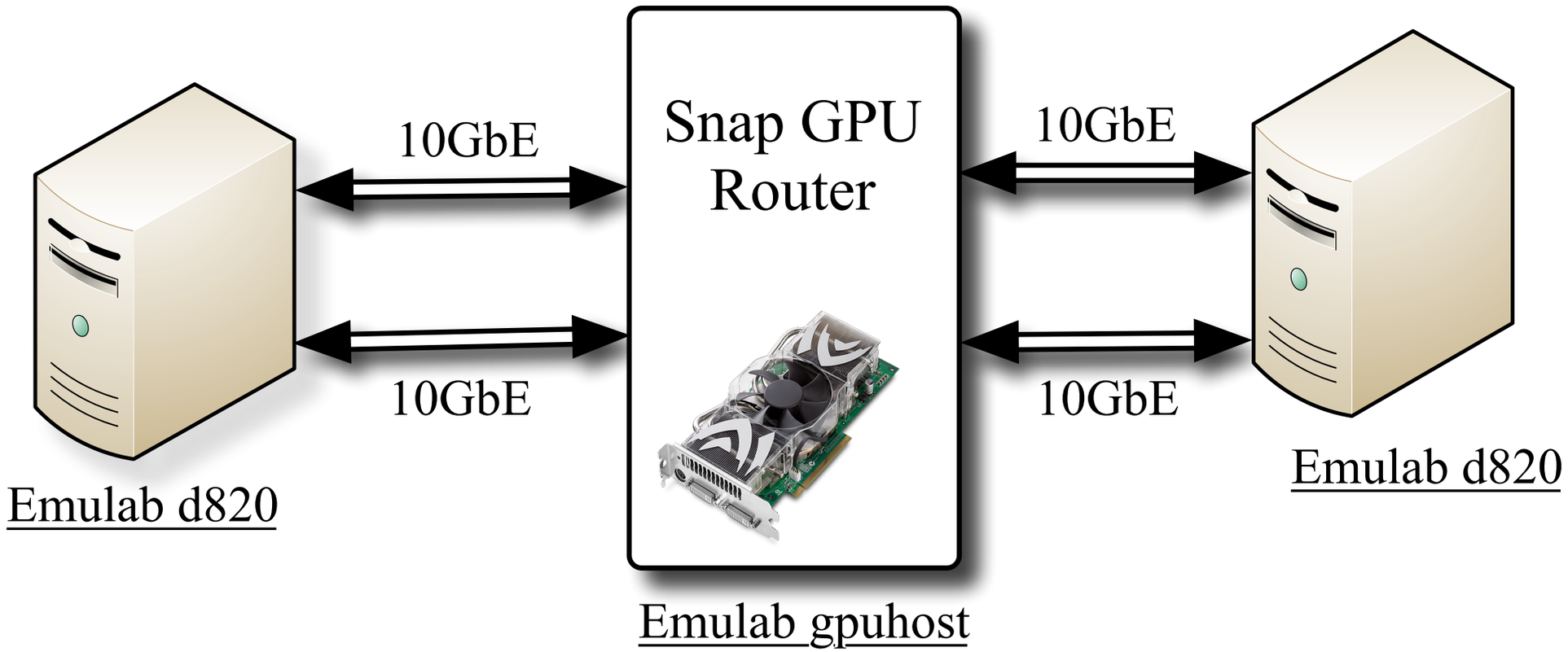
Implementation

- Netmap for packet I/O:
  - Zero-copy between kernel and user spaces
  - Multi-queue NIC support
  - 10Gb forwarding with a single low-end CPU core
  - Improved integration into Click with multi-threading support, more packet buffers
- Major packet processing elements:
  - IP Lookup, IDS Matcher, Classifier

Evaluation

- Basic forwarding
- Applications
- Full router
- Latency

Evaluation HW



Forwarding

With Netmap, 64B packets, four 10Gb NICs

Configuration	Throughput	
Click 1 Path	4.55 Gbps	6.5 Mpps
Click 4 Paths (1 thread)	8.28 Gbps	11.8 Mpps
Click 4 Paths (4 threads)	13.02 Gbps	18.5 Mpps
Snap 1 Path	8.59 Gbps	12.2 Mpps
Snap 4 Paths	30.97 Gbps	44.0 Mpps

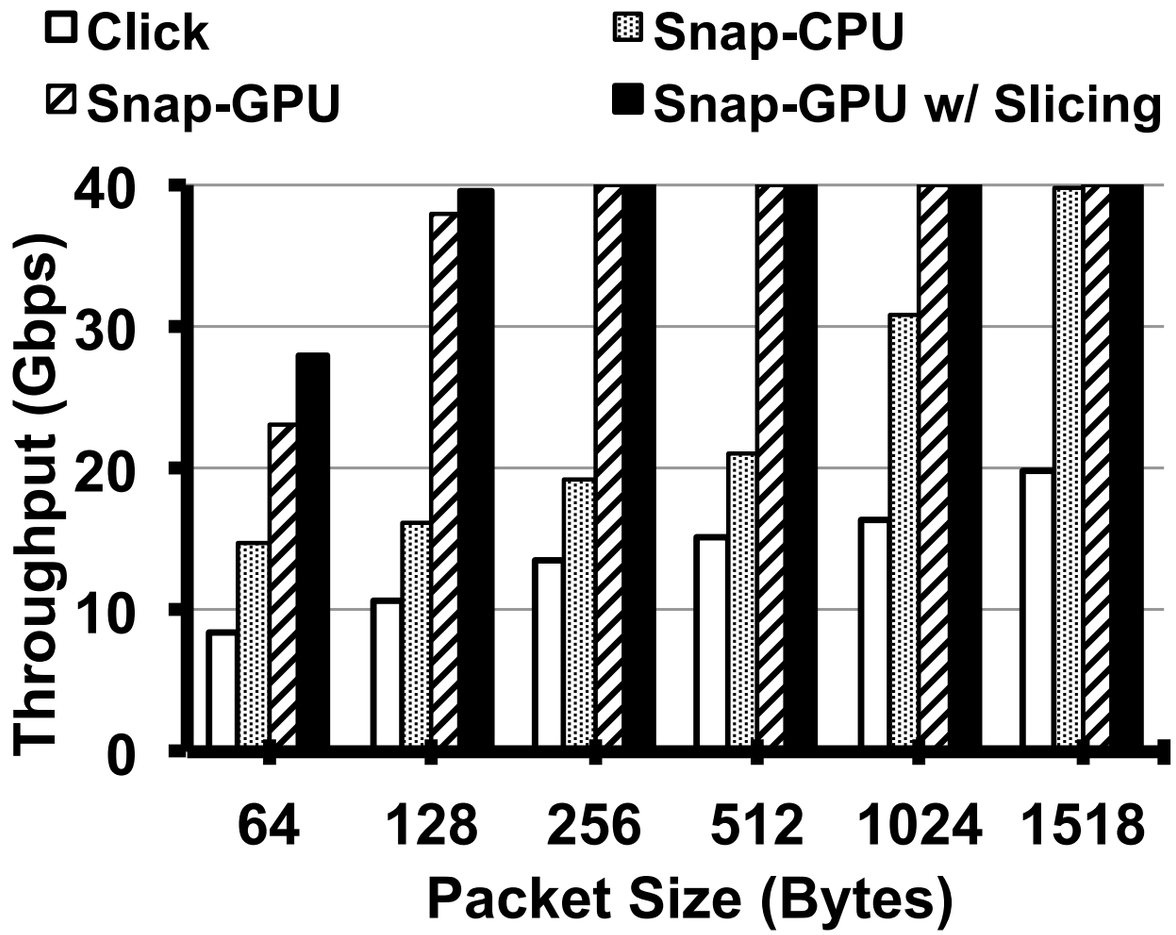
Forwarding

With Netmap, 64B packets, four 10Gb NICs

Configuration	Throughput	
Click 1 Path	4.55 Gbps	6.5 Mpps
Click 4 Paths (1 thread)	8.28 Gbps	11.8 Mpps
Click 4 Paths (4 threads)	13.02 Gbps	18.5 Mpps
Snap 1 Path	8.59 Gbps	12.2 Mpps
Snap 4 Paths	30.97 Gbps	44.0 Mpps

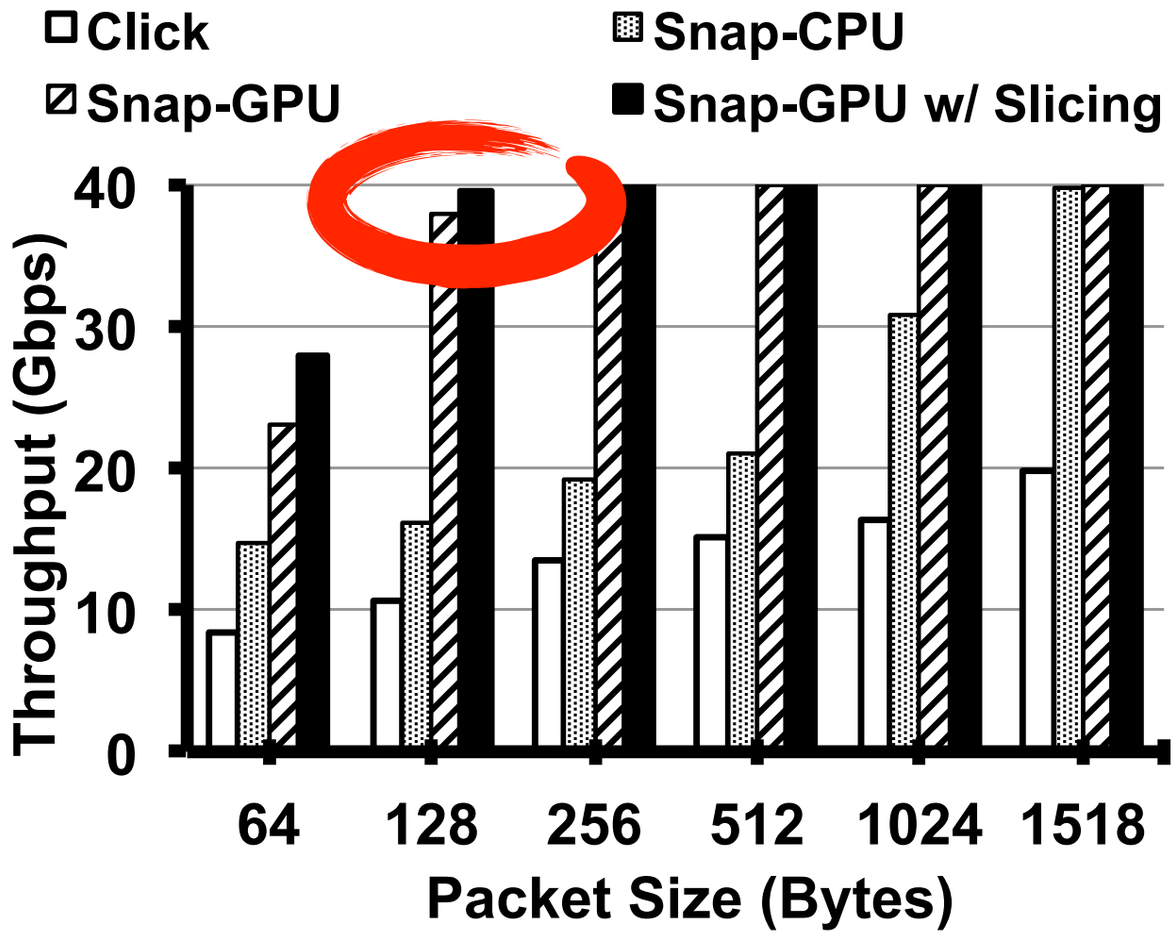
Router with IDS

- Classifier + IP Lookup + IDS Matcher
- More applications on the paper



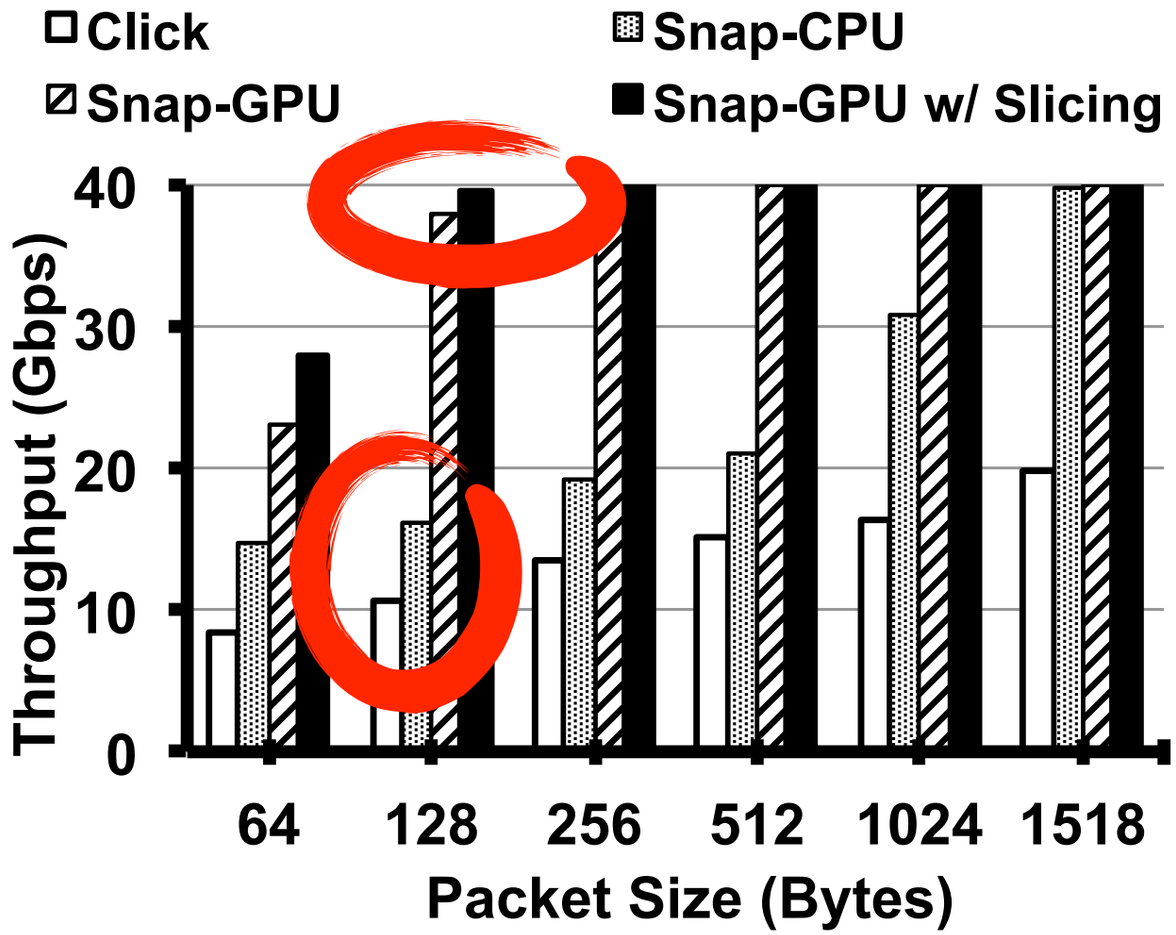
Router with IDS

- Classifier + IP Lookup + IDS Matcher
- More applications on the paper

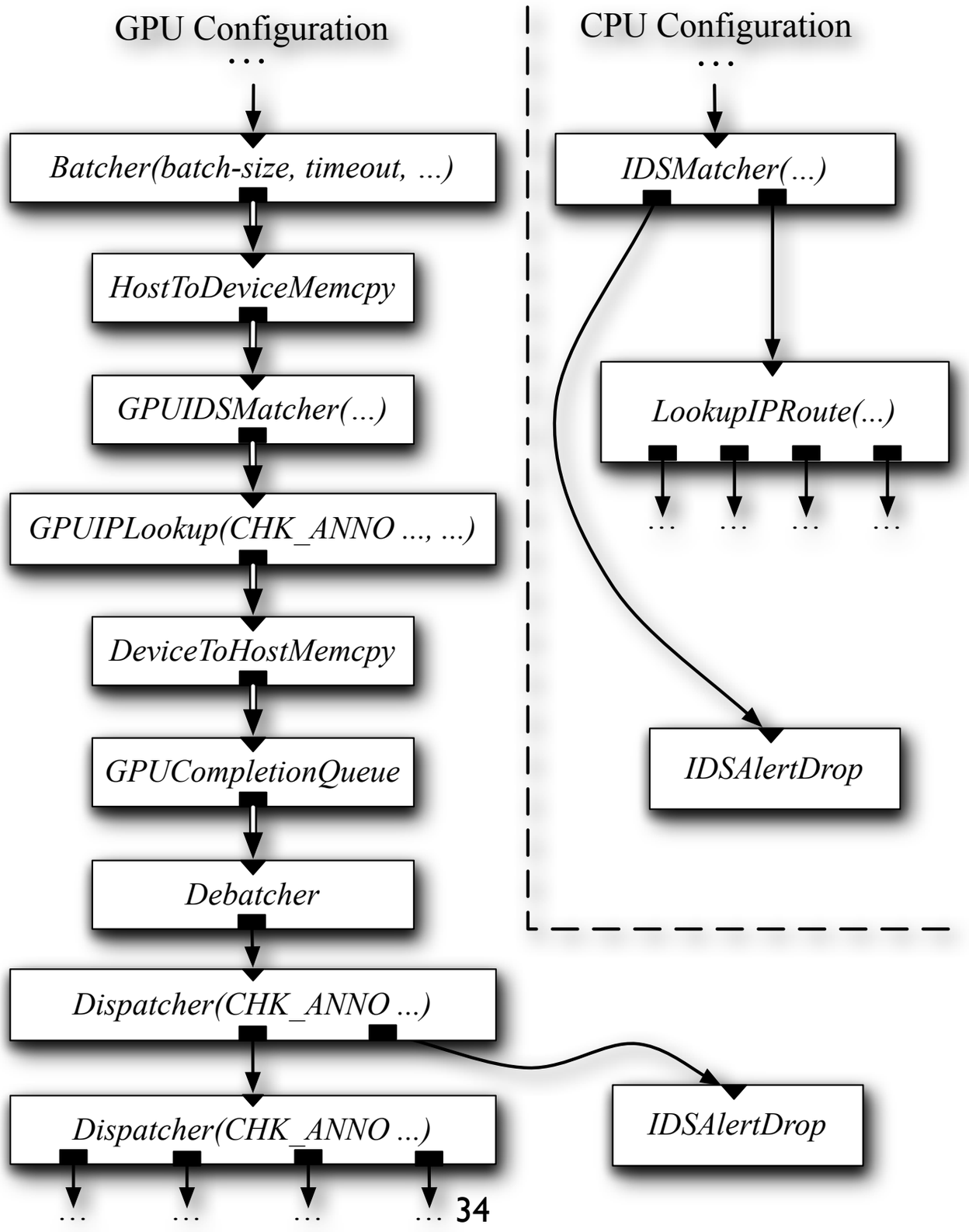


Router with IDS

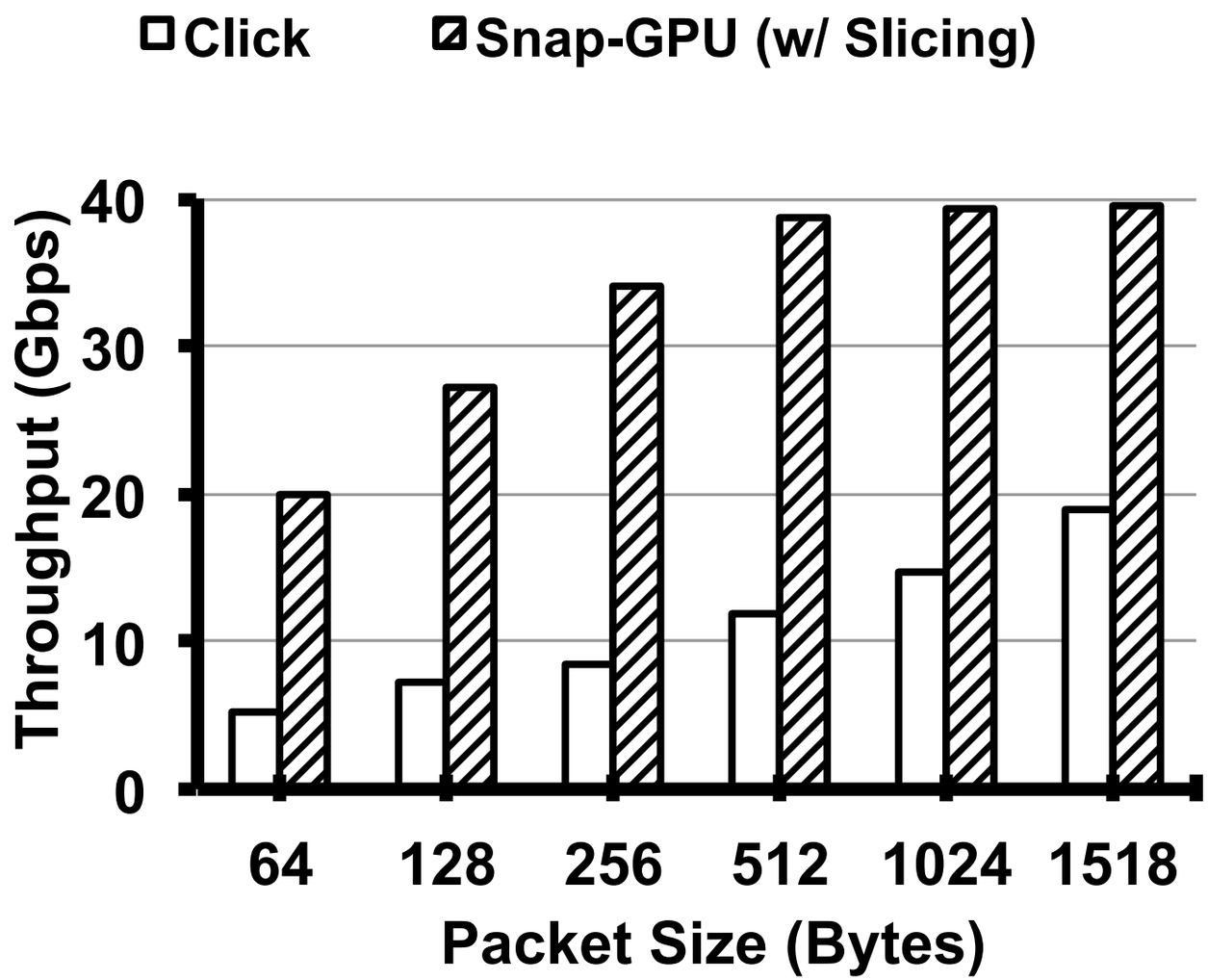
- Classifier + IP Lookup + IDS Matcher
- More applications on the paper



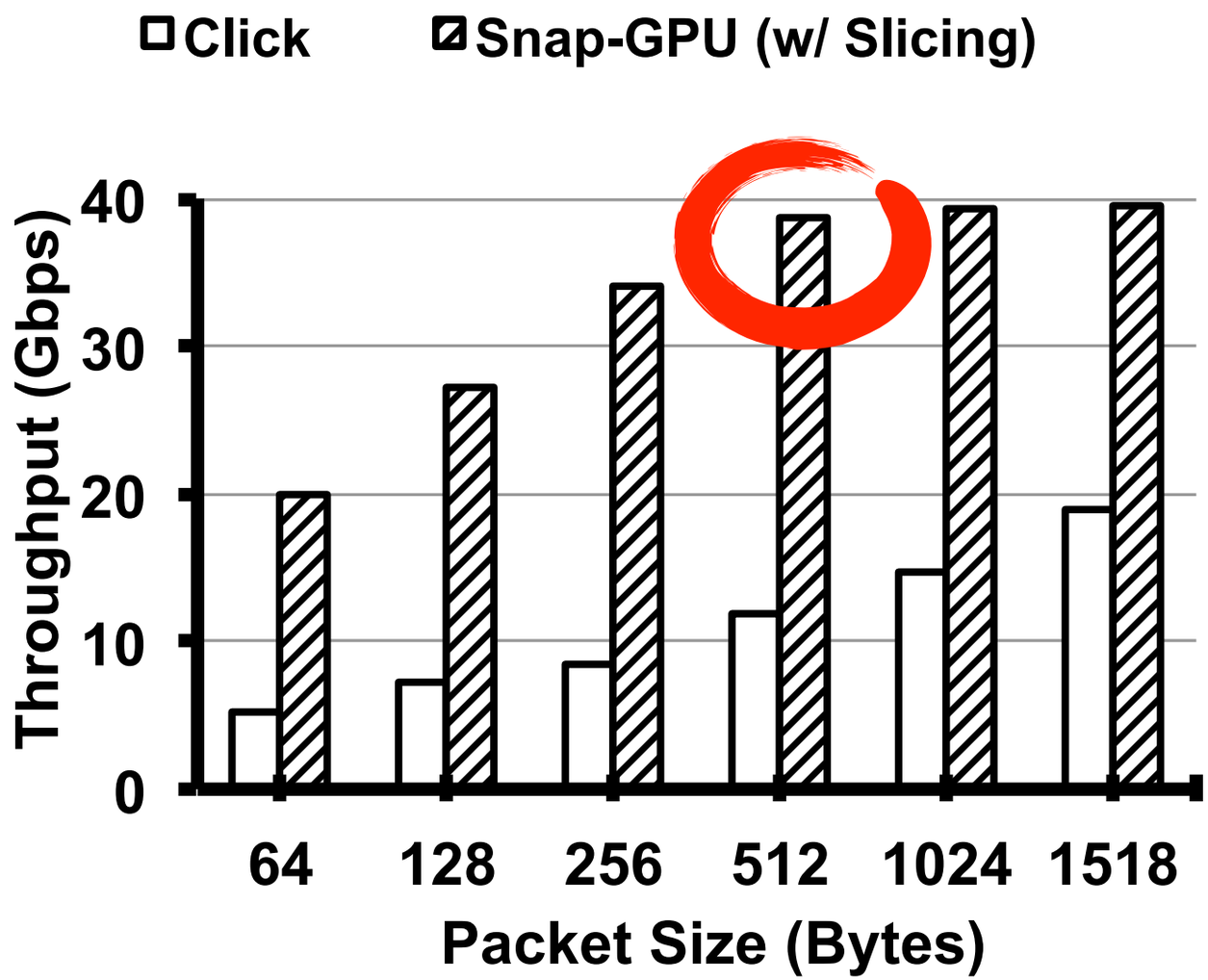
Full IDS Router



Full IDS Router



Full IDS Router



Latency

- 1024 batch:
  - CPU: 57.5us [31.4us, 320us], σ: 25.7us
  - GPU: 508us [292us, 606us], σ: 53us
- 512 batch: 380.4us average, but ~15% throughput slow down
- Tolerable in LAN, negligible in WAN.

Summary

- What we've got:
  - Fast = Fast packet I/O + Parallel GPU Processing
  - Flexible = Modular Click
- Snap: A GPU-accelerated packet processing framework
- Application elements

Future Work

- More NICs to saturate PCIe 3.0 GPU's algorithm performance.
- Investigating other overhead in the full router

Thanks!

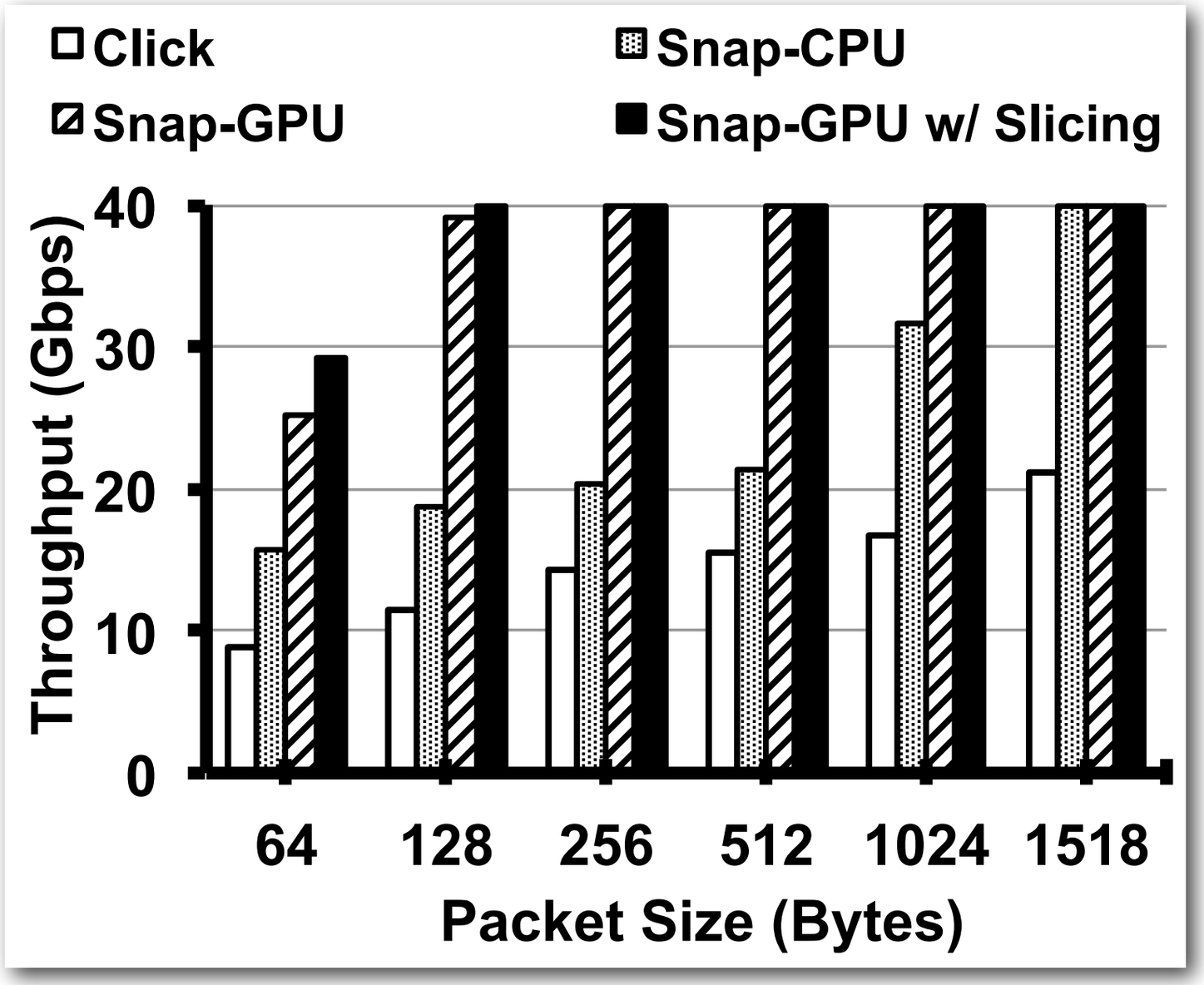
Questions?!

Snap Source: https://github.com/wbsun/snap

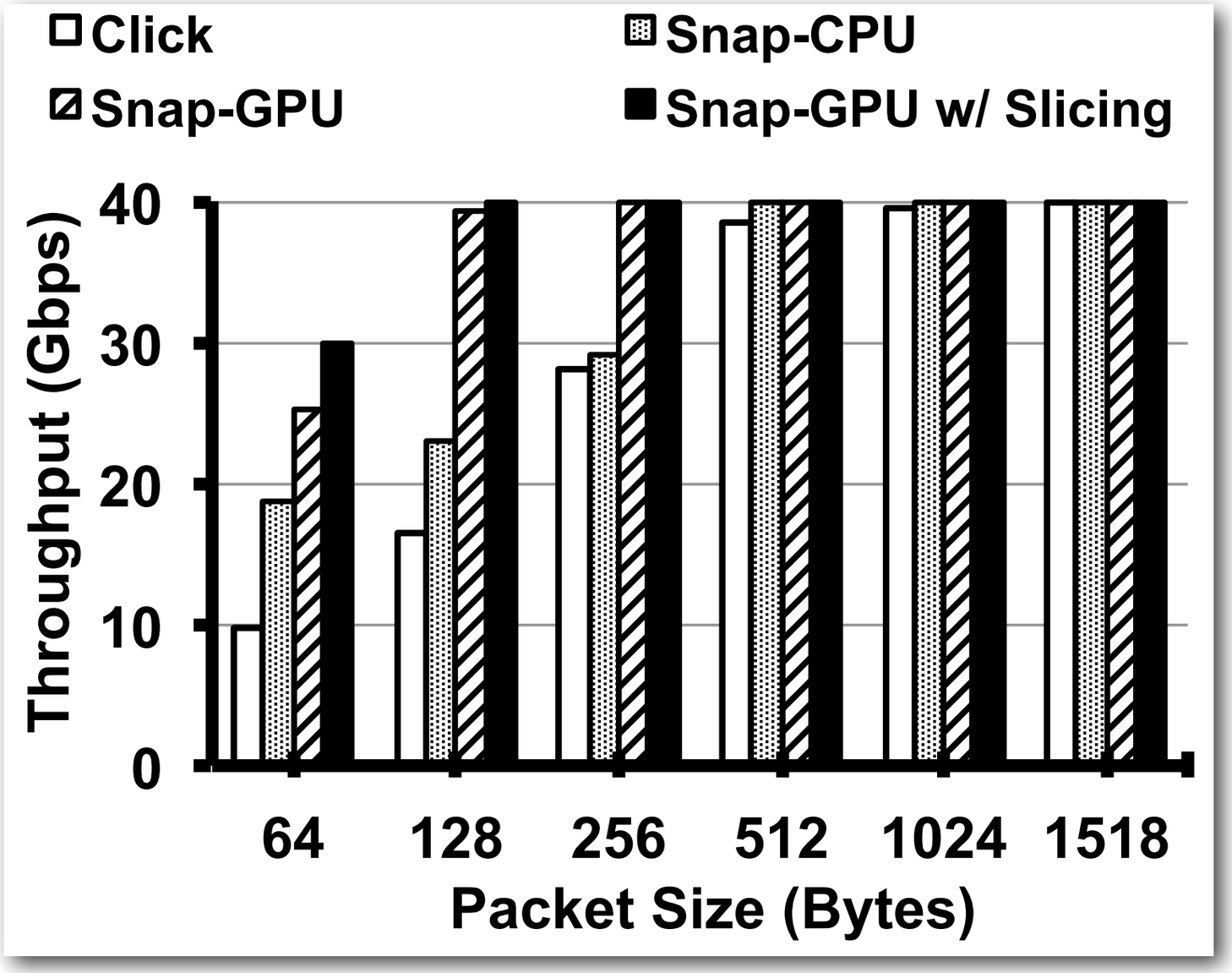
Snap Is Different...

- Based on Click, rather than an ad-hoc way
  - Complete TCP/UDP/IP/Ethernet/ARP/ ICMP/Wireless, etc...

DPI Router


Click
Snap-CPU
Snap-GPU
Snap-GPU w/ Slicing
Throughput (Gbps)
40
30
20
10
0
64
128
256
512
1024
1518
Packet Size (Bytes)

SDN Forwarder



Outline

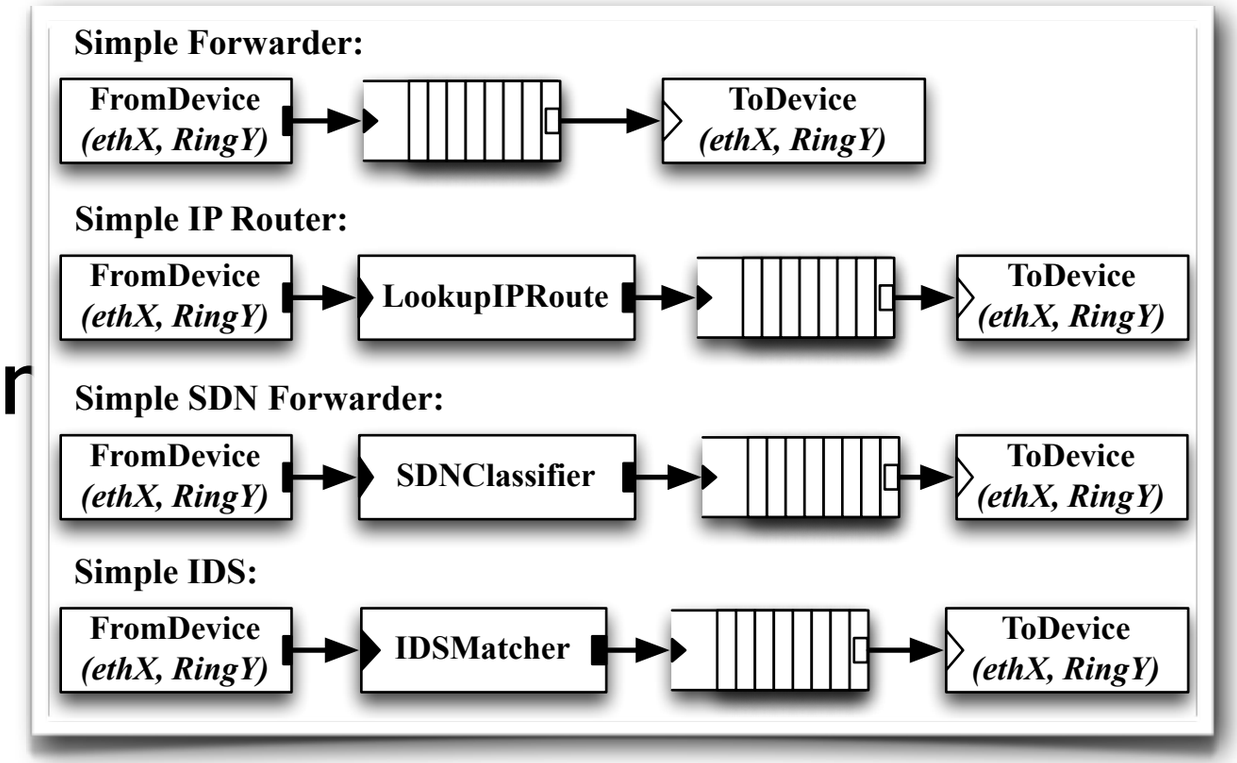
- Why GPU?
- Why Click?
- Snap Design and Implementation
- Snap Evaluation

Why GPU?

- Previous Work: PacketShader, Gnort, GPUstore
- Parallel GPU excels at SIMD computing
- Motivating experiments that compared the CPU and the GPU for packet processing algorithms.

Why GPU?

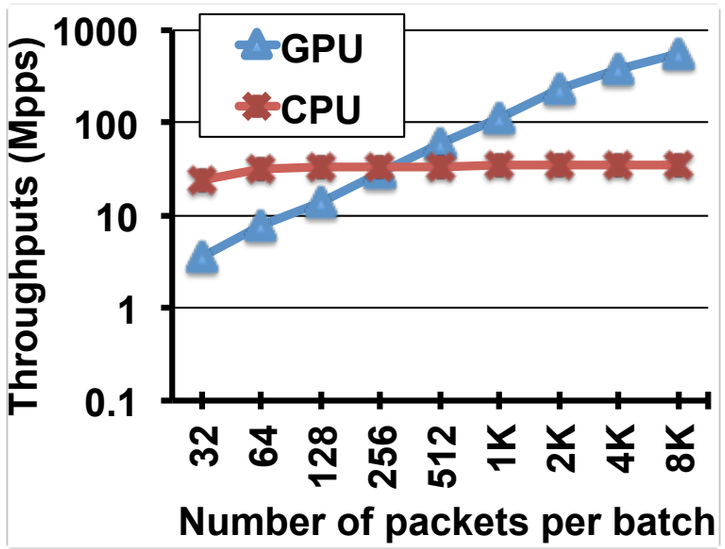
- Motivating experimer



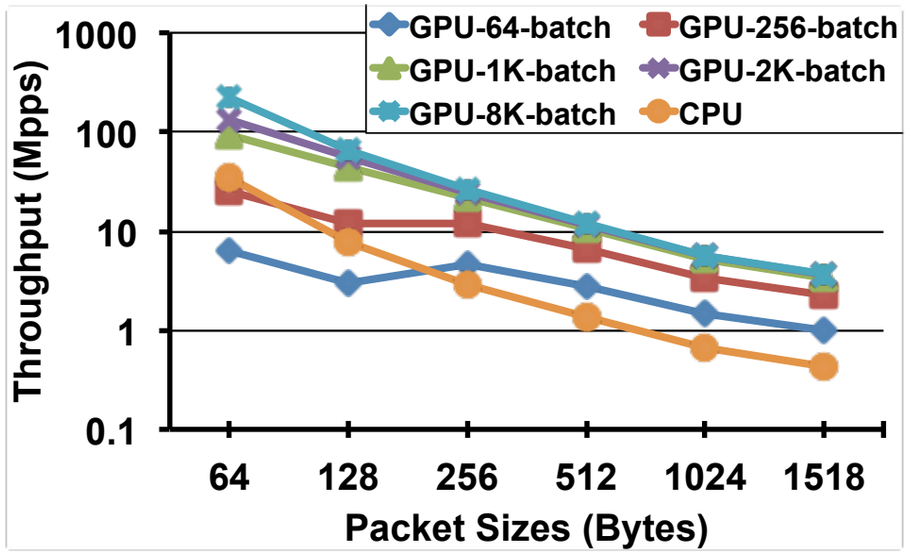
Configuration	Throughput	Relative TPut
Simple Forwarder	30.97 Gbps	100%
IP Router	19.4 Gbps	62.7%
IDS	17.7 Gbps	57.3%
SDN Forwarder	18.8 Gbps	60.7%

Why GPU?

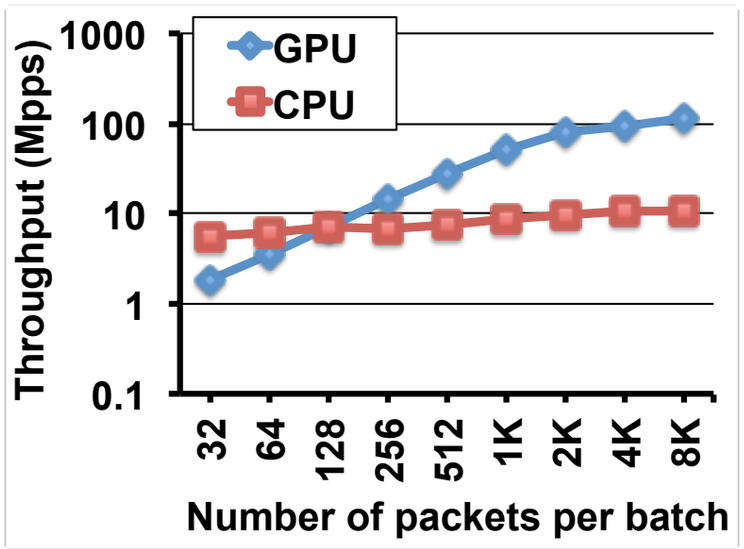
- Algorithm performance comparison:



IP Lookup



IDS Matcher



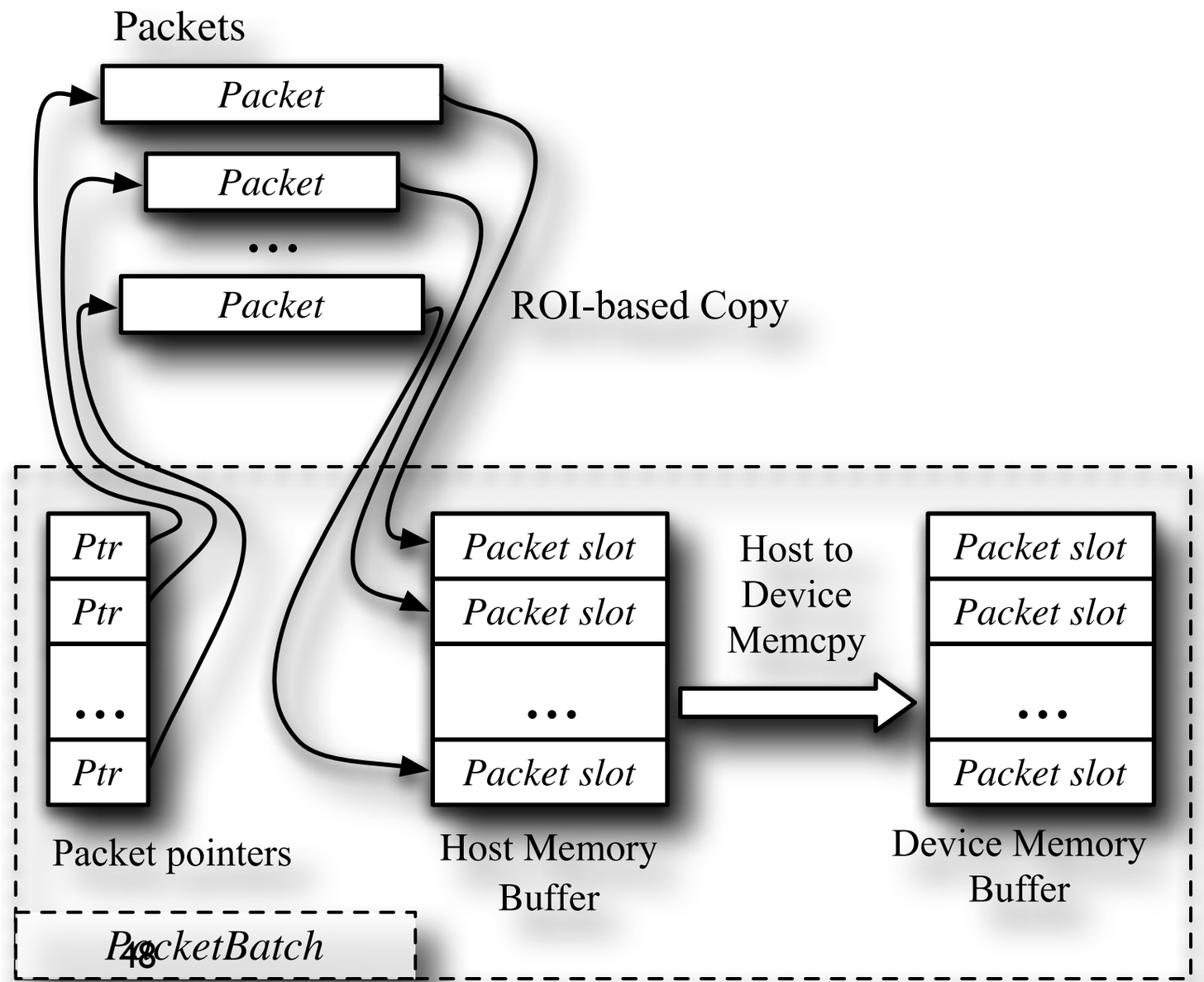
SDN Classifier

Why Click

- Modular, Flexible
  - Compassable processing pipelines with Elements.
- Easy to extend with CUDA for GPU computing
  - Simply linking with new CUDA code libs.
- Fast user-space packet processing
  - Netmap showed high performance user-space Click
- Many existing elements
  - All layers, including wireless!

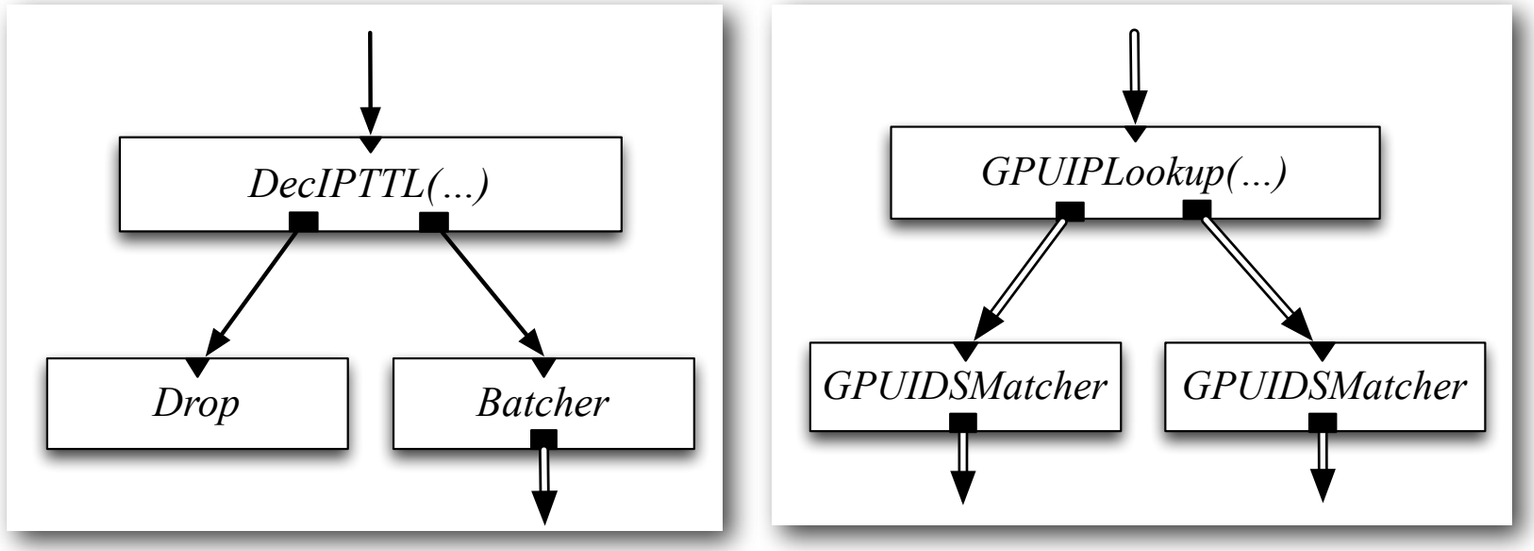
Batched Processing

- Wider pipeline: ***`BElement`*** with ***`bpush/bpull`***



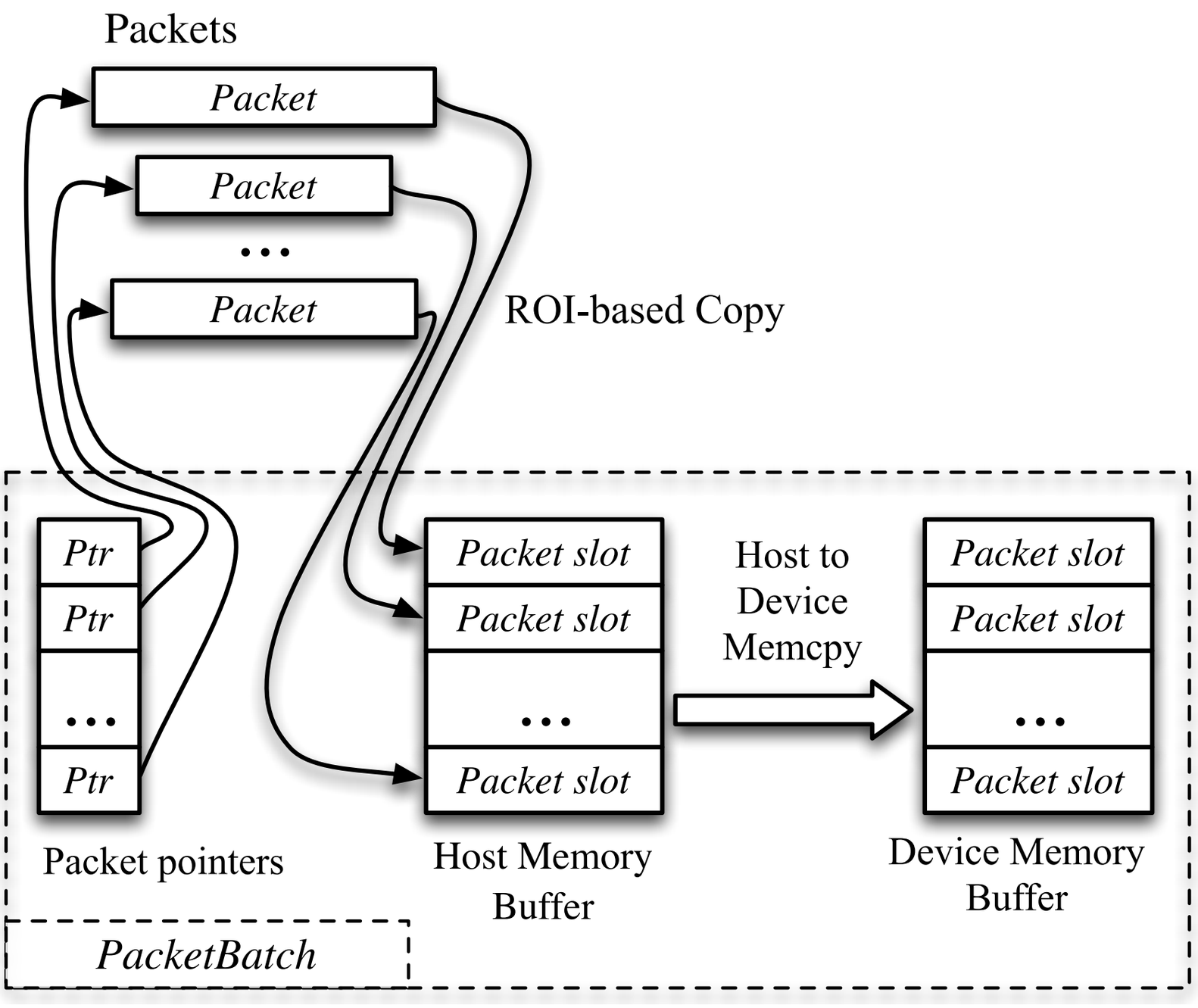
Packets Divergence

- Why diverge?



- Solution:
  - Return to CPU
  - Predicate bits

PacketBatch

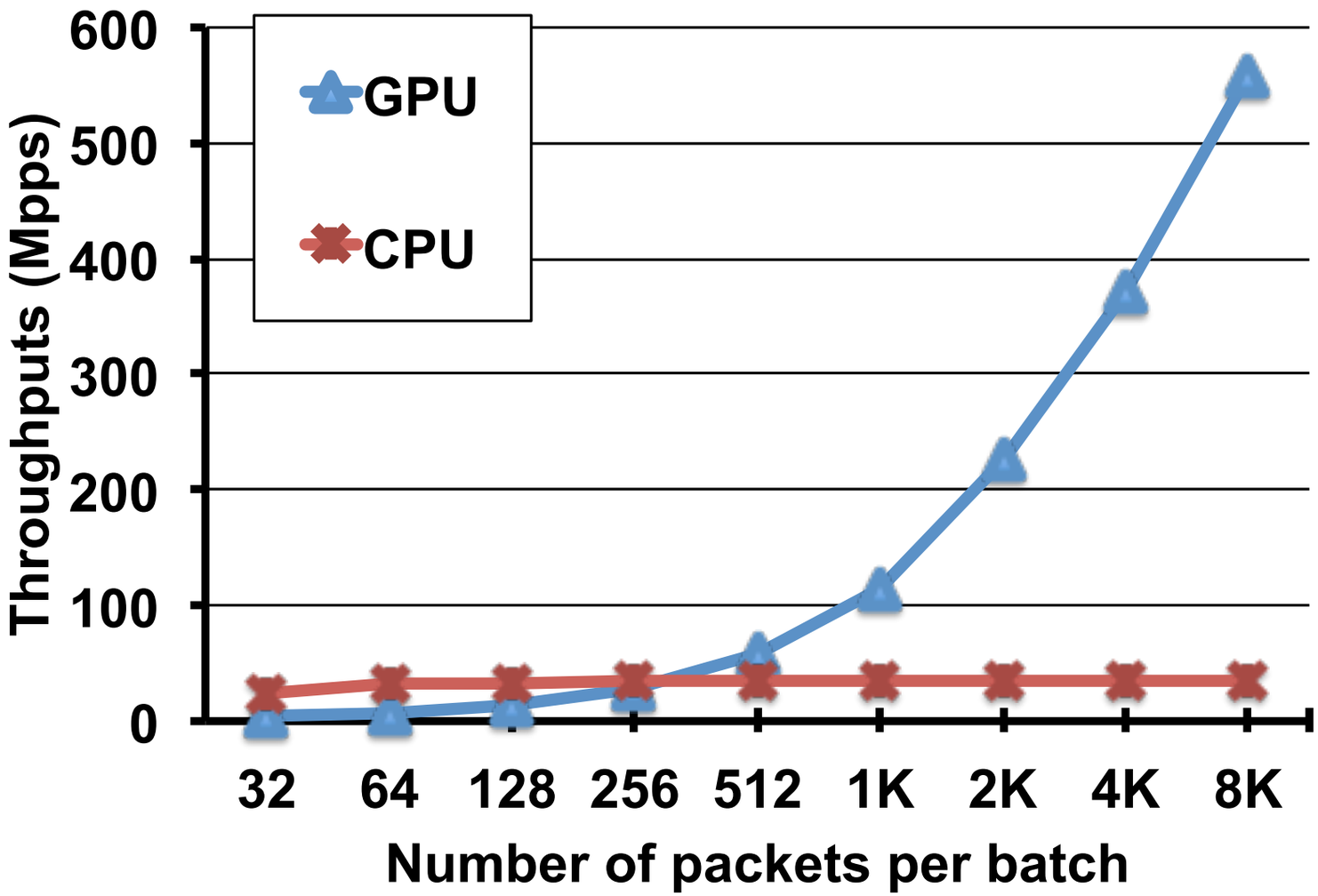


Packets Divergence

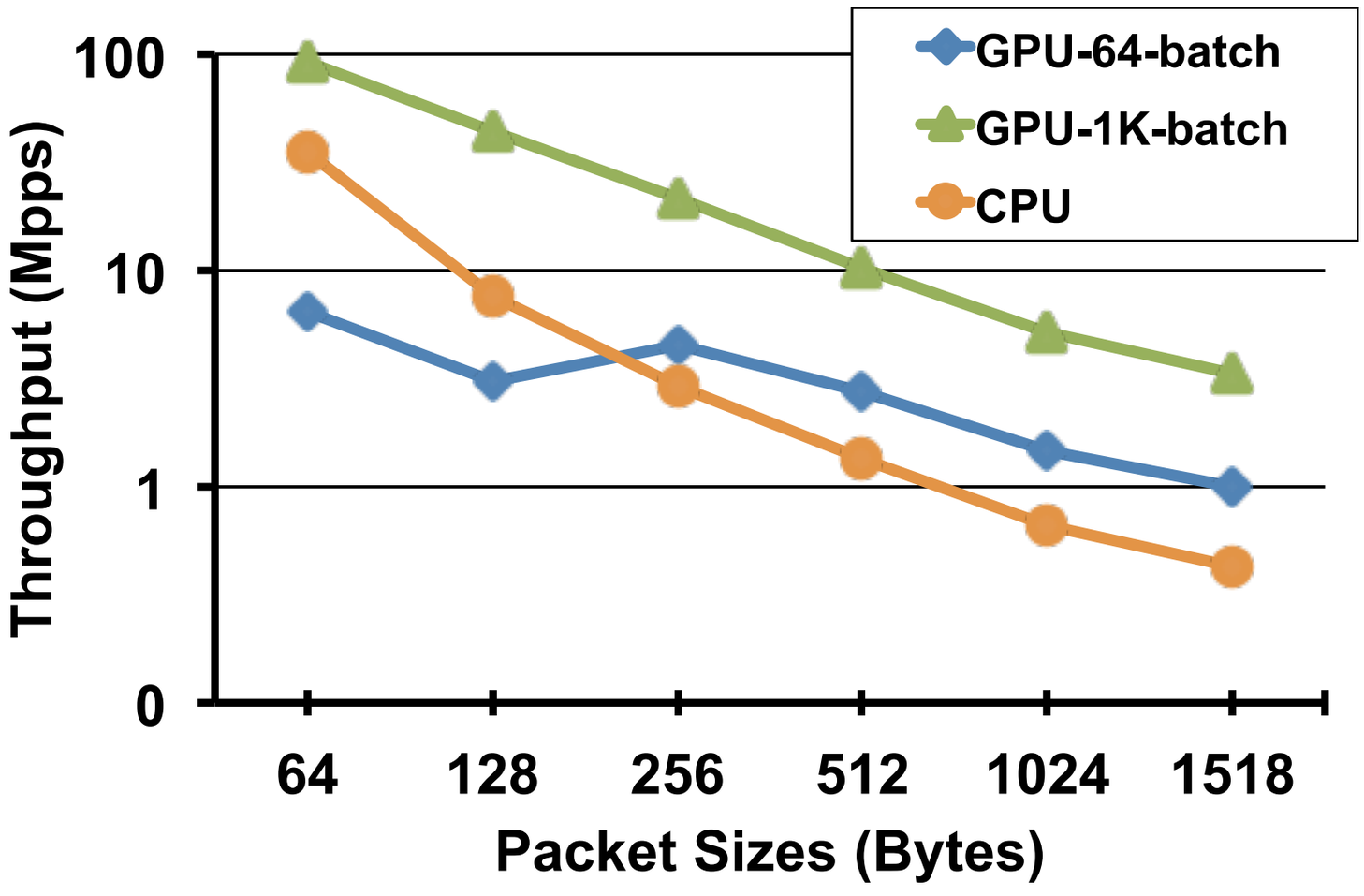
Packets in the same batch are sent to different elements

IP Lookup

IP Lookup


600
500
400
300
200
100
0
Throughputs (Mpps)
GPU
CPU
32
64
128
256
512
1K
2K
4K
8K
Number of packets per batch

IDS Matcher


GPU-64-batch
GPU-1K-batch
CPU
100
10
1
0
Throughput (Mpps)
64
128
256
512
1024
1518
Packet Sizes (Bytes)