Hardware-based Flow Offload in Suricata

Alfredo Cardigliano
cardigliano@ntop.org
@acardigliano

ntop

SURICON PRAGUE 2017

About ntop

- ntop develops open source network traffic monitoring applications

- ntop is also the name of the first app we released (1998), a web-based network monitoring application

- Today our products range from traffic monitoring (ntopng, nProbe), high-speed packet capture (PF_RING), deep-packet inspection (nDPI), traffic recording (n2disk), DDoS mitigation (nScrub), IDS/IPS acceleration (capture modules for Suricata, Bro, Snort)

Our Approach

- Operate at line rate, any packet size
- Promote scalability, by leveraging on modern multi-core, NUMA architectures
- Use commodity hardware for producing affordable, long-living (no vendor lock), scalable (use new hardware by the time it is becoming available) monitoring solutions
- Use open source to spread the software, allow contributions and let the community test it on uncharted places

PF_RING [1/2]

- PF_RING is a home-grown open source packet processing framework for Linux

- When it was born, PF_RING was designed to accelerate packet capture on commodity hardware without using costly FPGA-based network adapters

- Later we introduced DNA (Direct NIC Access) featuring kernel-bypass for line rate RX/TX packet processing

- After DNA we created the new generation kernel-bypass technology, named PF_RING ZC (Zero-Copy), improving efficiency and flexibility

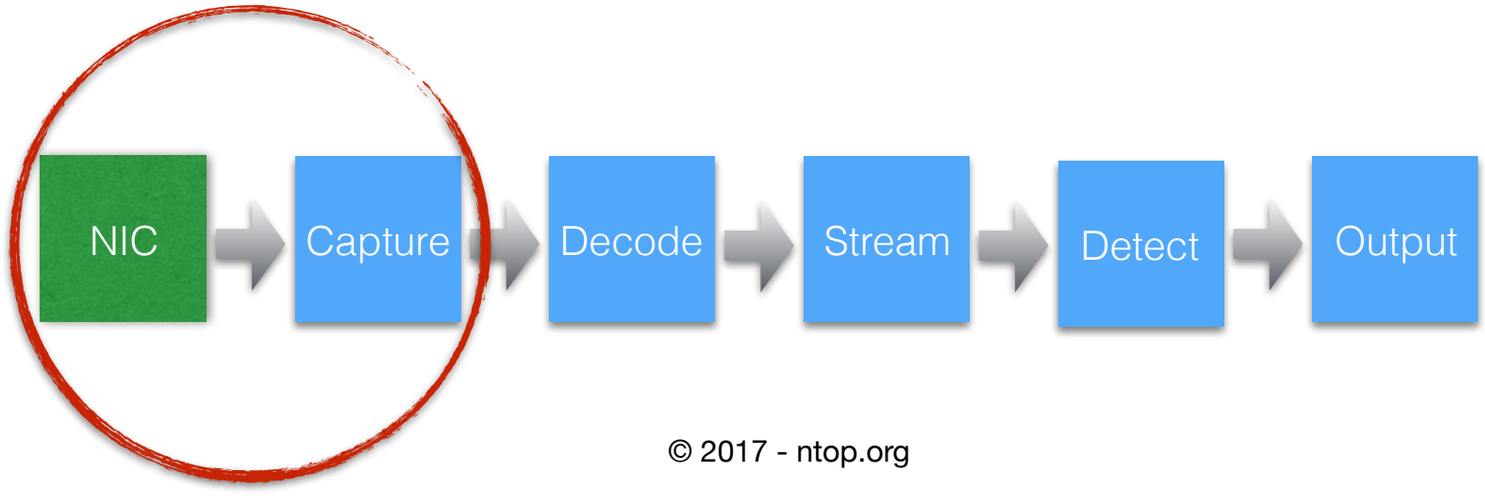
PF_RING [2/2]

- In the last years we worked hard on providing multi-vendor support, today PF_RING can be used with almost all the FPGA-based adapters on the market, providing an abstraction layer that makes packet-processing applications completely hw agnostic.

- Last month we released PF_RING 7.0 where we focused on improving hw offload, in order to help cpu-intensive applications to cope with high speed links, as we will see later in this talk.

In The Previous Episode..

- Last time we met at Suricon we focused on packet capture acceleration in Suricata, unlocking the capture component to be able to:
  - Capture at line-rate (14.88Mpps at 10 Gbit) on commodity hw (Intel NICs on a low end Xeon)
  - Free CPU cycles for the real processing

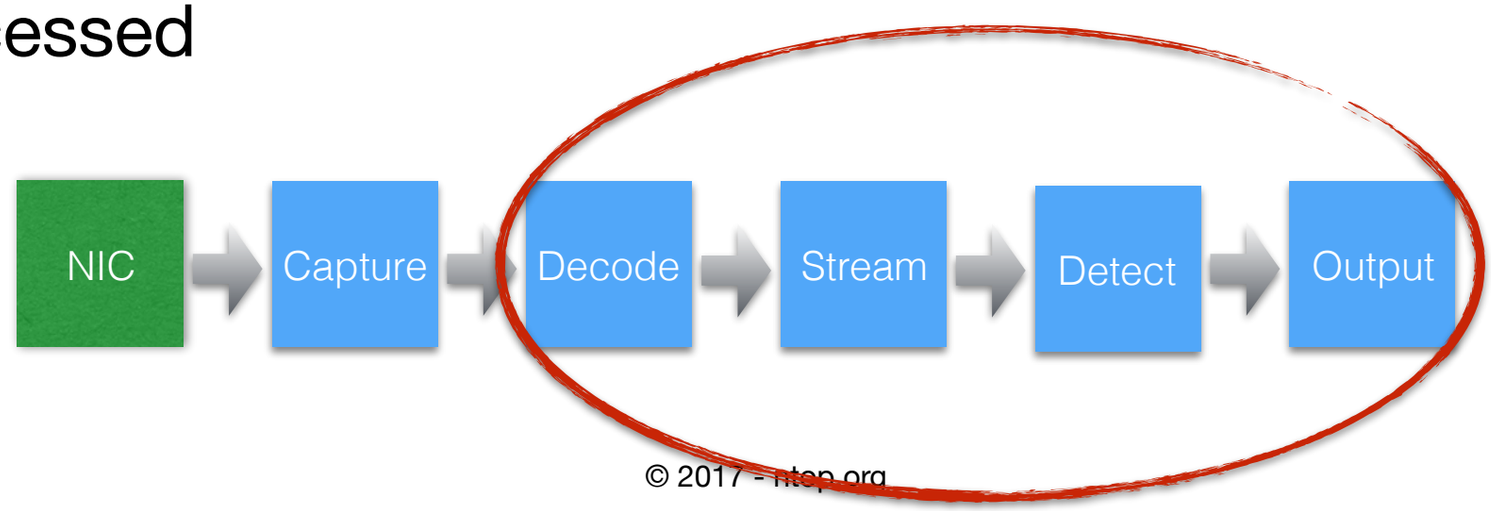


ntop

SURICON PRAGUE 2017

In This Episode

- Today we want to focus on accelerating the processing side
- Suricata is a CPU bound application, thousands of rules to be evaluated for every packet
- Content scanning is CPU intensive and presents significant challenges to network analysis applications
- Performance is affected by the number of packets per second to be processed

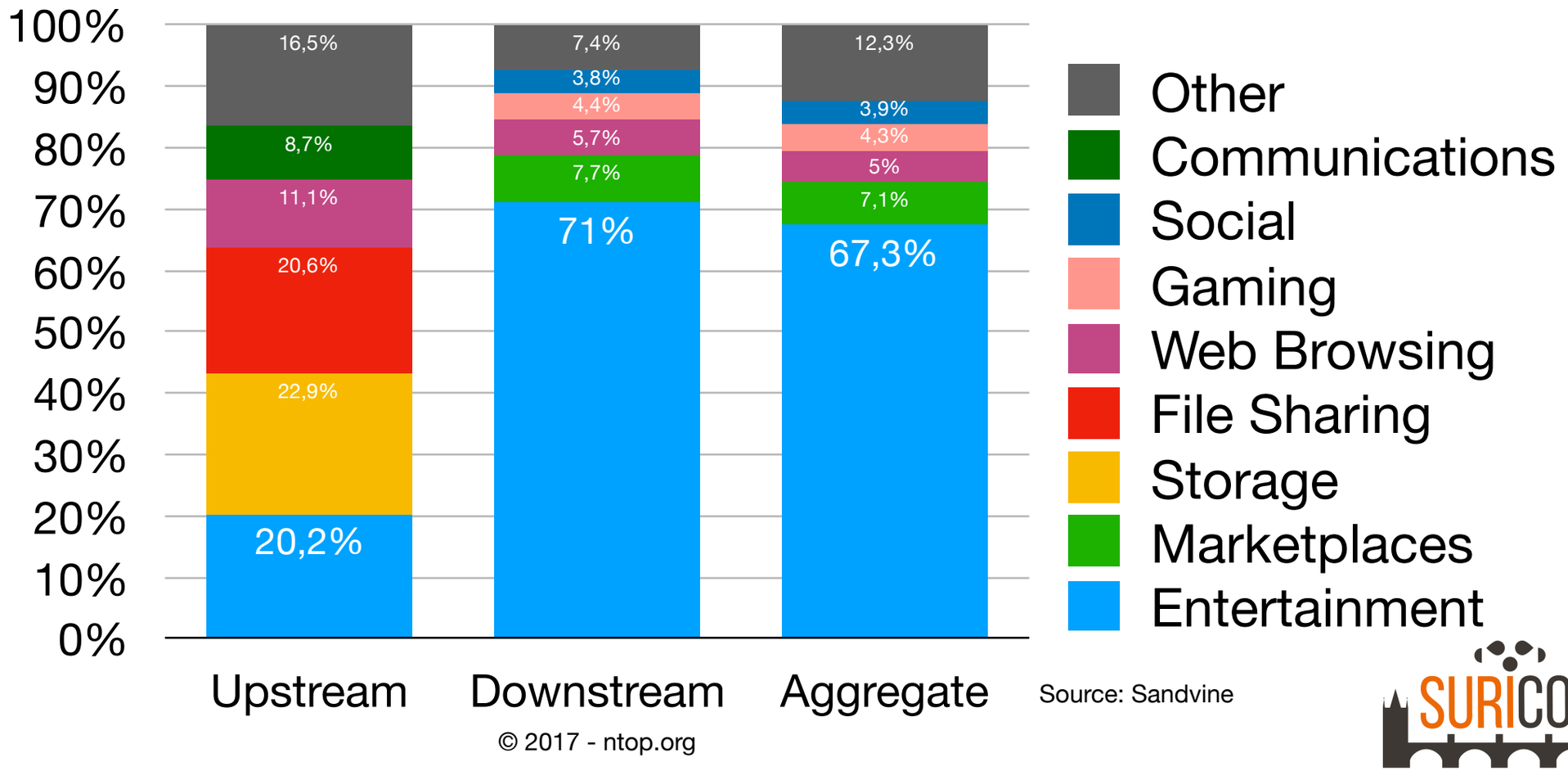


Processing Takes Time

- Thousands of rules to be evaluated for every packet
- Doing some math we have 3 options:
  - Reduce the number of rules (is this an option? :-))
  - Accelerate rule evaluation with pattern matching acceleration (GPU/CUDA, Hyperscan, ..)
  - Reduce the number of packets to be processed (static traffic filtering?)

Internet Traffic [1/2]

- ~70% of Internet traffic during peak hours in North America comes from video and music streaming



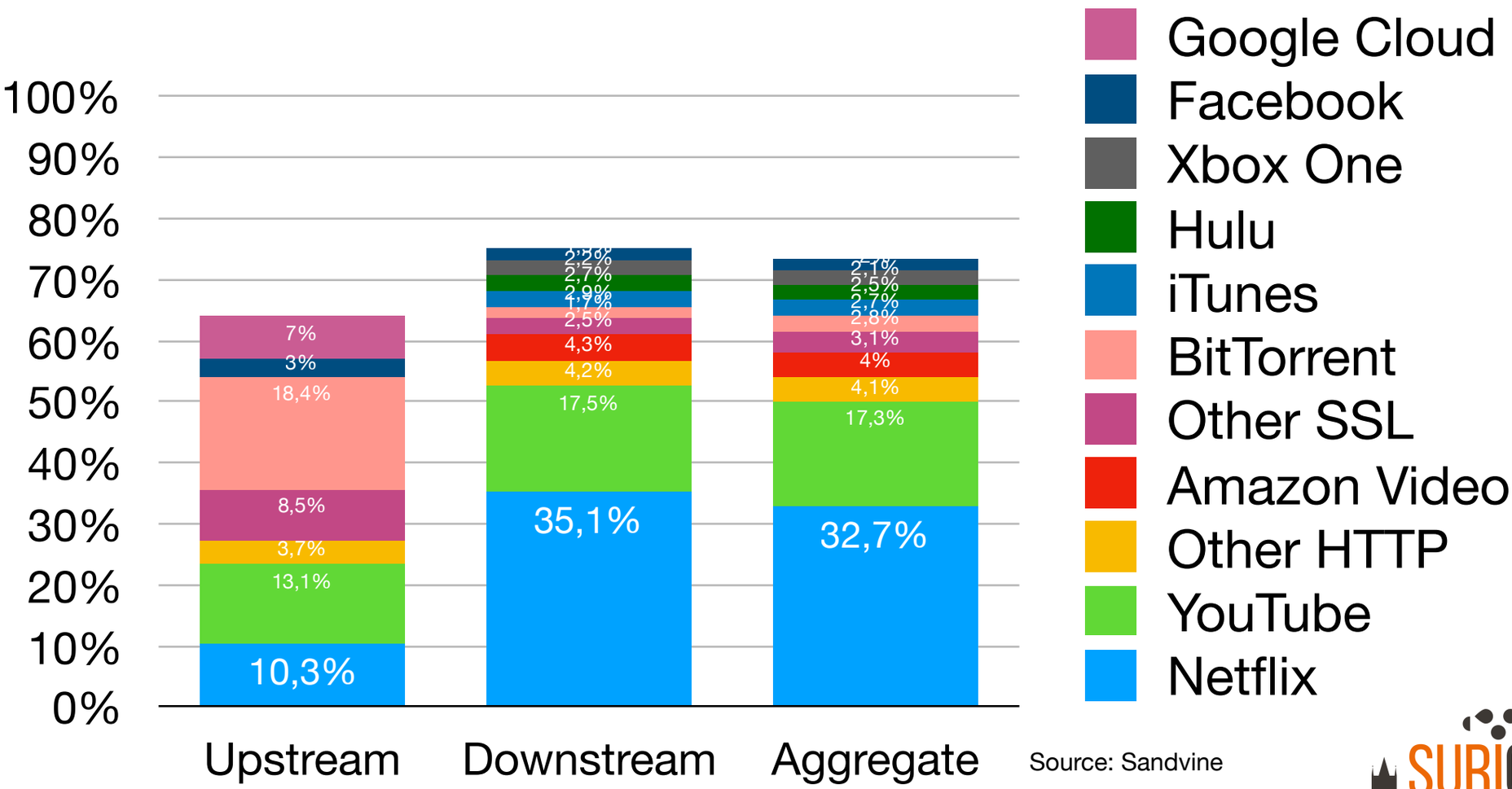
Source: Sandvine

ntop

SURICON PRAGUE 2017

Internet Traffic [2/2]

- Netflix reached ~35% share of Internet traffic in North America, half of the whole multimedia traffic



Source: Sandvine

ntop

SURICON PRAGUE 2017

The Netflix Example

- Watching a movie on Netflix with standard quality your browser exchanges more than 2M packets / 2 GByte of data

- Do we really need to inspect all Netflix traffic?

- The number of packets needed to identify all Netflix flows during a movie session is ~0.02% of the total traffic

- If we are able to discard Netflix flows as soon as they are recognised we remove 99.98% of the load needed for processing Netflix traffic (1/3 of total traffic)

Bypass Support in Suricata

- Suricata can be configured to enable `stream.bypass` based on `stream.reassembly.depth` or by means of `bypass` rules:

```
alert http any any -> any any (msg:"Bypass youtube”;
content:"youtube.com"; http_host; bypass; sid:1; rev:1;)
```

- Suricata can instruct the capture module to bypass a flow using a callback, local bypass is used otherwise:

```
if (p->BypassPacketsFlow && p->BypassPacketsFlow(p)) {
    FlowUpdateState(p->flow, FLOW_STATE_CAPTURE_BYPASSED);
} else {
    FlowUpdateState(p->flow, FLOW_STATE_LOCAL_BYPASSED);
}
```

Software Bypass [1/2]

- Flow bypass (aka offload) is currently implemented in Suricata in the NFQUEUE and AFPACKET (using eBPF) modules to reduce the overhead for packet capture and decoding when kernel drivers are in use

- Suricata relies on another software component at an earlier stage, reducing the overhead for moving packets to userspace, but still requiring CPU cycles for packet capture and evaluation

- “Local bypass” (with packet decoding in Suricata) is used as fallback in case the capture module does not support bypass

- If you want to boost the capture performance by installing a FPGA card, local bypass is the only option

Software Bypass [2/2]

- At 40 Gbit minimum frame inter-arrival time is ~16 nsec, at 100 Gbit ~6 nsec (148 Mpps!)

- Even considering real Internet traffic with avg packet size >900 bytes, you still have 13+ Mpps to process at 100G

- Overwhelmed by the traffic rates of 40/100G networks, it's likely to loose traffic!

Flow Processing and Hardware [1/3]

- In the past decade nothing really happened in networking hardware beside speed bump:
  - People talk about application protocols and metrics, NIC manufacturers about packet headers
  - NICs are mostly stateless and the most they can do is to filter packets at header level, provide metadata, load-balance based on packet hash

Flow Processing and Hardware [2/3]

- Most network monitoring applications are based on the concept of flow

- Firewalls, IPS/IDS, monitoring tools, all work the same way:
  - Decode packet headers
  - Compute a hash
  - Find the flow bucket in a hash table
  - Update the flow status, and do some action

Flow Processing and Hardware [3/3]

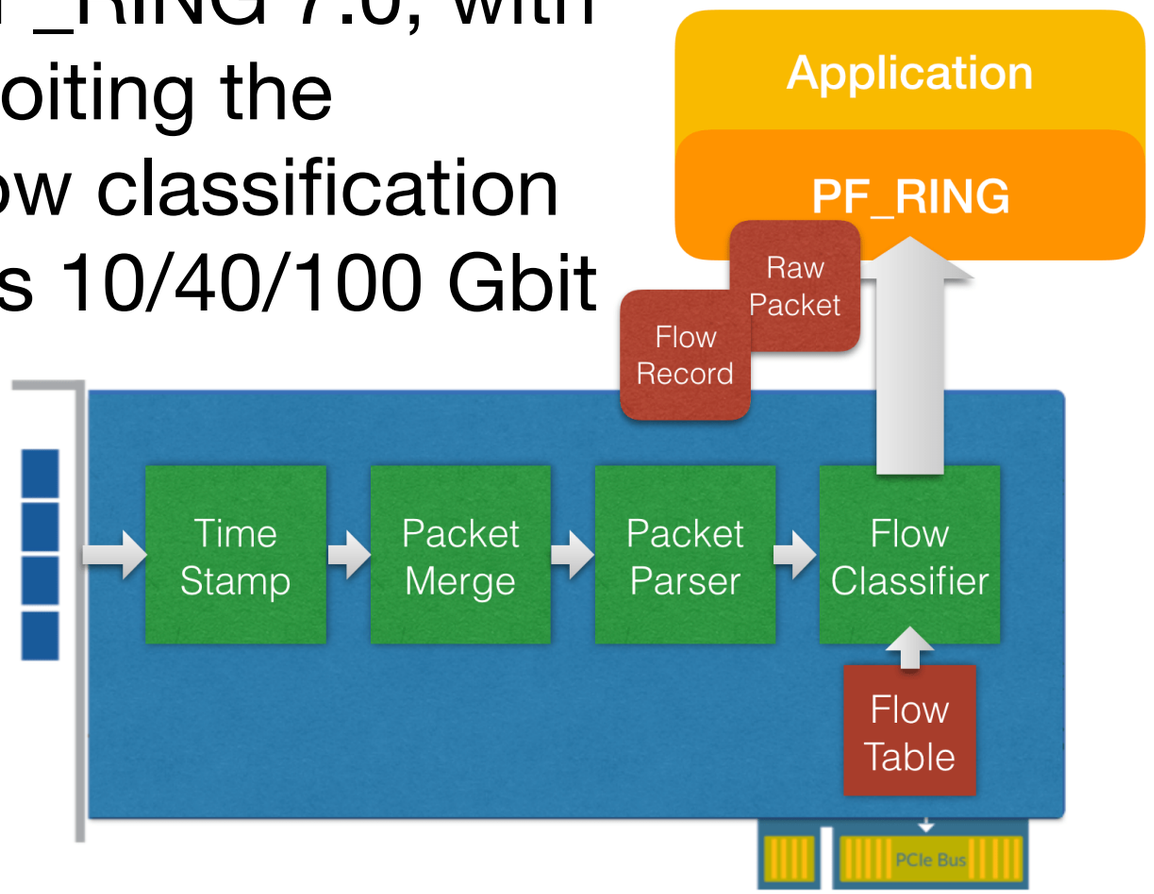
- In order to accelerate flow processing advanced hardware NICs provide metadata that include 5-tuple and packet hash

- Software applications are responsible for doing all the rest

- Question: can we offload some more flow-oriented tasks to the NIC, doing some flow classification and keeping flow state in hardware?

Flow Offload

- In the last couple of years we worked with Accolade on prototyping a network card able to classify flows in hw
- Last month we announced PF_RING 7.0, with hw flow offload support, exploiting the Accolade hardware-based flow classification engine, available on Ku Series 10/40/100 Gbit network adapters
- We created a Generic API in order to be hardware agnostic (support in other NICs will be transparent to the application)

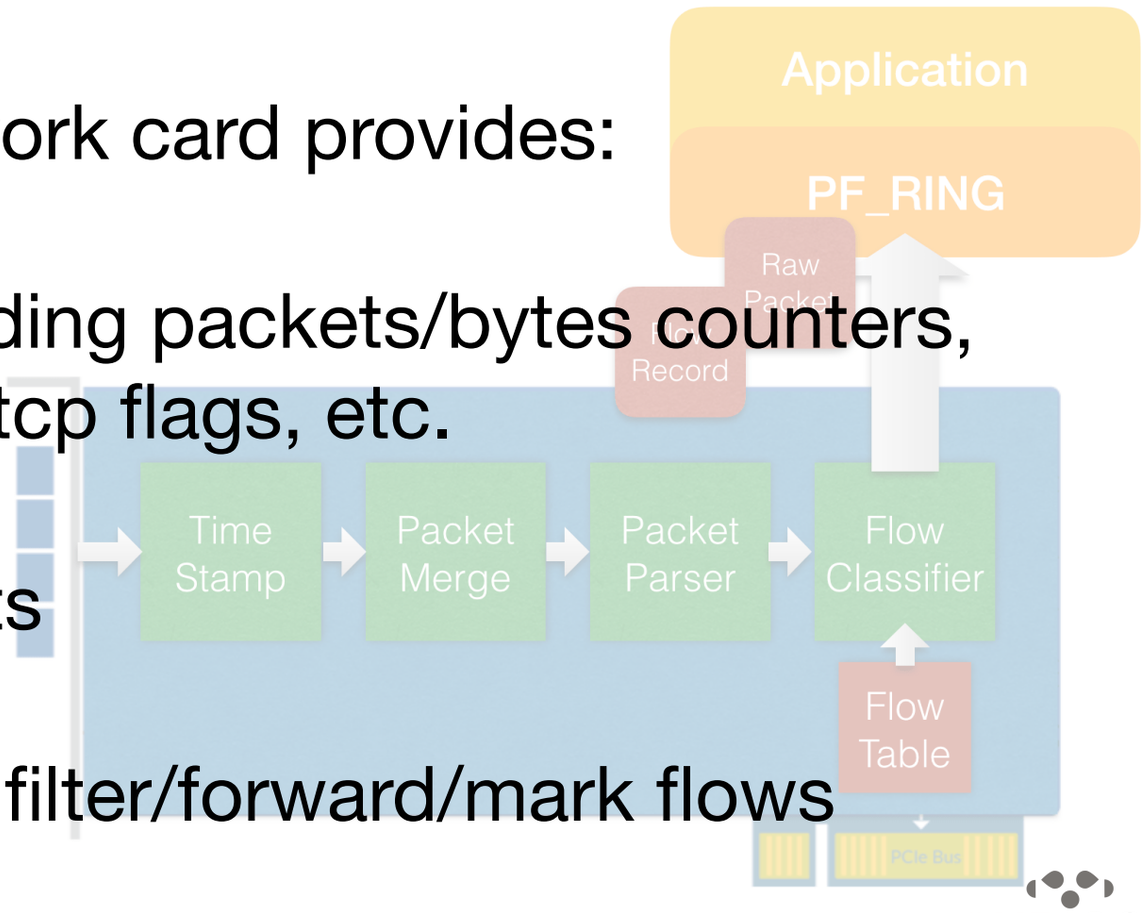


What Flow Offload Provides

- Packet decoding and flow classification happens in hw, where flow stats are updated locally (hw is really good with this!)

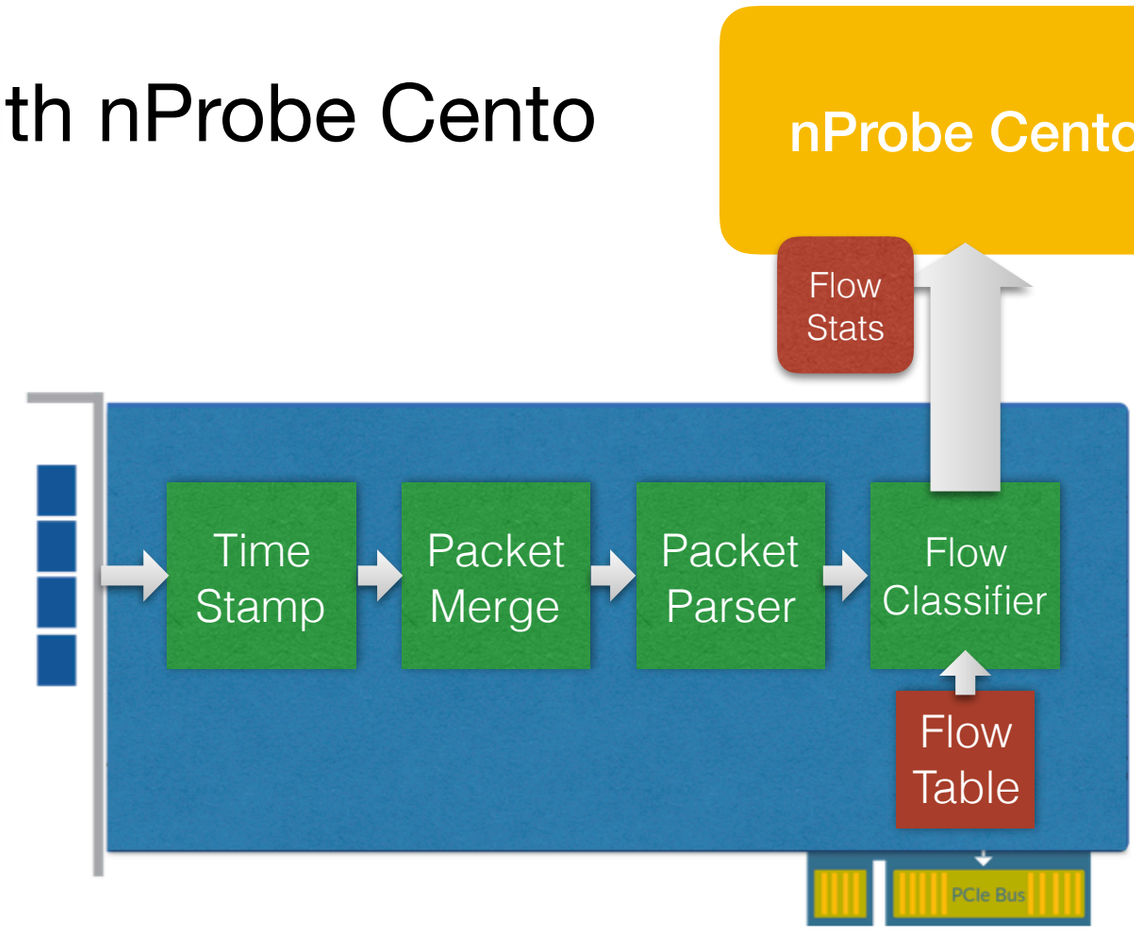
- Enabling flow offload, the network card provides:
  - Periodic flow updates, including packets/bytes counters, first/last packet timestamp, tcp flags, etc.
  - Tagged (Flow ID) raw packets

- Hardware can be instructed to filter/forward/mark flows (enabling shunting!)



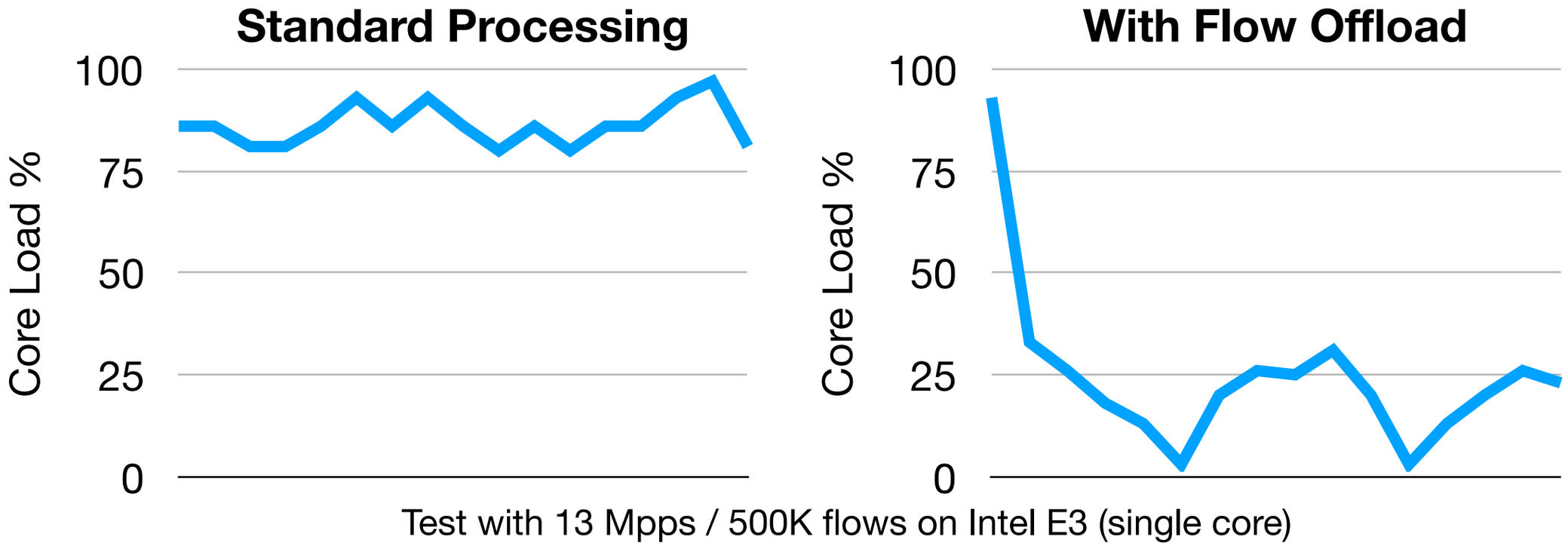
Flow Offload Applications [1/3]

- Initially supported for accelerating our traffic monitoring applications:
  - Netflow offload with nProbe Cento



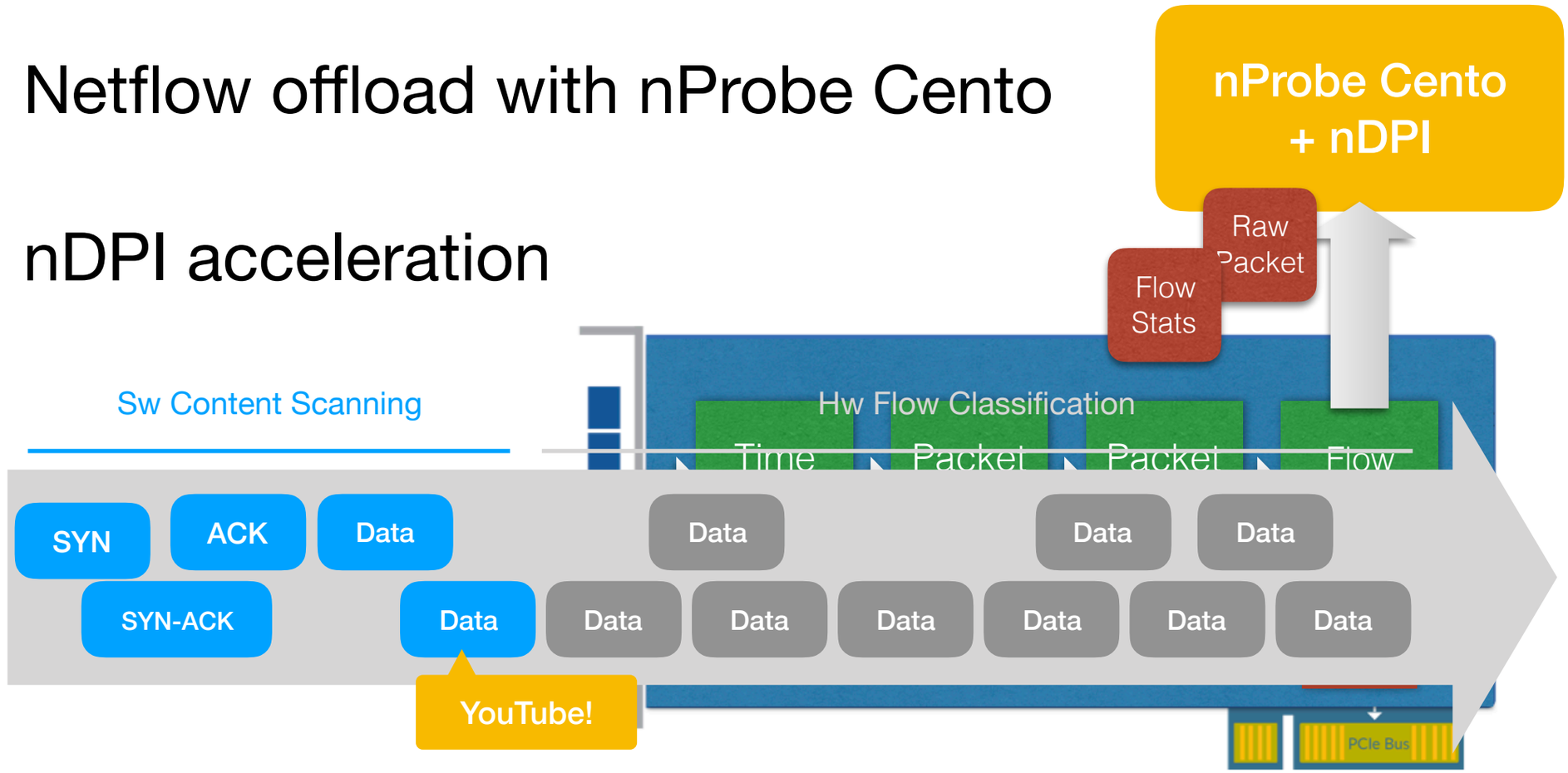
Flow Offload Applications [1/3]

- Initially supported for accelerating our traffic monitoring applications:
  - Netflow offload with nProbe Cento



Flow Offload Applications [2/3]

- Initially supported for accelerating our traffic monitoring applications:
  - Netflow offload with nProbe Cento
  - nDPI acceleration

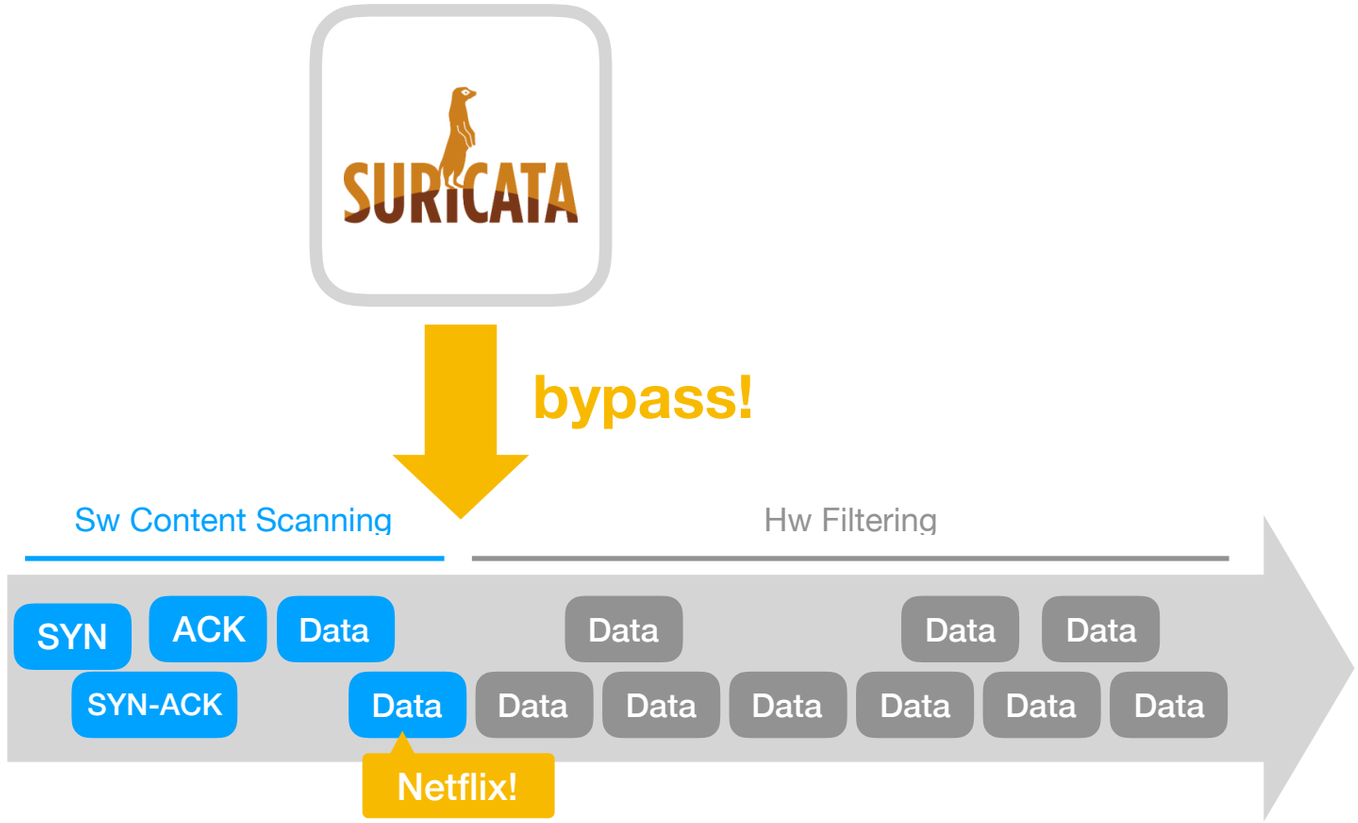


Flow Offload Applications [3/3]

- Initially supported for accelerating our traffic monitoring applications:
  - Netflow offload with nProbe Cento
  - nDPI acceleration
  - Packet-to-disk with n2disk + Netflow on a single box

Enabling Hardware Bypass

- Then we realised it was also a good fit for Suricata..
- The flow classifier can be instructed to filter/forward a flow according to the bypass verdict from Suricata



Under The Hood

```
opflag |= PF_RING_FLOW_OFFLOAD |           /* enable hw flow offload */
          PF_RING_FLOW_OFFLOAD_NOUPDATES; /* disable periodic updates */

ptv->pd = pfring_open(ptv->interface, default_packet_size, opflag);

…

p->pfring_v.flow_id = hdr.extended_hdr.pkt_hash; /* read the packet flow id */
p->BypassPacketsFlow = PfringBypassCallback;     /* set the bypass callback */
```

```
…

static int PfringBypassCallback(Packet *p) /* bypass callback */
{
  hw_filtering_rule r;
  r.rule_family_type = generic_flow_id_rule;
  r.rule_family.flow_id_rule.action = flow_drop_rule;   /* discard future packets */
  r.rule_family.flow_id_rule.flow_id = p->pfring_v.flow_id; /* flow id to discard */

  if (pfring_add_hw_rule(p->pfring_v.ptv->pd, &r) < 0) {
    return 0;
  }

  return 1;
}
```

ntop

SURICON PRAGUE 2017

Coming Back To Numbers..

- Considering real Internet traffic, avg packet size >900 bytes, 13+ Mpps at 100G

- 70+% of Internet traffic in North America comes from video and music streaming

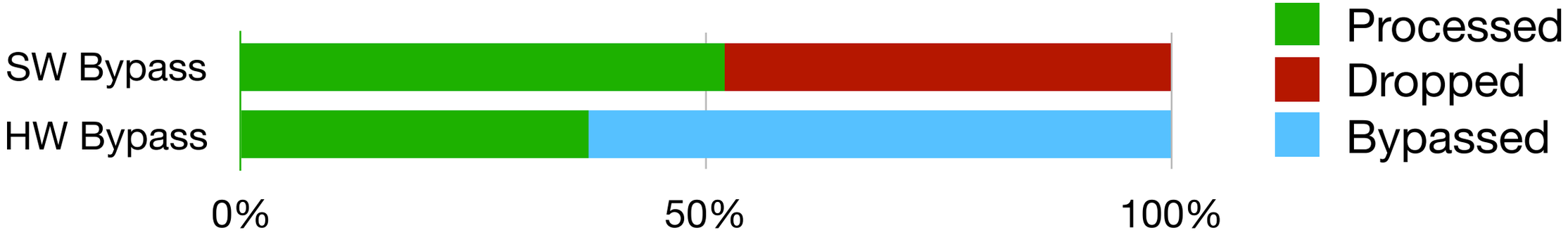
- If we are able to discard entertainment traffic in hw, less than 4 Mpps hit the CPU at 100G!

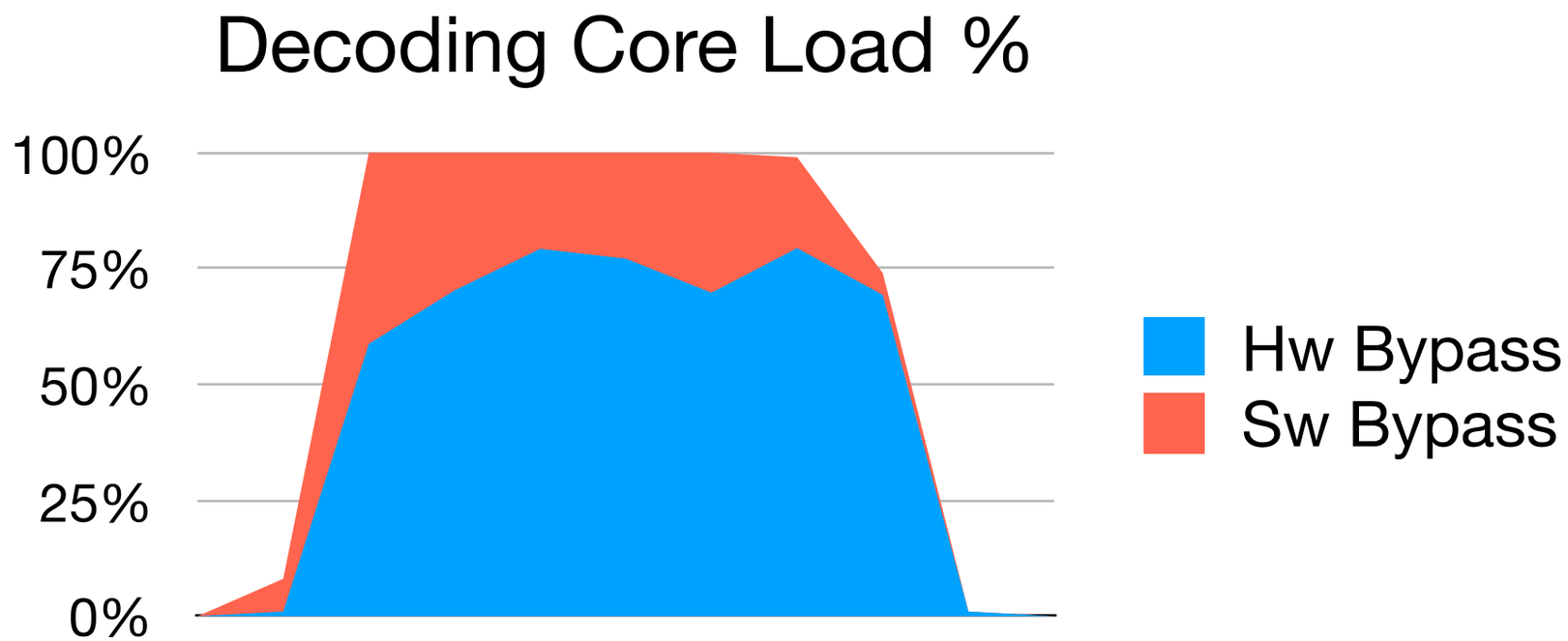
Validation Testbed

- Hardware:
  - Intel Xeon E3-1230 v3 4-cores
  - ANIC Ku Series Quad 10 Gbit
- Suricata configuration:
  - Single capture thread, with CPU affinity enabled, to check the core load on the decoding thread
  - Ruleset stripped down to avoid bottlenecks on detection (we do not have enough cores on this CPU)
  - Bypass enabled: we want to test local bypass vs hardware bypass

Validation Results

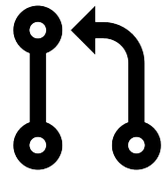
- 14M Internet packets @ 18 Gbit (2.1Mpps), bypass rules to discard multimedia





Final Remarks

- Link speed is becoming really fast compared to CPU clock speed
- Hardware offload is becoming more and more important for CPU intensive applications
- A pull request is under review (#2978), hopefully hw bypass will be available mainline soon



- Future work: IPS flow forwarding offload

Accolade

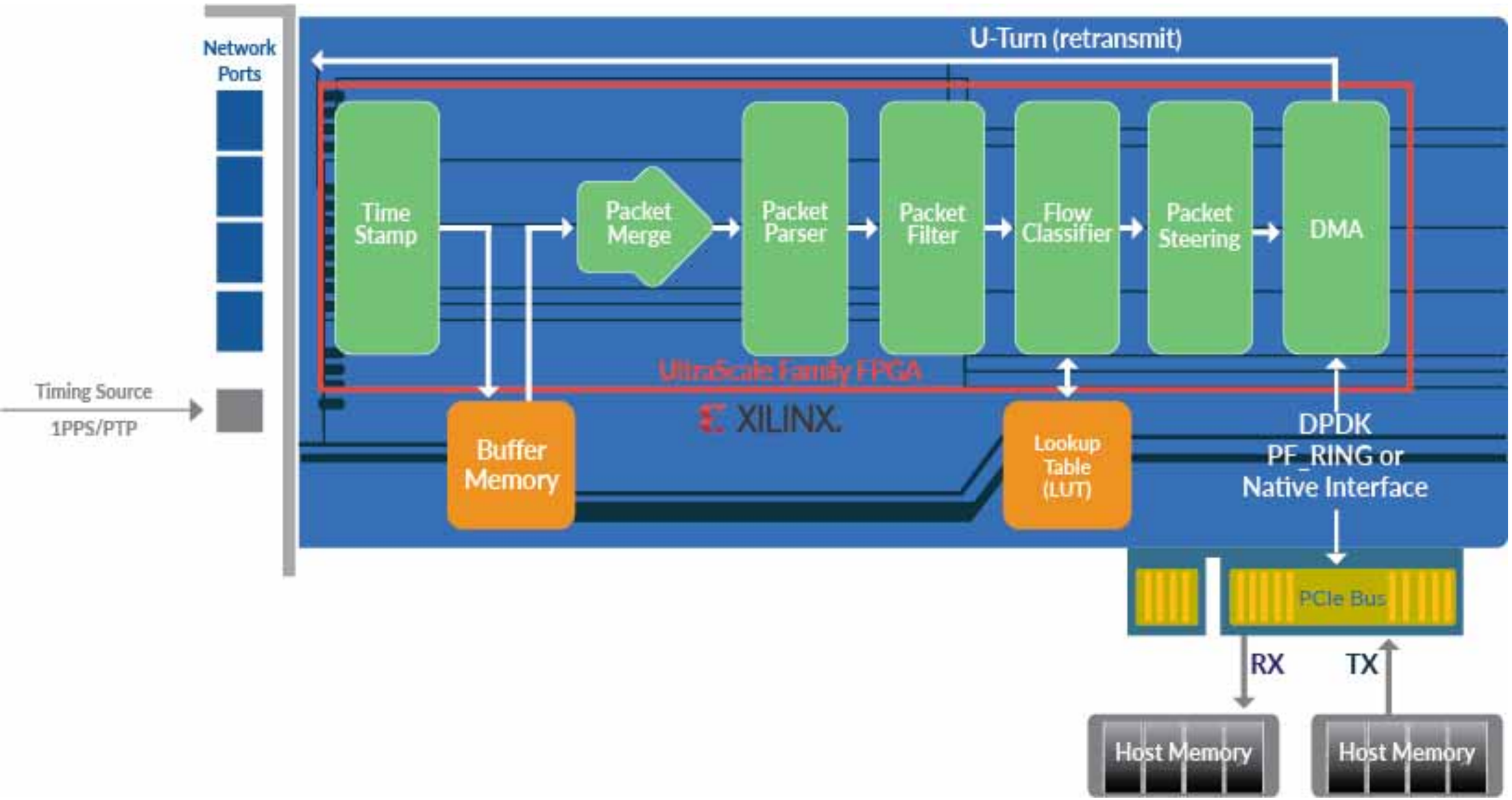
Accolade Overview

- FPGA-based, offload NICs & acceleration platforms
- ANIC (PCIe adapter) product line since 2007 (1-100GbE)
- ATLAS series acceleration platforms
- Focused on developing innovative, high-value products
- Target Applications: Network Monitoring (APM/DPI/IDS/IPS/Forensics)
- Headquarters: Franklin, MA (Boston Area)
- www.accoladetechnology.com

Accolade Technology

SURICON PRAGUE 2017

ANIC Packet Processing Flow



Accolade Technology

SURICON PRAGUE 2017

Flow Classification Offload

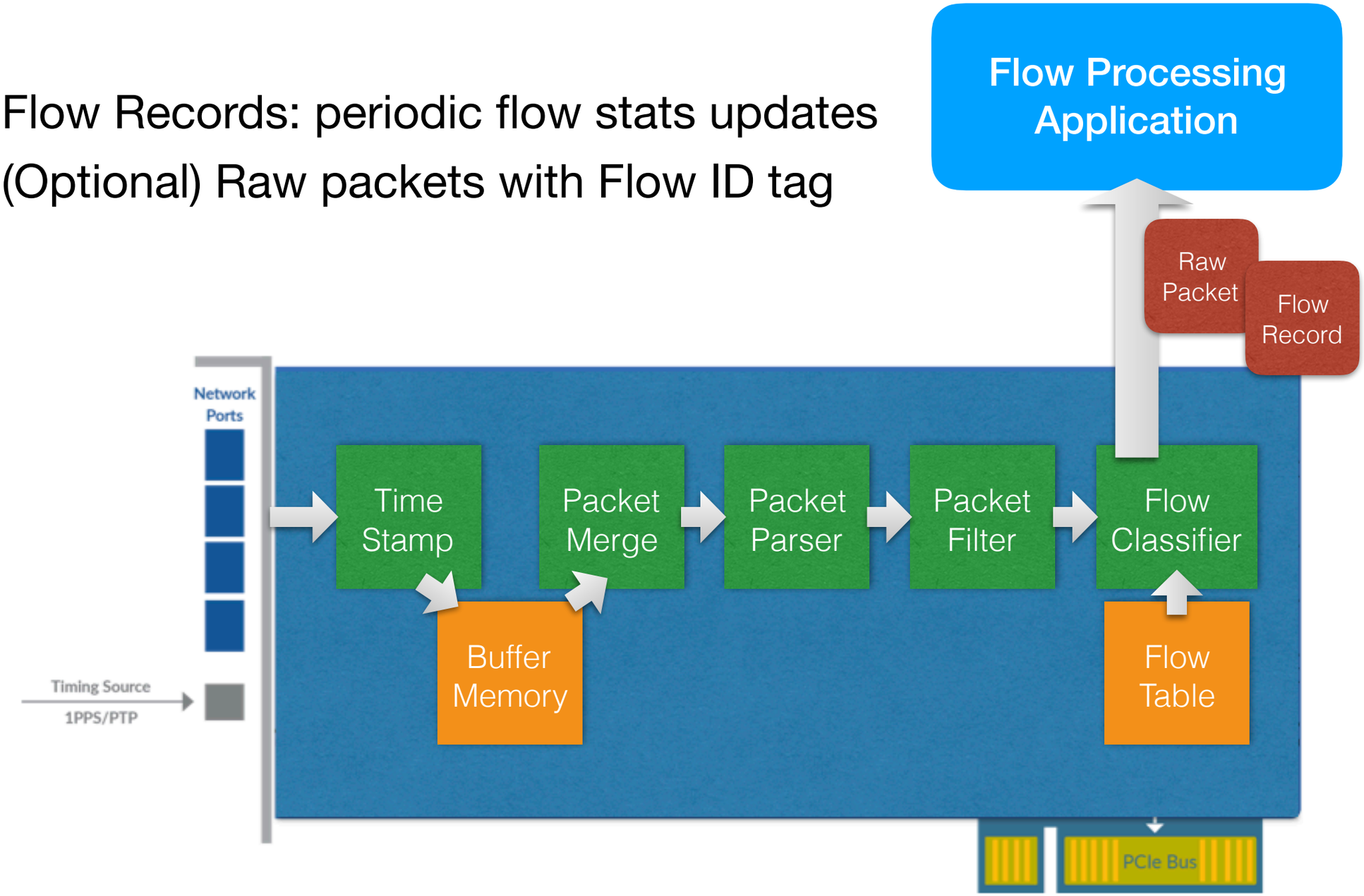
- Flow classification uses a standard hash table based approach
  - Parser isolates the inner IP header of interest, walking past L2/L3 encapsulations.
  - A CRC is calculated on the IPV4 or IPV6 5-tuple.
  - This CRC is used to index into the DRAM-based flow table. If the location is in use, a collision chain is walked.
    - For the first packet of a new flow, a table item is created. For subsequent packets, the same table item is used for tracking & counts.
    - A walker detects idle flows, and frees up the table item
- Currently sized for 16M flow table size on 4x10G, and 64M on 100G
  - Good performance (low collision ratio) with up to about 1 / 4 table full
- Benchmarked at
  - 4 x 10GE full bandwidth with average 128B sized packets
  - 1 x 100GE full bandwidth with average 200B sized packets

Accolade Technology

SURICON PRAGUE 2017

Flow Offload, Usage

- Flow Records: periodic flow stats updates
- (Optional) Raw packets with Flow ID tag



Flow Table Offload, Use cases

- Use case 1: adding a flow tag to original packets
  - User applications can use the flowid presented by FPGA, hence offloading classification in software

- Use case 2: only receive flow reports from FPGA
  - Dramatic reduction in packet load to host software
  - As a modified use case: host software can receive and examine the first few packets of a given flow, and then turn them off

- Use case 3: fine grained filtering for RX into Host or Inline (Bump-in-Wire)

- Use case 4: Inline packet header or payload modification on per-flow basis

Accolade Technology

SURICON PRAGUE 2017

Accolade Products

ANIC (PCIe) Adapters

- 100G
  - ANIC-200Ku/Kq (2-port)
  - ANIC-200KFlex (2-port)
- 40G
  - ANIC-80Ku (2-port)
- 10G
  - ANIC-20Ku (2-port)
  - ANIC-40Ku (4-port)
- 1G
  - ANIC-2KL (2-port)
  - ANIC-4KL (4-port)

Common Architecture

ATLAS Series OEM Platforms

- Common architecture with PCIe adapters
- Compact Design (1U x 8.25” x 14”)
- 2 units fit side-by-side in standard 19” rack
- FPGAs on motherboard
- x86 COTS ComExpress Module
- 4 x 10G or 2 x 40G pluggable interface
- Multiple integrated timing sources: GPS, IEEE-1588/PTP and 1PPS. 2 x SSD = 1TB storage capacity

Accolade Technology

SURICON PRAGUE 2017

Thank you

ntop

SURICON PRAGUE 2017