Hey, You Darned Counters! Get Off My ASIC!

Jeff Mogul, Paul Congdon
HP Labs, Networking and Communication Lab

August 13, 2012

OpenFlow counters: good news and bad news

Good news about OpenFlow counters:

- There are counters for every flow-table rule (packets, bytes, duration)
- So the controller has visibility into every flow

Bad news about OpenFlow counters:

- Every flow's counters act exactly the same way
- Sending all counters to the controller is expensive
- … as is polling all of those counters

Specific issue from the DevoFlow paper (SIGCOMM 2011):

- How can the controller ask to see only "significant" flows?
- DevoFlow solution: add "triggers" to OpenFlow rules
  - E.g., "send report when this flow exceeds 500 packets"
  - Or, "send report when this flow exceeds 500MB/sec."
  - **But: it's hard to add this kind of novel counter function to an ASIC**

Why do we keep the counters on the ASIC?

Historically, the ASIC has been the only place where we can count fast enough

- And prior to SDNs, we didn't interact with the counters very often

But on-ASIC counters create several problems:

1. Lack of flexibility
   - Hard to introduce new kinds of counters, such as DevoFlow
2. Cost of reading and manipulating counters
   - The switch-local CPU has to poke around in the ASIC to do this
3. ASIC area
   - Counters take space, and DevoFlow triggers can take a lot more
4. ASIC complexity
   - Counter-related functions complicate the chip

Why do we keep the counters on the ASIC?

Historically, the ASIC has been the only place where we can count fast enough

- And prior to SDNs, we didn't interact with the counters very often

But on-ASIC counters create several problems:

1. Lack of flexibility
   - Hard to introduce new kinds of counters, such as DevoFlow
2. Cost of reading and manipulating counters
   - The switch-local CPU has to poke around in the ASIC to do this
3. ASIC area
   - Counters take space, and DevoFlow triggers can take a lot more
4. ASIC complexity
   - Counter-related functions complicate the chip

These are the reasons that initially got us started thinking about counters

Why do we keep the counters on the ASIC?

Historically, the ASIC has been the only place where we can count fast enough

- And prior to SDNs, we didn't interact with the counters very often

But on-ASIC counters create several problems:

1. Lack of flexibility
   - Hard to introduce new kinds of counters, such as DevoFlow
2. Cost of reading and manipulating counters
   - The switch-local CPU has to poke around in the ASIC to do this
3. ASIC area
   - Counters take space, and DevoFlow triggers can take a lot more
4. ASIC complexity
   - Counter-related functions complicate the chip

This is the reason that actually seems to matter.

ASICs vs. rapid innovation

ASICs have long design cycles

- Multiple years from “we should add this feature” to shipping products
- New trains leave the station only once in a while

It’s hard to anticipate future uses of SDN counters

- So it’s hard to figure out what to put on an ASIC to support future uses

OpenFlow/SDN: **largely a reaction to the slow pace of ASIC innovation**

- So: move the locus of innovation to software

Software-Defined Counters (in one slide)

No counters on the ASIC!

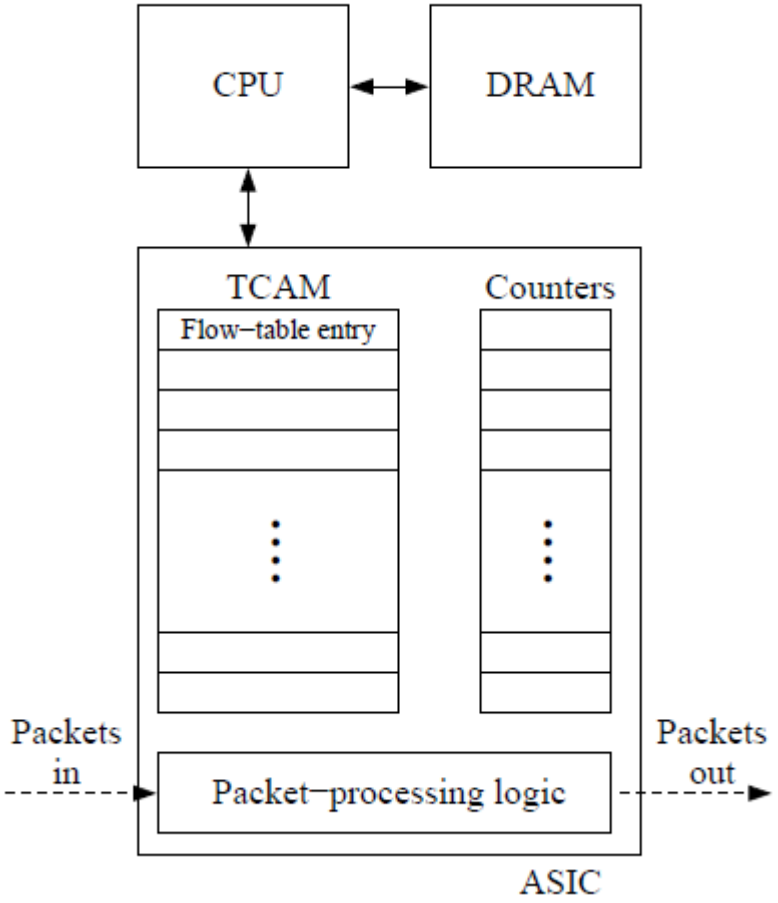
ASIC generates an *event record* on each packet arrival

- Fields: **`ruleNumber, byteCount`**
- 32 bits is enough for ~500K rules
- Small on-ASIC buffer for event records
- Full buffer blocks are DMAed to off-ASIC DRAM

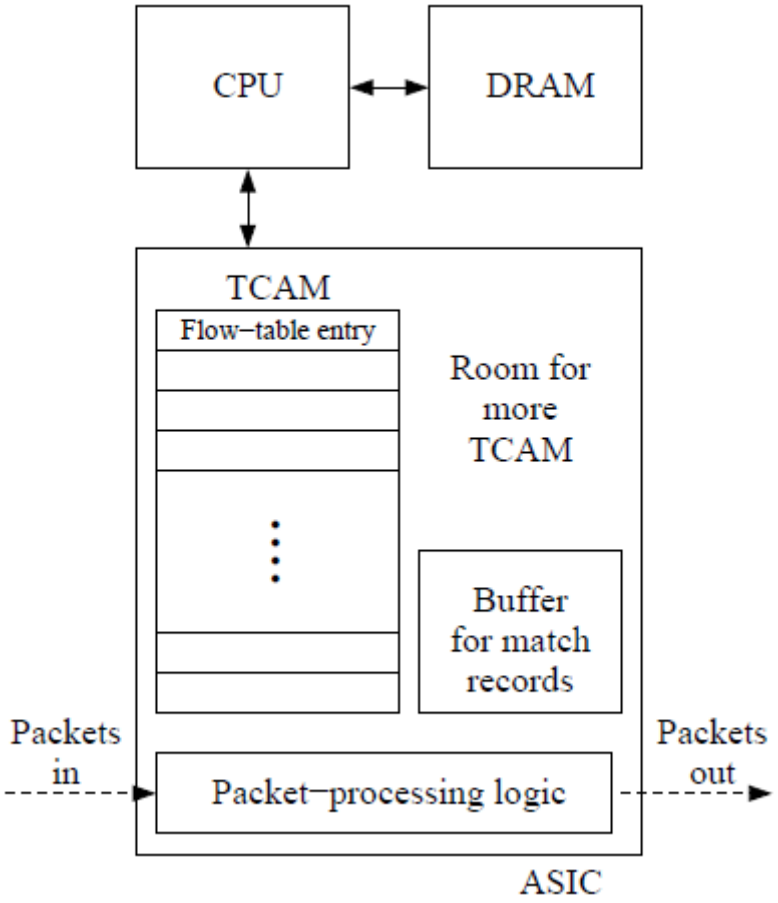
Switch-local CPU reads the event-record stream from DRAM

- Updates standard OpenFlow counters, also stored in DRAM
- Can implement novel counter features with SW-only changes

System-level design

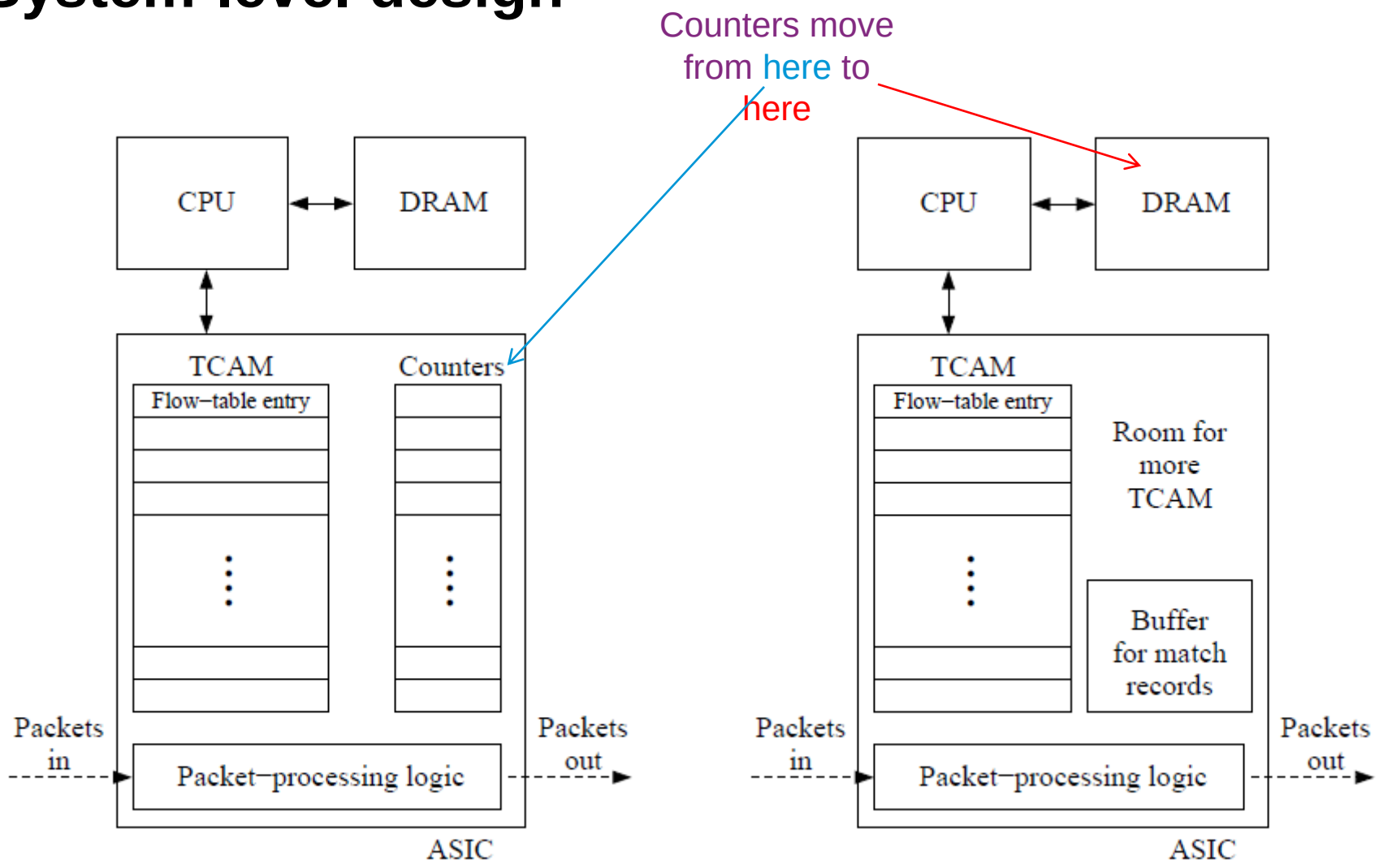


Traditional SDN switch



SDC switch with off-ASIC CPU

System-level design



Traditional SDN switch

SDC switch with off-ASIC CPU

Examples of features that SDC could support

DevoFlow extensions:

- Triggers: store trigger values in DRAM; check trigger on each update
- Rate-based triggers: keep EWMA of bytes/sec in DRAM (perhaps)

Immediately discard “uninteresting” records:

- Possibly useful for implementing `Limit` clause in the Frenetic Language

Multi-flow triggers: send report when some subset of flows hits threshold:

- E.g., “connections to Web server together account for 1 Gbit/sec”

Flow-creation-rate triggers:

- E.g., “more than 100 new flows/sec into a laptop”

Emulate sFlow, NetFlow, or RMON

- Makes it easier to integrate OpenFlow into a legacy environment

How could this possibly work?

We need to worry about:

1. ASIC-to-CPU bandwidth: how many bytes/sec can we transfer via DRAM?
2. CPU processing costs: how many event records/sec can the CPU process?

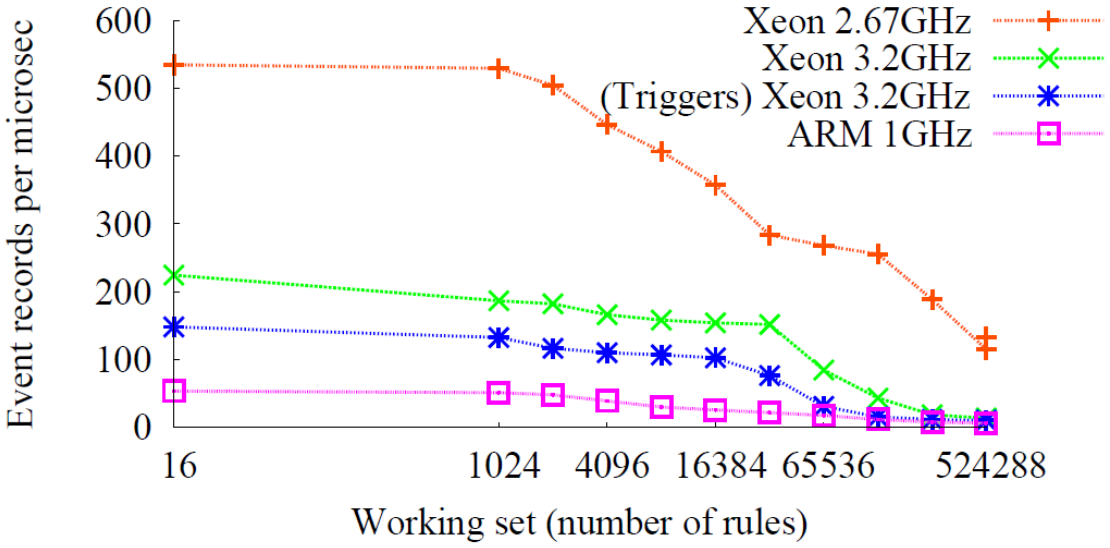
Fundamental issue: what event rates can we expect?

- (assumes: 48-port 10GbE switch)
- Worst case: 714M events/sec (2.9GByte/sec)
  - Even this worst-case bandwidth is achievable with DDR3 DRAM
- "realistic" full-load case (400B/pkt): 150M events/sec (572MByte/sec)

How could this possibly work?

Experiments (in paper) suggest older Xeon should handle 150M events/sec

- *If* working set of rules (counters) is fairly small (~1024 – 4096)
  - Likely to be true for a data-center network; not likely for core Internet router
  - Newer CPUs (e.g., Xeon/Nehalem) can handle much higher rates
  - 3 C(



Techniques to reduce bandwidth and event rates

Bandwidth reduction: compress event records

- Huffman-encoding of rule numbers
- Short encodings of popular packet sizes

Event-rate reduction: exploit locality

- Keep a small on-ASIC cache, indexed by rule number
- Coalesce event records in this cache
  - Adds a **`packetCount`** field to the event record format
  - Tag certain flow-table rules as “do not coalesce”, if necessary
- Stream evicted entries to CPU (via DRAM)

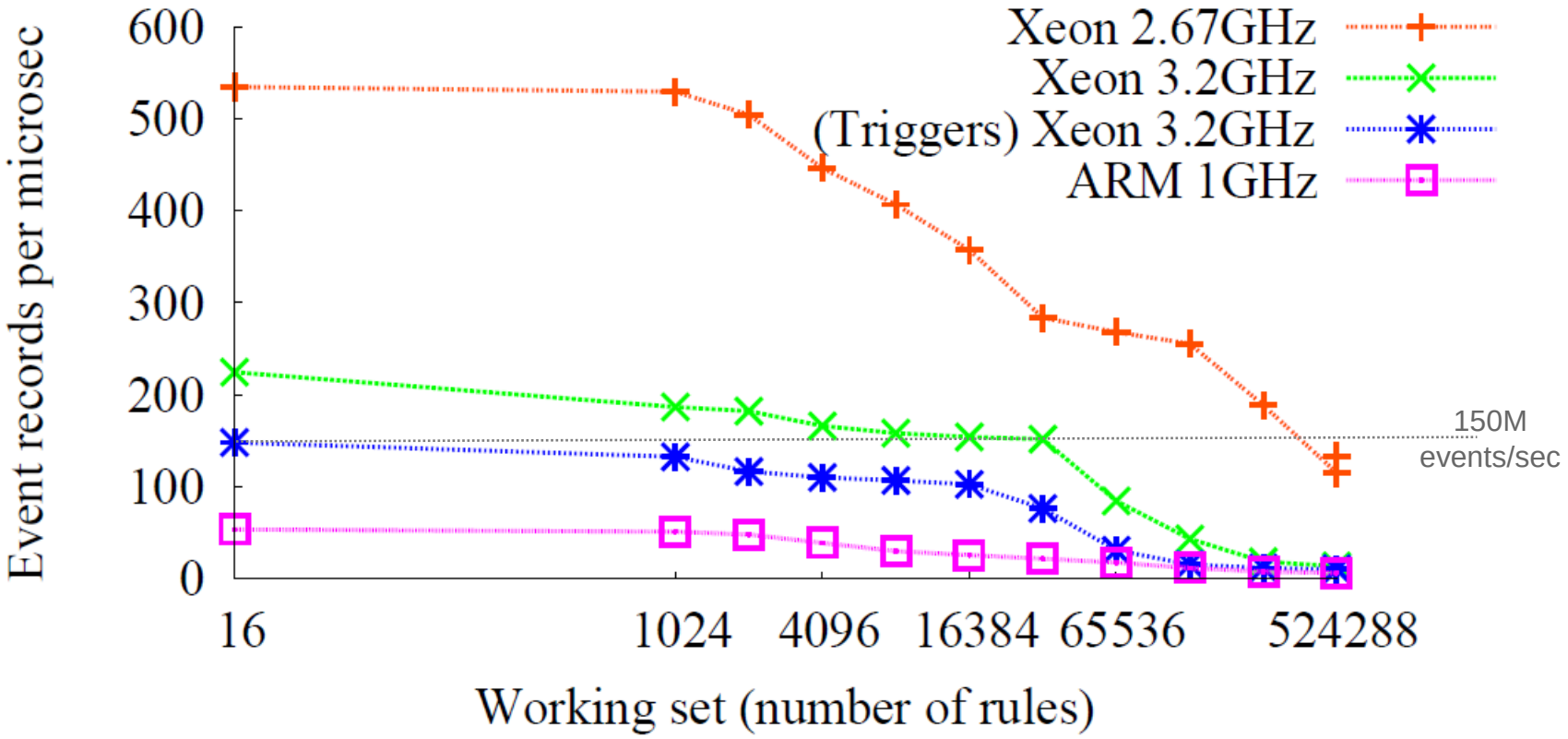
Worst-case: discard excess event records:

- Drop-tail, or randomly (to reduce sampling bias)
- Or tag some flow-table rules as “important” or “boring”

Summary: Software-Defined Counters

- Enable more flexible SDN counters
- Get stuff off of the ASIC
- Ride trends in DRAM and embedded-CPU technology
- Simplistic analysis suggests it might actually work

Microbenchmark results



"Triggers" means checking against byte-count and packet-count thresholds

Locality analysis

Previous work

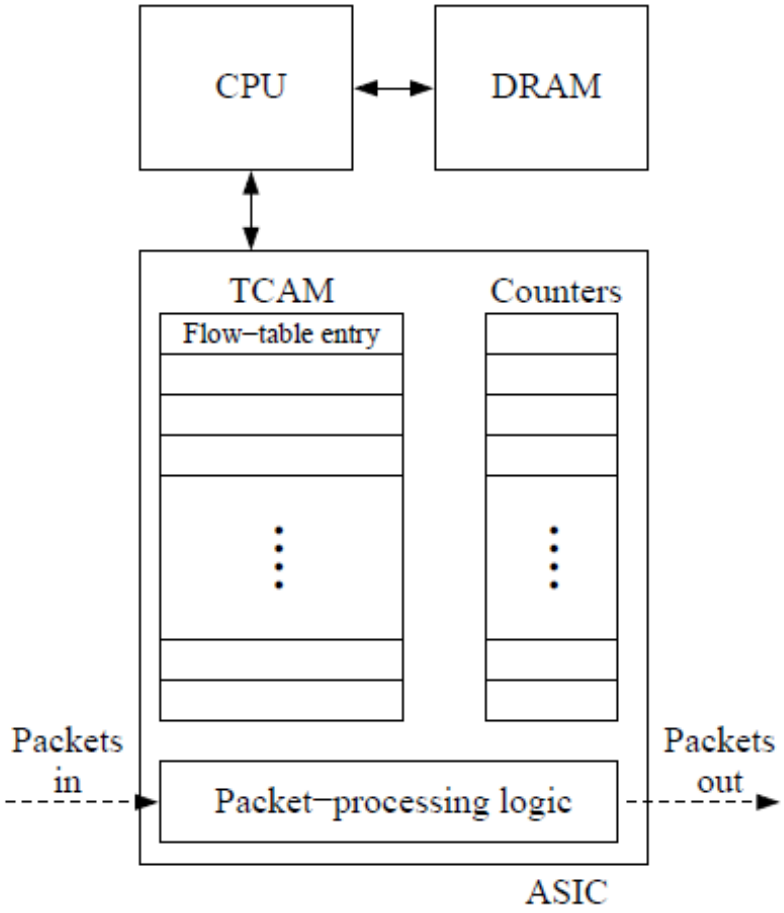
- Benson *et al.* 2009: no more than 10K active flows at any switch
  - Usually much lower
- Tavakoli *et al.* 2009: data-center OpenFlow switches require <=5K entries

Trace-based analysis using 1K-entry cache

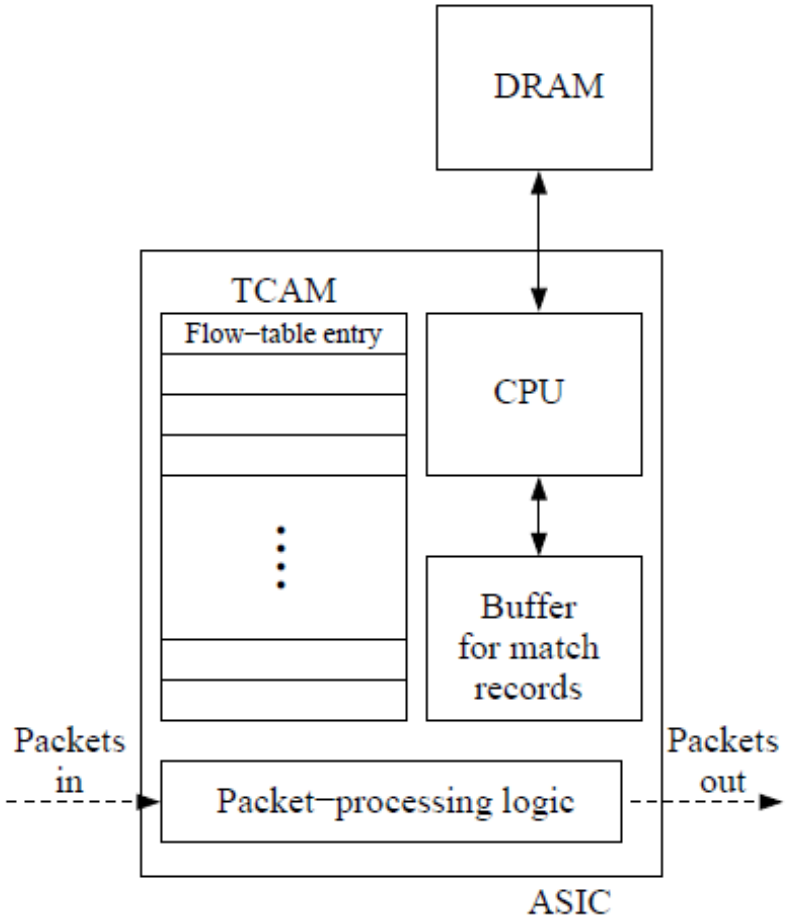
- Benson's data-center traces: hit rates between 78% - 97%
- Jan 2009 CAIDA traces: hit rates between 43% - 52%

Thank you

System-level design



Traditional SDN switch



SDC switch with on-ASIC CPU