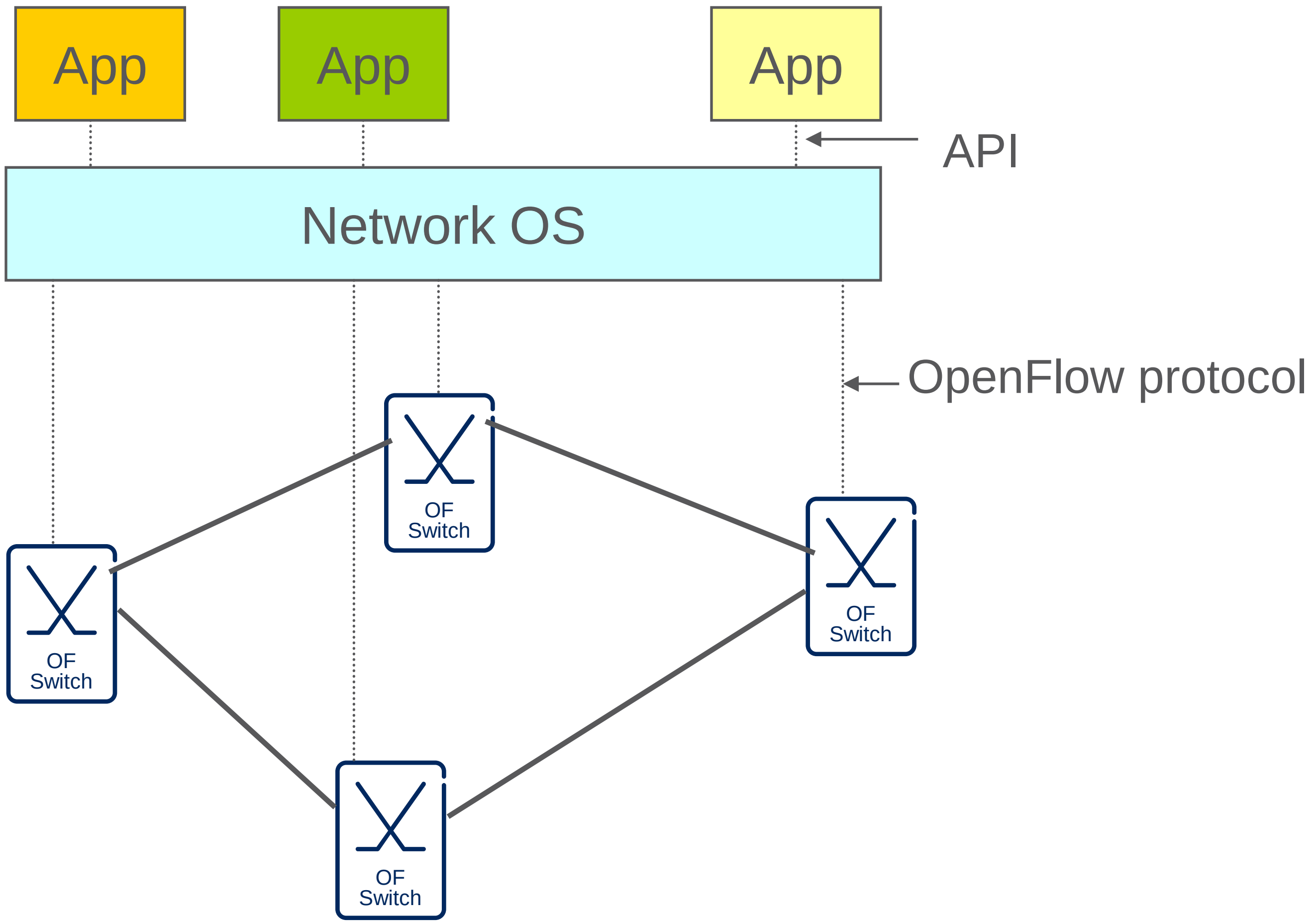
Design of an OpenFlow Switch on a Multi-core Platform

Ritun Patney, Erik Rubow, Ludovic Beliveau, Ramesh Mishra

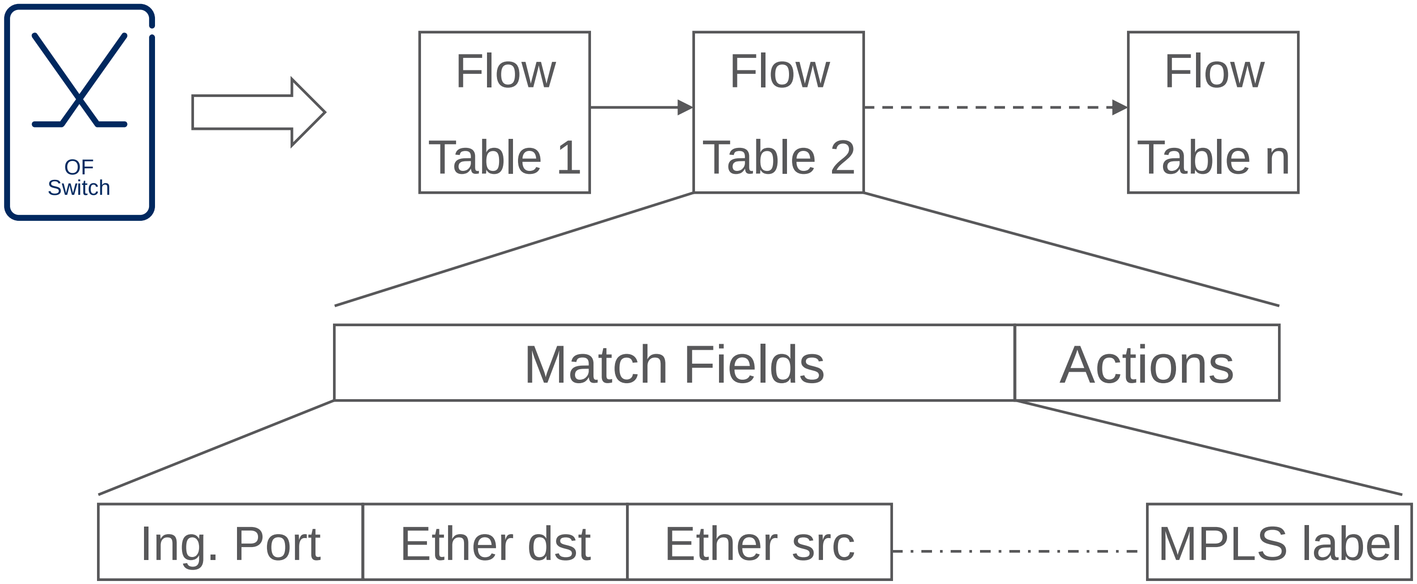
Ericsson Research, San Jose

{ritun.patney, erik.rubow, ludovic.beliveau, ramesh.mishra}
@ericsson.com

Introduction to OpenFlow

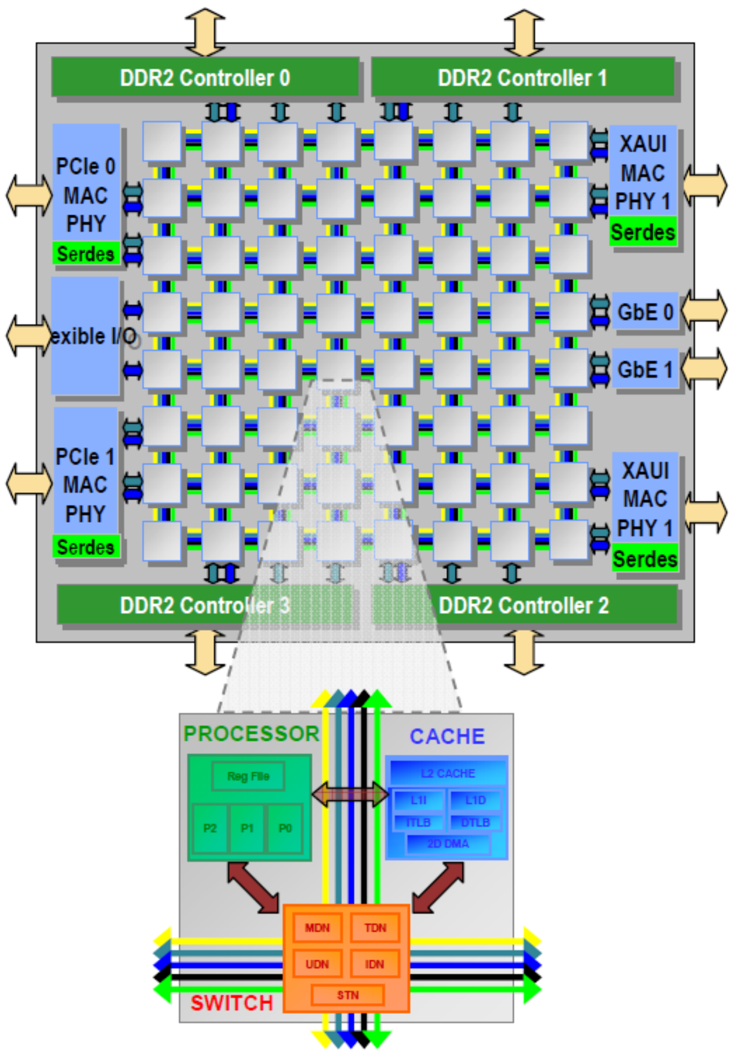


Forwarding Abstraction (OF 1.1)



- Forwarding abstraction contains multiple flow tables
- Each table has a set of fields and a set of actions
- Each table is generalized to contain 14/15 match fields

Platform Details

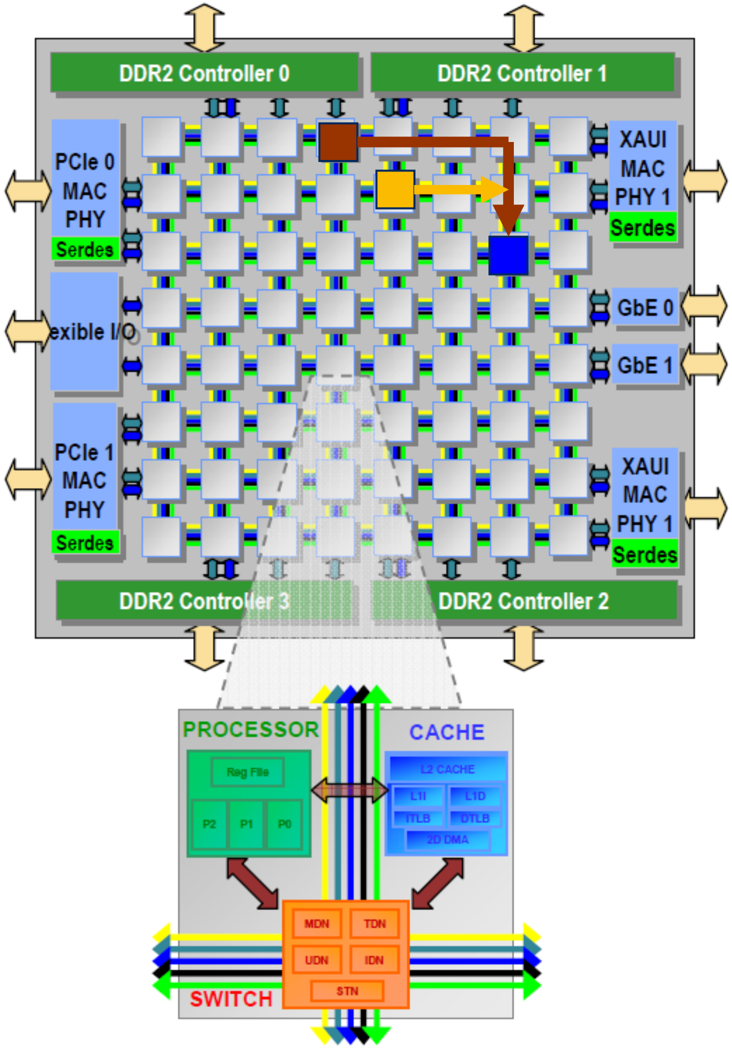


› **Highlights**

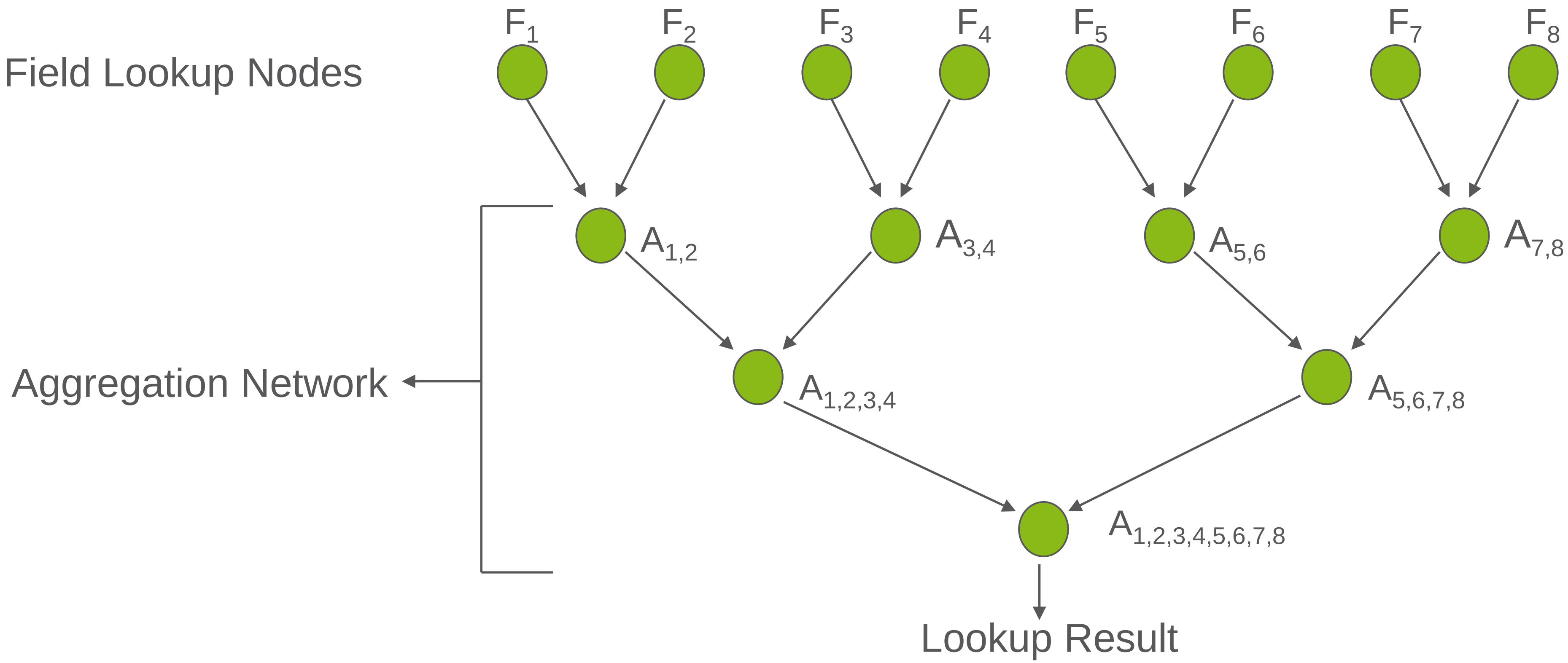
- 64 Cores, 866MHz
- On chip interconnect
- Caches
  - Each core has 16KB L1 I-Cache, 8KB L1 D-Cache, and 64KB combined L2 cache per tile
- Single Thread per Core

Challenges in Many Core Platforms

› Challenges
- Splitting packet processing into tasks
- Hiding memory latency
  - Single threaded model
  - Caches are not extremely large
  - Effectively apply pre-fetching
    - ❖Work on multiple packets in the same code loop
- Sharing Data across Cores
  - Shared memory consumes cycles on locking and cache misses
  - On chip communication network prone to errors



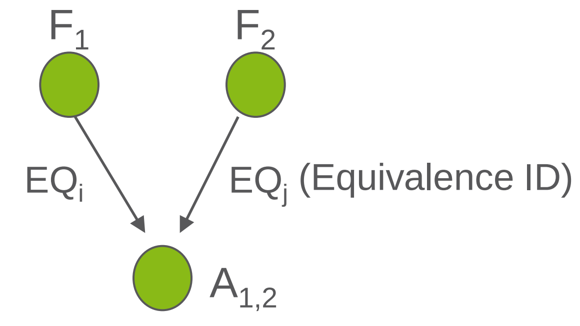
Algorithmic Packet Classification



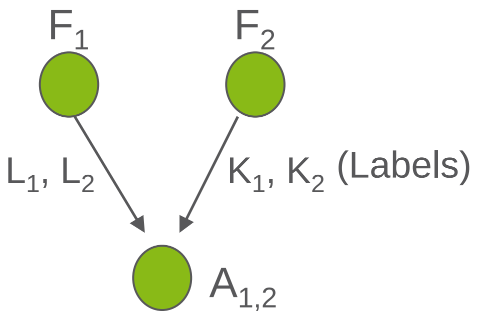
- Packet header is decomposed into individual fields and fed to the classifier

RFC vs DCFL

	RFC	DCFL
Rule Set	Pre-computed, stored as 2d array	Cross-product taken at run time
Memory Accesses	Constant	Variable
Memory Latency	Easier to hide via Pipelining	Harder to Pipeline
Rule Set Scaling	Poor	Scales Well
Incremental Updates	Hard	Efficient



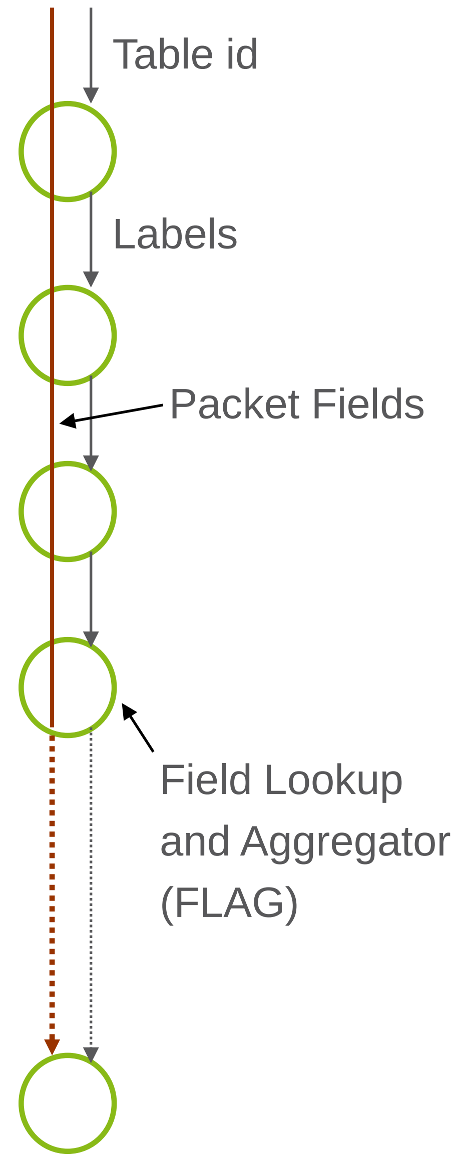
RFC



DCFL

DCFL Lookup Architecture

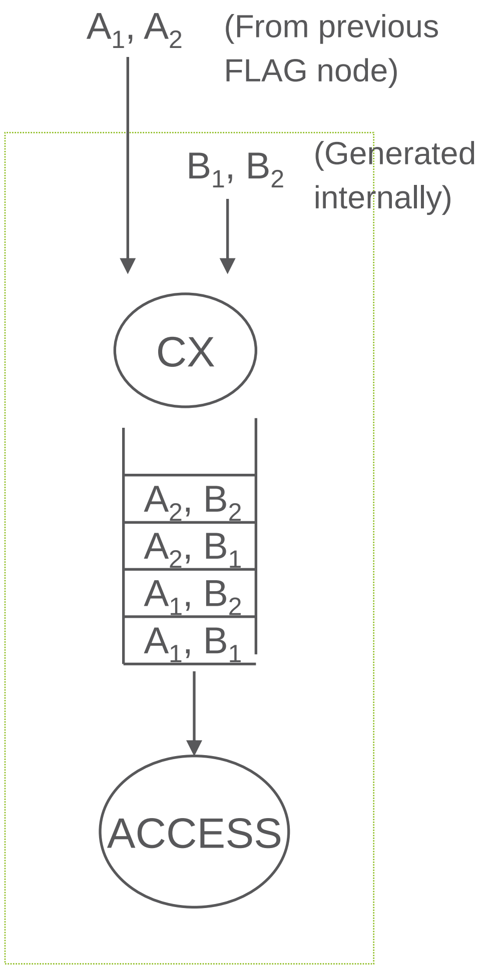
- Tree topology hard to map to a grid
- Main Problem => Deadlocks while distributing packet fields via the mesh
- Solution => Linear Topology
- Advantages
  - Easy to map, avoids deadlocks
- Cons
  - Consumes several cores
  - Spend cycles in receiving and passing packet fields



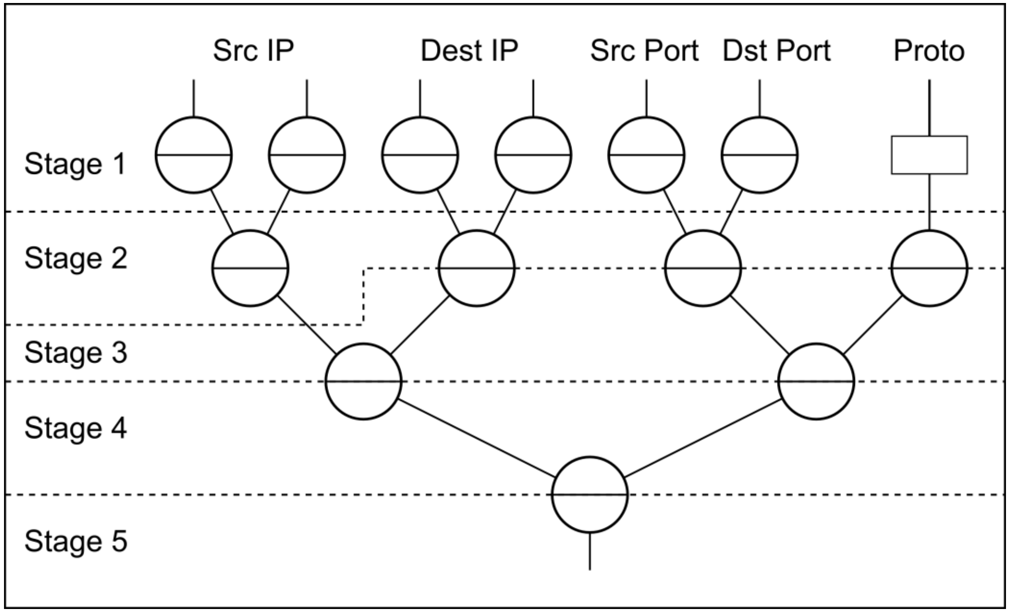
DCFL Internal Node Pipeline

- CX
  - Cross-products the labels to produce keys
  - Performs one pre-fetch operation for mem[key_j]
- ACCESS
  - Performs one access operation to load mem[key_j]
- Scheduling
  - CX, ACCESS scheduled in a tight code loop
  - Constant number of outstanding pre-fetches maintained to maximize memory performance

- Non trivial to implement
  - Primarily because of variable memory accesses per packet
  - Extra logic keeps state per packet, next key to prefetch, etc.

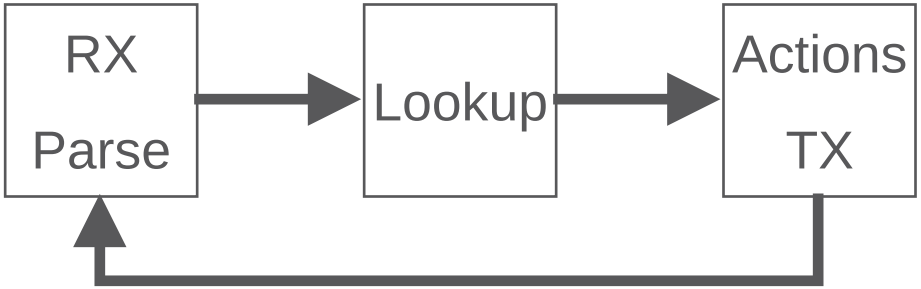


RFC: Lookup Pipeline



- Because RFC involves a fixed number of memory accesses, it can be easily pipelined
- Each circle is composed of pre-access (pre-fetch) and post-access operations
- Each operation is executed once per iteration of a tight code loop
- Each stage operates on a different packet, allowing for more parallelism
- Pre-access and post-access operations are scheduled in order to maximize the number of outstanding accesses per core and to maximize DRAM throughput
  - The core can continually maintain a high number of outstanding prefetches across loop iterations

RFC: System Architecture



- The pipelined and highly optimized structure of the lookup code leads to an overall architecture where packet parsing and action processing are handled on separate cores
- Lookup requests and responses are sent between cores
- The load on each core is roughly balanced, depending on the match fields supported and actions applied
- Several instances of the above pipeline run on the chip

Results

- Rule Set Scalability
  - DCFL designed for 1 million flow entries. Easily Extensible.
  - RFC limited by the algorithm
- Performance
  - Data Path - Single Table
    - DCFL - 3.5 Mpps for a single pipeline using 20 cores
    - RFC
      - 10.7 Mpps for 5-tuple classification, 15 pipelines
      - 4.5 Mpps for full 14-tuple classification, 15 pipelines

Conclusion

- The high-level architecture of this implementation was driven by the fact that we were using cores with a single hardware thread. This was the source of most of the complexity and optimization effort.
- The very wide multi-dimensional lookups in OpenFlow are fundamentally expensive
  - The problem is magnified by multiple tables
  - The number of standard match fields has only increased with each OpenFlow version
  - A flow cache is one possible work-around for this, but isn't always appropriate

ERICSSON