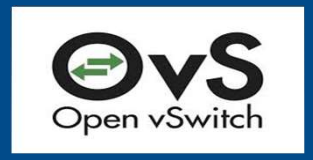


CLASSIFICATION OPTIMIZATIONS TO ENABLE MEGAFLOW OFFLOAD ONTO LIGHTWEIGHT HW PIPELINES

James Choi, Anbuvelu Venkataraman, Kshitij Gupta, Ajay Dubey,
Sathya Narayana Pottimurthy

Notices & Disclaimers

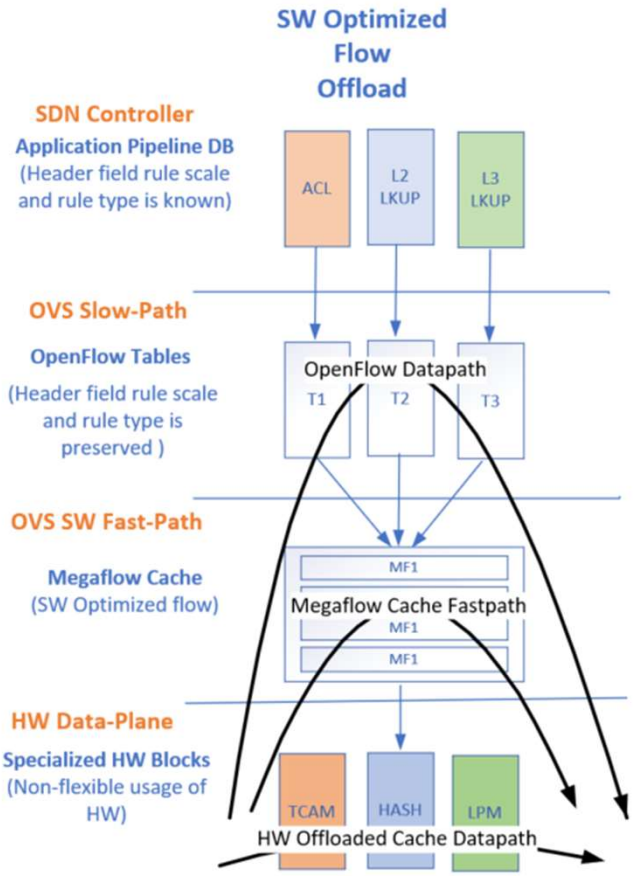
- Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.
- Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.
- Your costs and results may vary.
- Intel technologies may require enabled hardware, software or service activation.
-

Agenda

1. What is megaflow?
2. Difficulty in offloading megaflows to HW.
3. Why is the megaflow formed this way?
4. What can help and why?
5. Prototypes
    1. Deriving field subtables from DPCLS subtables
    2. Partial offload using field subtables in driver
    3. Full Offload to FPGA Pipeline for Field Sub-tables

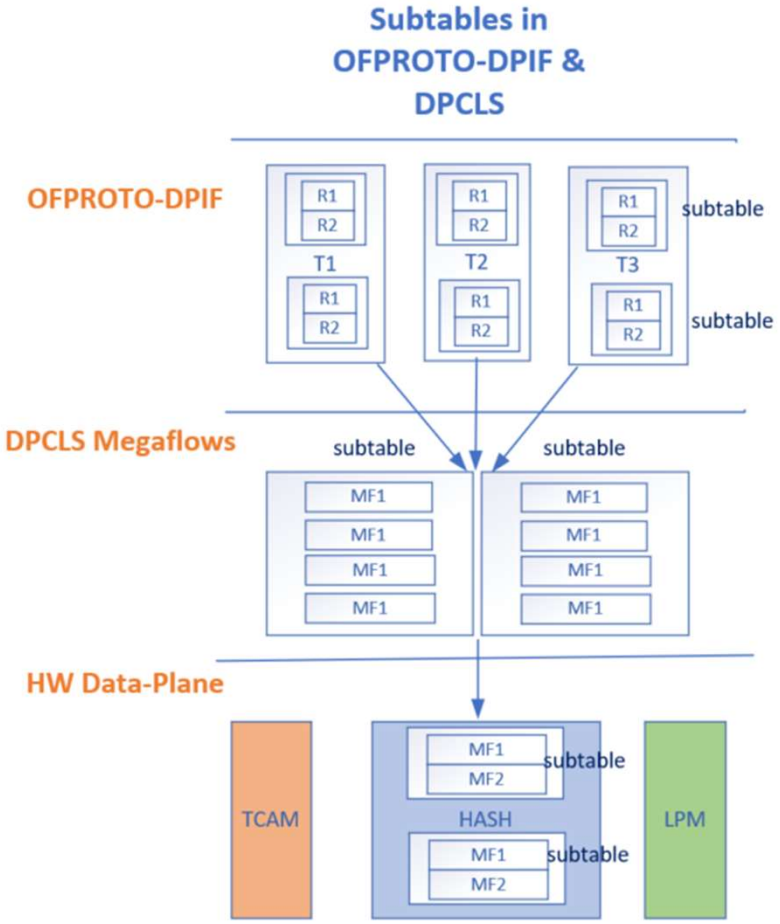
What is OVS Megaflow cache?

1. Cache of flows from OpenFlow rules
   - Doesn't represent all OpenFlow table (ofproto) rules.
   - Created using headers & metadata of 1st packet of a flow through OFPROTO datapath (ofproto-dpif).
2. Flow key fields with mask for aggregate of u-flows.
   - Each key field has mask derived from OF rules. (megaflow)
   - Same flows with same masks for all fields are put into a "subtable" (ie. Same tuple space)
3. Subtables implemented with exact match tables
   - Each subtable implemented with exact match / hash table with unique key mask.
4. Entries from all subtables are non-overlapping without priorities
   - OpenFlow datapath (ofproto-dpif) unwildcarding algorithm creates non-overlapping megaflows.
   - First match == only possible match
5. Each subtable for all fields of all headers and metadata
   - Subtables hold generic key/mask for all fields.
6. Each subtable entry is associated with its action set.



Subtables in OFPROTO-DPIF vs DPCLS

	Subtable Characteristics
OFPROTO-DPIF	OpenFlow Tables - Each OF table w/ multiple subtables - Each subtable w/ multiple rules with same key mask - Each rules with priority - Entries overlapping
DPCLS Megaflow	Megaflow Subtables - Each subtable w/ multiple flows with same key mask - Each flow with no priority - Entries non-overlapping
HW	Megaflow Subtables - Each subtable w/ multiple flows with same key mask - Each flow with no priority - Entries non-overlapping



How are the megaflows derived from OFPROTO-DPIF Classifier

How does classifier work? (From classifier.h)

** This is how the classifier works. In a "struct classifier", each form of*

** "struct cls_rule" present (based on its ->match.mask) goes into a separate*

** "struct cls_subtable". A lookup does a hash lookup in every "struct*

** cls_subtable" in the classifier and tracks the highest-priority match that*

** it finds. The subtables are kept in a descending priority order according*

** to the highest priority rule in each subtable, which allows lookup to skip*

** over subtables that can't possibly have a higher-priority match than already*

** found. Eliminating lookups through priority ordering aids both classifier*

** primary design goals:* ***skipping lookups saves time and avoids un-wildcarding***

**** fields that those lookups would have examined****.*

Steps:

1. Use exact match tables to search through all OFPROTO tables.
    1. Each OFPROTO table is organized as subtables with associated mask.
2. Calculate single non-overlapping match mask to avoid priority-based lookup in dataplane.
    1. **Match all possible/relevant subtables to find rule with highest priority.**
    2. **Create flow wildcard based both matching and not matching rules evaluated.**
    3. Find single tuple space that includes headers from all headers.
3. **Calculate single non-overlapping match masks across all relevant OFPROTO tables by un-wildcarding flow mask.**

How are the megaflows derived from OFPROTO-DPIF Classifier - Optimizations

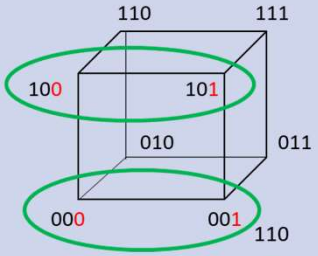
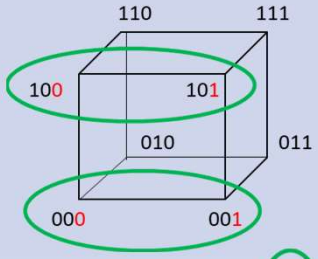
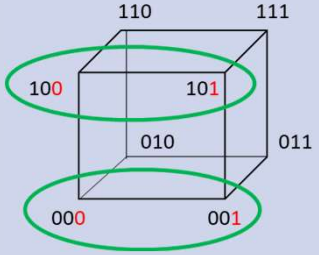
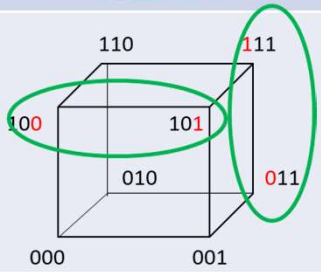
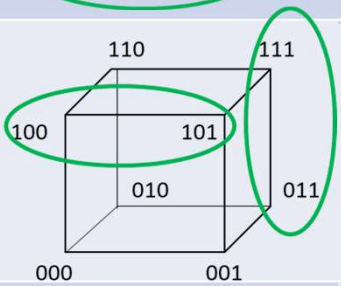
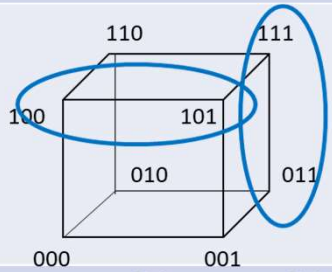
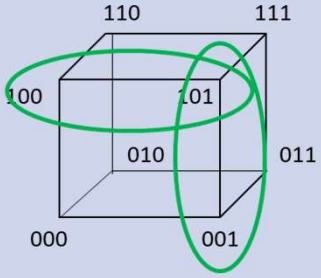
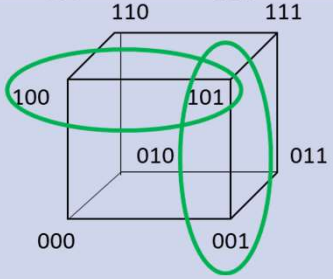
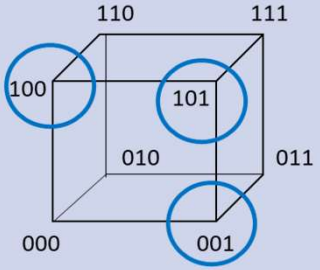
Classifier Optimizations

1. Priority Sorting (Lookup Time) - The subtables are kept in a descending priority order according to the highest priority rule in each subtable, which allows lookup to skip over subtables that can't possibly have a higher-priority match than already found.
2. **Staged Lookup (Wildcard Optimization)** – Classifier subtables divided into 4 hash tables: metadata, +L2, +L3, +all fields. Helps with exact low granularity L4 port values from being looked up. Reduce total hash entries and helps with creating larger wildcard fields.
3. Prefix Tracking (Wildcard Optimization) – Utilize longest prefix match lookup to eliminate subtables lookups based on possible match or non-match prefix masks.

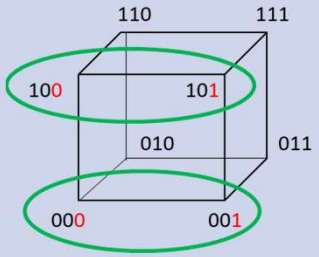
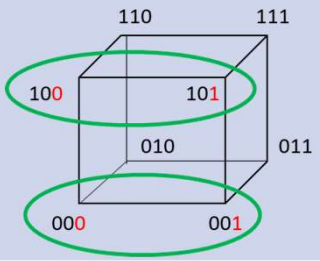
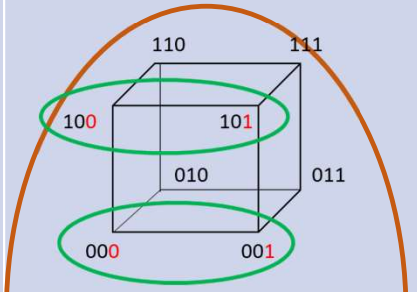
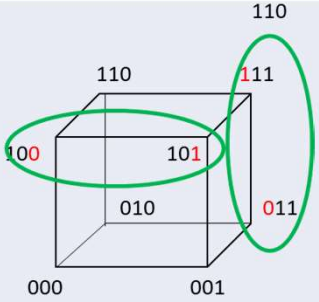
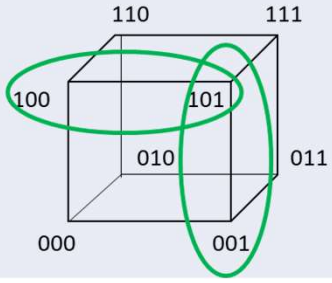
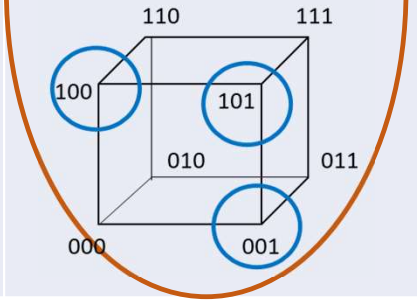
Goals

1. Wildcard optimizations
    1. Reduces # of megaflow subtables and # of entries.
    2. Segment lookups by headers to wildcard whole headers (e.g. like L4 headers with exact match rules).

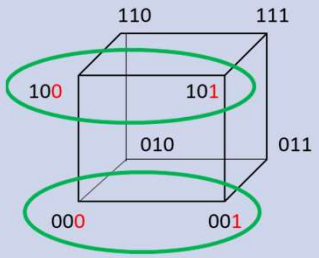
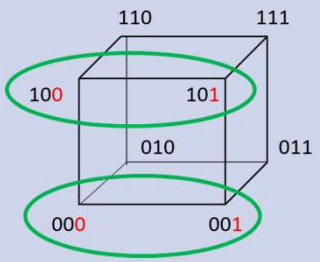
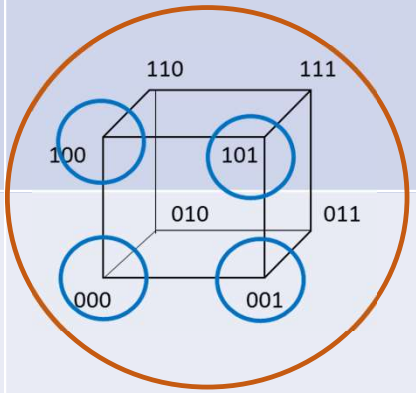
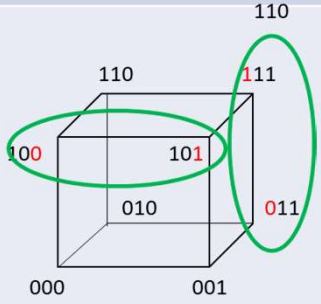
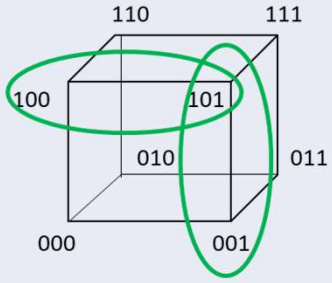
Megaflow: Non-overlapping subtables and entries

	OFPROTO Rules	OFPROTO-DPIF Rules	DPCLS / Megaflow Rules
OFPROTO-DPIF single priority: 1 subtable (mask=b'110) 2 entries (entry=b'10x, b'00x) **DPCLS (w/ 4 packets covered in rules):** 1 subtable (mask=b'110) 2 entries (entry=b'10x, b'00x)			
OFPROTO-DPIF single priority: 2 subtable (mask=b'110, b'011) 1 entrie each (entry=b'10x, b'x11) DPCLS (w/ 4 packets covered in rules): 2 subtable (mask=b'110, b'011) 1 entry each (entry=b'10x, b'x11)			
OFPROTO-DPIF single priority: 2 subtable (mask=b'110, b'x11) 2 entries (entry=b'10x, b'x01) **DPCLS (w/ 3 packets covered in rules):** 1 subtable (mask=b'111=b'110|b'011) 3 entries (b'100, b'001, b'101)			

Megaflow: Non-overlapping subtables and entries in 2 header fields

	OFPROTO Rules	OFPROTO-DPIF Rules	DPCLS / Megaflow Rules Megaflow Subtable Mask is longer = b'110 111
Header field 1: OFPROTO-DPIF single priority: 1 subtable (mask=b'110) 2 entries (entry=b'10x, b'00x) DPCLS (w/ 4 packets covered in rules): 1 subtable (mask=b'110) 2 entries (entry=b'10x, b'00x)			
Header field 2: OFPROTO-DPIF single priority: 2 subtable (mask=b'110, b'011) 1 entrie each (entry=b'10x, b'x11) DPCLS (w/ 4 packets covered in rules): 2 subtable (mask=b'110, b'011) 1 entry each (entry=b'10x, b'x11)			

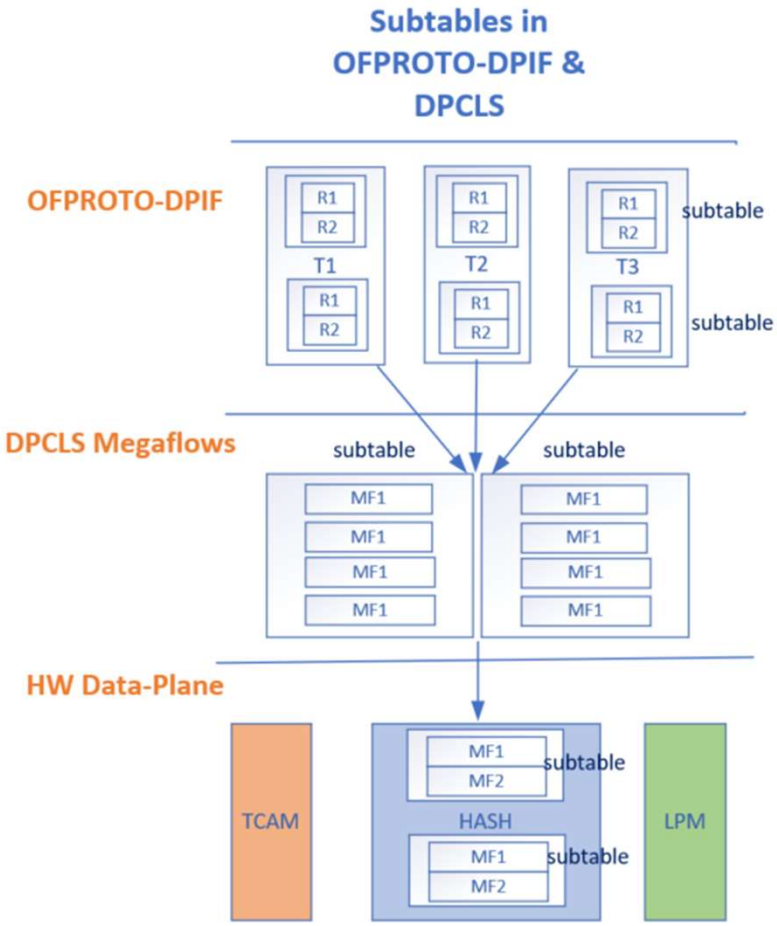
Megaflow: Non-overlapping subtables and entries in 2 OF Tables

	OFPROTO Rules	OFPROTO-DPIF Rules	DPCLS / Megaflow Rules Flow Keys from both Tables. Megaflow subtable mask is smaller b'111 = (b'110 | b'011) requiring more entries
Table 1: **OFPROTO-DPIF single priority:** 1 subtable (mask=b'110) 2 entries (entry=b'10x, b'00x) **DPCLS (w/ 4 packets covered in rules):** 1 subtable (mask=b'110) 2 entries (entry=b'10x, b'00x)			
Table 2: **OFPROTO-DPIF single priority:** 2 subtable (mask=b'110, b'011) 1 entrie each (entry=b'10x, b'x11) **DPCLS (w/ 4 packets covered in rules):** 2 subtable (mask=b'110, b'011) 1 entry each (entry=b'10x, b'x11)			

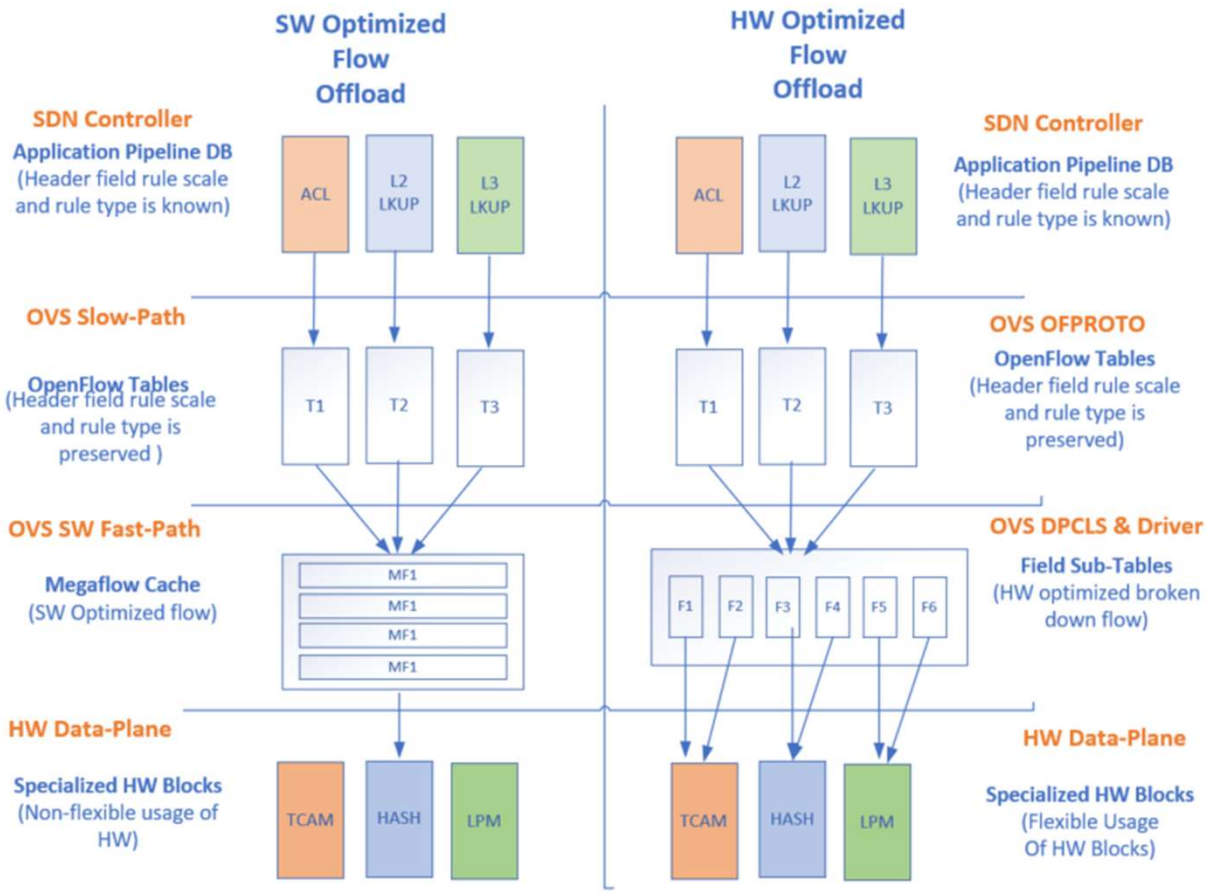
Difficulties in offloading megaflows to HW

Megaflow entries chracteristics	Difficulties in offloading to HW
Longer mask length	• More resource/space in HW to hold longer keys. • More permutations possible for more subtables == parallel hash table lookups with different key/mask in HW. (Tuple Space Explosion)
Smaller mask	• More resource/space in HW to hold more entries per subtable.

For better HW utilization, need flexibility to decomposition into smaller units to utilize all HW block based on usage of fields.

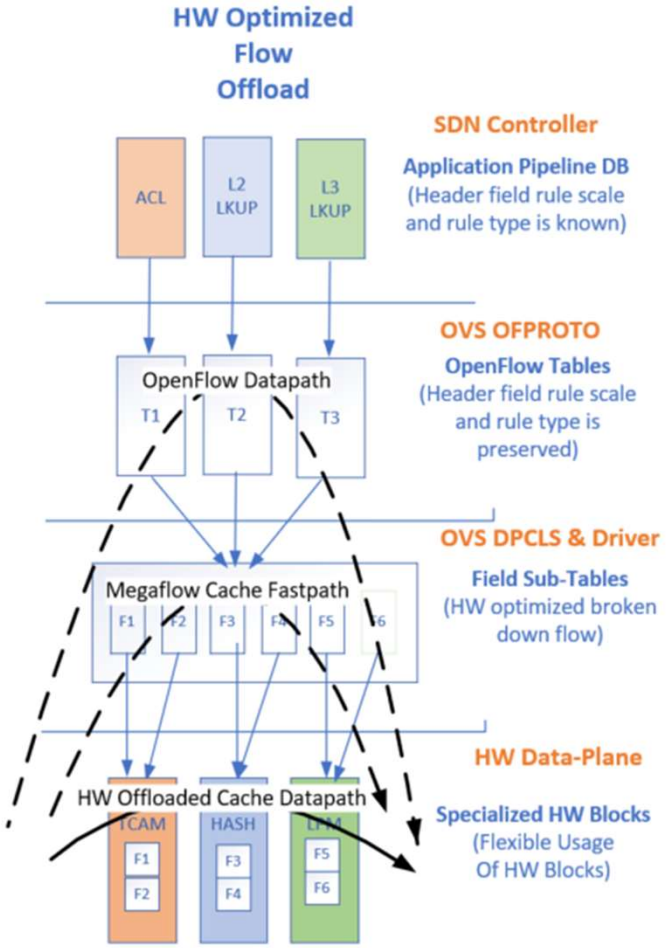


HW Optimized Flow Offload



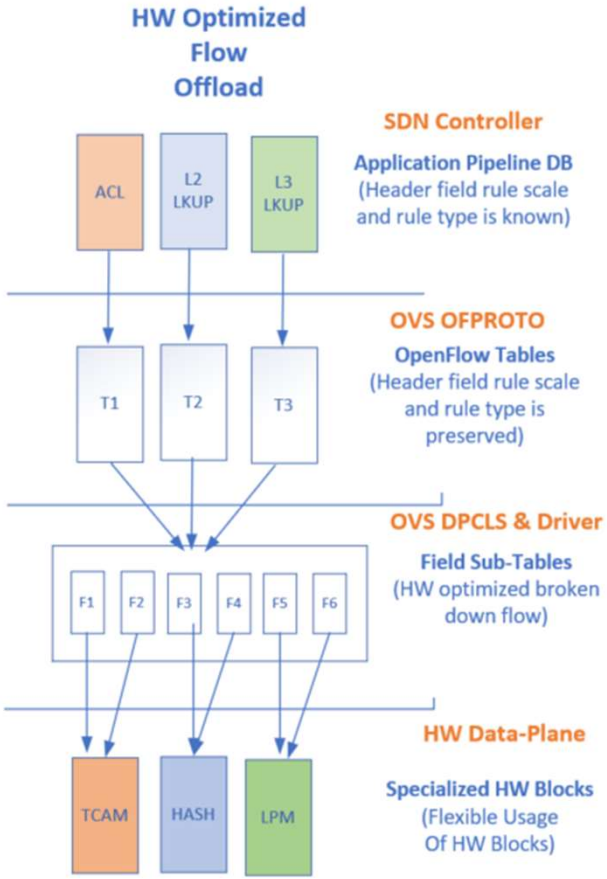
What is HW Optimized Field Subtable cache?

1. Cache of flows from OpenFlow rules
   - Doesn't represent all OpenFlow table (ofproto) rules.
   - Created using headers & metadata of 1st packet of a flow through OFPROTO datapath (ofproto-dpif).
2. Flow key fields with mask for aggregate of u-flows.
   - Each key field has mask derived from OF rules.
   - Same flows with same masks for individual fields are put into a " field subtable"
3. Field subtables implemented with different tables types.
   - Each field subtable need not be implemented with per-mask hash table alone.
4. Entries from all field subtables are non-overlapping without priorities
   - OpenFlow datapath (ofproto-dpif) unwildcarding algorithm creates non-overlapping entries.
   - First match == only possible match
5. Each field subtable is for single field of a header
   - Subtables hold generic key/mask for single fields.
6. A set of field subtable entries maps to an action set.
   - Need field subtable lookup for each field subtable.



Field Subtables in OFPROTO-DPIF & DPCLS

	Fields Subtable Characteristics
OFPROTO-DPIF	OpenFlow Tables - Each OF table w/ multiple subtables - Each subtable w/ segments of rules with segmented keys and masks by header fields. - Each rules with priority - Entries overlapping
DPCLS Megaflow	Field Subtables - Each field subtable w/ organized with data structure to hold smaller scale entries. - Each field subtable entries with no priority - Entries non-overlapping
HW	Field Subtables - Each field subtable maps to different HW blocks, some of which can handle multiple masks. - Each flow with no priority - Entries non-overlapping



Tuple Space Explosion(TSE)

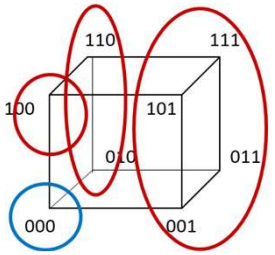
Tuple Space Explosion(TSE) Scenario in “Discrepancy of the MegaFlow Cache in OVS, Part II” by L. Csikor. (https://www.openvswitch.org/support/ovscon2019/day1/0924-levente_csikor_ovs_con.pdf)

Attack Case for megaflow:

- Single allow rule on SRC_IP, SRC_PORT, and DST_PORT with overlapping default drop rule.
- Produced 8192 (=32 * 16 * 16) masks.
  - Mask accommodates all fields
  - Each mask with single non-wildcard bit in all of SRC_IP, SRC_PORT, and DST_PORT fields.
- Performance:
  - Dramatic drop even in full offload case after 100 masks

Pipeline with field subtables:

- Produce 32 SRC_IP masks, 16 SRC_PORT masks, 16 DST_PORT masks
- Mask length fitting for each field length.
- All field subtables will fit in HW.



Prototypes

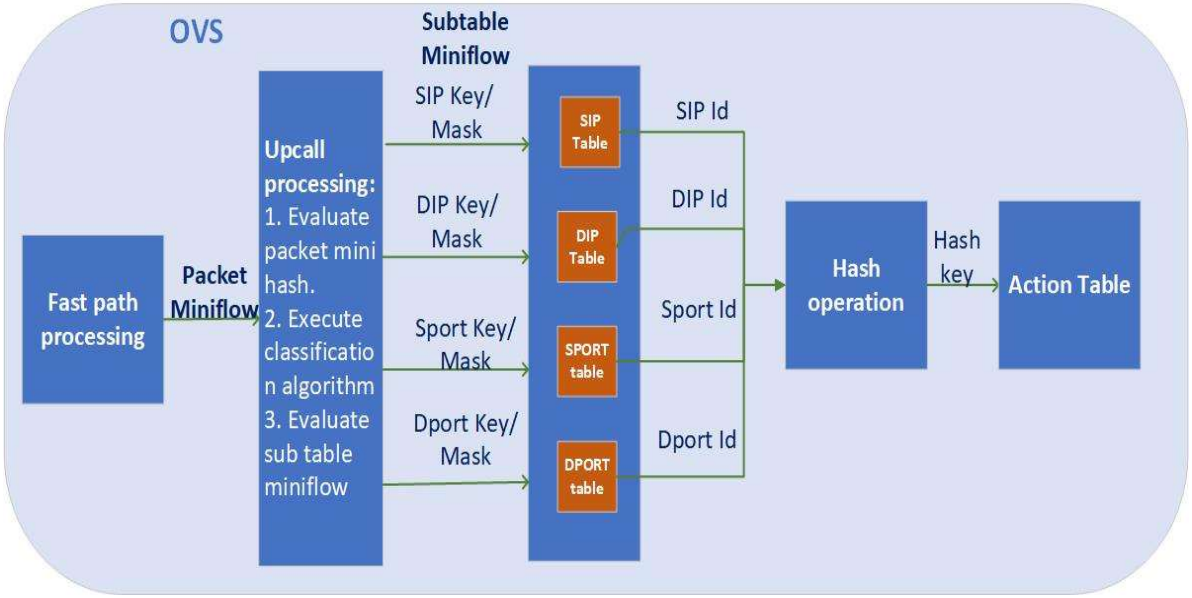
- Deriving field subtable from DPCLS subtable
- Partial OVS flow offload using Intel Ethernet 810 card.
- Full OVS flow offload using FPGA based NIC

Prototype: Deriving Field Subtable from DPCLS_SUBTABLE

- To verify the classification optimization algorithm, decomposition of the megaflow is done to form set of field specific sub tables in DPCLS and offload action table in PMD context. This prototype is implemented to ensure existing functionality of megaflow action and the action derived from this optimization algorithm is same.

Data path rule add flow: (Use subtable miniflow)

- OVS fastpath calls ofproto-dpif upcall processing upon miss of DPCLS megaflow cache.
- Decompose the offloaded megaflow from ofptofo classifier output into set of field key and mask.
- Add each field key and mask into respective field subtable and associate an identifier.
- Compute the hash key with combination of all field-specific identifiers and store action set from associated megaflow into new offload_action table



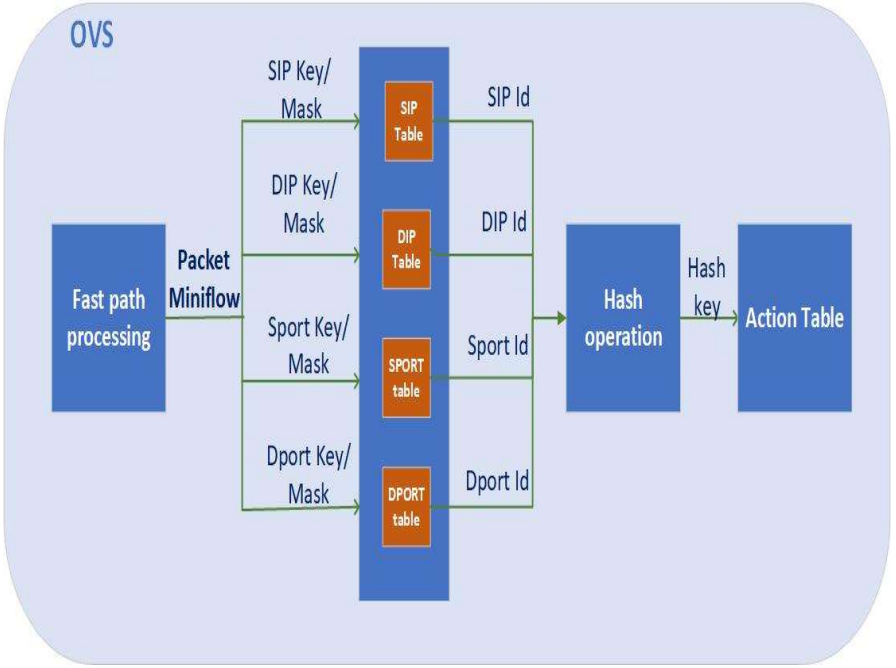
Prototype: Deriving Field Sub table from DPCLS_SUBTABLE

Lookup flow: (Use packet mini flow)

- Retrieve packet fields from packet miniflow
- Lookup each field subtable and find the identifiers for each field.
- Generate hash key from all the field-specific identifiers and retrieve the action from new offload_action table

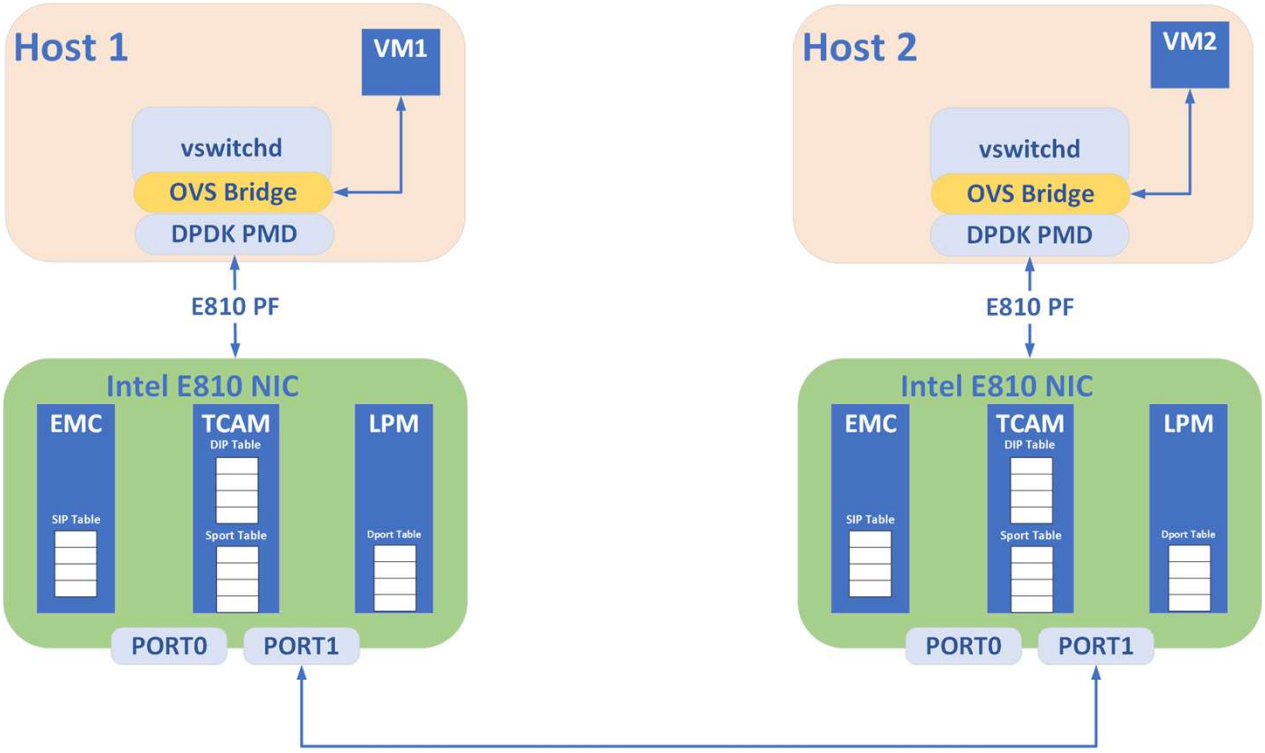
Delete flow:

- Lookup each field subtable and delete corresponding entries if reference count is 0.
- Cleanup the associated action in offload action table.

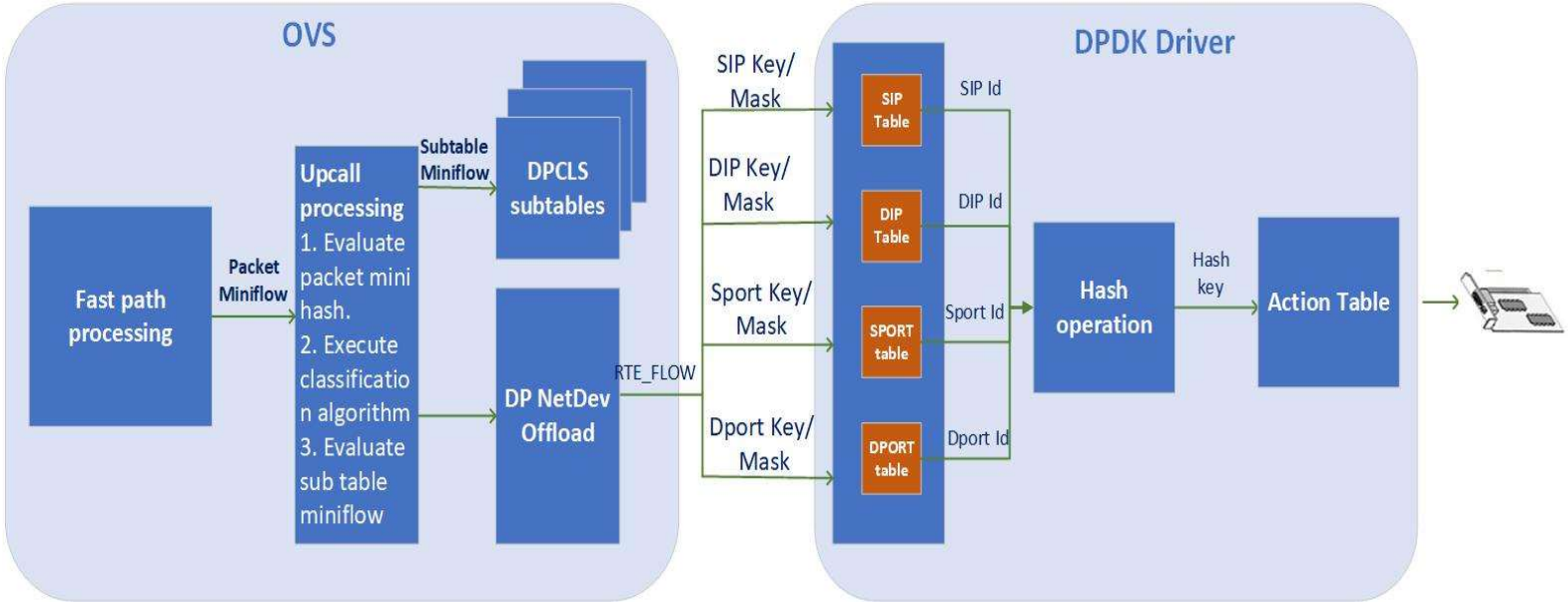


Prototype: Partial HW offload

- OVS Partial offload with mark action.
- Without changes in ofproto-dpif or dpcls, field subtable decomposition still can be done in the driver layer using the standard RTE_FLOW interface.



Implementation in Driver layer



Driver:

Data path rule add flow: (Use subtable miniflow)

- DPCLS offloads a megaflow to driver through RTE_FLOW.
- Driver decomposes the RTE_FLOW offload request into set of field key and mask.
- Add each field key and mask into respective field subtable and associate an identifier.
- Compute the hash key with combination of all field-specific identifiers and store action set from associated megaflow into new offload_action table

Implementation in Driver Layer

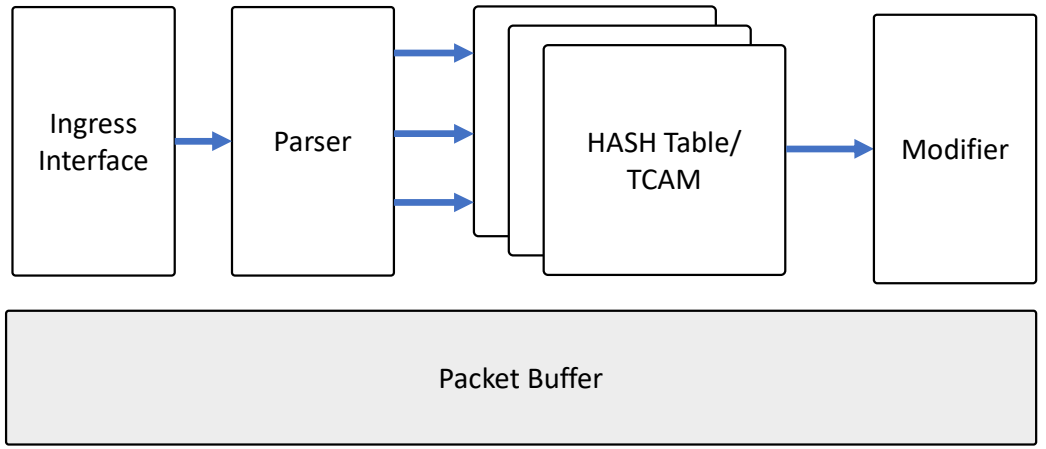
Driver:

Delete flow:

a) When revalidator thread deletes a megaflow through RTE_FLOW.
b) Driver looks up individual field subtables and deletes corresponding entries if the reference count is zero.
c) Driver deletes the corresponding action set from offload action table.
d) Sync individual subtables and action with HW tables.

Prototype: Full Offload to FPGA Pipeline for Field Sub-tables

- Offloading Field Sub tables on FPGA yet keeping it lightweight requires accommodating a large number of fast Lookup and hash tables with a variety of configurations
- It requires a variety of fast Hash and Lookup tables to be populated in a specific manner
- FPGAs allow a combination of static and dynamic configuration of memory size as well as their interconnection
- Latest Agilex FPGA(s) provide abundance of SRAMs (259Mb) running at over 500MHz to realize a high performance Megaflow pipeline in the FPGA.
- For a typical use case (example: local address: 1K, remote address: 10K, and Source & Destination Ports: 64K)
  - Proposed tables can fit on the chip with low utilization thereby reducing the latency, and maximizing search rate and performance



References

1. https://sites.google.com/view/tuple-space-explosion
2. The Discrepancy of the MegaFlow Cache in OVS Part II (https://www.openvswitch.org/support/ovscon2019/day1/0924-levente_csikor_ovs_con.pdf
3. The Design and Implementation of Open vSwitch https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-pfaff.pdf

BACKUP SLIDES

Implementation in driver

Driver:

1. Megaflows getting offloaded through RTE_FLOW
2. Maintain each RTE_FLOW_ITEM_TYPE_xxx in a field-subtables.
3. Maintain a scheme to map the list of IDs assigned to each field-subtable-entry for all fields to a set of action profile.
4. Each field-subtable-entries maintains the pointer back to the megaflow entries.
5. Each megaflow entries maintain pointers to the subtable-entries.
6. Mapping of each field-subtable-entries to HW is determined by driver.

	Driver
Update	As new rules are offloaded.
Benefit / Goal	Reactive megaflow offload mechanism

Scenarios to verify - 1

Scenarios	Description	Observation
Multiple masks in same OVS sub table with L3 parameters in key and rules	Send the traffic such a way that multiple packets contains same packet fields and hit the different OpenFlow rules.	1. Observed that different entries added in field specific sub tables 2. Action in the driver and OVS DPCLS is same.
Multiple masks in different OVS sub table with L3 parameters in key and rules	Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table	1. Observed that different entries added in field specific sub tables 2. Action in the driver and OVS DPCLS is same for further data packets.
Create multiple masks in same sub table, where overlapping is there and with same priority	Add open flow rules such that one rule contains <ip>/16 as key and other rule contains <ip>/24 as key and send the packets to hit both rules	1. OvS: Classifier already handling the overlapping and returning action corresponding <ip>/16 rule. 2. Action in the driver and OVS DPCLS are same.
Create multiple masks in same sub table with overlapping and with different priority(With longest prefix)	Add open flow rules such that one rule contains <ip>/16 as key and other rule contains <ip>/24 as keyand priority as high. send the packets to hit both rules	TBD

Scenarios to verify - 2

Scenarios	Description	Expectation
Delete OpenFlow when multiple masks installed in different OVS sub table	1. Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table 2. Delete the OpenFlow rules	When actions associated with mega flows being changed, offload actions also need to be changed in driver.
Delete Port when multiple masks installed in different OVS sub table	1. Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table 2. Delete the port associated with action	When mega flow rules being deleted, corresponding entries in offload table also need to be changed in driver.
Modify OpenFlow action when multiple masks installed in different OVS sub table	1. Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table 2. Modify the action associated with the rule.	When mega flow action being modified, offload action also need to be modified in driver.
Create LAG with LACP enabled in active-backup mode. Perform failover	1. Send traffic before failover 2. Perform failover 3. Send traffic after failover	After failover HW offload and DPCLS actions should be same.

Prototype at DPIF-NETDEV - 1

Scenarios	Description	Observation
Multiple masks in same OVS sub table with L3 parameters in key and rules	Send the traffic such a way that multiple packets contains same packet fields and hit the different OpenFlow rules.	1. Observed that different entries added in field specific sub tables 2. DPCLS and mega flow offload action same.
Multiple masks in different OVS sub table with L3 parameters in key and rules	Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table	1. Observed that different entries added in field specific sub tables 2. DPCLS and mega flow offload action same for further data packets.
Create multiple masks in same sub table, where overlapping is there and with same priority	Add open flow rules such that one rule contains <ip>/16 as key and other rule contains <ip>/24 as key and send the packets to hit both rules	1. OvS: Classifier already handling the overlapping and returning action corresponding <ip>/16 rule. 2. DPCLS and offload actions are same.
Create multiple masks in same sub table with overlapping and with different priority(With longest prefix)	Add open flow rules such that one rule contains <ip>/16 as key and other rule contains <ip>/24 as keyand priority as high. send the packets to hit both rules	TBD

Prototype at DPIF-NETDEV - 2

Scenarios	Description	Expectation
Delete OpenFlow when multiple masks installed in different OVS sub table	1. Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table 2. Delete the OpenFlow rules	When actions associated with mega flows being changed, offload actions also need to be changed
Delete Port when multiple masks installed in different OVS sub table	1. Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table 2. Delete the port associated with action	When mega flow rules being deleted, corresponding entries in offload table also need to be changed
Modify OpenFlow action when multiple masks installed in different OVS sub table	1. Send the traffic such a way that multiple packets contains different packet fields and each packet create a different sub table 2. Modify the action associated with the rule.	When mega flow action being modified, offload action also need to be modified.
Create LAG with LACP enabled in active-backup mode. Perform failover	1. Send traffic before failover 2. Perform failover 3. Send traffic after failover	After failover offload and DPCLS actions should be same.