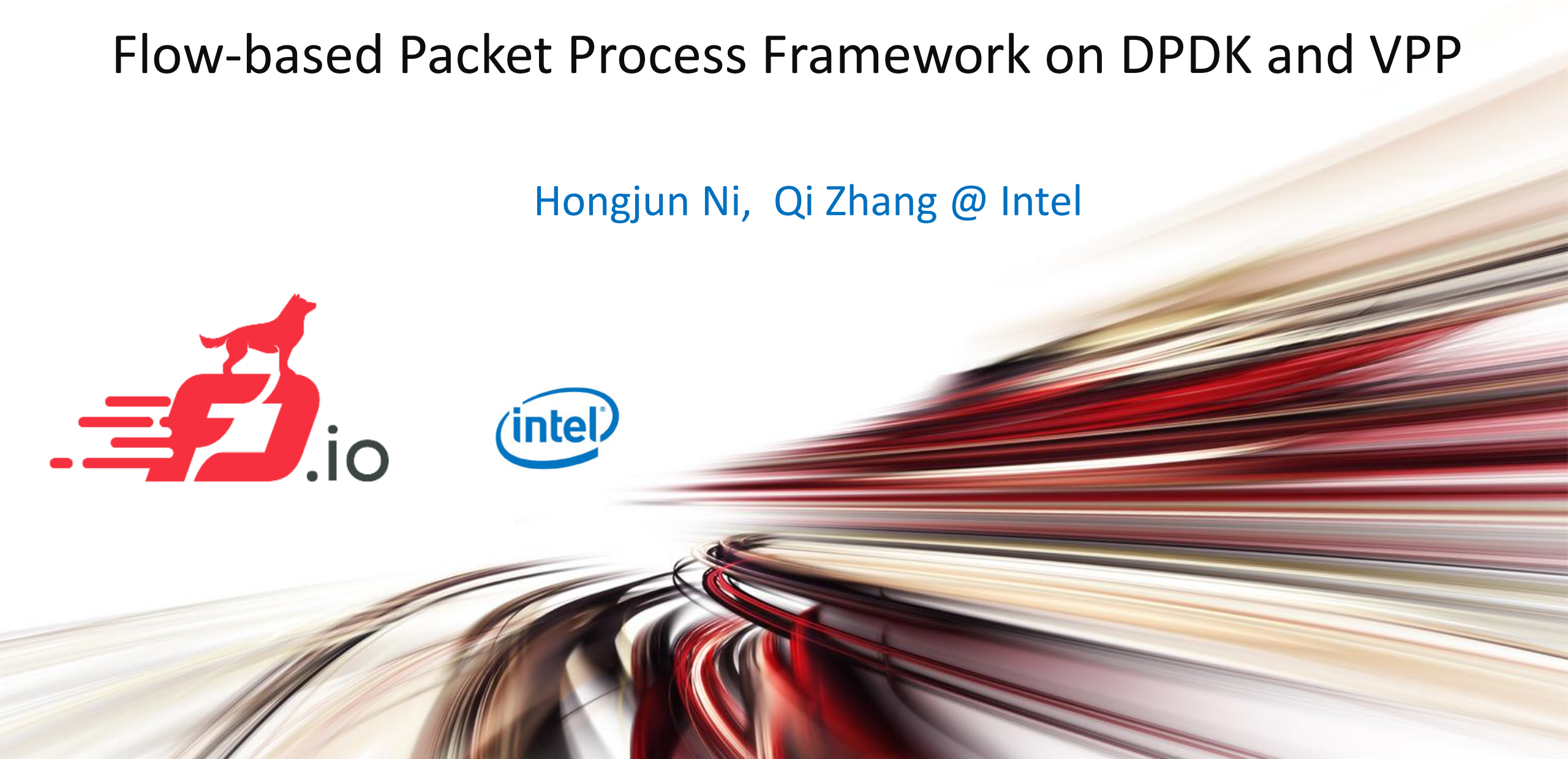
Flow-based Packet Process Framework on DPDK and VPP

Hongjun Ni, Qi Zhang @ Intel



Acknowledgement

VPP Community

John DiGiglio @ Intel

Ray Kinsella @ Intel

Jokul Li @ Intel

Xiang Wang @ Intel

Jerome Tollet @ Cisco

Dave Barach @ Cisco

Damjan Marion @ Cisco

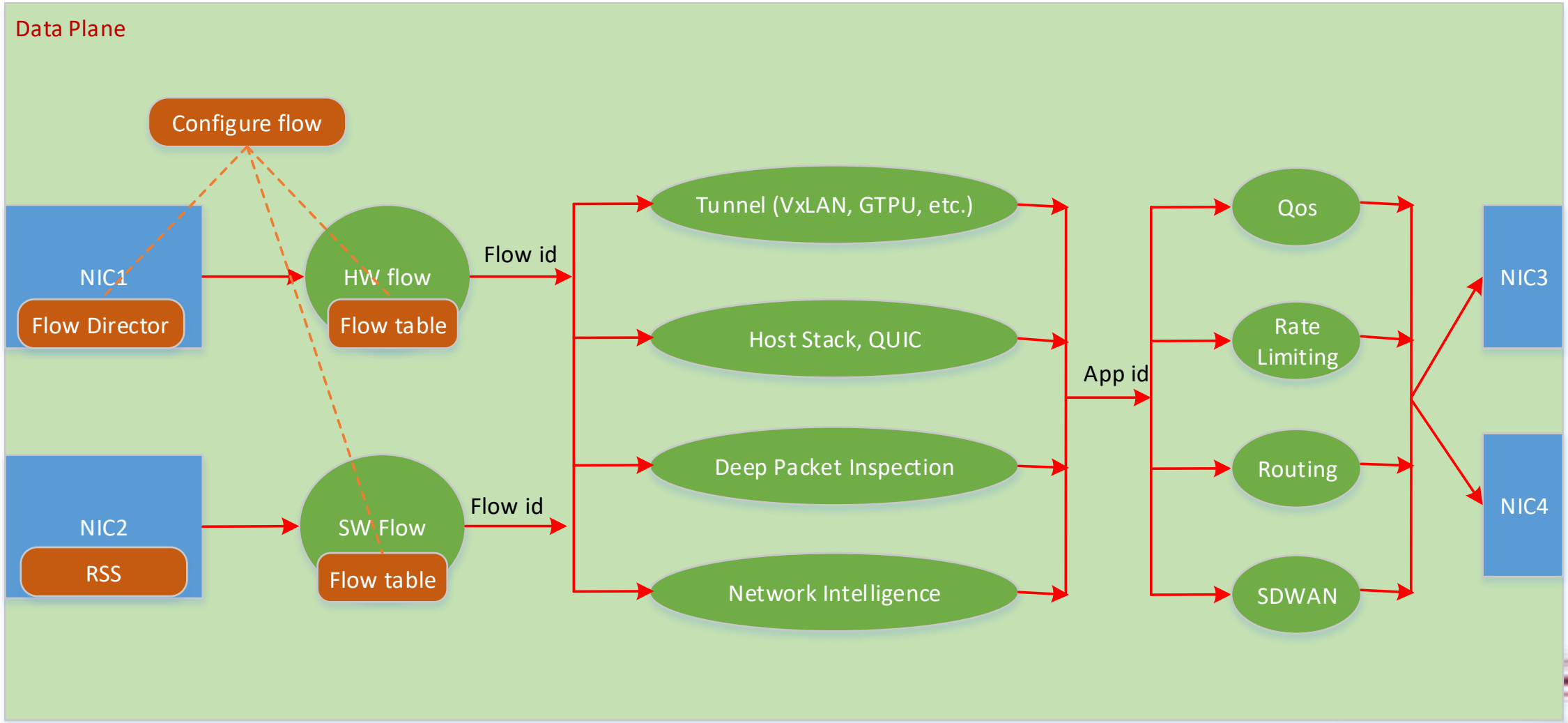
Andreas Schultz @ Travelping

Mathias Gumz @ Travelping

Agenda

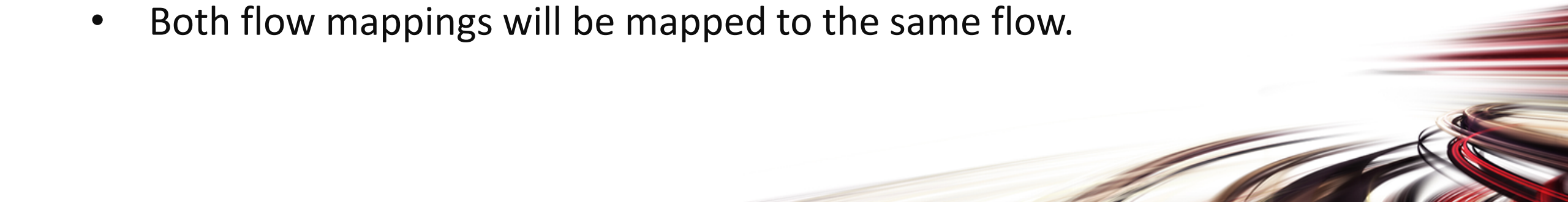
- Flow-based Solution
- HW Offload and SW Flow Process
- TCP Segments Reassembly
- Hyperscan Block and Stream Mode
- Identifying Layer 7 Applications
- Key Takeaway

Flow-based Solution



Flow Configuration

- Configures static flows with 5-tuple and VRF-aware.
- Supports both ipv4 and ipv6 flows.
- First try HW offload to NIC based on DPDK rte_flow mechanism.
- If failed, then will create SW flow mappings.
- Each flow creates two HW or SW flow mappings, i.e. bi-directional traffic.
- Both flow mappings will be mapped to the same flow.



HW Flow Offload

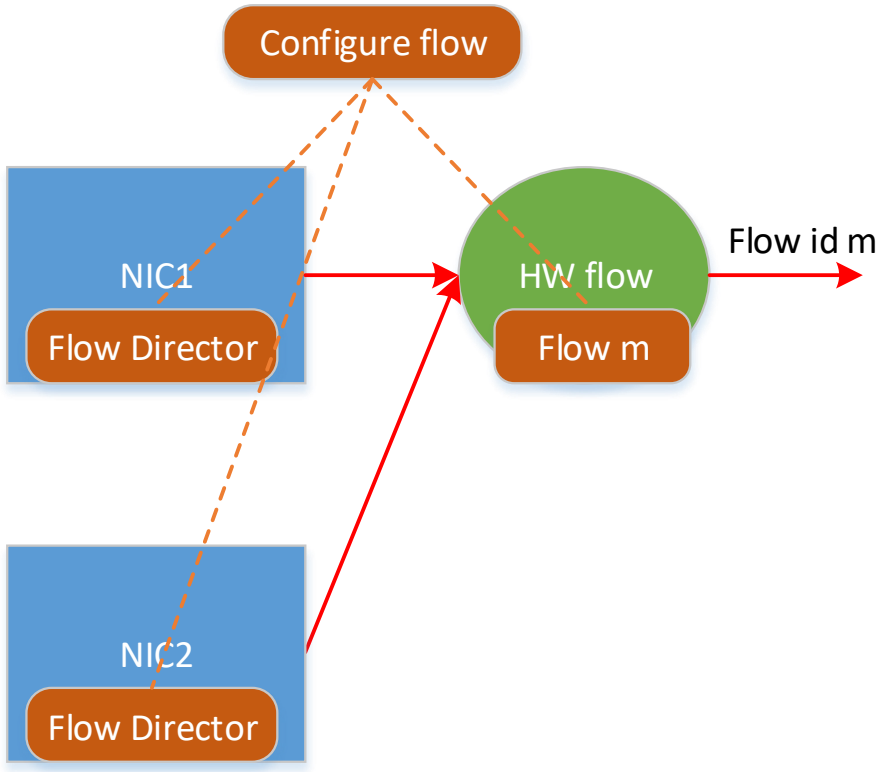
- Leverages rte_flow mechanism from DPDK:
  - Supports ipv4-n-tuple, ipv6-n-tuple, ipv4-vxlan, ipv6-vxlan, etc.
  - Supports rte_flow_item, rte_flow_action, etc.
  - Using rte_flow_create to create a flow.
  - If failed, then create a SW flow session.

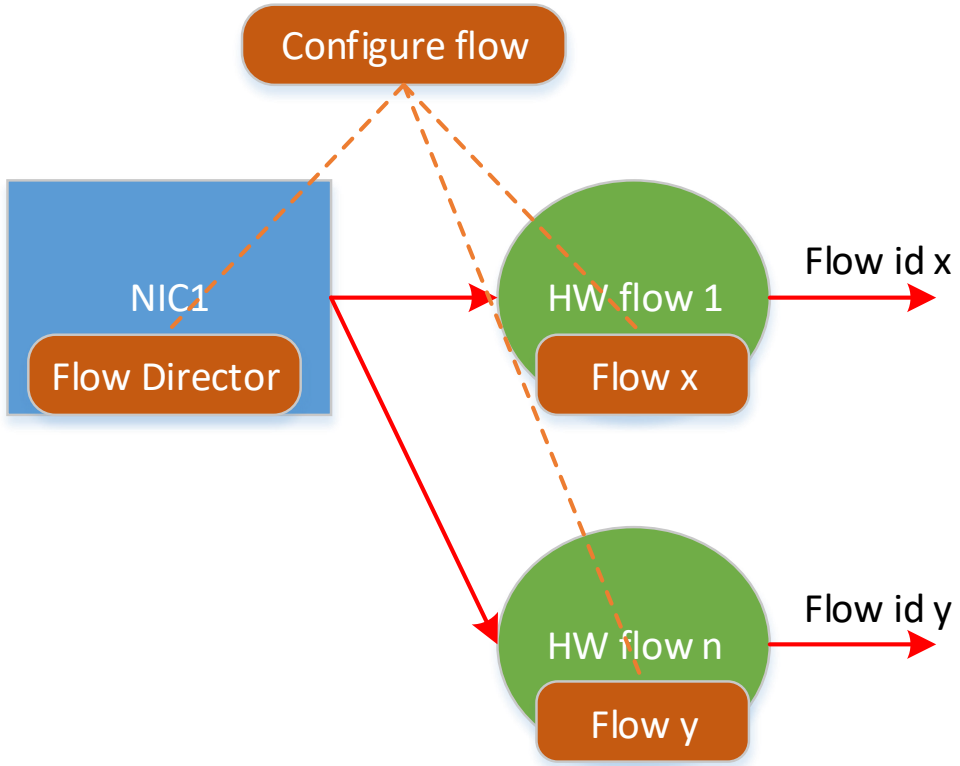
- HW flow matching:
  - If one packet matches a flow, then flow ID is marked by HW.
  - DPDK drivers sets flow ID to metadata of packet descriptor.
  - Subsequent features could retrieve flow ID from packet descriptor.

SW Flow Process

- Leverages RSS mechanism:
- Supports ipv4-n-tuple, ipv6-n-tuple, etc.
- Calculate hash value and look up SW flow table.
- If matched, set flow ID to metadata of packet descriptor.
- Subsequent features could retrieve flow ID from packet descriptor.

NICs & Flows





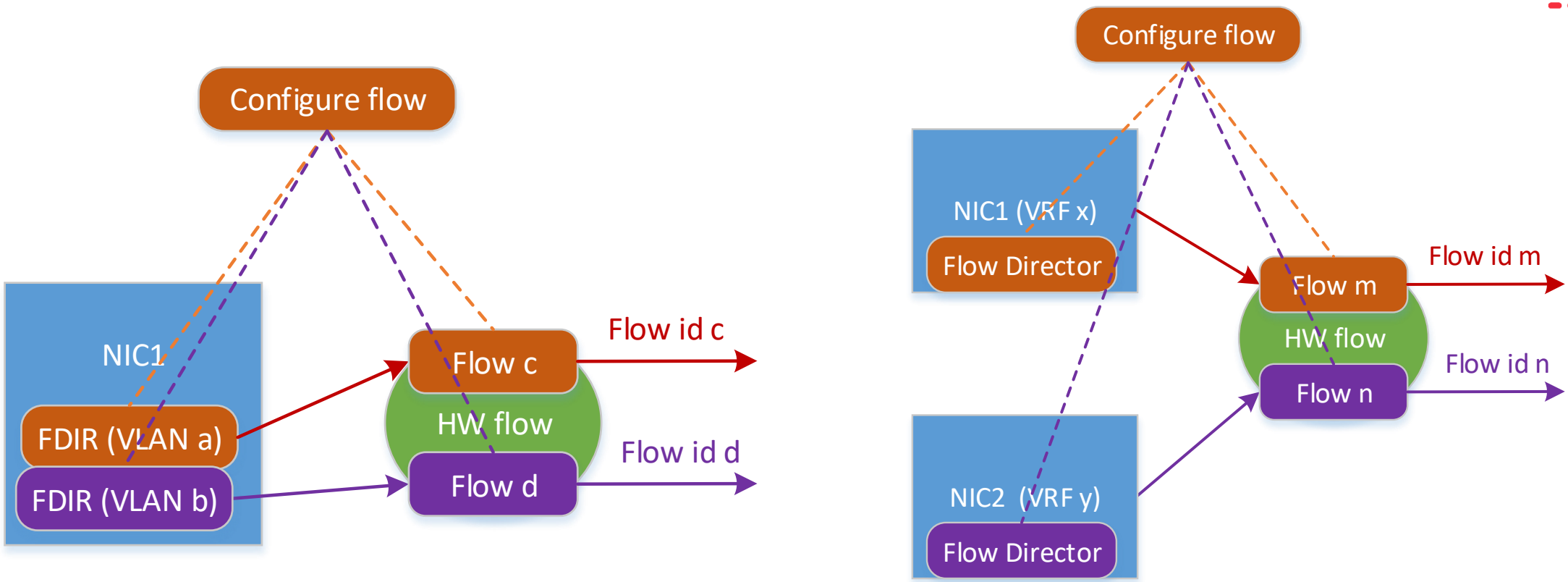
One flow is mapping to many NICs.

- Same flow from all NICs are sent to same core to handle.

One NIC contains many flows.

- Different flows from one NICs are sent to different core to handle.

BD/VRF Aware



If having same 5-tuple packet matching on 2 different VLANs or even on 2 different interfaces, they should be treated as different flows.

DPI Flow Process

- When HW flow offload matched, packets will be redirected to DPI plugin with dpi flow id in packet descriptor.

- If not, packets will be bypassed to DPI plugin from ip-input, and then lookup SW flow mapping table.

TCP Connection Track

- Tracks TCP three-way handshakes.
- Identify TCP traffic direction.
- Tracks TCP send sequence and ack sequence.

AppID Database

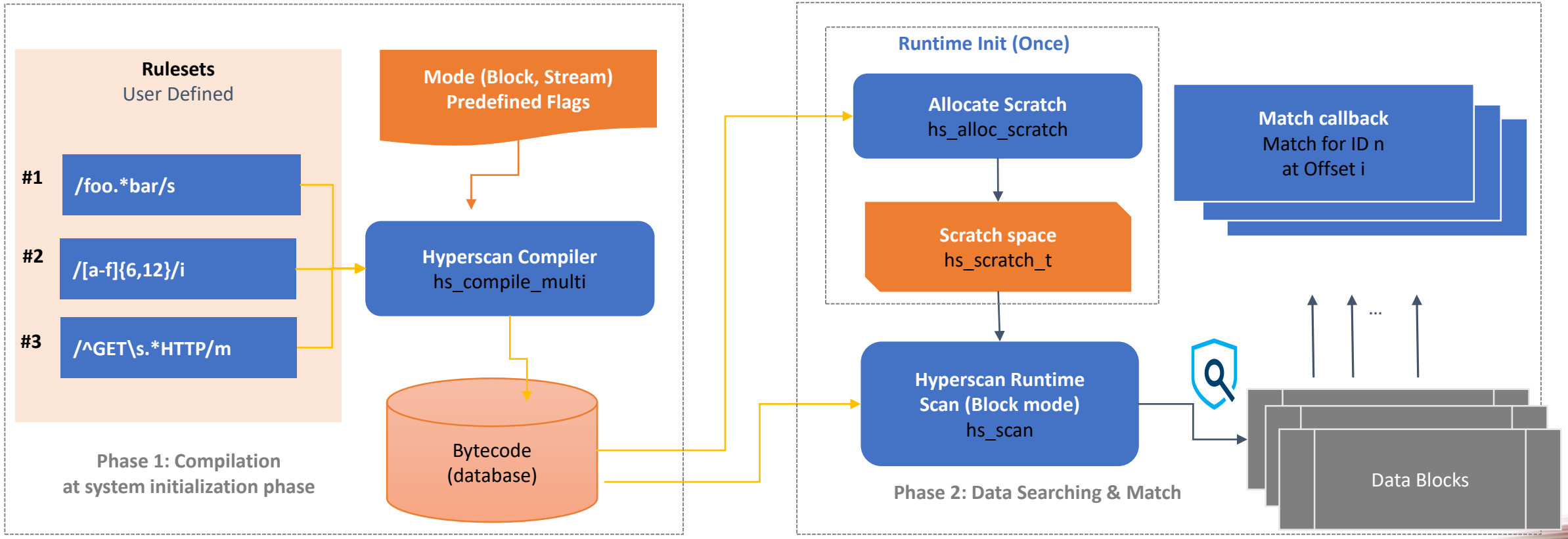
```
typedef struct dpi_app_match_rule_
{
  char *host;
  char *pattern;
  char *app_name;
  u32 app_id;
} dpi_app_match_rule;
```

```
typedef enum
{
  DPI_APP_CISCO = 1,
  …
  DPI_APP_INTEL = 7,
  …
} dpi_application_id_t;
```

```
dpi_app_match_rule app_match_rules[] = {

  {"www.cisco.com", NULL, "Cisco", DPI_APP_CISCO}
  ,
  …
  ,
  {"www.intel.com", NULL, "Intel", DPI_APP_INTEL}
  ,
  …
  }
```

How Hyperscan Works

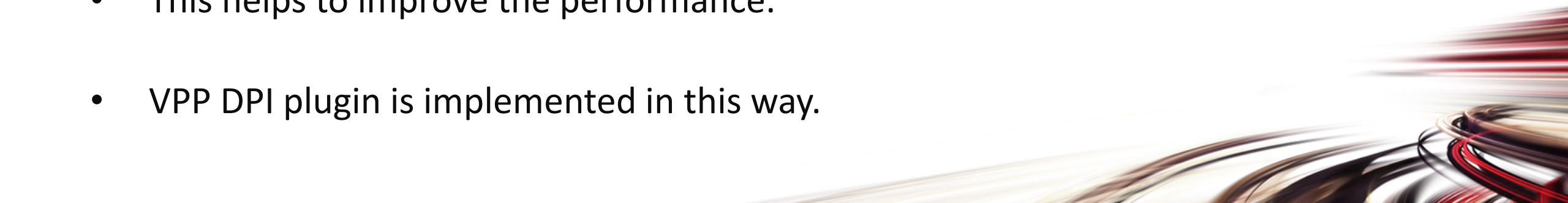


Block Mode and TCP Segments Reassembly

- Block mode can scan rules only in a complete payload.
- If defining a rule "abcde", then for Block Mode, "abcde“ should be in a complete PDU payload.
- Requirements for TCP Segments process:
  - Reassembly TCP segments first to a complete PDU payload.
  - Scan PDU payload through Block mode.
  - Fragment TCP segments again.
- This degrades the performance.
- Most DPI open source projects leveraging Hyperscan performs in above way.

Stream Mode and TCP Segments Reassembly

- Stream mode can scan rules straddling into different TCP segments.
- If defining a rule "abcde", then for Stream Mode, then "abc" can be reside in packet 1, and "de" can be in packet 2.
- Requirements for TCP Segments process:
  - Reassemble TCP segments reassembly on the fly.
  - Can handle out-of-order tcp segments.
  - Can handle overlapping segments.
- This helps to improve the performance.
- VPP DPI plugin is implemented in this way.



Identifying Layer 7 Applications

- Identify SSL/TLS certificate message and subsequent segments.
- Scan SSL/TLS certificate message through hyperscan, and get application ID if hit.
- If maximum packets for this flow are scanned and not matched, the detection will end up.

Test Stream

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.67.118.188	10.240.252.16	TCP	74	53282 → 912 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=226840610 TSecr=0 WS=128
2	0.002603	10.240.252.16	10.67.118.188	TCP	78	912 → 53282 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM=1 TSval=2110335135 TSecr=226840610 WS=64
3	0.002659	10.67.118.188	10.240.252.16	TCP	66	53282 → 912 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=226840611 TSecr=2110335135
4	0.002907	10.67.118.188	10.240.252.16	HTTP	271	CONNECT www.intel.cn:443 HTTP/1.1
5	0.005291	10.240.252.16	10.67.118.188	TCP	66	[TCP Window Update] 912 → 53282 [ACK] Seq=1 Ack=1 Win=262848 Len=0 TSval=2110335138 TSecr=226840611
6	0.006498	10.240.252.16	10.67.118.188	HTTP	105	HTTP/1.1 200 Connection established
7	0.006539	10.67.118.188	10.240.252.16	TCP	66	53282 → 912 [ACK] Seq=206 Ack=40 Win=29312 Len=0 TSval=226840612 TSecr=2110335139
8	0.008935	10.67.118.188	10.240.252.16	TLSv1.2	583	Client Hello
9	0.082730	10.240.252.16	10.67.118.188	TLSv1.2	1414	Server Hello
10	0.082770	10.240.252.16	10.67.118.188	TCP	1414	912 → 53282 [ACK] Seq=1388 Ack=723 Win=262848 Len=1348 TSval=2110335215 TSecr=226840613 [TCP segment of a reassembl
11	0.082789	10.67.118.188	10.240.252.16	TCP	66	53282 → 912 [ACK] Seq=723 Ack=2736 Win=34688 Len=0 TSval=226840631 TSecr=2110335215
12	0.082802	10.240.252.16	10.67.118.188	TCP	1466	912 → 53282 [PSH, ACK] Seq=2736 Ack=723 Win=262848 Len=1400 TSval=2110335215 TSecr=226840613 [TCP segment of a reas
13	0.082811	10.67.118.188	10.240.252.16	TCP	66	53282 → 912 [ACK] Seq=723 Ack=4136 Win=37504 Len=0 TSval=226840631 TSecr=2110335215
14	0.082823	10.240.252.16	10.67.118.188	TLSv1.2	1159	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
15	0.089214	10.67.118.188	10.240.252.16	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
16	0.089795	10.67.118.188	10.240.252.16	TLSv1.2	243	Application Data
17	0.089867	10.67.118.188	10.240.252.16	TLSv1.2	1172	Application Data
18	0.092239	10.240.252.16	10.67.118.188	TCP	66	912 → 53282 [ACK] Seq=5229 Ack=1026 Win=262656 Len=0 TSval=2110335225 TSecr=226840633
19	0.109791	10.240.252.16	10.67.118.188	TLSv1.2	308	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
20	0.109826	10.240.252.16	10.67.118.188	TLSv1.2	143	Application Data
21	0.110345	10.67.118.188	10.240.252.16	TCP	66	53282 → 912 [ACK] Seq=2132 Ack=5548 Win=42880 Len=0 TSval=226840638 TSecr=2110335242
22	0.110396	10.67.118.188	10.240.252.16	TLSv1.2	104	Application Data
23	0.123400	10.240.252.16	10.67.118.188	TLSv1.2	1119	Application Data
24	0.123448	10.240.252.16	10.67.118.188	TLSv1.2	1697	Application Data
25	0.123466	10.67.118.188	10.240.252.16	TCP	66	53282 → 912 [ACK] Seq=2170 Ack=8232 Win=48768 Len=0 TSval=226840641 TSecr=2110335256

Test Result

```
Packet 8

00:01:16:174624: pg-input
  stream cap, 583 bytes, 1 sw_if_index
  current data 0, length 583, buffer-pool 0, ref-cc
  IP4: 00:1e:67:e6:fe:18 -> 00:00:5e:00:01:01
  TCP: 10.67.118.188 -> 10.240.252.16
    tos 0x00, ttl 64, length 569, checksum 0xa22f
    fragment id 0x0e90, flags DONT_FRAGMENT
  TCP: 53282 -> 912
    seq. 0x0e375f7c ack 0x2dcbd00a
    flags 0x18 PSH ACK, tcp header: 32 bytes
    window 229, checksum 0x8a2b
00:01:16:174652: ethernet-input
  frame: flags 0x1, hw-if-index 3, sw-if-index 3
  IP4: 00:1e:67:e6:fe:18 -> 00:00:5e:00:01:01
00:01:16:174669: ip4-input
  TCP: 10.67.118.188 -> 10.240.252.16
    tos 0x00, ttl 64, length 569, checksum 0xa22f
    fragment id 0x0e90, flags DONT_FRAGMENT
  TCP: 53282 -> 912
    seq. 0x0e375f7c ack 0x2dcbd00a
    flags 0x18 PSH ACK, tcp header: 32 bytes
    window 229, checksum 0x8a2b
00:01:16:174679: dpi4-input
  DPI from flow 0 app_id -1 next 0 error 0
```

```
Packet 9

00:01:16:174624: pg-input
  stream cap, 1414 bytes, 1 sw_if_index
  current data 0, length 1414, buffer-pool 0, ref-count 1,
  IP4: 00:04:96:9b:aa:83 -> 00:1e:67:e6:fe:18
  TCP: 10.240.252.16 -> 10.67.118.188
    tos 0x00, ttl 56, length 1400, checksum 0x2f5f
    fragment id 0xc621
  TCP: 912 -> 53282
    seq. 0x2dcbd00a ack 0x0e376181
    flags 0x10 ACK, tcp header: 32 bytes
    window 4107, checksum 0xa32c
00:01:16:174652: ethernet-input
  frame: flags 0x1, hw-if-index 3, sw-if-index 3
  IP4: 00:04:96:9b:aa:83 -> 00:1e:67:e6:fe:18
00:01:16:174669: ip4-input
  TCP: 10.240.252.16 -> 10.67.118.188
    tos 0x00, ttl 56, length 1400, checksum 0x2f5f
    fragment id 0xc621
  TCP: 912 -> 53282
    seq. 0x2dcbd00a ack 0x0e376181
    flags 0x10 ACK, tcp header: 32 bytes
    window 4107, checksum 0xa32c
00:01:16:174679: dpi4-input
  DPI from flow 0 app id 7 next 1 error 0
00:01:16:174743: ip4-lookup
  fib 0 dpo-idx 1 flow hash: 0x00000000
  TCP: 10.240.252.16 -> 10.67.118.188
    tos 0x00, ttl 56, length 1400, checksum 0x2f5f
    fragment id 0xc621
```

Key Takeaway

- Provides a flow-based framework for advanced packet processing.
- Supports HW flow offloading and SW flow process.
- Supports Hyperscan Stream Mode.
- Supports TCP segments reassembly on the fly.

Thank you !

Q & A

Email : hongjun.ni@intel.com
qi.z.zhang@intel.com