DPDK SUMMIT
Powering the Future of Networking Software

12–13 May 2026

Cryptographic Offloading to DPU/XPU PCI Cards

Akhil Goyal, Anoob Joseph
Marvell Technology

@handle

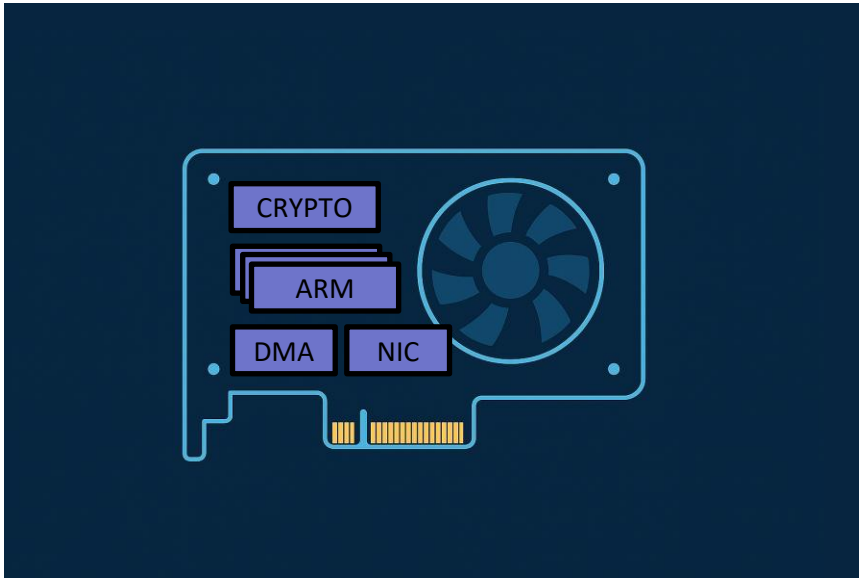
Agenda

- DPU/XPU cards for Offloading Crypto
- VirtIO-Crypto for Offloading
- Liquid-Crypto as an alternative
  - Comparison of the approaches
  - Suggested improvements

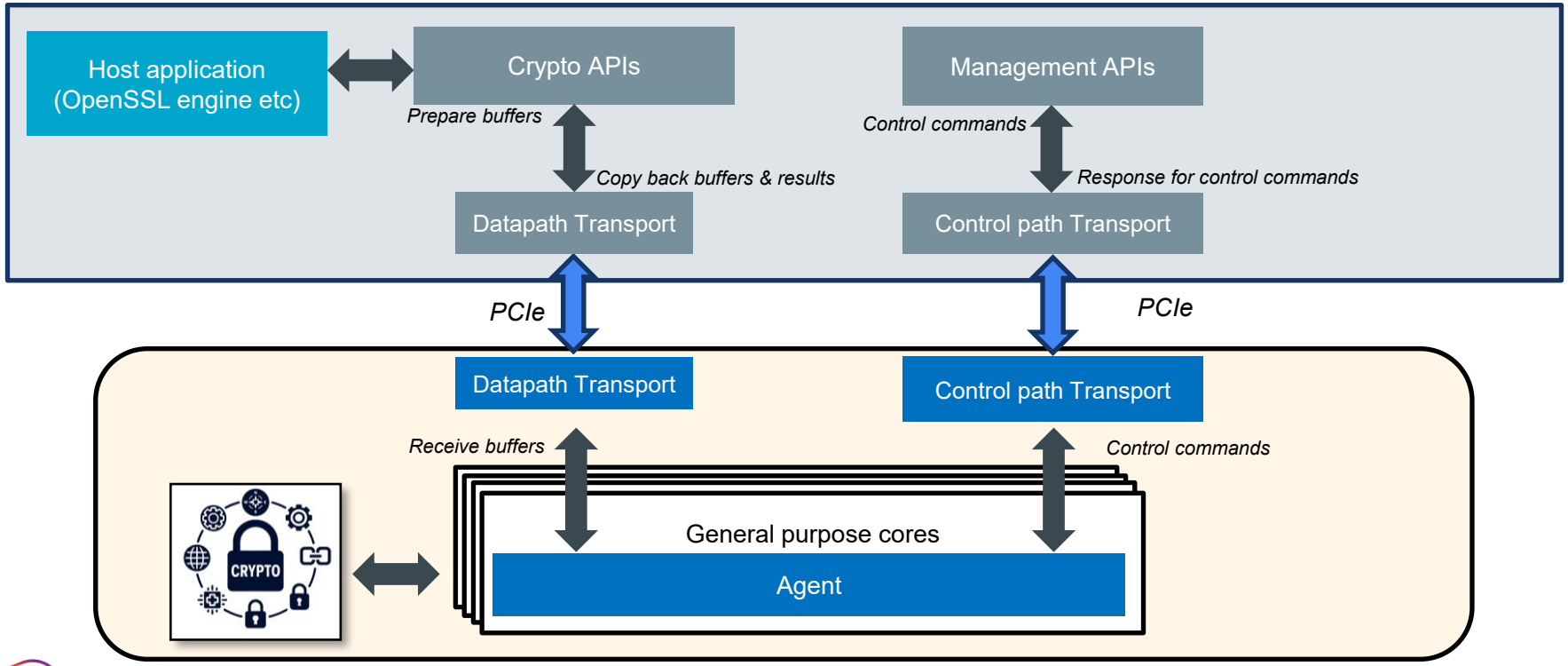
DPU/XPU cards for Offloading Crypto

DPU/XPU cards for Offloading Crypto

- Offload heavy cryptographic operations to PCIe card
- Efficient and low power ARM cores for handling datapath
- DMA (rte_dmadev) or NIC (rte_ethdev) for handling datapath



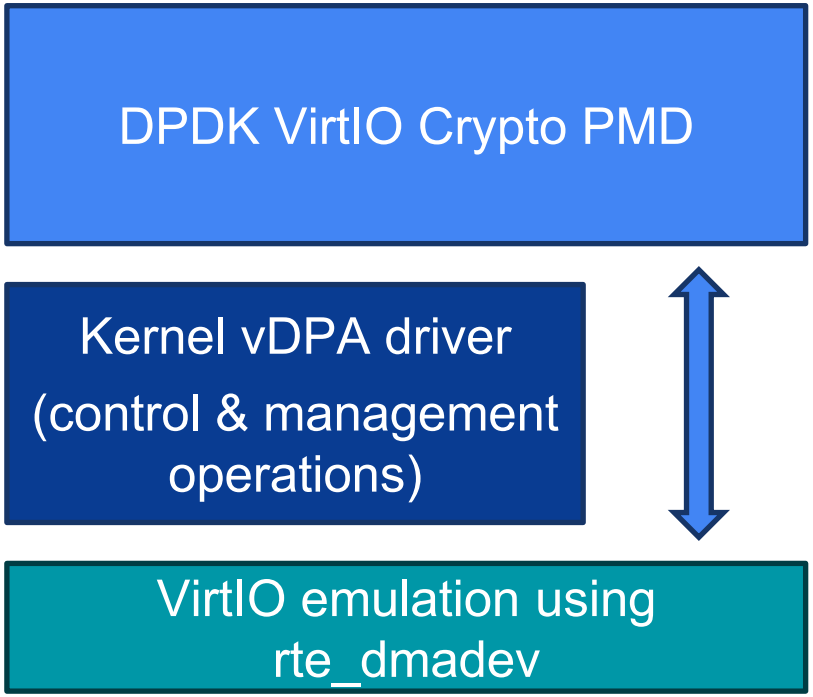
Software Architecture




Virtio Crypto

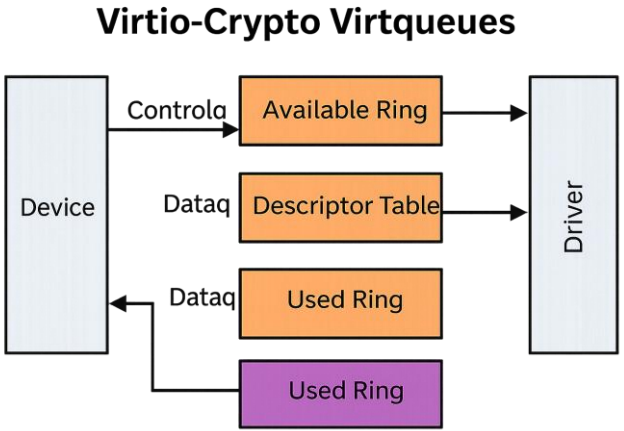
VirtIO Crypto - components

- Custom vDPA driver for very low level hardware access
- Standard VirtIO configuration by DPDK VirtIO Crypto PMD
- Can also support kernel VirtIO Crypto PMD



Software Interface – VirtIO Crypto queues

- Control Queue (controlq) - slow path for Crypto session management (Algo/key/other params)
- Descriptor Table - Used by both controlq and dataq. Fixed-size descriptor chains representing Control messages (session ops) and Crypto data ops (scatter/gather)
- Data Queue(s) (dataq) - fast path for actual crypto operations (Encrypt / Decrypt / Authenticate / Verify) with per lcore queue. Batched descriptor chains for throughput
- Used Ring - Written by the host virtio backend. Indicates completed requests, Driver polls used ring, reclaims descriptors and buffers



VirtIO Crypto PMD in DPDK Challenges

- Lack of vDPA backend – addressed
- Lack of packed queue support – addressed
- Lack of asymmetric crypto support – addressed

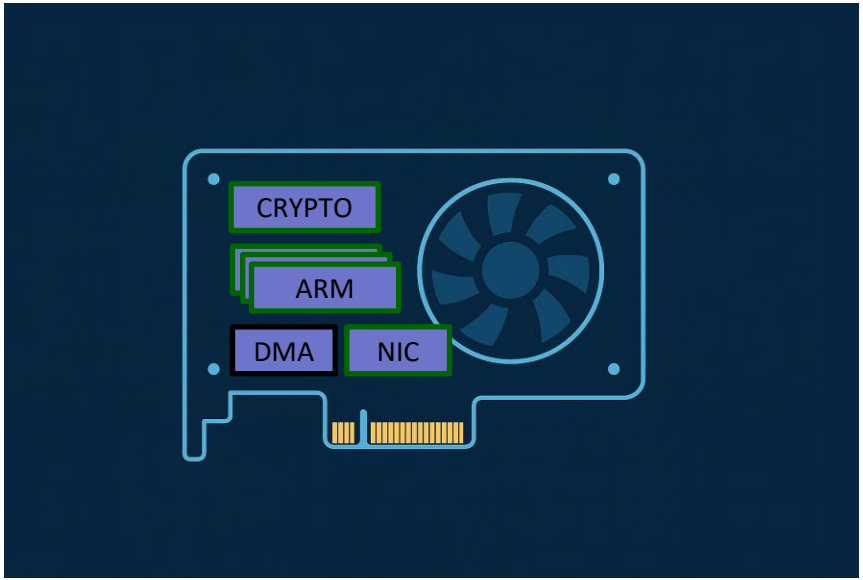
VirtIO Crypto PMD in DPDK – Disadvantages

- VirtIO handles control path. Management plane is still required
- Lack of algorithmic support
- Session create happen via blocking command queue call.
  - Asymmetric operations typically only use one session for one operation
- Large transport overhead
- Multiple Translations

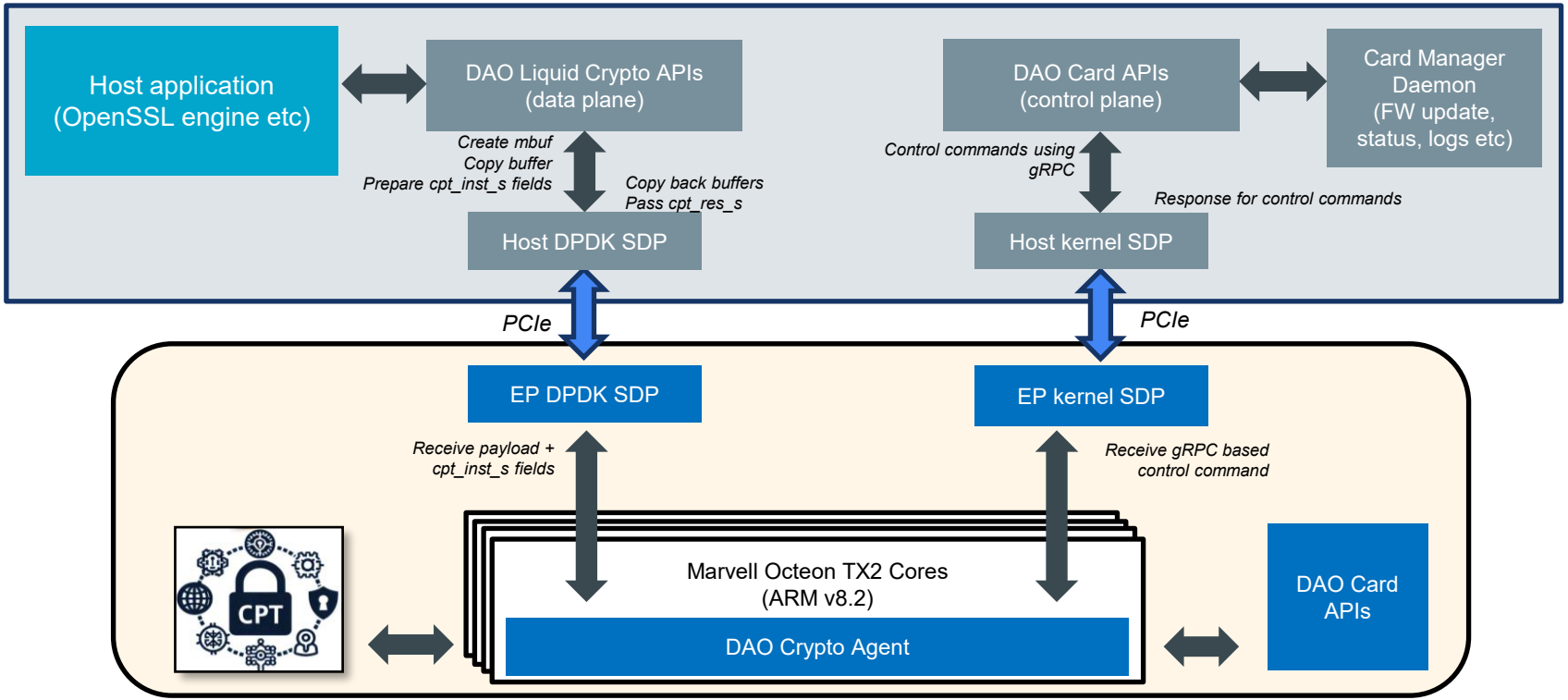

Liquid Crypto

Liquid Crypto

- Use rte_ethdev as transport
- Introduce custom-scalable packet formats
- Introduce crypto APIs on top of rte_ethdev APIs
- Use gRPC as transport for control & management plane



Architecture

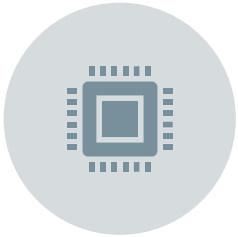


Liquid Crypto

Zero copy using huge pages

Standard DPDK ethdev APIs for scalability

Crypto APIs inspired from DPDK cryptodev



Reuse kernel networking & gRPC for control - management plane

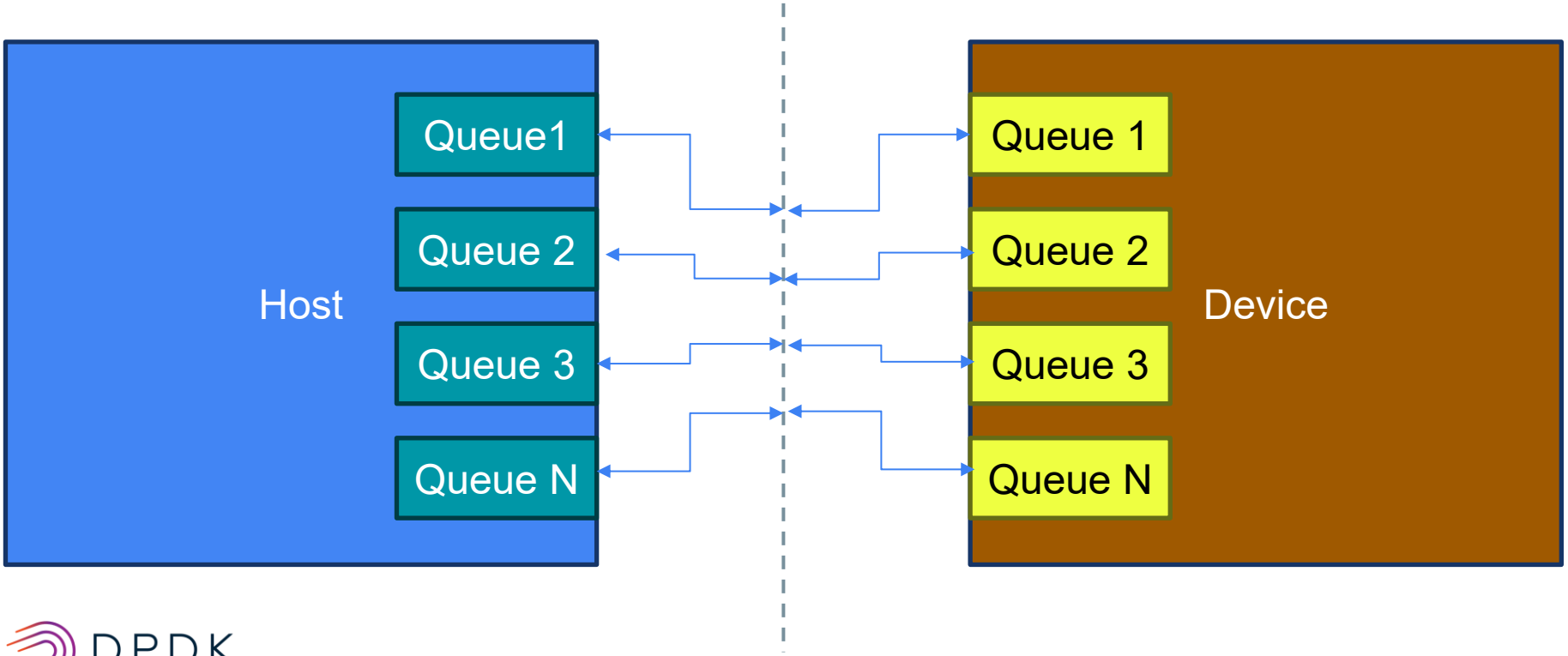
Liquid Crypto APIs – symmetric crypto

```
/**
* Enqueue a burst of requests to perform symmetric crypto operations on the
* crypto device.
*
* Note: The max number of requests that can be enqueued at once is 128.
*
* @param dev_id
* The identifier of the device.
* @param qp_id
* The index of the queue pair on which the operations are to be enqueued.
* @param op
* The array of pointers to *dao_lc_sym_op* structures containing the operations
* to be enqueued.
* @param nb_ops
* The number of operations to enqueue.
*
* @return
* The number of operations enqueued. The return value can be less than
* the number of operations requested to be enqueued if the queue is full or
* if there are any errors in the operations. In case of errors, 'rte_errno'
* will be set to indicate the cause.
* - EINVAL, indicating an invalid argument.
* - ENOMEM, indicating an out of memory error.
* - ENOSPC, indicating that there is no space left in the queue.
* - EIO, indicating an I/O error.
*/
uint16_t dao_liquid_crypto_sym_enqueue_burst(uint8_t dev_id, uint16_t qp_id,
                                             struct dao_lc_sym_op *op, uint16_t nb_ops);
```

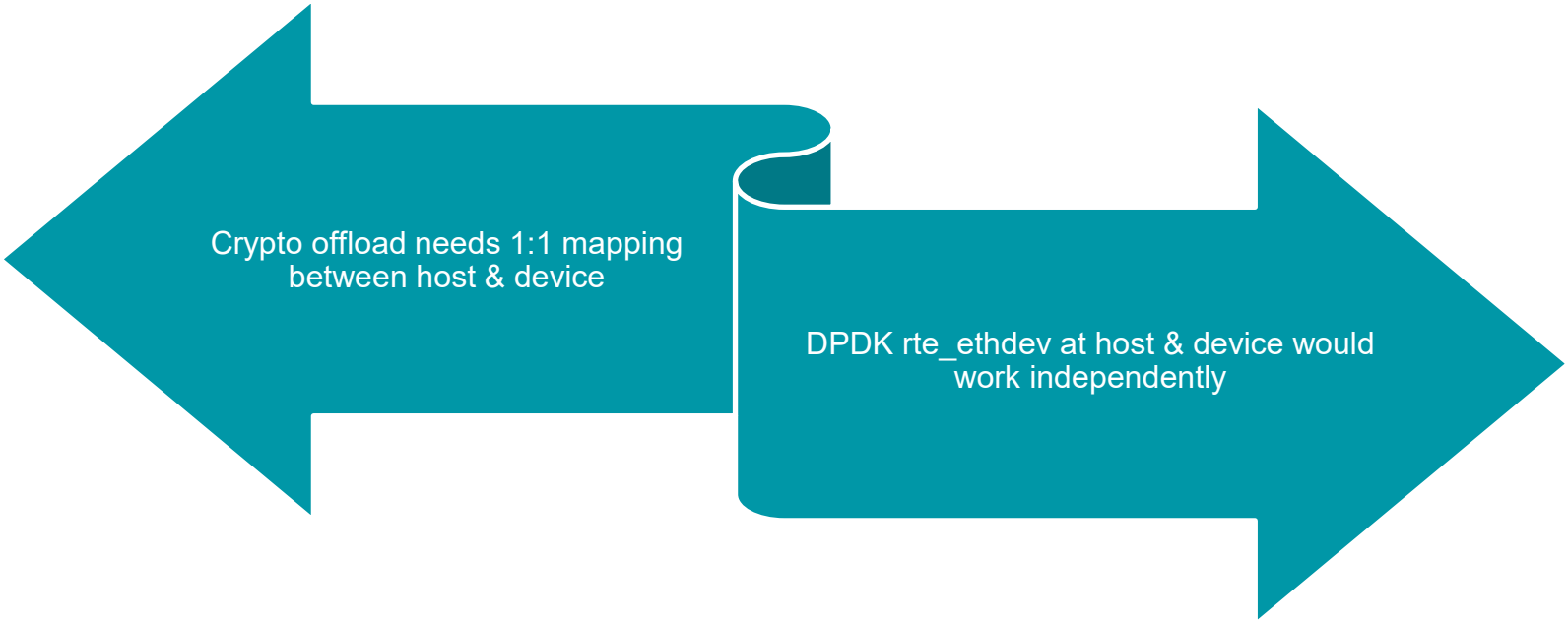
Liquid Crypto APIs – asymmetric crypto

```
/**
* Enqueue request to perform RSA CRT encrypt operation on the crypto device.
*
* @param dev_id
* The identifier of the device.
* @param qp_id
* The index of the queue pair on which the operation is to be enqueued.
* @param mod_len
* The length of the modulus. Value must be even and should be at least 34
* bytes and at most 1024 bytes.
* @param msg_len
* The length of the message in bytes. Value must be at most mod_len - 11.
* @param q
* The address of the buffer containing the first factor. Length of this buffer
* must be mod_len/2 bytes and the value must be odd.
* @param dQ
* The address of the buffer containing the first factor's CRT exponent. Length
* of this buffer must be mod_len/2 bytes.
* @param p
<snip>
*/
int dao_liquid_crypto_enq_op_pkcs1v15enc_crt(uint8_t dev_id, uint16_t qp_id, uint16_t mod_len,
               uint16_t msg_len, const uint8_t *q, const uint8_t *dQ,
               const uint8_t *p, const uint8_t *dP,
               const uint8_t *qInv, const uint8_t *msg, uint8_t *em,
               uint64_t op_cookie);
```

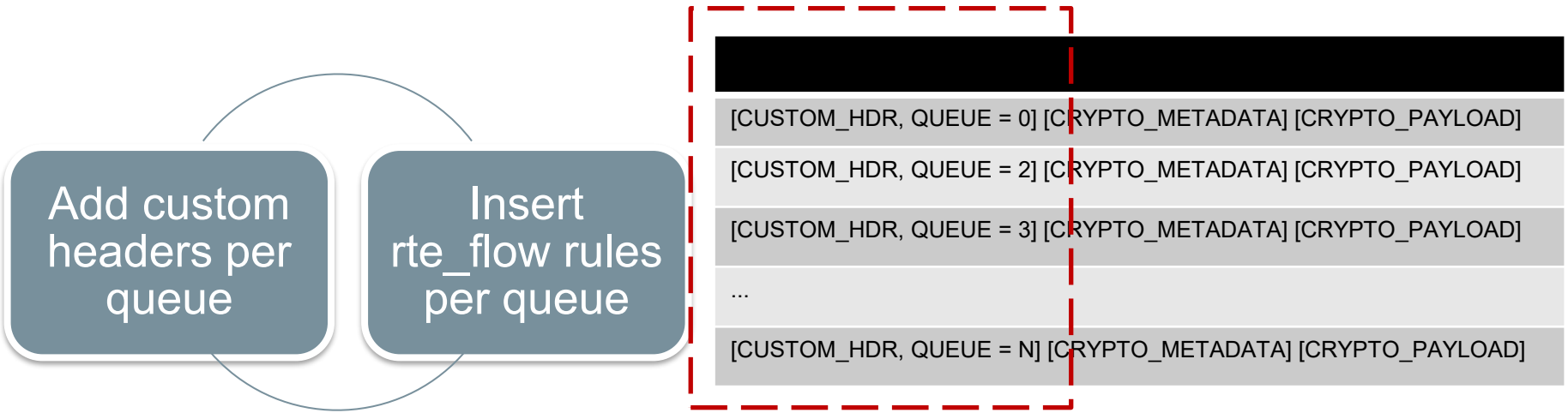
1:1 mapping using RSS – example of advanced use



1:1 mapping using RSS – example of advanced use



1:1 mapping: Option 1 – rte_flow



- ✓ Simple to implement
- ✓ Most PMDs support rte_flow

- – rte_flow configuration can take time
- – Per packet transport overhead

1:1 mapping: Option 2 – RSS to get fixed mapping

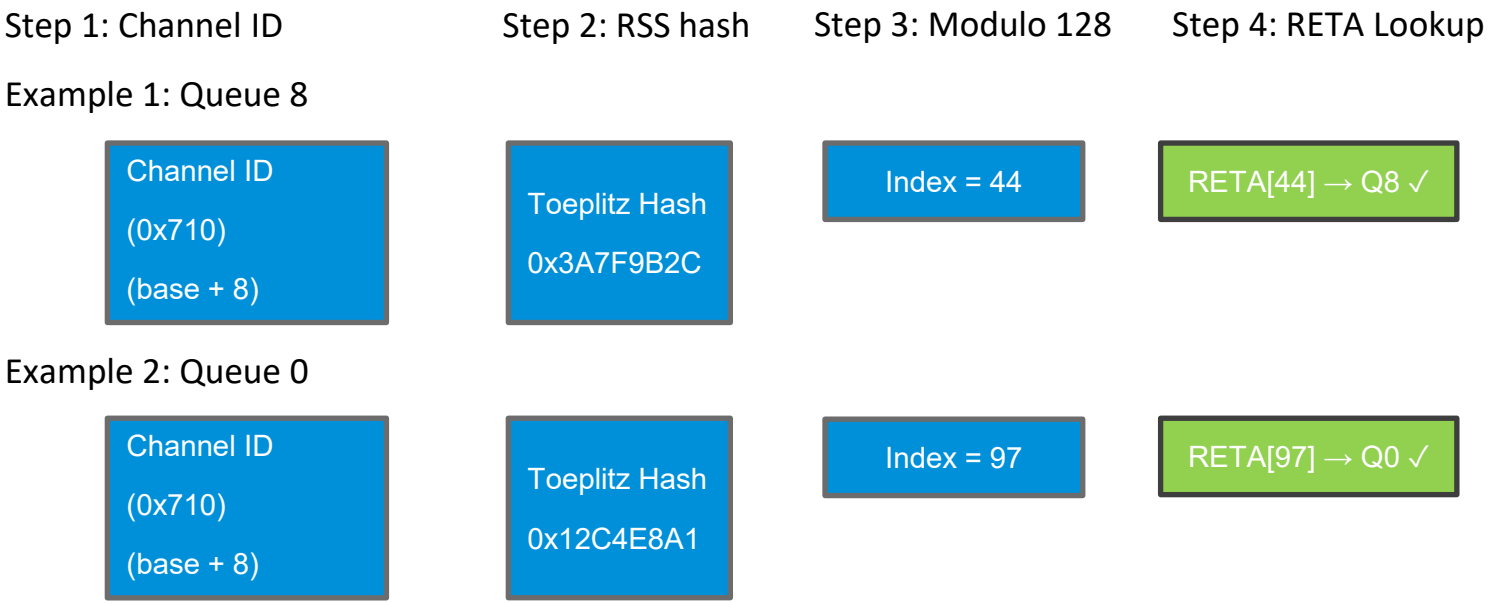
- Reverse engineer RSS to achieve fixed mapping
- Pre-configuration (done once at initialization):

```
For each queue N (N = 0 to 7):
        channel[N] = base + (port x 8) + N
        hash = rss_hash(channel[N])
        reta[hash % 128] = N
```

Index	Queue
0	
1	
2	
3	
…	
44	8
…	
97	0
…	
127	

Sample reta table

1:1 mapping: Option 2 – Flow Diagram

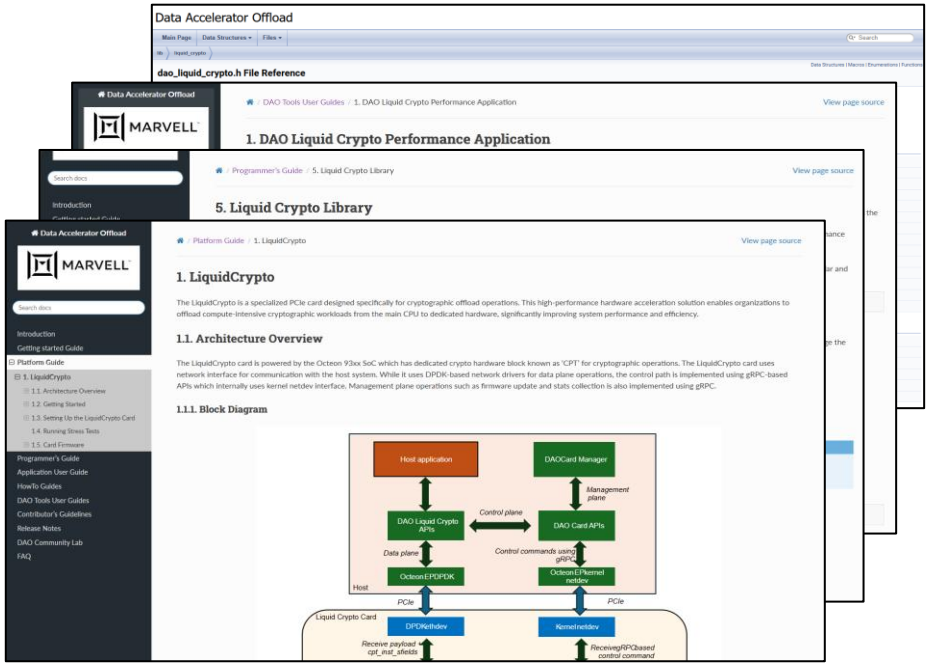


Liquid Crypto Components

- Management plane
  - gRPC based
- Control plane
  - gRPC for messaging
  - rte_cryptodev & rte_ethdev in device
  - rte_ethdev in host
- Data plane
  - rte_cryptodev & PMD APIs for interacting with crypto device
  - rte_ethdev for transport (both host & device)
  - DPDK infra for generic services (RCU QSBR, buffer management etc)

Liquid Crypto Development Strategy

- Fully opensource code via GitHub
- Platform-independent code: Develop once, deploy everywhere.
- Rich documentation
  - Programing Guide
  - API Documentation
  - Platform Guide
  - Performance Application



Liquid Crypto v/s VirtIO Crypto

VirtIO Crypto	*Emulated using DMA device* *Implements standard VirtIO Crypto specification* *Supported with standard kernel & DPDK VirtIO drivers* *Requires higher number of DMA channels and ARM core cycles to initiate DMAs*
Liquid Crypto	*Optimized PCIe transport* *Custom API to expose all capabilities of HW & firmware* *Allows usage of advanced capabilities in hardware* *VirtIO Crypto for standard kernel & DPDK drivers*


Q & A


DPDK
SUMMIT
Powering the Future of Networking Software