Implementing ChaCha Based Crypto Primitives on Programmable SmartNICs

Shaguftha Zuveria Kottur

Krishna Kadiyala, Praveen Tammana, and Rinku Shah

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

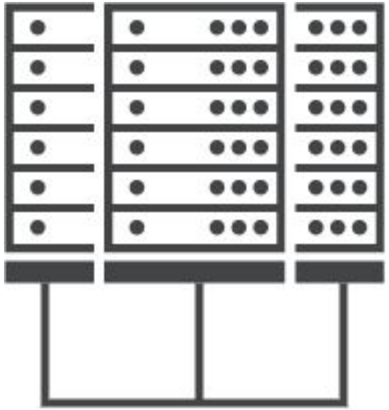
TCU

भारतीय प्रौद्योगिकी संस्थान
हैदराबाद
Indian Institute of Technology
Hyderabad

ACM SIGCOMM Workshop on
Formal Foundations and Security of Programmable Network Infrastructures (FFSPIN)

August 22, 2022

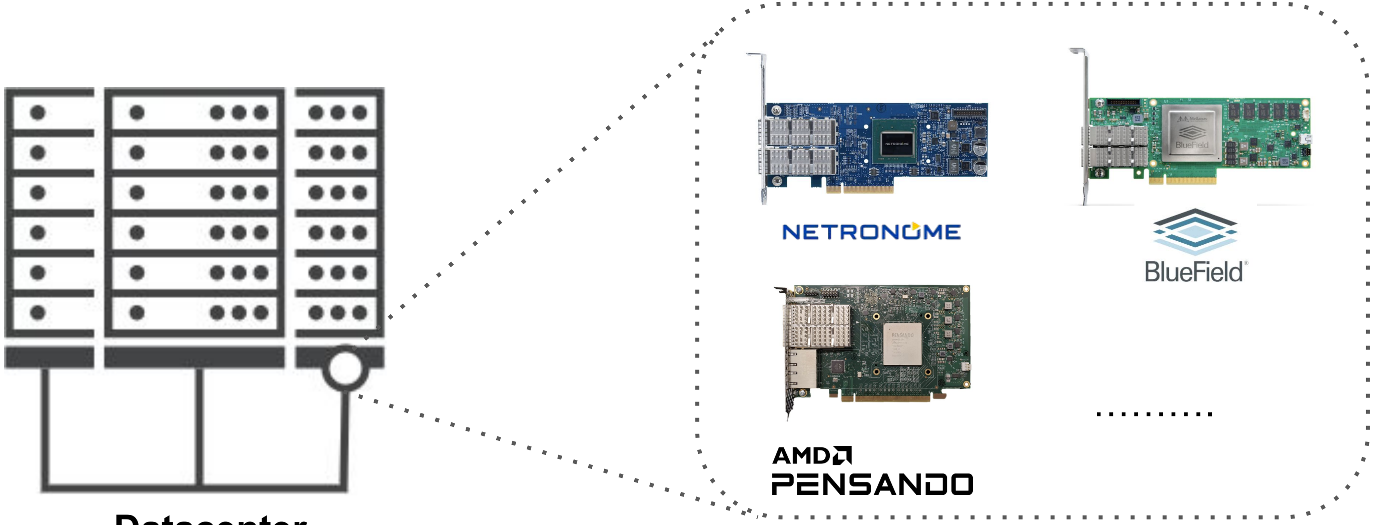
Datacenter control applications offloaded to PDPs



Datacenter

PDP - Programmable Data Plane

Datacenter control applications offloaded to PDPs



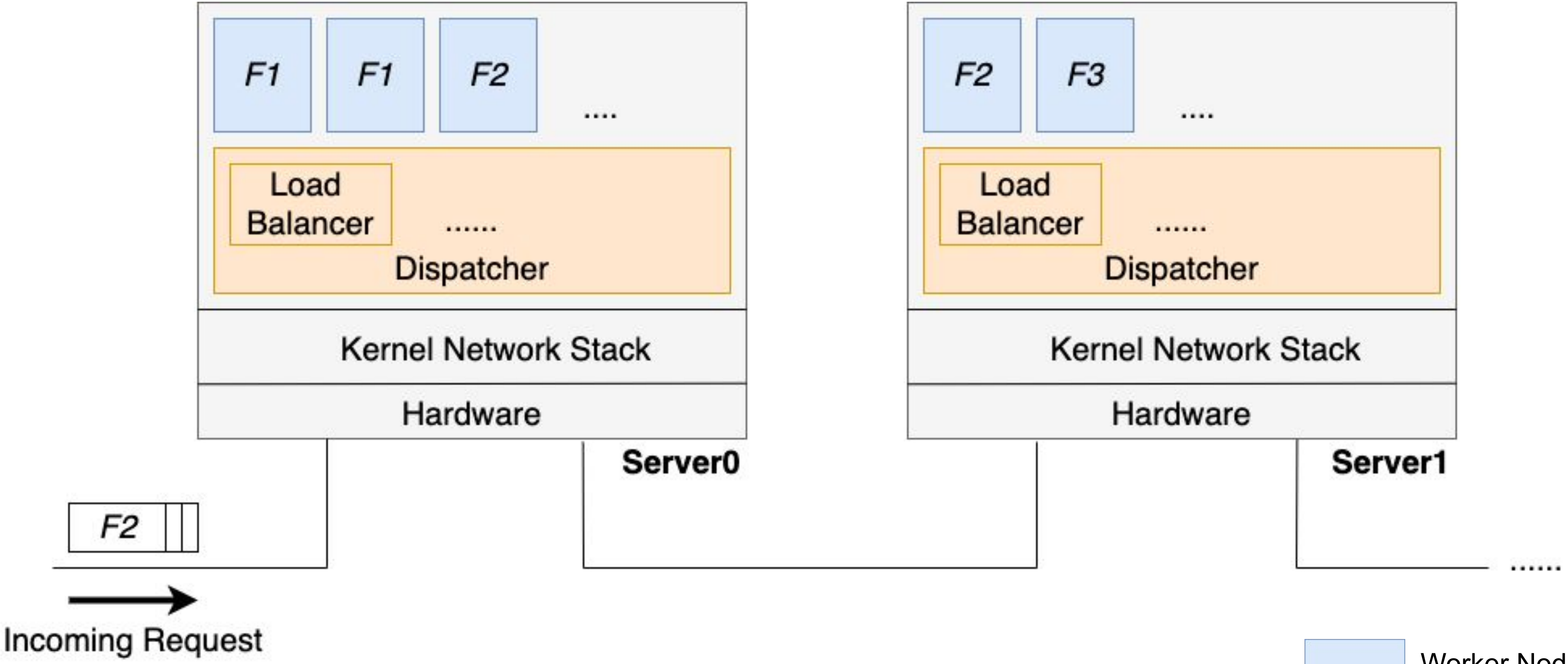
Datacenter

PDP devices - smartNICs

Applications offloaded to PDPs
- reduce latency
- increase server CPU savings

PDP - Programmable Data Plane

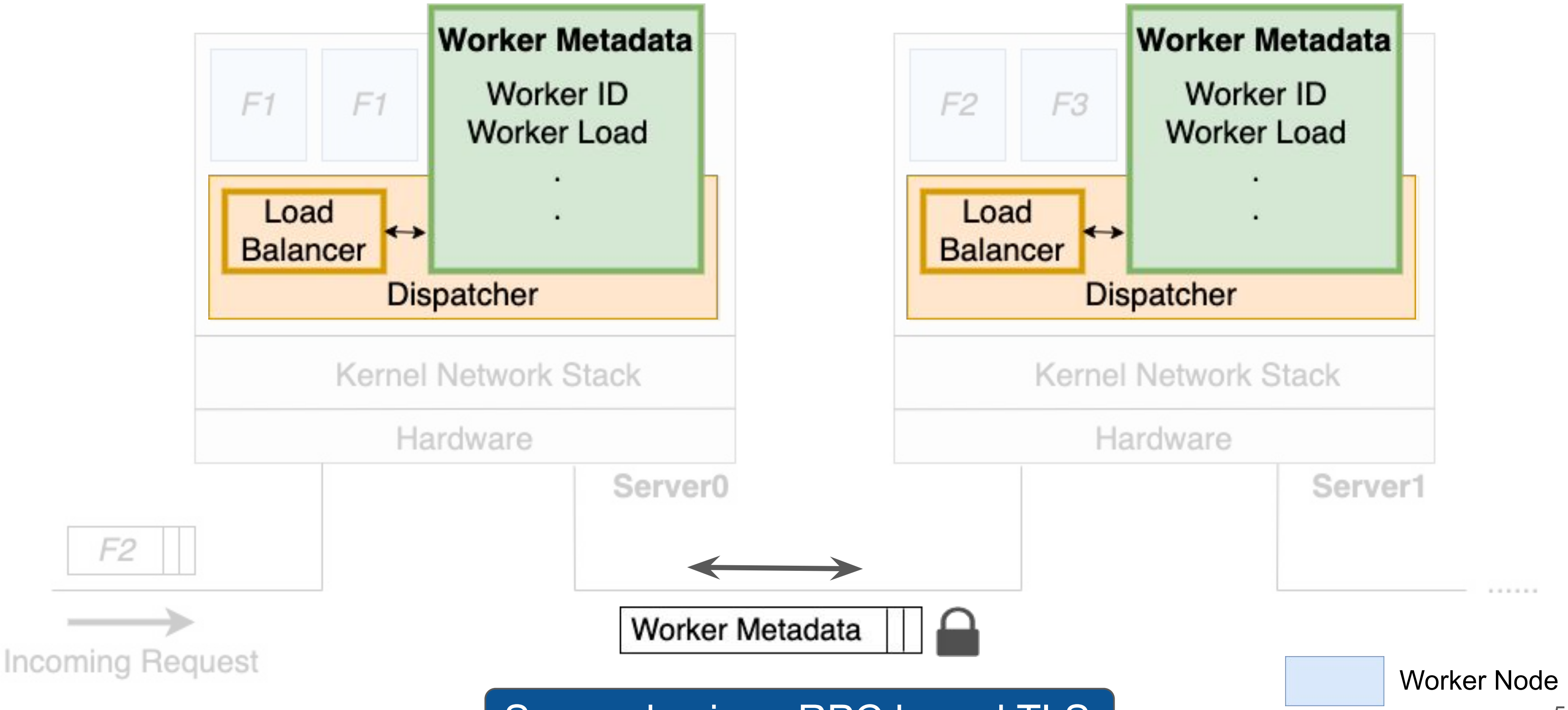
Example: Dispatcher in Serverless computing[1]



🟦 Worker Node

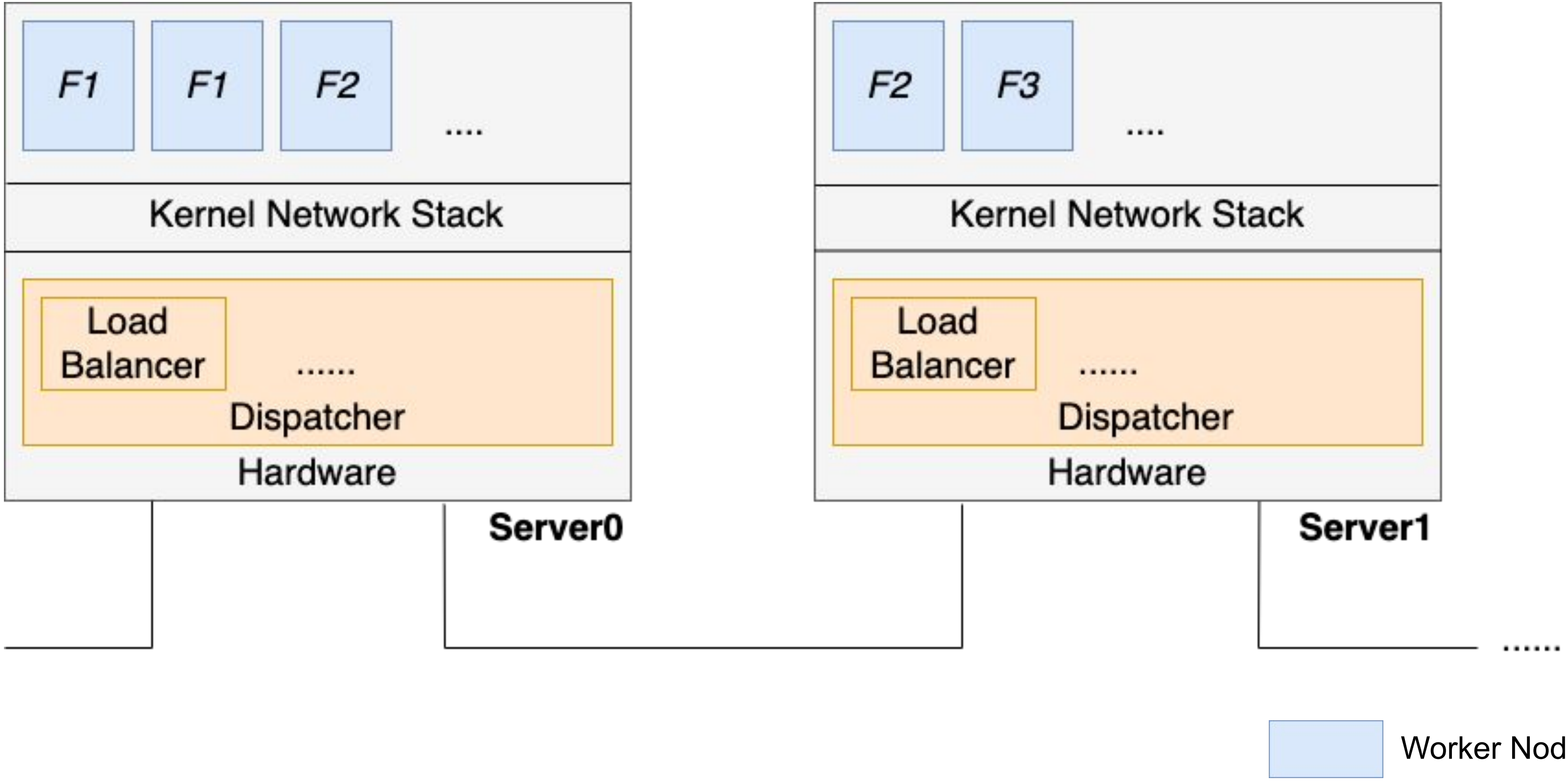
[1] Nilanjan Daw et.al Speedo: Fast Dispatch and Orchestration of Serverless Workflows. In Proceedings of SoCC ’21.

Dispatcher in Serverless computing

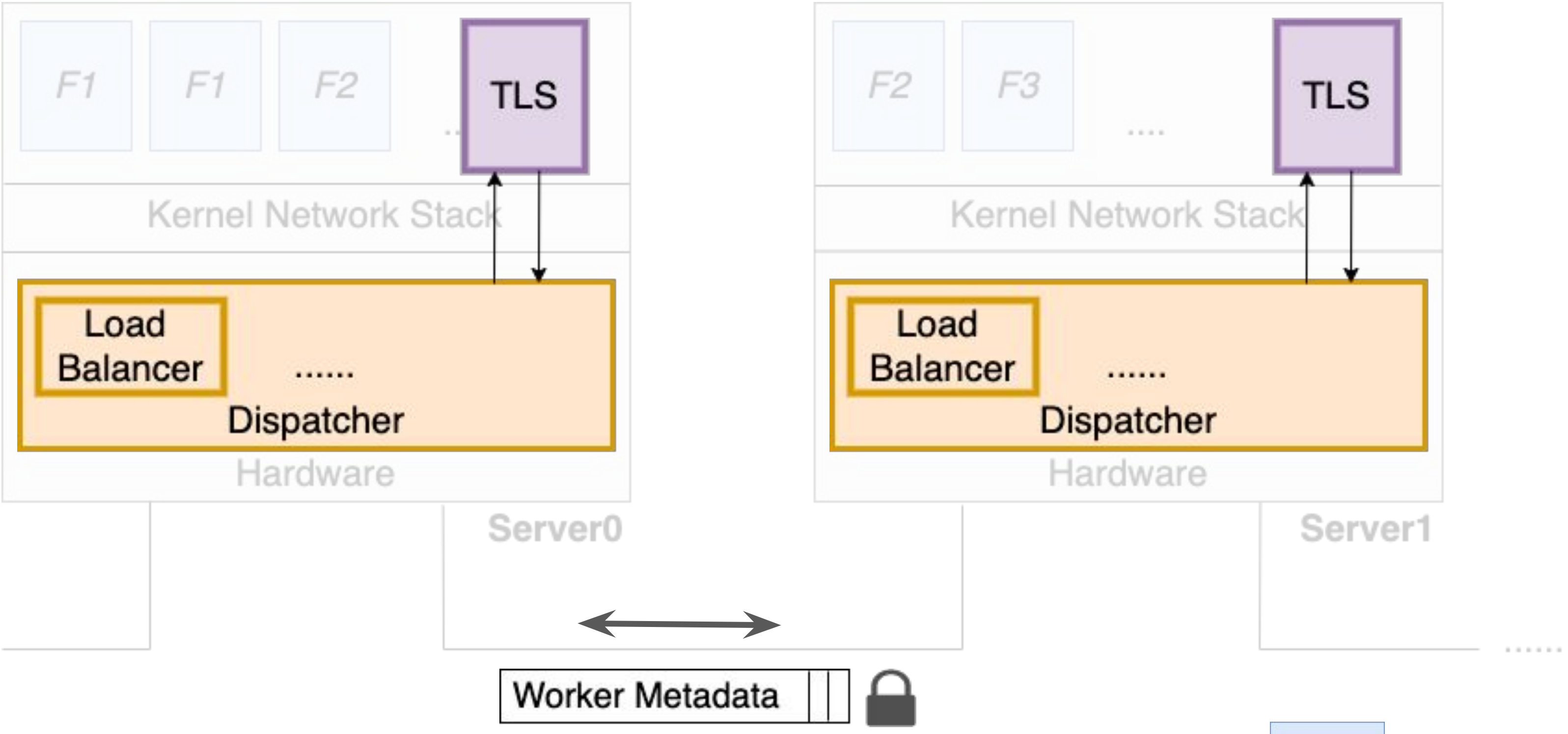


Secured using gRPC based TLS

Dispatcher in Serverless computing - **Offloaded**



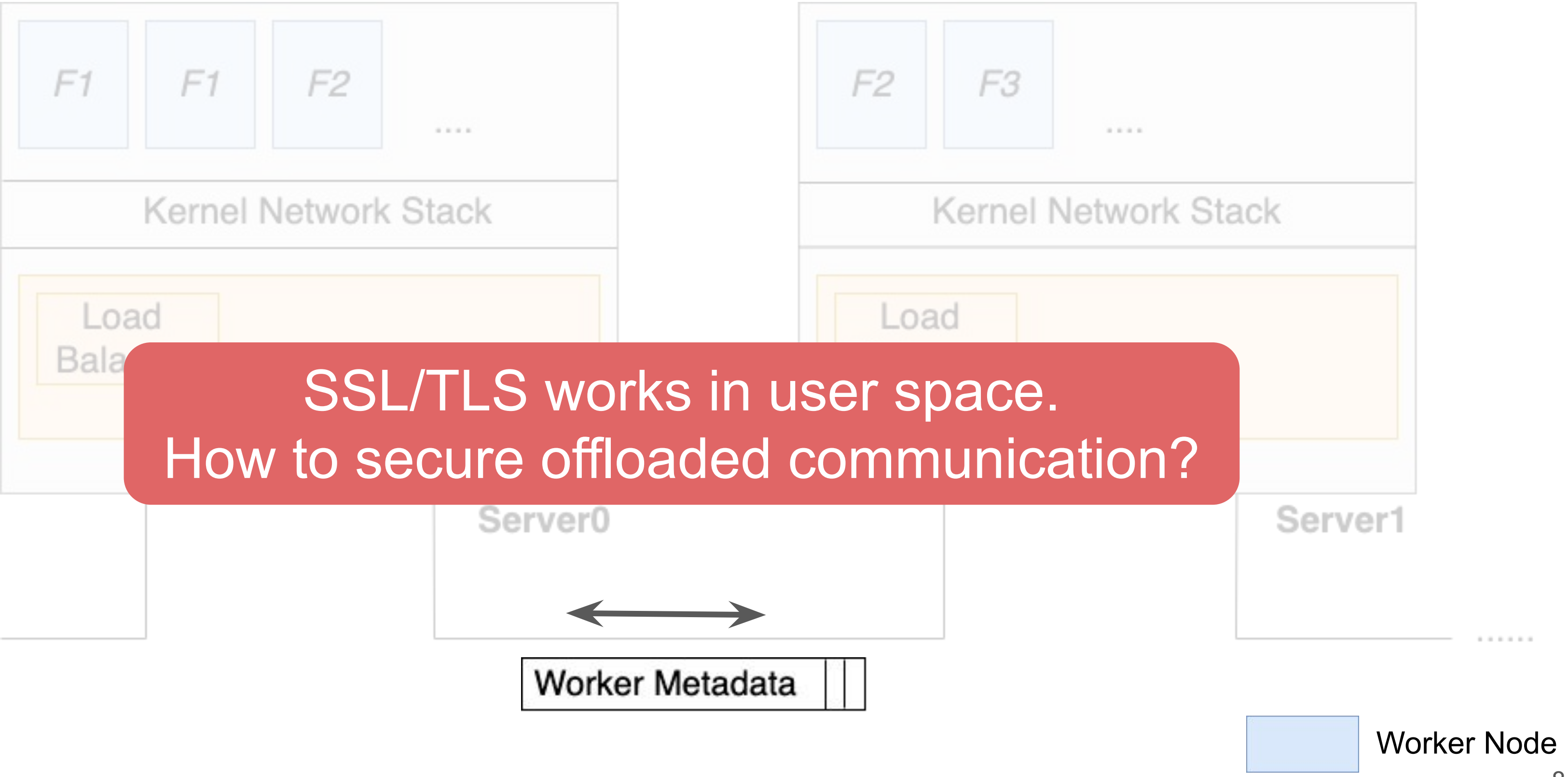
Dispatcher in Serverless computing - **Offloaded**



Negates offload benefits

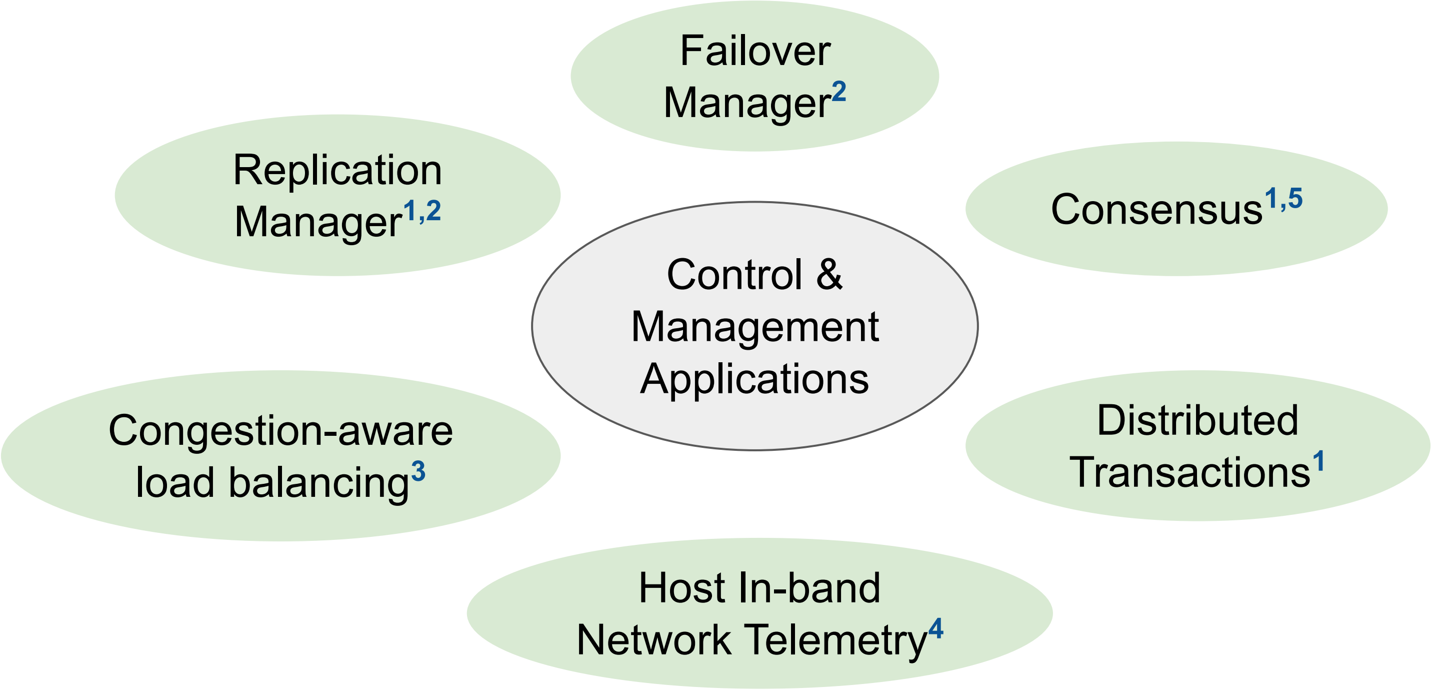
🟦 Worker Node

Dispatcher in Serverless computing - **Offloaded**



SSL/TLS works in user space.
How to secure offloaded communication?

Other offloaded applications



Failover Manager[2]

Replication Manager[1,2]

Consensus[1,5]

Control & Management Applications

Congestion-aware load balancing[3]

Distributed Transactions[1]

Host In-band Network Telemetry[4]

[1] Ming Liu et.al Offloading Distributed Applications onto SmartNICs Using IPipe. In Proceedings of the ACM SIGCOMM 2019..
[2] Ming Liu et.al E3: Energy-Efficient Microservices on SmartNIC Accelerated Servers. In USENIX ATC 2019.
[3] Naga Katta et.al Clove: Congestion-Aware Load Balancing at the Virtual Edge. In Proceedings of CoNEXT 2017.
[4] Tomasz Osiński et.al Achieving End-to-End Network Visibility with Host-INT. In Proceedings of ANCS 2021.
[5] Huynh Tu Dang et.al Partitioned Paxos via the Network Data Plane. arXiv:1901.08806 http://arxiv.org/abs/1901.08806

Existing in-network crypto processing solutions

Accelerators

- AES accelerators
  - Nvidia Bluefield, Pensando DSC[1]
- TLS handshake offload to Nvidia Bluefield[2]

Offloads

- AES/GCM offload to Mellanox ASIC NICs[3]
- AES offload to Intel Tofino
  - using Scrambled Lookup Tables[4]

Focus is primarily on AES!

[1] S. VenkataKeerthy et. al. Packet Processing Algorithm Identification using Program Embeddings. In APNet 2022.
[2] Duckwoo Kim et. al. A Case for SmartNIC-accelerated Private Communication. In APNet 2020
[3] Boris Pismenny et. al. Autonomous NIC Offloads. In Proceedings of ASPLOS 2021.
[4] Xiaoqi Chen. Implementing AES Encryption on Programmable Switches via Scrambled Lookup Tables. In ACM SIGCOMM SPIN 2020.

Are there other cipher suites?

TLS 1.3 supports TWO ciphersuites

- AES - GCM
- ChaCha20 - Poly1305

ChaCha stream cipher

- Processor friendly Add-Rotate-XOR operations
- Resistant to side channel cache timing attacks[1]

[1] Zakaria Najm et. al. On Comparing Side-channel Properties of AES and ChaCha20 on Microcontrollers. In IEEE APCCAS 2018.

Key idea

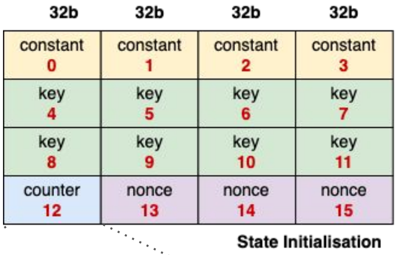
Offload ChaCha based crypto primitives to smartNIC without using accelerators/co-processors

Key contributions

1. Identification of applications that benefit from offloaded crypto primitives
2. Implementation of ChaCha based crypto primitives on **Netronome smartNIC**
3. Performance evaluation of proposed implementation

ChaCha Overview

ChaCha Stream Cipher: State Initialization

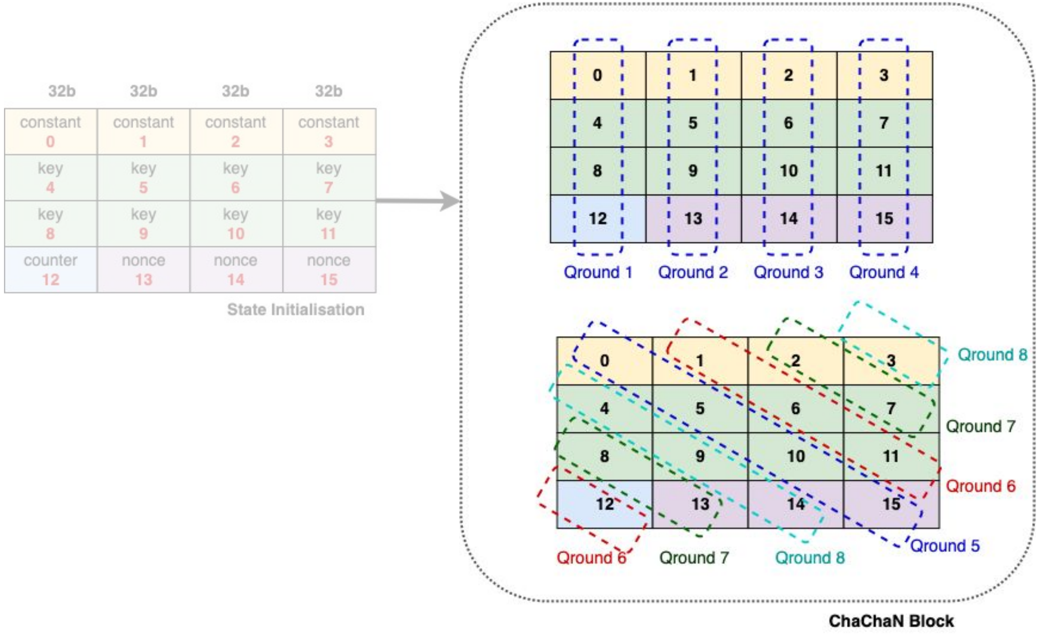


32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

State Initialisation

Increment for each 512 bit of pkt

ChaCha Stream Cipher: ChaChaN block



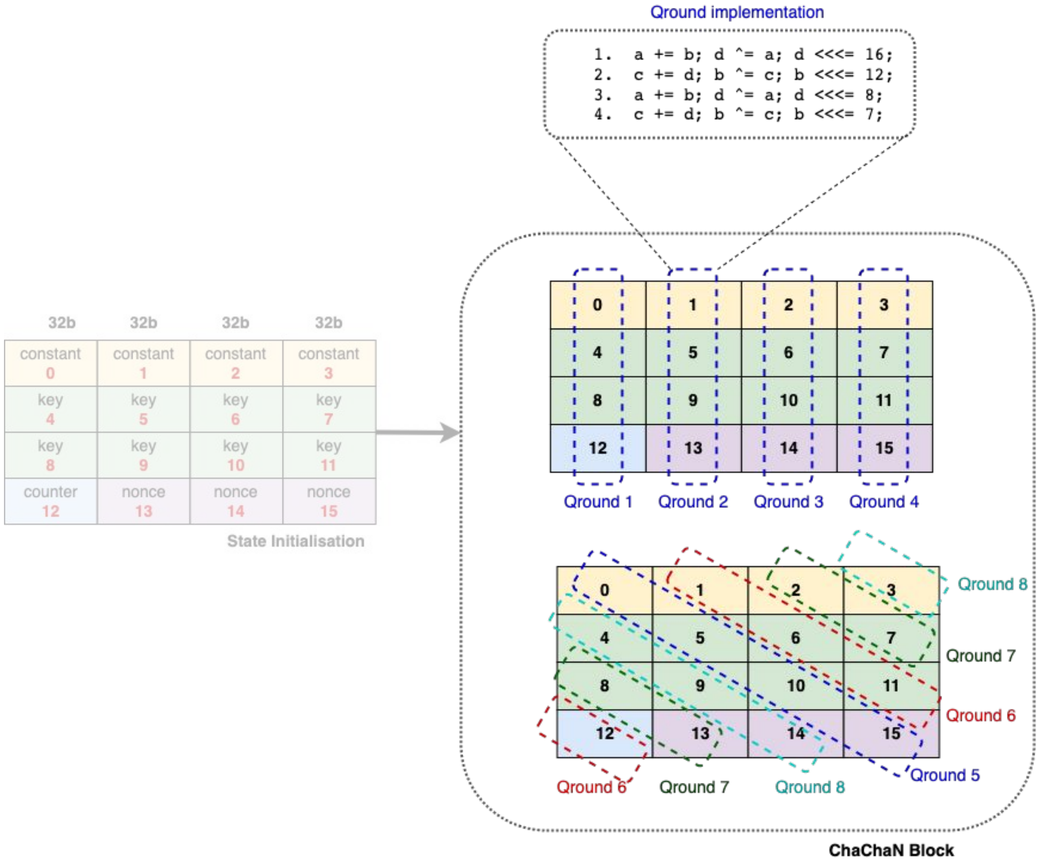
32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

State Initialisation

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

ChaChaN Block

ChaCha Stream Cipher: ChaChaN block



32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

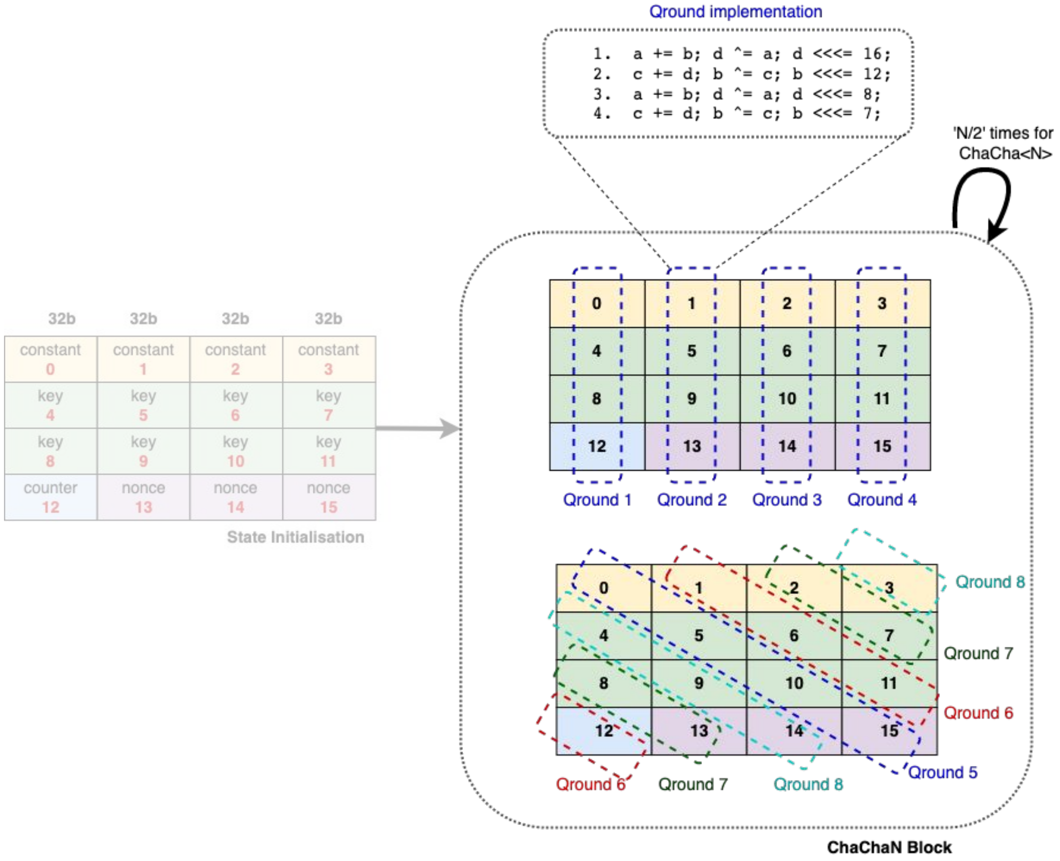
State Initialisation

Qround implementation

```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

ChaChaN Block

ChaCha Stream Cipher: ChaChaN block



32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

State Initialisation

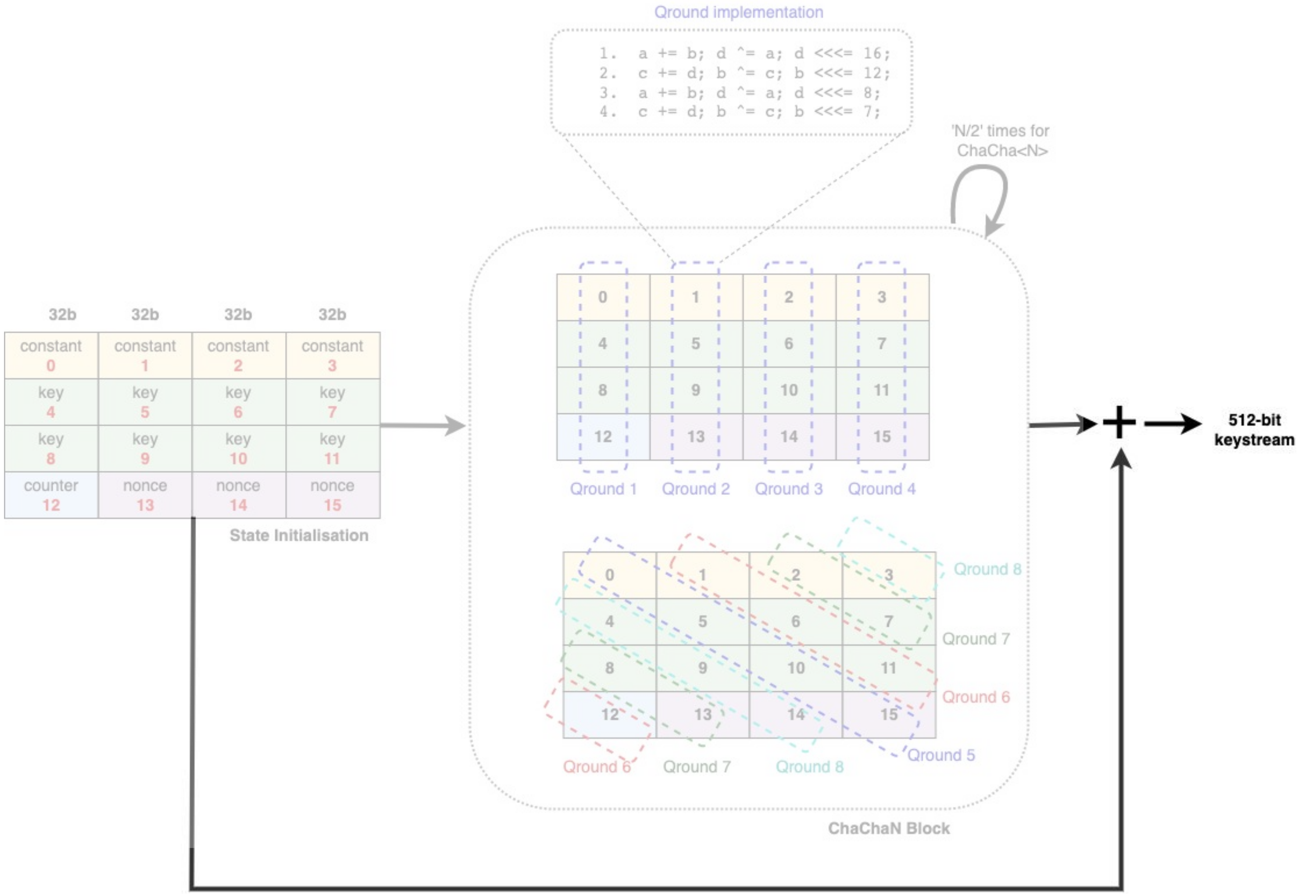
Qround implementation

```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

'N/2' times for ChaCha<N>

ChaChaN Block

ChaCha Stream Cipher: Keystream



32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

State Initialisation

Qround implementation

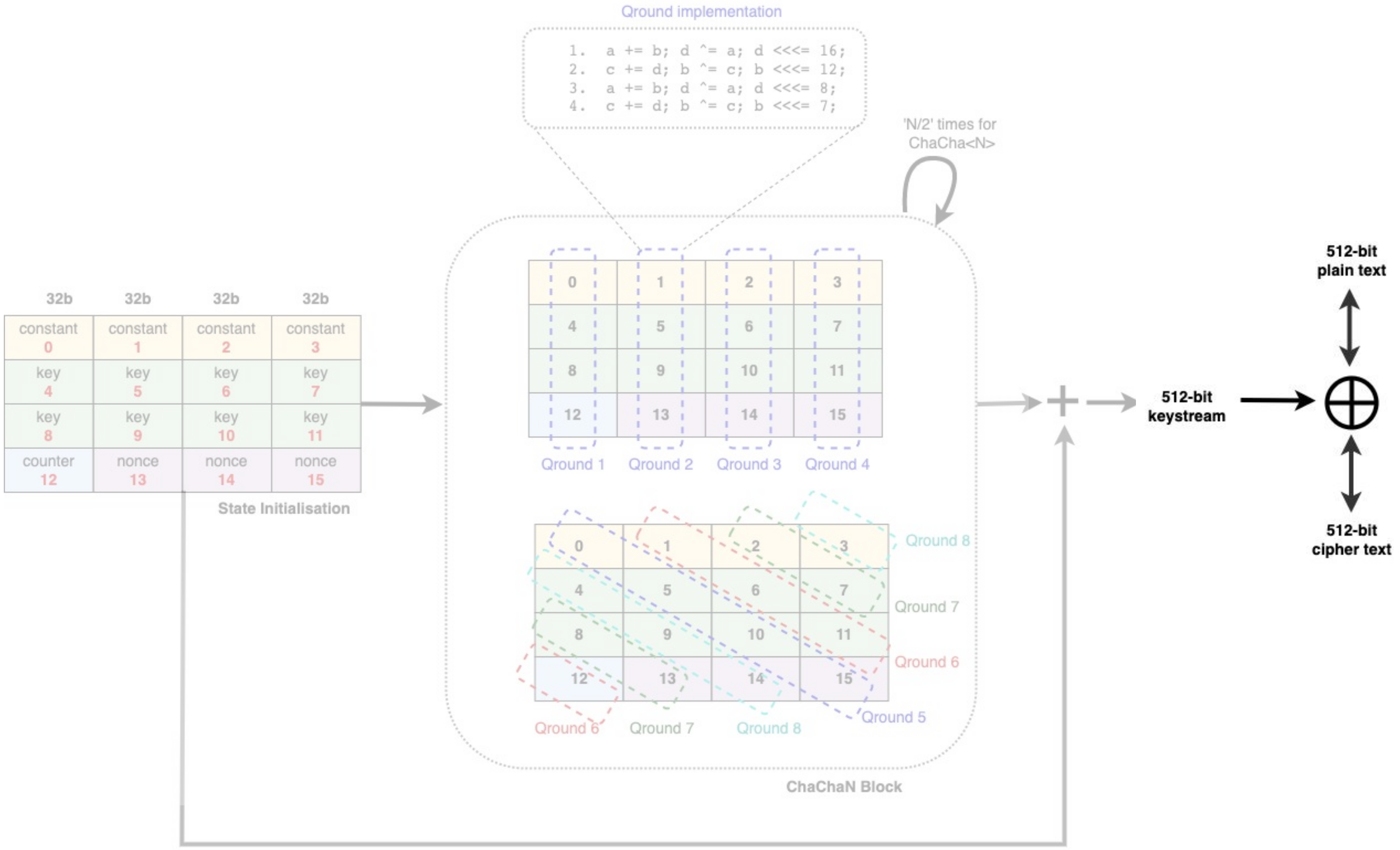
```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

'N/2' times for ChaCha<N>

ChaChaN Block

512-bit keystream

ChaCha Stream Cipher: Encryption/Decryption



Qround implementation

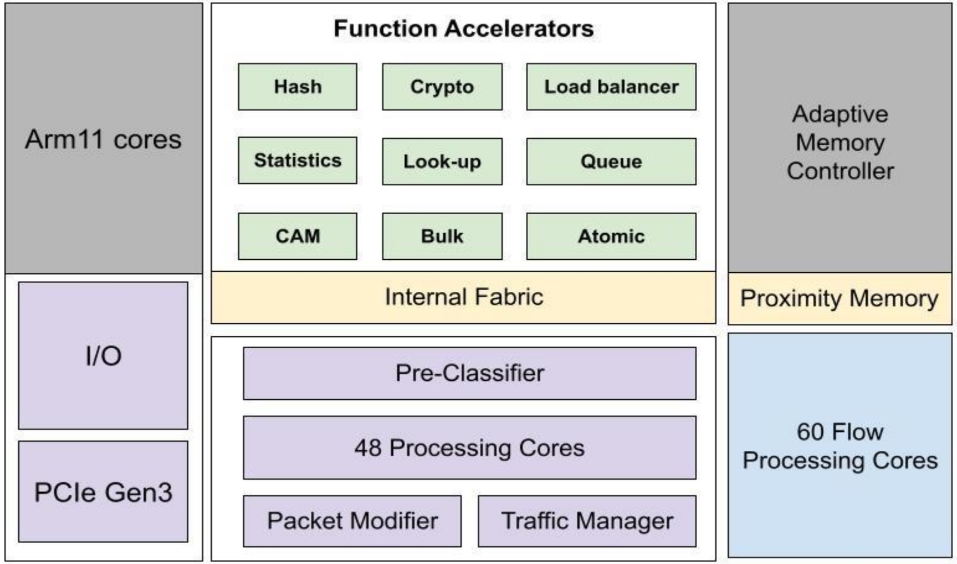
```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

State Initialisation

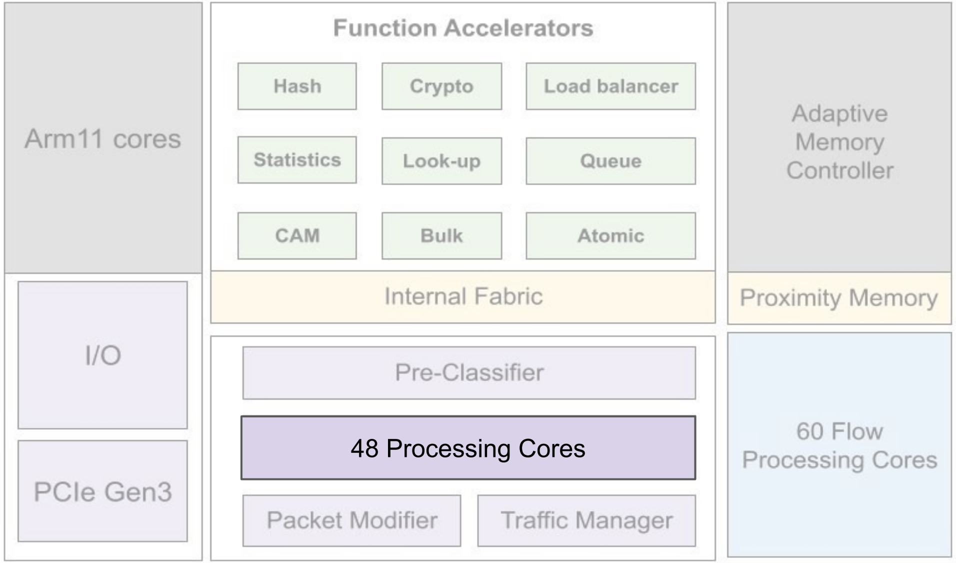
Design Challenges

Challenge 1: Initial Nonce



Netronome NFP-4000 Flow Processor Block Diagram

Challenge 1: Initial Nonce



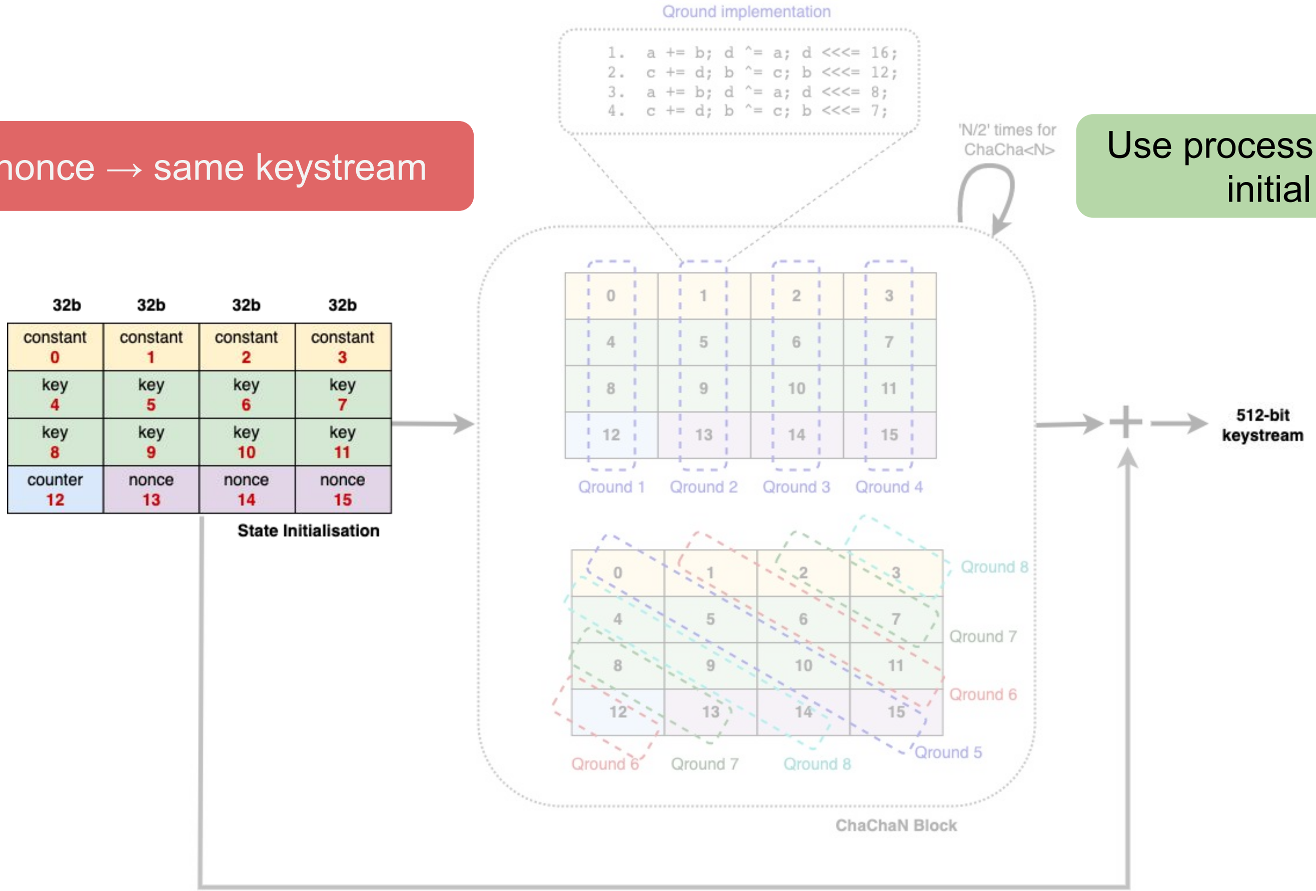
Each core can process packets in parallel

Netronome NFP-4000 Flow Processor Block Diagram

Solution 1: Use core ID as Initial Nonce

Same initial nonce → same keystream

Use processing core ID as initial nonce



32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

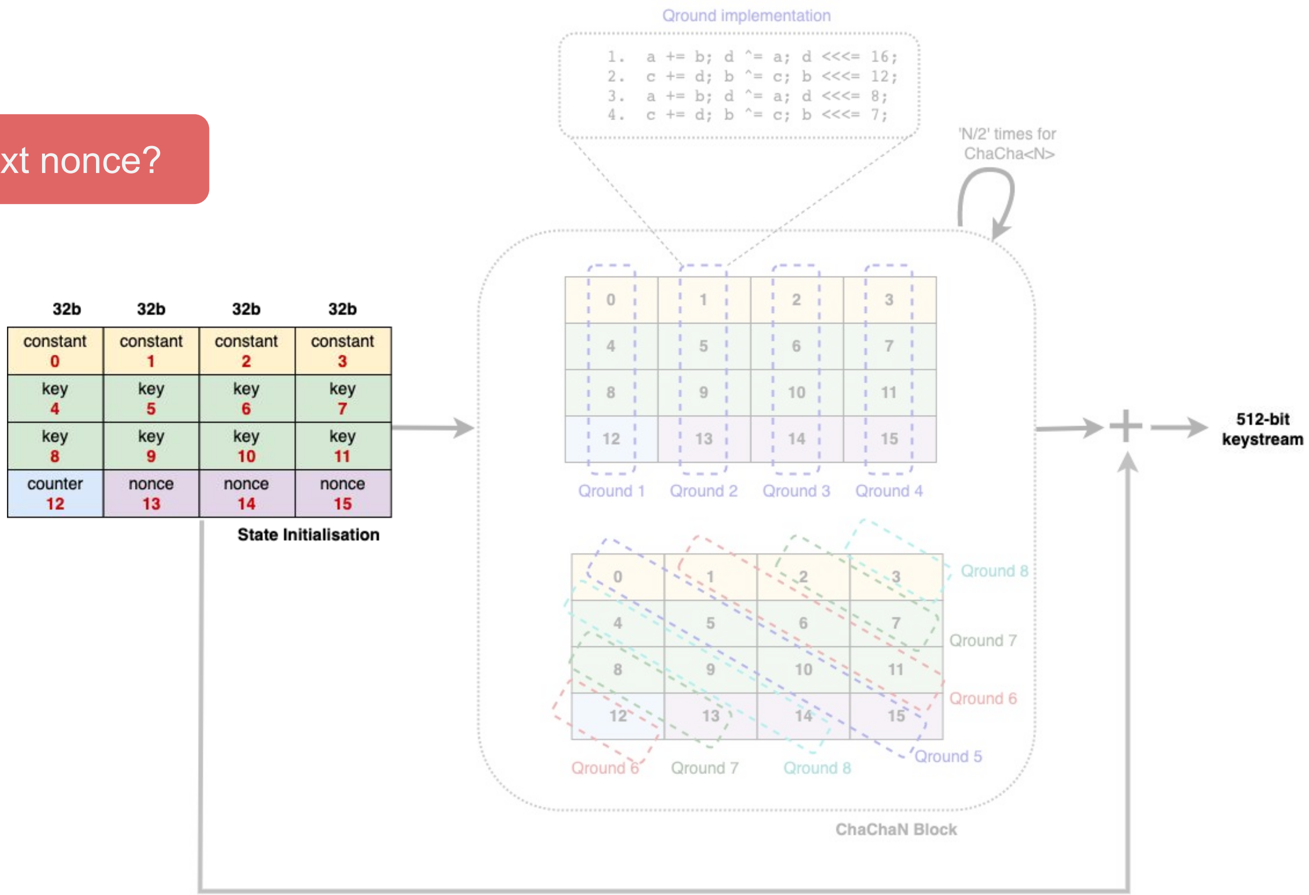
State Initialisation

Qround implementation

```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

Challenge 2: Nonce Generation

Next nonce?



Qround implementation

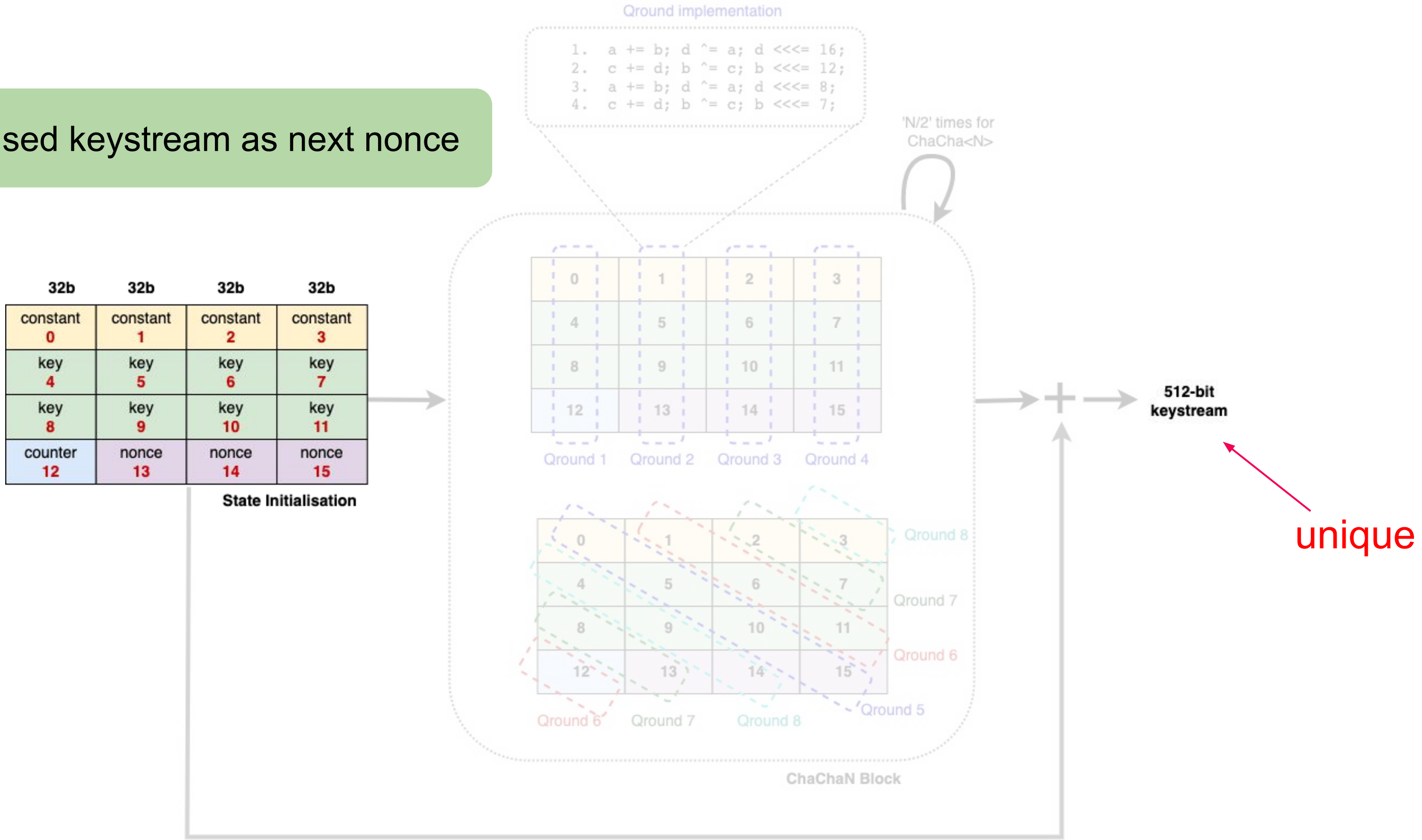
```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

State Initialisation

Solution 2: Use previous keystream

Use 96-bit of unused keystream as next nonce



32b	32b	32b	32b
constant 0	constant 1	constant 2	constant 3
key 4	key 5	key 6	key 7
key 8	key 9	key 10	key 11
counter 12	nonce 13	nonce 14	nonce 15

State Initialisation

Qround implementation

```
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

unique

Implementation

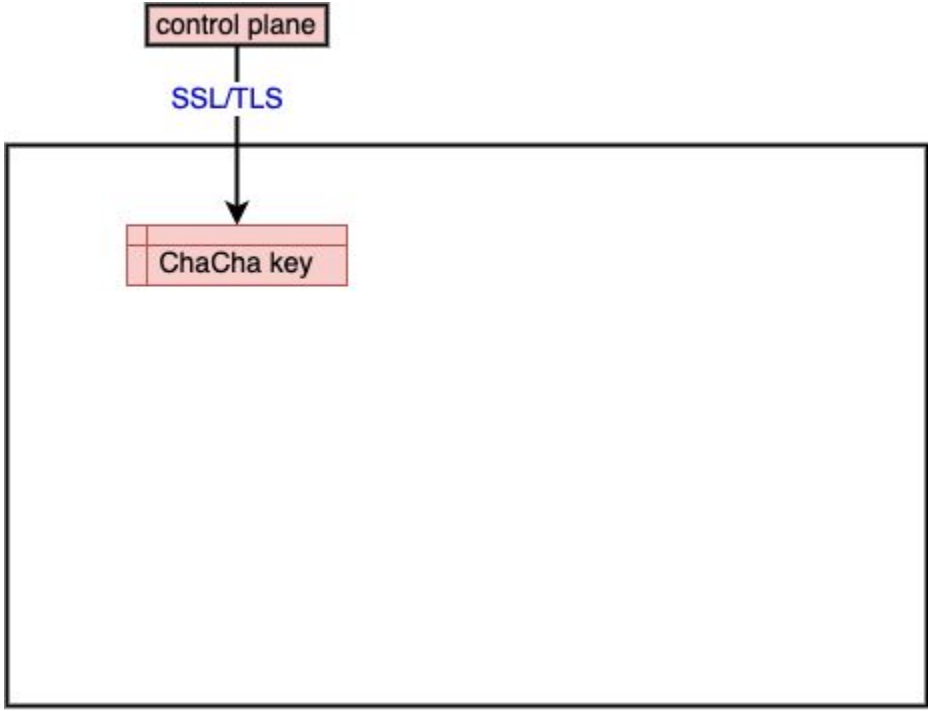
Implementation

- Implemented on Netronome Agilio smartNIC
- Crypto primitives offered:
    - ENC - Encryption
    - DEC - Decryption

      ChaCha10
    - AUTH_set
    - AUTH_test

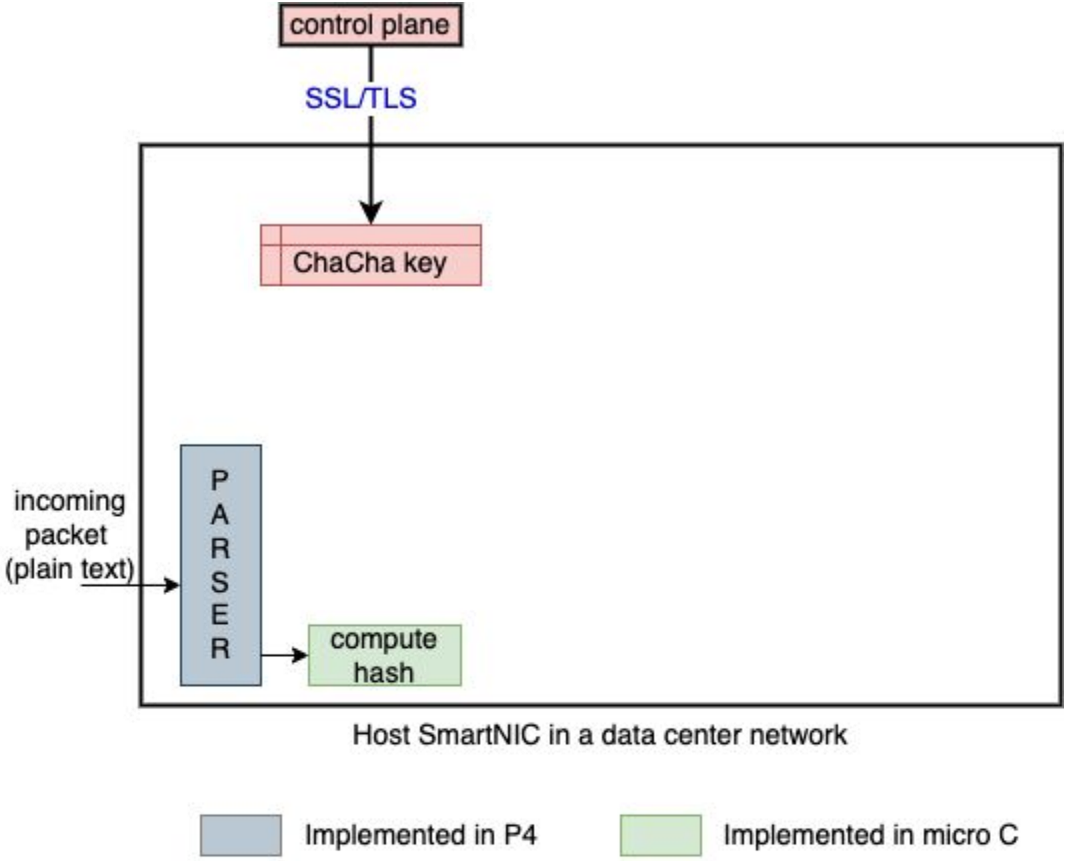
      custom crc32 + ChaCha10
    - Compound primitives
        - ENC+AUTH_set
        - DEC+AUTH_test

Implementation

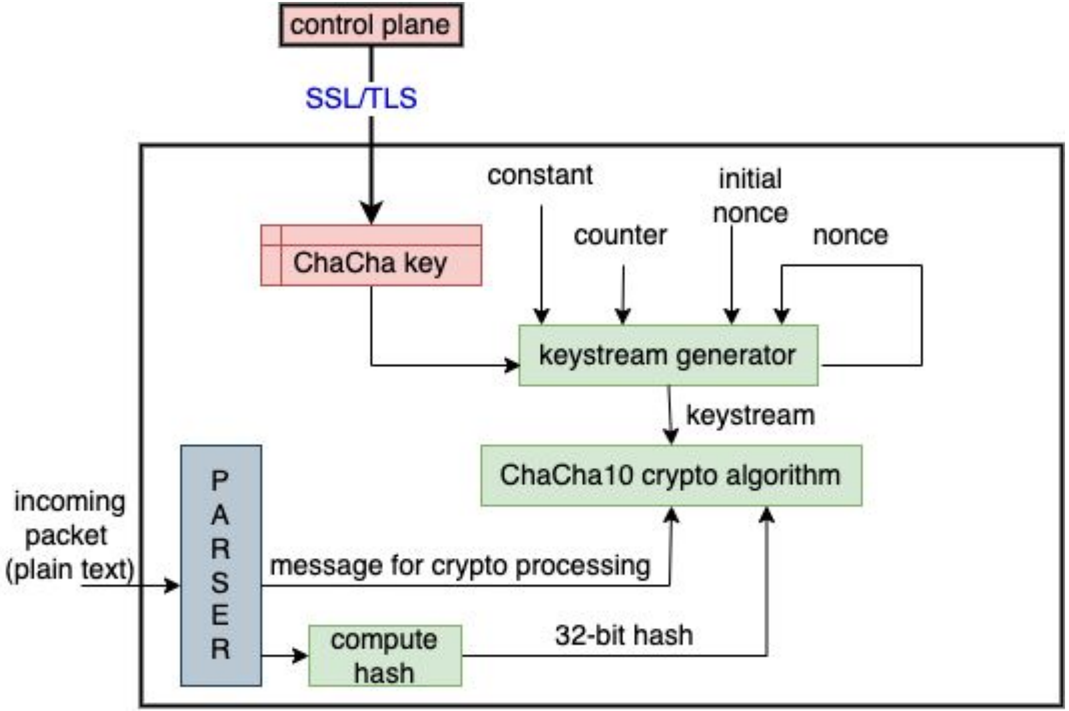


Host SmartNIC in a data center network

Implementation: ENC + AUTH_set



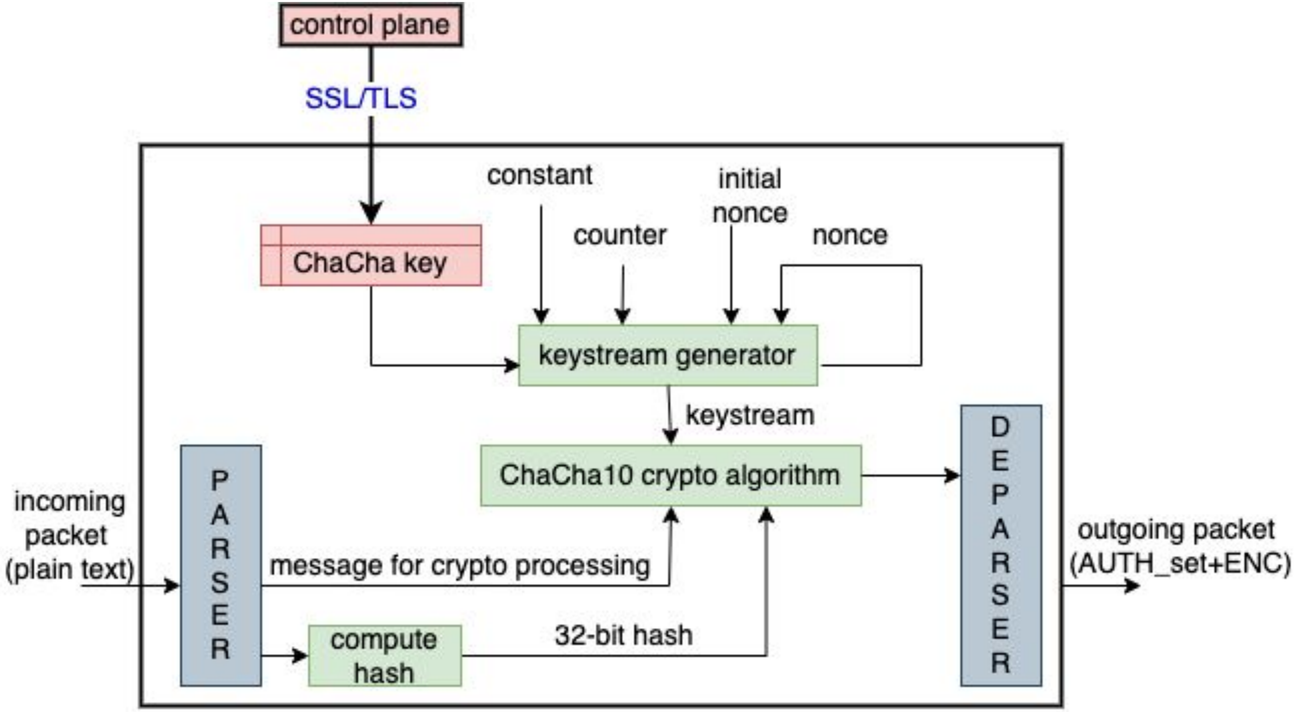
Implementation: ENC + AUTH_set



Host SmartNIC in a data center network

🟦 Implemented in P4 🟩 Implemented in micro C

Implementation: ENC + AUTH_set

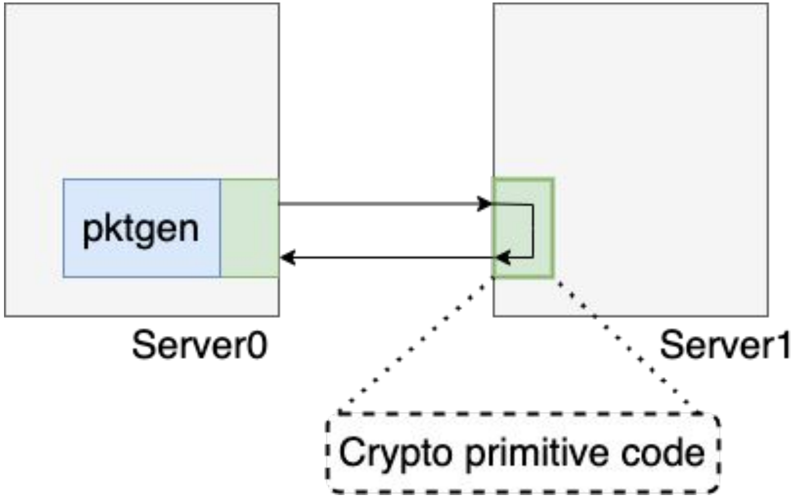


Host SmartNIC in a data center network

🟦 Implemented in P4 🟩 Implemented in micro C

Evaluation

Evaluation: Setup

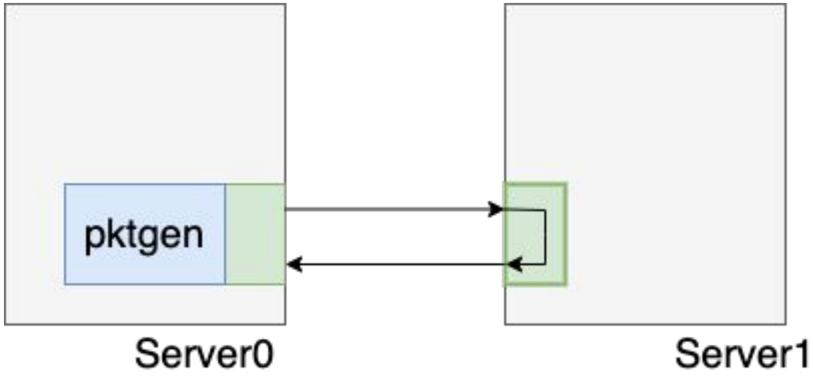


SmartNIC offload Setup

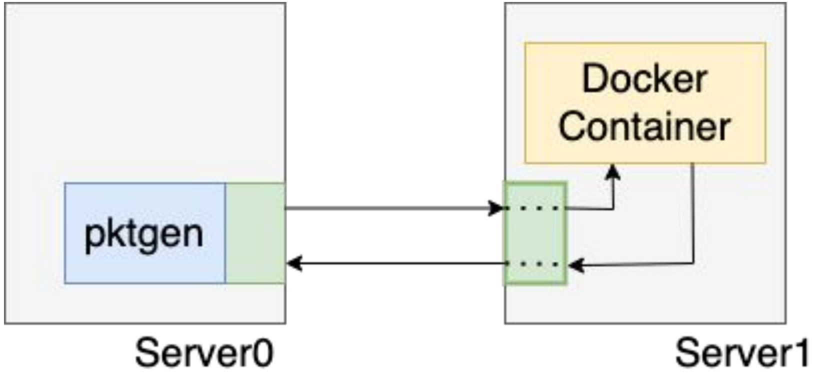
⬜ AMD Ryzen 9 5950X (3.4 GHz, 16 cores, 32 threads) processor and 32GB RAM

🟩 Netronome Agilio CX 40 Gbit/s dual-port SmartNIC

Baseline setup



Baseline:NIC - L2 forwarding



Baseline:Host - ENC on host

⬜ AMD Ryzen 9 5950X (3.4 GHz, 16 cores, 32 threads) processor and 32GB RAM

🟩 Netronome Agilio CX 40 Gbit/s dual-port SmartNIC

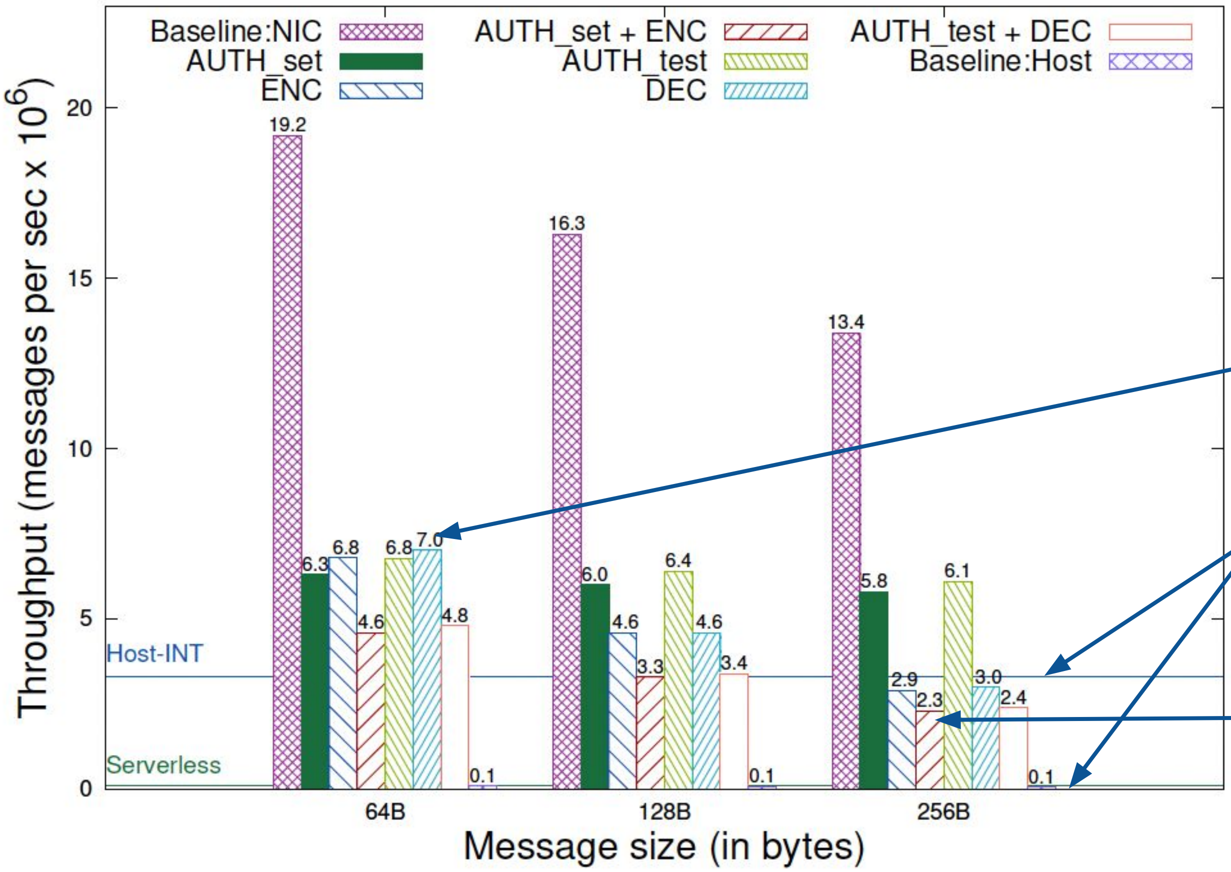
Questions to answer

1. How does our implementation perform compared to the baselines?

2. Which applications will benefit by leveraging these crypto primitives?

3. How much memory is available to offload other applications?

Throughput: ChaCha based crypto primitive vs. Baseline



Message Size	Improvement over Baseline:Host	
	Min	Max
64 B	46x	70x
128B	33x	64x
256B	23x	61x

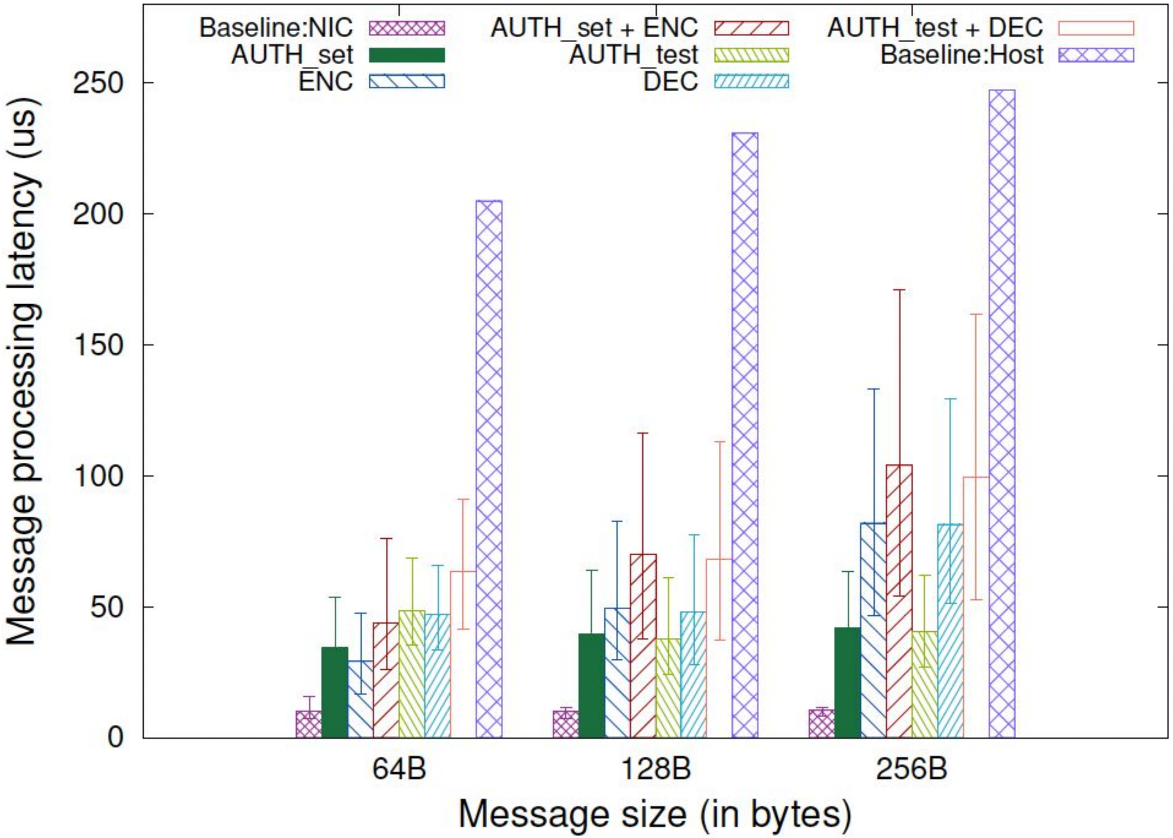
70x of Baseline:Host

Meets application requirements

- Serverless: 20K to 100K pps
- Host-INT: 3.3M pps (18B pkt)

23x of Baseline:Host

Latency: ChaCha based crypto primitive vs. Baseline



Message Size	Latency reduction over Baseline:Host Min	Max
64 B	69%	84%
128B	70%	83%
256B	59%	83%

Summary

- Implemented in-network ChaCha crypto without using co-processor
- Solution meets crypto processing requirements of control applications

Future Work

- Implementing Poly-1305 authentication algorithm
- Crypto processing for MTU-sized messages
- Crypto primitive APIs for P4/C programmers

Contact: *shagufttha21079@iiitd.ac.in*