In-Network Compression Challenges and Opportunities

Daniel E. Lucani Rötter

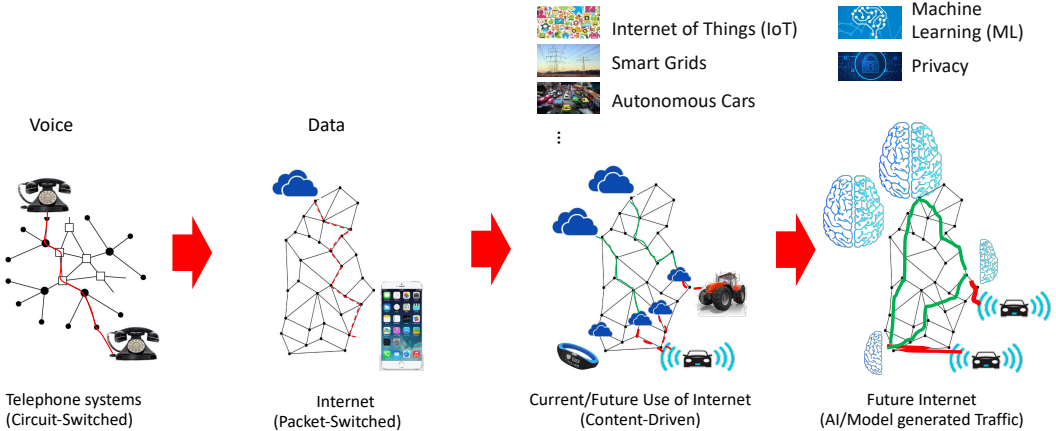
Aarhus Universitet, Denmark
daniel.lucani@ece.au.dk

Some results presented in CoNEXT 2020

Setting the Stage

Unexploited compression potential from considerable correlation in network traffic data

- Temporal (same source): well understood
- Spatial (across sources): more difficult to address



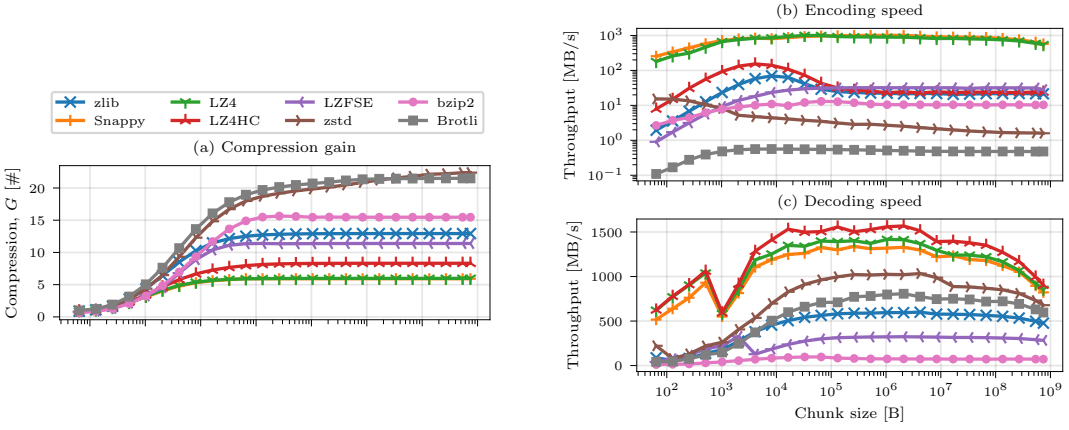
Setting the Stage

- State-of-the-art compression not suitable to In-Network Processing
  - Sequential process $\rightarrow$ not fast enough
    - Example 1: Good compressors like Brotli, Bzip2 $\rightarrow$ 5 - 100 Mbps
    - Example 2: Reasonable compressors like Snappy $\rightarrow$ 5 - 10 Gbps
  - Most efficient for large blocks of data (larger than network packets)
  - Large use of memory
- Contrast to INP
  - Network switches $\rightarrow$ 100 Gbps
  - Ideally, operating on packet by packet basis (latency, speed, memory)
  - Limited memory

Setting the Stage

- State-of-the-art compression not suitable to In-Network Processing
  - Sequential process $\rightarrow$ not fast enough
    - Example 1: Good compressors like Brotli, Bzip2 $\rightarrow$ 5 - 100 Mbps
    - Example 2: Reasonable compressors like Snappy $\rightarrow$ 5 - 10 Gbps
  - Most efficient for large blocks of data (larger than network packets)
  - Large use of memory
- Contrast to INP
  - Network switches $\rightarrow$ 100 Gbps
  - Ideally, operating on packet by packet basis (latency, speed, memory)
  - Limited memory

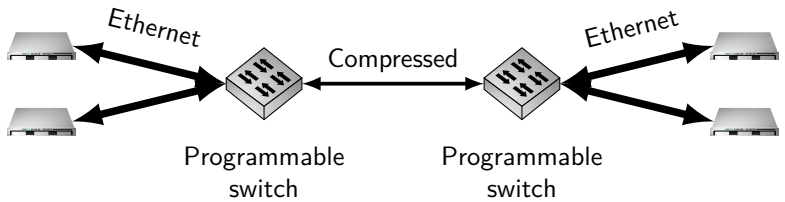
Setting the Stage



R. Vestergaard, Q. Zhang, and D. Lucani. Enabling Random Access in Universal Compressors. In *IEEE INFOCOM WKSHPS*, 2021.

Research problem

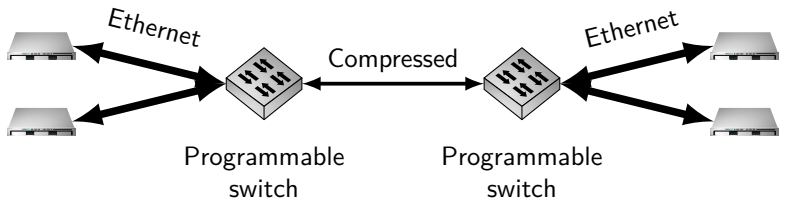
- Compression of arbitrary packets on programmable switches
  - Guaranteed line-speed processing
  - Specialized programming language (P4)
  - Resource-constrained



Our solution: Use Generalized Deduplication (GD) algorithm on Intel Tofino platform

Research problem

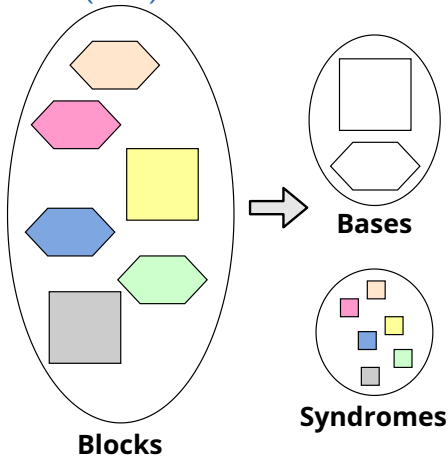
- Compression of arbitrary packets on programmable switches
  - Guaranteed line-speed processing
  - Specialized programming language (P4)
  - Resource-constrained



Our solution: Use Generalized Deduplication (GD) algorithm on Intel Tofino platform

Generalized deduplication (GD)

- Transforms each block in a *basis* and a *syndrome*
  - → GD **en**coding $\Longrightarrow$ Hamming **de**coding
  - → GD **de**coding $\Longrightarrow$ Hamming **en**coding
- 1-bit variations of a piece of data share the same basis
- **Goal:** amplify benefits of memory (tables) used in switch



Idea first described in: R. Vestergaard, D. Lucani, and Q. Zhang. Generalized Deduplication: Lossless Compression for Large Amounts of Small IoT Data. In *European Wireless*, May 2019.

Implementing Generalized Deduplication in $P4_{16}$

- Hamming codes are equivalent to particular CRCs: Native support on Tofino
- GD requires several table look-ups in its algorithm
  - Tables are first-class citizens in P4
- Compression by replacing bases with shorter identifiers
  - Arbitrary identifiers
  - State synchronization between switches
- Do no harm if bases are not part of tables
- Details
  - Encoder and decoder work on actual switch
  - Bases up to 511 bits (limited by hardware resources)
  - Can run at line speed (100 Gbit/s)

Computing Hamming codes on Tofino

- Hamming codes are *linear* codes
- It is possible to compute them using CRCs

Bit error	Bit sequence	Syndrome
0	(0000001)	(001)
1	(0000010)	(010)
2	(0000100)	(100)
3	(0001000)	(011)
4	(0010000)	(110)
5	(0100000)	(111)
6	(1000000)	(101)

(a) Hamming code $(7, 4)$ syndromes

Poly.	Bit sequence	CRC-3
x^0	(0000001)	(001)
x^1	(0000010)	(010)
x^2	(0000100)	(100)
x^3	(0001000)	(011)
x^4	(0010000)	(110)
x^5	(0100000)	(111)
x^6	(1000000)	(101)

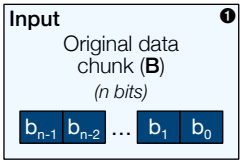
(b) CRC-3 of bit sequences with one non-zero bit

Particular Hamming codes

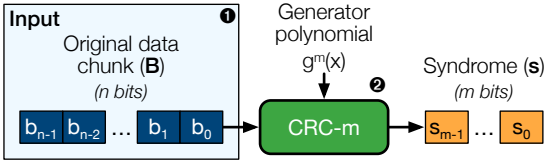
Code	Generator Polynomial	Corresponding CRC	Tofino
$(7, 4)$	$x^3 + x + 1$	CRC-3-GSM	$0x3$
$(15, 11)$	$x^4 + x + 1$	CRC-4-ITU	$0x3$
$(31, 26)$	$x^5 + x^2 + 1$	CRC-5-USB	$0x05$
$(63, 57)$	$x^6 + x + 1$	CRC-6-ITU	$0x03$
$(127, 120)$	$x^7 + x^3 + 1$	CRC-7	$0x09$
$(255, 247)$	$x^8 + x^4 + x^3 + x^2 + 1$	CRC-8-SAE J1850	$0x1D$

Table: Generator Polynomials

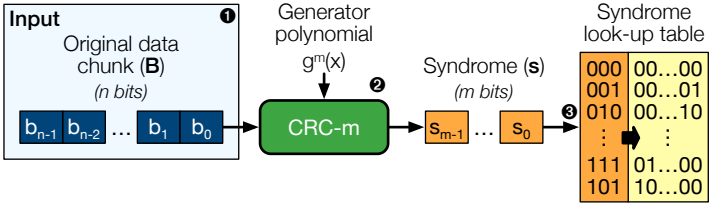
ZipLine: Encoding



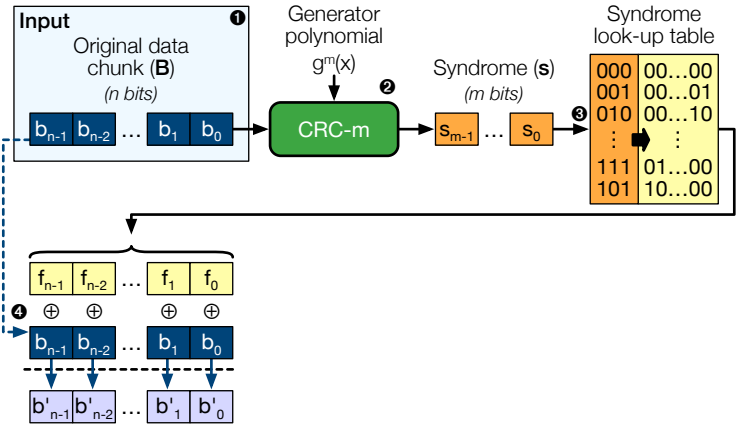
ZipLine: Encoding



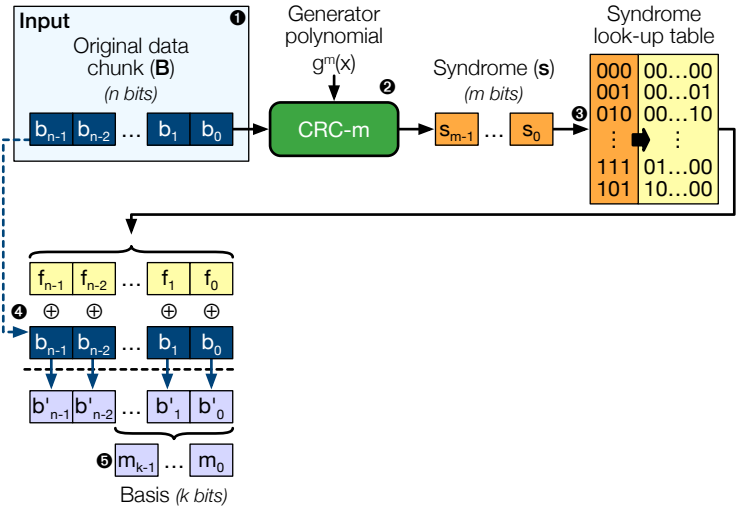
ZipLine: Encoding



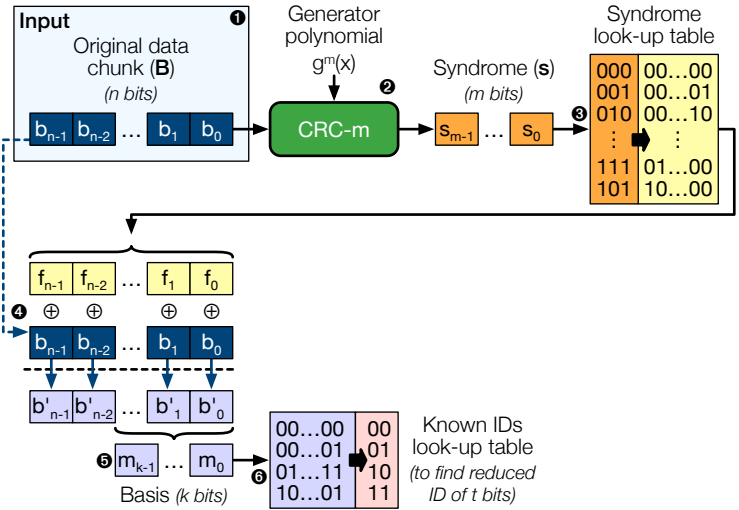
ZipLine: Encoding



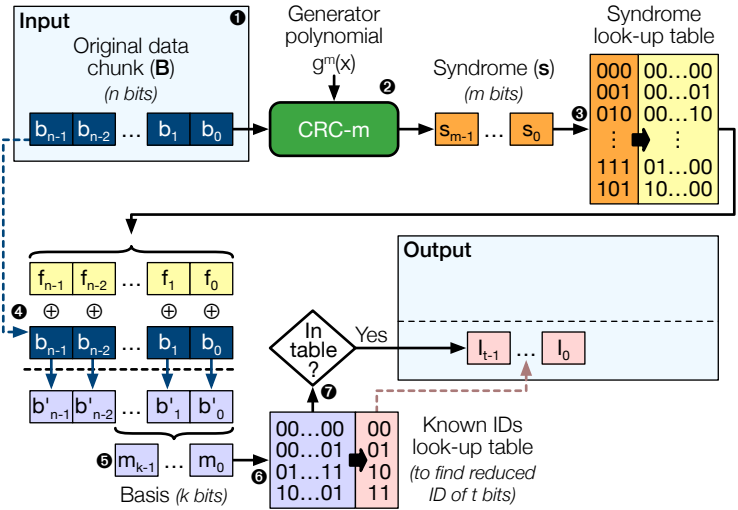
ZipLine: Encoding



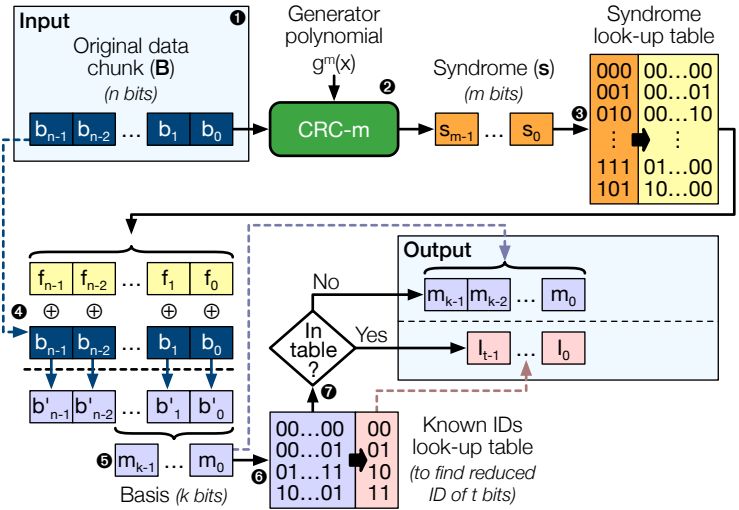
ZipLine: Encoding



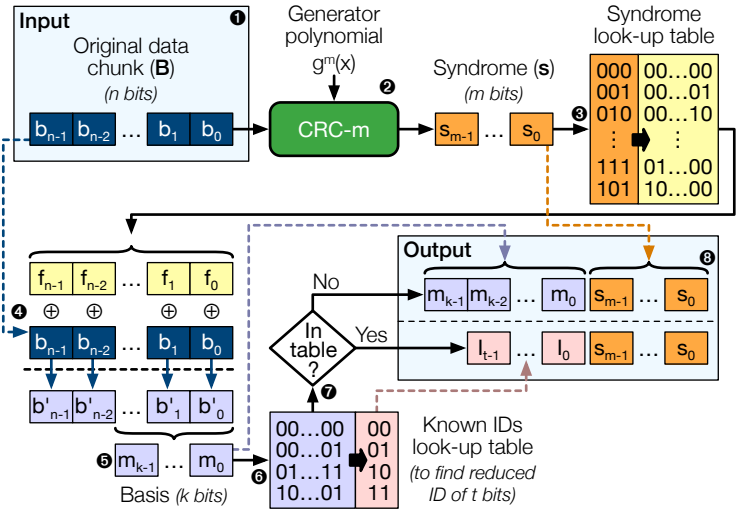
ZipLine: Encoding



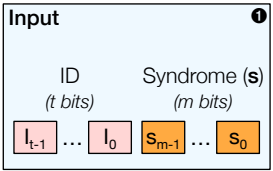
ZipLine: Encoding



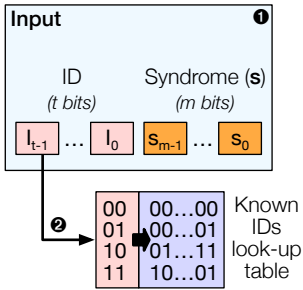
ZipLine: Encoding



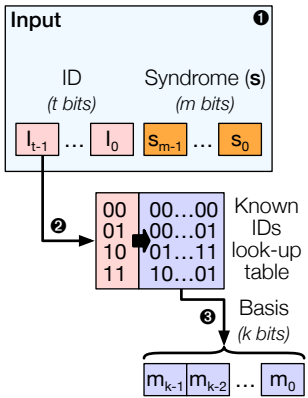
ZipLine: Decoding



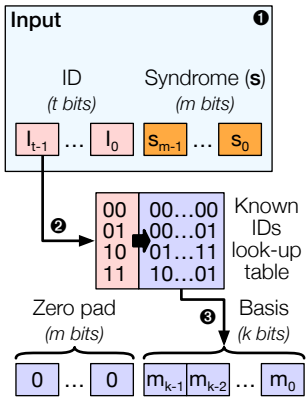
ZipLine: Decoding



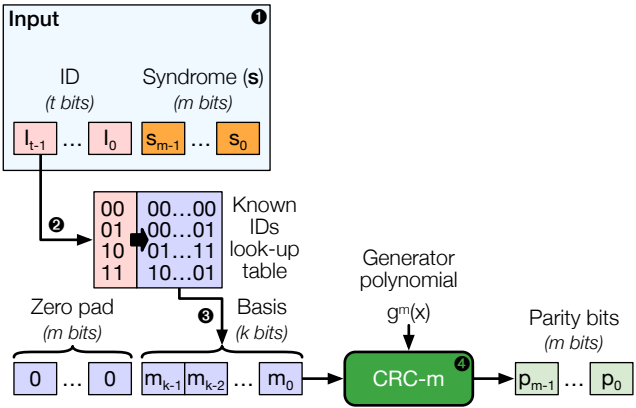
ZipLine: Decoding



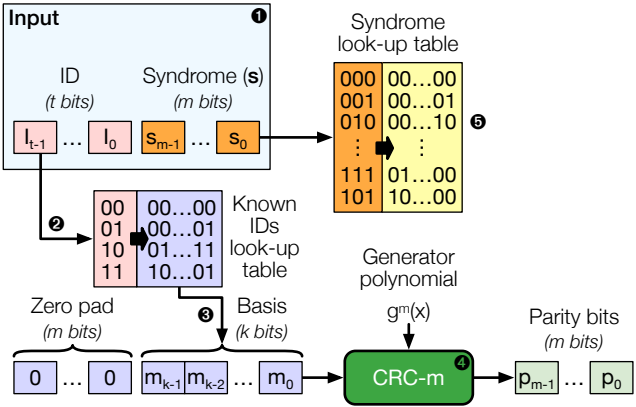
ZipLine: Decoding



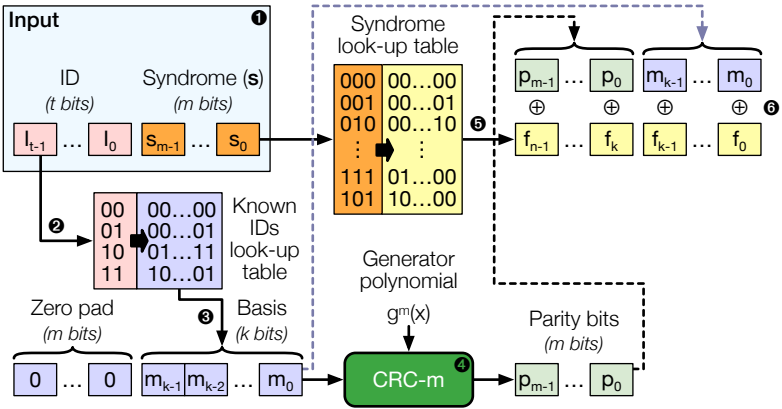
ZipLine: Decoding



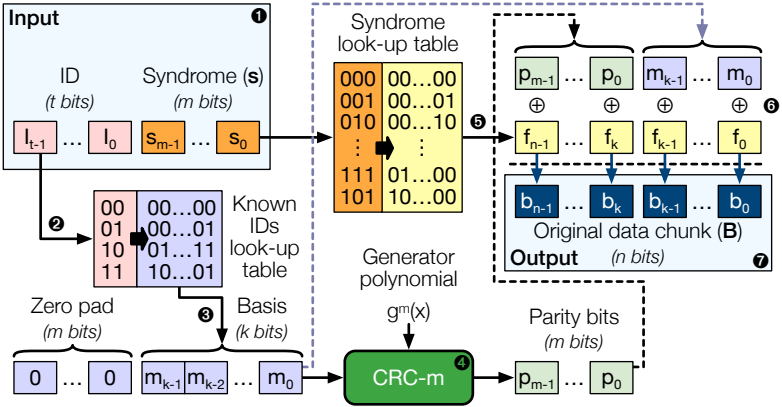
ZipLine: Decoding



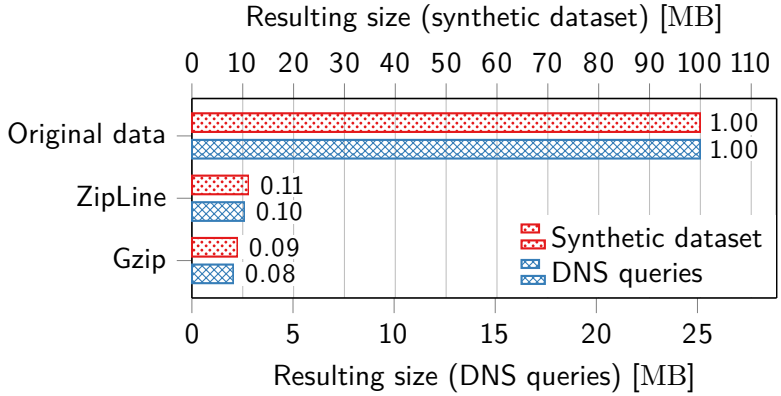
ZipLine: Decoding



ZipLine: Decoding

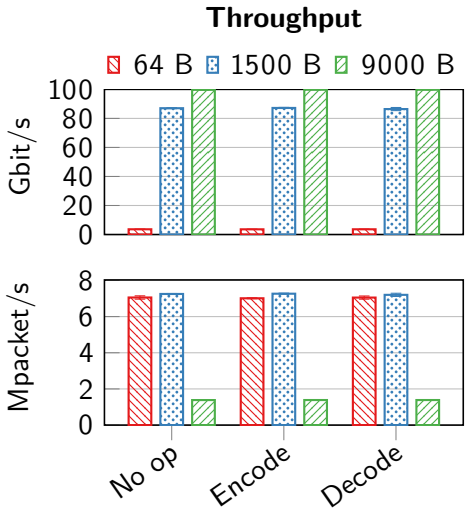


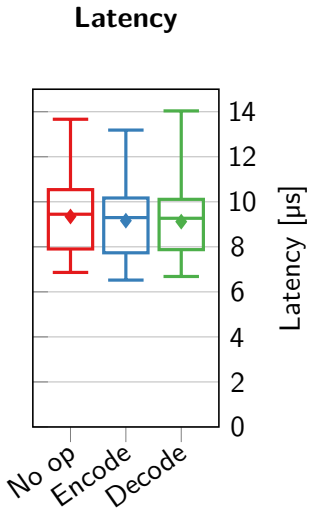
Evaluation: Compression ratio



Synthetic dataset: Data that maps to 2000 distinct bases
DNS queries: 1 day of DNS queries from 4000 active users

Evaluation: Performance





Limitations

- P4 only processes packet **headers**
- Largest **byte-aligned** parameters we can handle **in one pass**

$m = 8$	syndrome bits
$k = 247$	basis bits
$n = 255\,(+1)$	data bits

- 1.77 ms delay before a new basis gets compressed
  1. Data plane informs control plane
  2. Control plane adds ID $\rightarrow$ Basis mapping in receiver switch
  3. Control plane adds Basis $\rightarrow$ ID mapping in local switch

Limitations

- P4 only processes packet **headers**
- Largest **byte-aligned** parameters we can handle **in one pass**

$m = 8$	syndrome bits
$k = 247$	basis bits
$n = 255\,(+1)$	data bits

- 1.77 ms delay before a new basis gets compressed
  1. Data plane informs control plane
  2. Control plane adds ID $\rightarrow$ Basis mapping in receiver switch
  3. Control plane adds Basis $\rightarrow$ ID mapping in local switch

Conclusions

- Compression of network packets with Generalized Deduplication
- 90 % packet size reduction in real-world scenario
  - Control plane updates tables dynamically
  - Decisions based on the current packet
  - Generalized Deduplication amplifies impact of tables: better memory use
- Implementation on Intel Tofino platform
  - Operates at 100 Gbit/s
  - No extra latency

More details: S. Vaucher, N. Yazdani, P. Felber, D. E. Lucani, V. Schiavoni, ZipLine: in-network compression at line speed, *ACM CoNEXT*, pp. 399–405, 2020

Future Opportunities

- Demonstrating potential for **multiple sources** communicating to one receiver
  - Generalized deduplication shown to provide good performance in these settings
  - Common table can be used across multiple flows
- Generalized Deduplication well suited for **approximate analytics**
  - Various analytics can be performed on-the-fly
  - Statistics as well as more complex tasks (e.g., anomaly detection)
  - No need to store all data: bases (& times repeated) are fairly good statistics