Estimation of logarithmic and exponential functions entirely in P4-programmable data planes

Damu Ding[1,2], **Marco Savi**[1], **Domenico Siracusa**[1]

[1] *FBK CREATE-NET Research Center, Trento, Italy*
[2] *University of Bologna, Bologna, Italy*

2nd EuroP4 Workshop
23rd September, 2019

Advanced network monitoring functionalities in data planes

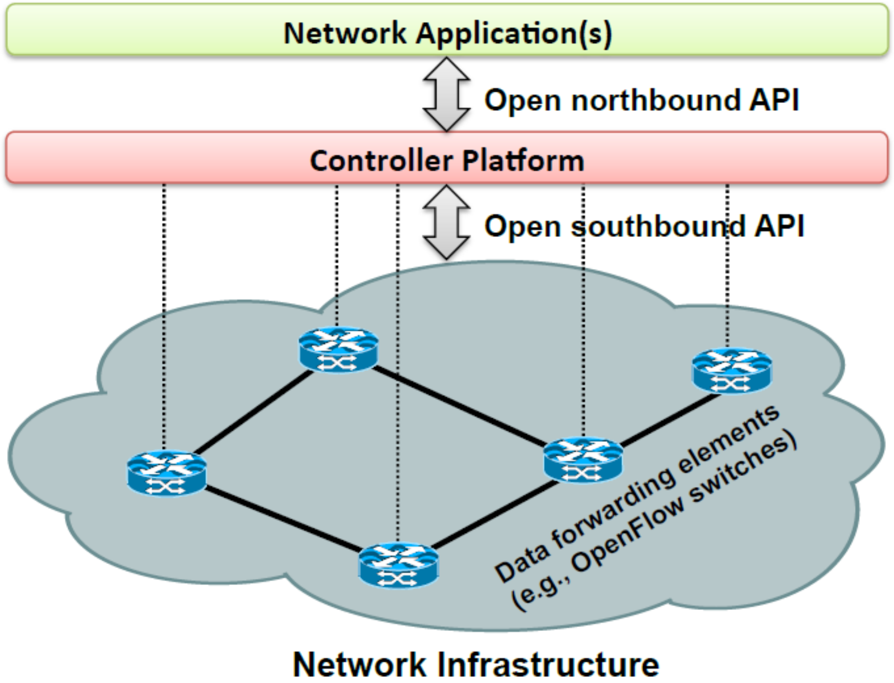
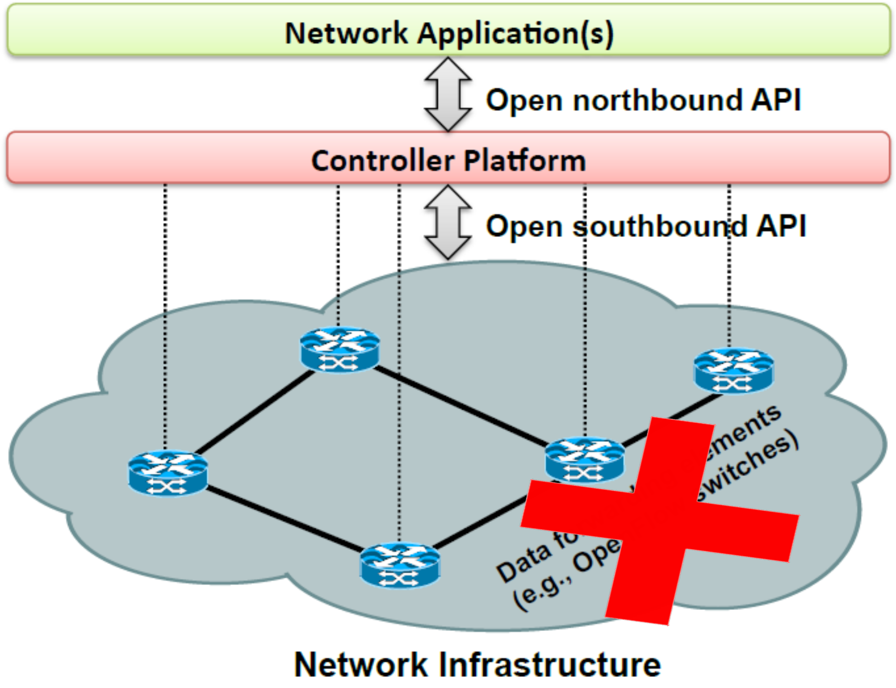


Figure source: Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103.1 (2015): 14-76. and https://n0where.net/real-time-network-monitoring-cyberprobe

Advanced network monitoring functionalities in data planes

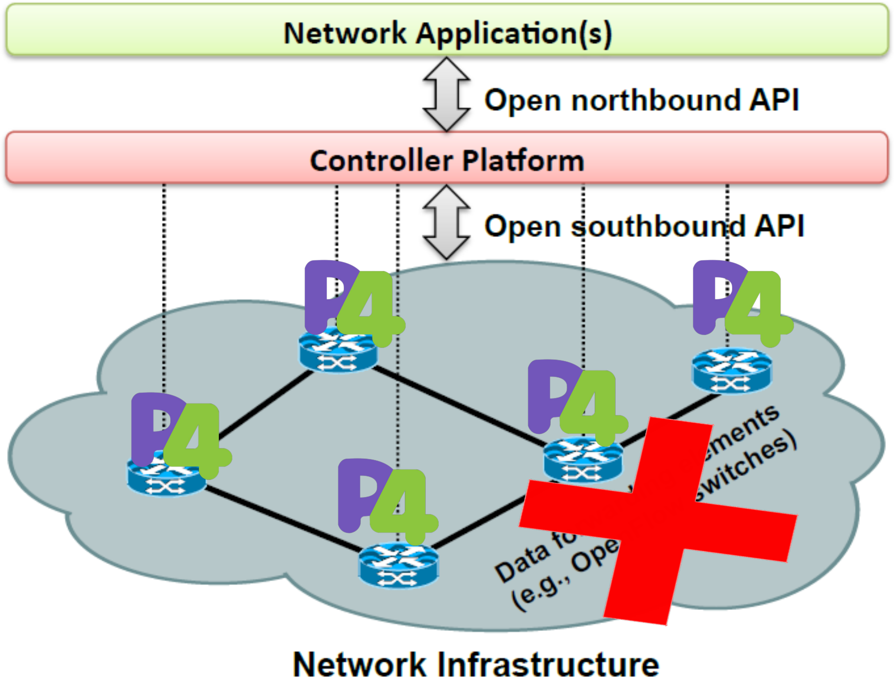


↑

Programmable switches

Figure source: Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103.1 (2015): 14-76. and https://n0where.net/real-time-network-monitoring-cyberprobe

Advanced network monitoring functionalities in data planes

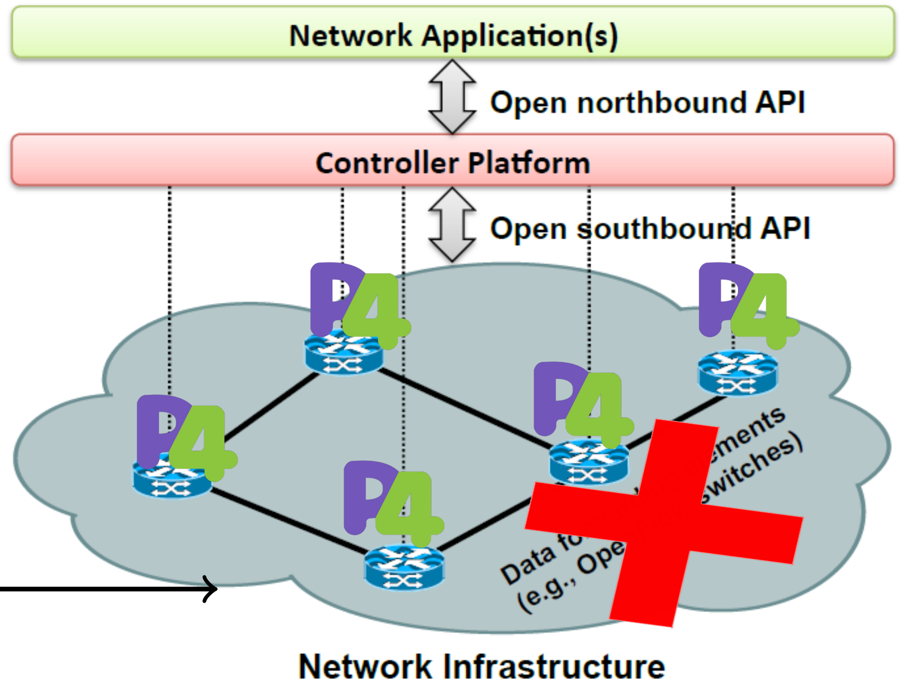


↑

Programmable switches

Figure source: Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103.1 (2015): 14-76. and https://n0where.net/real-time-network-monitoring-cyberprobe

Advanced network monitoring functionalities in data planes



Programmable switches

Figure source: Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103.1 (2015): 14-76. and https://n0where.net/real-time-network-monitoring-cyberprobe

P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.

P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.

Logarithmic function

P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.

| Exponential function |

| Logarithmic function |

P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.

Loops (*For*/*While*)

Exponential function

Logarithmic function

P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.

Division

Loops (*For*/*While*)

Exponential function

Logarithmic function

P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.

Floating numbers

Division

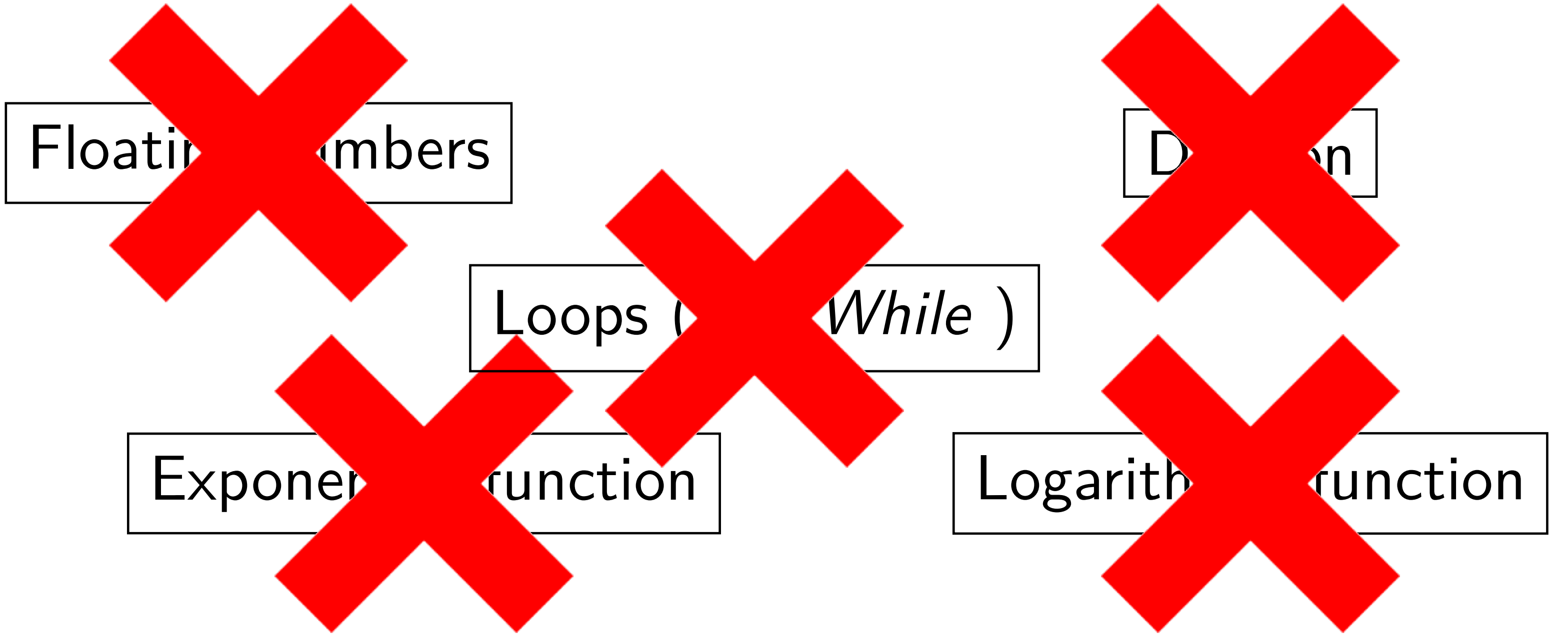
Loops (*For*/*While*)

Exponential function

Logarithmic function

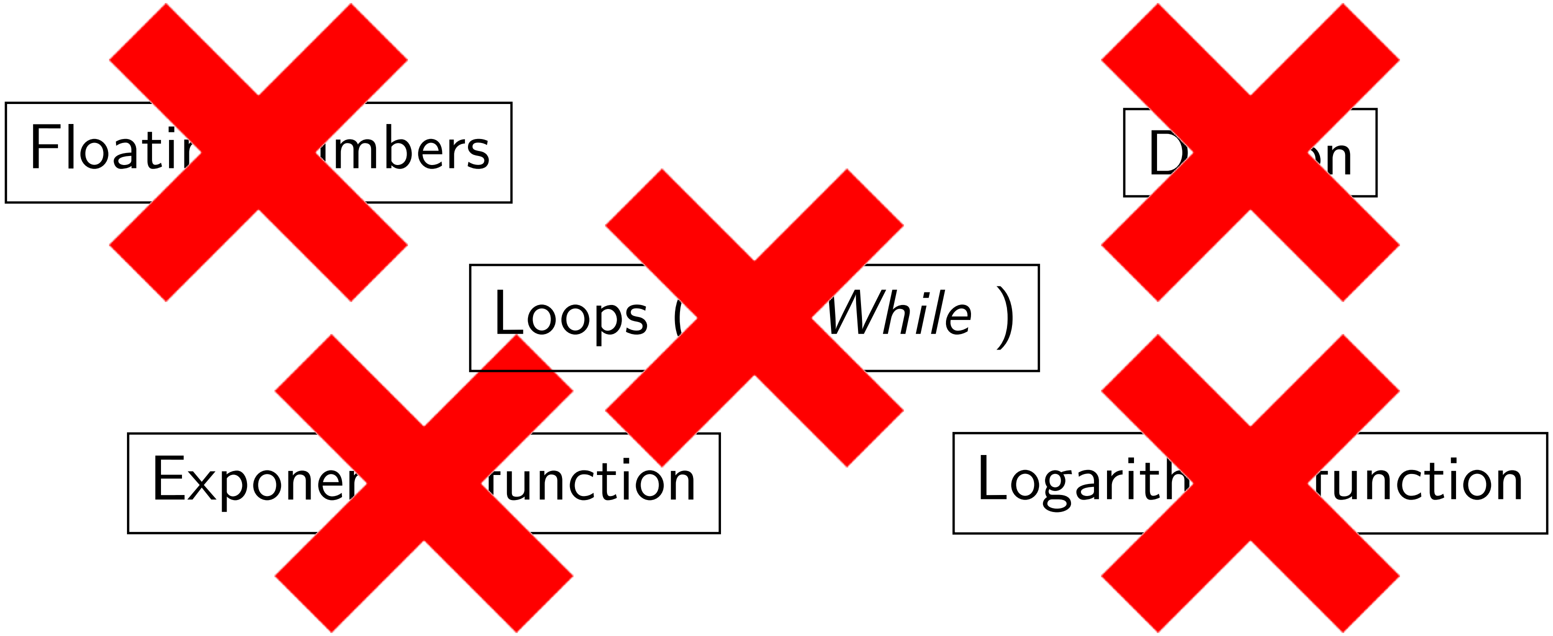
P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.



P4 language limitations

- Supports $+$, $-$, $*$, $\gg$, $\ll$, ^ (XOR), | (OR), & (AND), if-else statements, etc.



- Enable logarithmic and exponential-function estimation entirely in P4 language

Why logarithmic and exponential-function estimations?

Why logarithmic and exponential-function estimations?

- Logarithmic-function estimation
  - Bitmap-based cardinality estimation ($E = m \ln(m/V)$)
    - Linear counting algorithm
    - Range corrections in HyperLogLog algorithm
    - Number of items in Bloom filter
  - Network traffic entropy estimation
    - $H = -\sum_{i=1}^{n} \frac{m_i}{m} \log_d \frac{m_i}{m}$

Why logarithmic and exponential-function estimations?

- Logarithmic-function estimation
  - Bitmap-based cardinality estimation ($E = m\ln(m/V)$)
    - Linear counting algorithm
    - Range corrections in HyperLogLog algorithm
    - Number of items in Bloom filter
  - Network traffic entropy estimation
    - $H = -\sum_{i=1}^{n} \frac{m_i}{m} \log_d \frac{m_i}{m}$
- Exponential-function estimation
  - Division estimation
    - $\frac{A}{B} = 2^{(\log_2 A - \log_2 B)}$
  - More advanced cardinality estimation
    - LogLog algorithm: $E = \alpha_m m 2^{\frac{1}{m}\sum_j (M(j))}$
    - HyperLogLog algorithm: $E = \alpha_m m^2 2^{\sum_j (2^{-M(j)})}$

Our contributions

1. **P4Log** algorithm for the estimation of logarithmic function
2. **P4Exp** algorithm for the estimation of exponential function
3. We implemented a prototype of the proposed algorithms in the P4 behavioral model [1], proving that they can be **entirely** executed in the programmable data plane.

[1] https://github.com/p4lang/behavioral-model

P4Log algorithm

- **INPUT:** An L-bit integer x (L $\in \{16, 32, 64\}$) and a given logarithmic base d
- **OUTPUT:** Estimation of $\log_d x \ll 10$ (i.e., $\log_d x \cdot 2^{10}$)

P4Log algorithm

- **INPUT:** An L-bit integer x (L $\in \{16, 32, 64\}$) and a given logarithmic base d
- **OUTPUT:** Estimation of $\log_d x \ll 10$ (i.e., $\log_d x \cdot 2^{10}$)

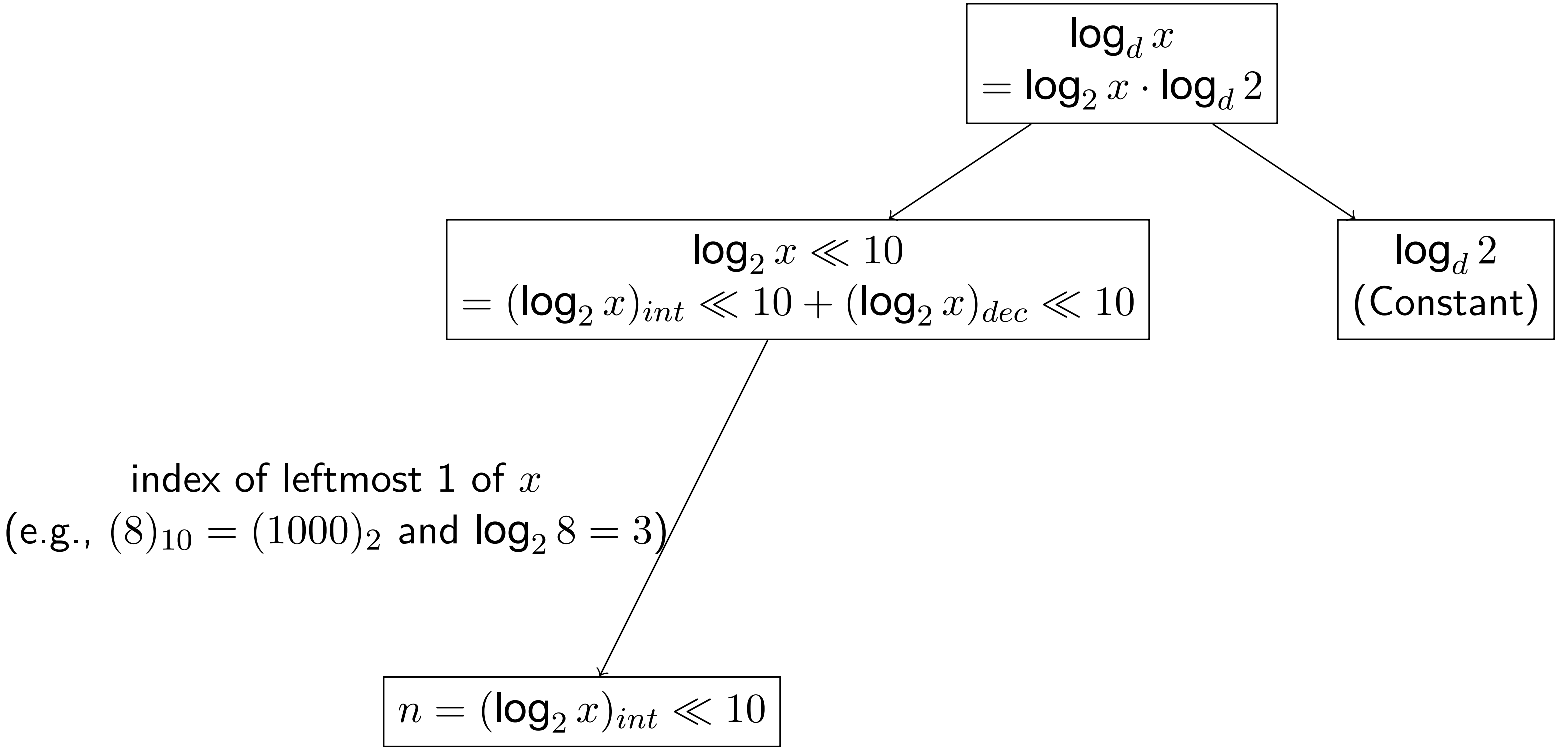
$$\log_d x = \log_2 x \cdot \log_d 2$$

- $\log_2 x \ll 10 = (\log_2 x)_{int} \ll 10 + (\log_2 x)_{dec} \ll 10$
- $\log_d 2$ (Constant)

D.Ding et al. ding@fbk.eu

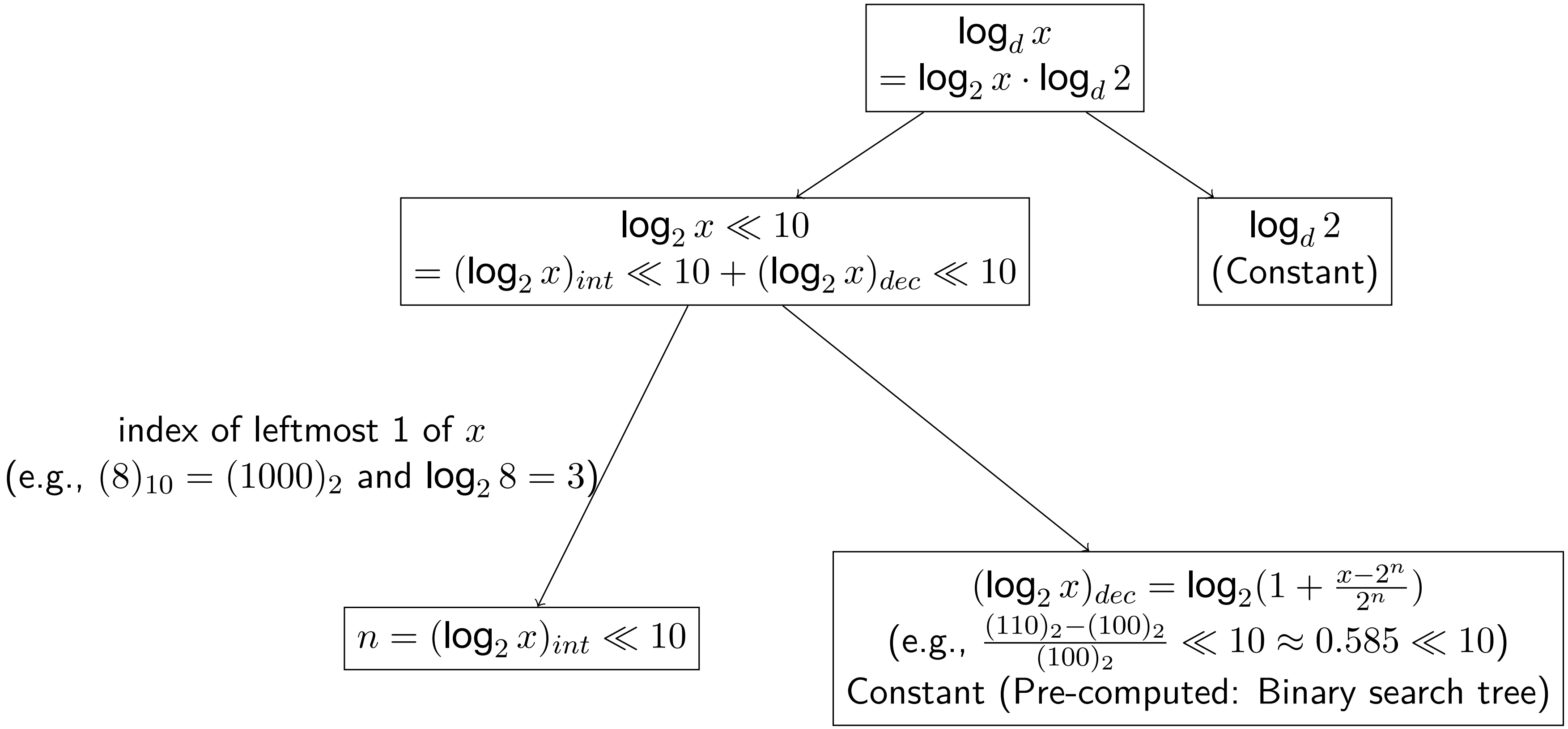
P4Log algorithm

- **INPUT:** An L-bit integer x (L $\in \{16, 32, 64\}$) and a given logarithmic base d
- **OUTPUT:** Estimation of $\log_d x \ll 10$ (i.e., $\log_d x \cdot 2^{10}$)



P4Log algorithm

- **INPUT:** An L-bit integer x (L $\in \{16, 32, 64\}$) and a given logarithmic base d
- **OUTPUT:** Estimation of $\log_d x \ll 10$ (i.e., $\log_d x \cdot 2^{10}$)



P4Exp algorithm

- **INPUT:** An integer base x and an exponent d
- **OUTPUT:** Estimation of x^d

P4Exp algorithm

- **INPUT:** An integer base x and an exponent d
- **OUTPUT:** Estimation of x^d

$$x^d = 2^{d \log_2 x} = 2^{(d \log_2 x)_{int}} \cdot 2^{(d \log_2 x)_{dec}}$$

$$2^{(d \log_2 x)_{int}} = 1 \ll ((d \log_2 x)_{int})$$

$$2^{(d \log_2 x)_{dec}}$$

P4Exp algorithm

- **INPUT:** An integer base x and an exponent d
- **OUTPUT:** Estimation of x^d

$$x^d = 2^{d \log_2 x} = 2^{(d \log_2 x)_{int}} \cdot 2^{(d \log_2 x)_{dec}}$$

$$2^{(d \log_2 x)_{int}} = 1 \ll ((d \log_2 x)_{int})$$

$$2^{(d \log_2 x)_{dec}}$$

- **Binomial series expansion**:
$$2^y = 1 + y + \frac{y(y-1)}{2!} + \frac{y(y-1)(y-2)}{3!} + \cdots$$

P4Exp algorithm

- **INPUT:** An integer base x and an exponent d
- **OUTPUT:** Estimation of x^d

$$x^d = 2^{d \log_2 x} = 2^{(d \log_2 x)_{int}} \cdot 2^{(d \log_2 x)_{dec}}$$

$$2^{(d \log_2 x)_{int}} = 1 \ll ((d \log_2 x)_{int})$$

$$2^{(d \log_2 x)_{dec}}$$

- **Binomial series expansion**:
$$2^y = 1 + y + \frac{y(y-1)}{2!} + \frac{y(y-1)(y-2)}{3!} + \cdots$$

So it holds that:

$$2^{(d \log_2 x)_{dec}} = 1 + (d \log_2 x)_{dec} + \frac{(d \log_2 x)_{dec}((d \log_2 x)_{dec}-1)}{2!} + \frac{(d \log_2 x)_{dec}((d \log_2 x)_{dec}-1)((d \log_2 x)_{dec}-2)}{3!} + \cdots$$

Comparison with SOTA

- SOTA[2] needs thousands of table entries in TCAM for the logarithmic and exponential-function estimations to assure the relative error is under 1%
- By properly setting the parameters (e.g., Terms in Binomial series expansion) in the P4Log and P4Exp algorithms, our algorithms could reach the same accuracy as SOTA
- No TCAM and extra stateful memories (e.g., registers, counters and meters)
  - TCAM is expensive and power-hungry
  - TCAM needs Communication overhead for populating the lookup tables in the switches
  - Programmable switches have limited memory
- Only relies on ALU instructions

[2] Sharma, N. K., Kaufmann et al. "Evaluating the power of flexible packet processing for network resource allocation" Symposium on Networked Systems Design and Implementation (NSDI 17) (pp. 67-82).

Future work

1. Implement advanced cardinality estimation (e.g., LogLog and HyperLogLog)-based and network traffic entropy-based DDoS detection entirely in programmable data plane
2. Test proposed algorithms and strategies on a real testbed

Acknowledgement:

The research leading to these results has received funding from the EC within the H2020 Research and Innovation program, Grant Agreement No. 856726 (GN4-3 project).