Sketch-based DDoS detection and monitoring using P4 data plane programming

Damian Parniewicz[1], Marco Savi[2], Damu Ding[2], Domenico Siracusa[2], Mikołaj Nowaczyk[1], Pavel Benacek[3], Mauro Campanella[4], Xavier Jeannin[5]

[1] PSNC, Poznań, Poland

[2] FBK, CREATE-NET Research Center, Trento, Italy

[3] CESNET, Prague, Czech Republic

4 GARR, Rome, Italy

5 RENATER, Paris, France

TNC 2019

20/06/2019

Tallin

Agenda

Introduction on data plane programming in GN3-4

- Telemetry use case
- DDoS use case

Sketches and sketching algorithms

- General pros and cons
- Usage advantages in the network monitoring

DDoS detection and monitoring implementation

- Problem formalization
- Sketch-based algorithms
- P4 data plane and controller workflows
- Memory requirements
- Experienced problems

Demonstrations and conclusion

Agenda

Introduction on data plane programming in GN3-4

- Telemetry use case
- DDoS use case

Sketches and sketching algorithms

- General pros and cons
- Usage advantages in the network monitoring

DDoS detection and monitoring implementation

- Problem formalization
- Sketch-based algorithms
- P4 data plane and controller workflows
- Memory requirements
- Experienced problems

Demonstrations and conclusion

Introduction of GN4-3 WP6 task 1 – Data Plane Programmability

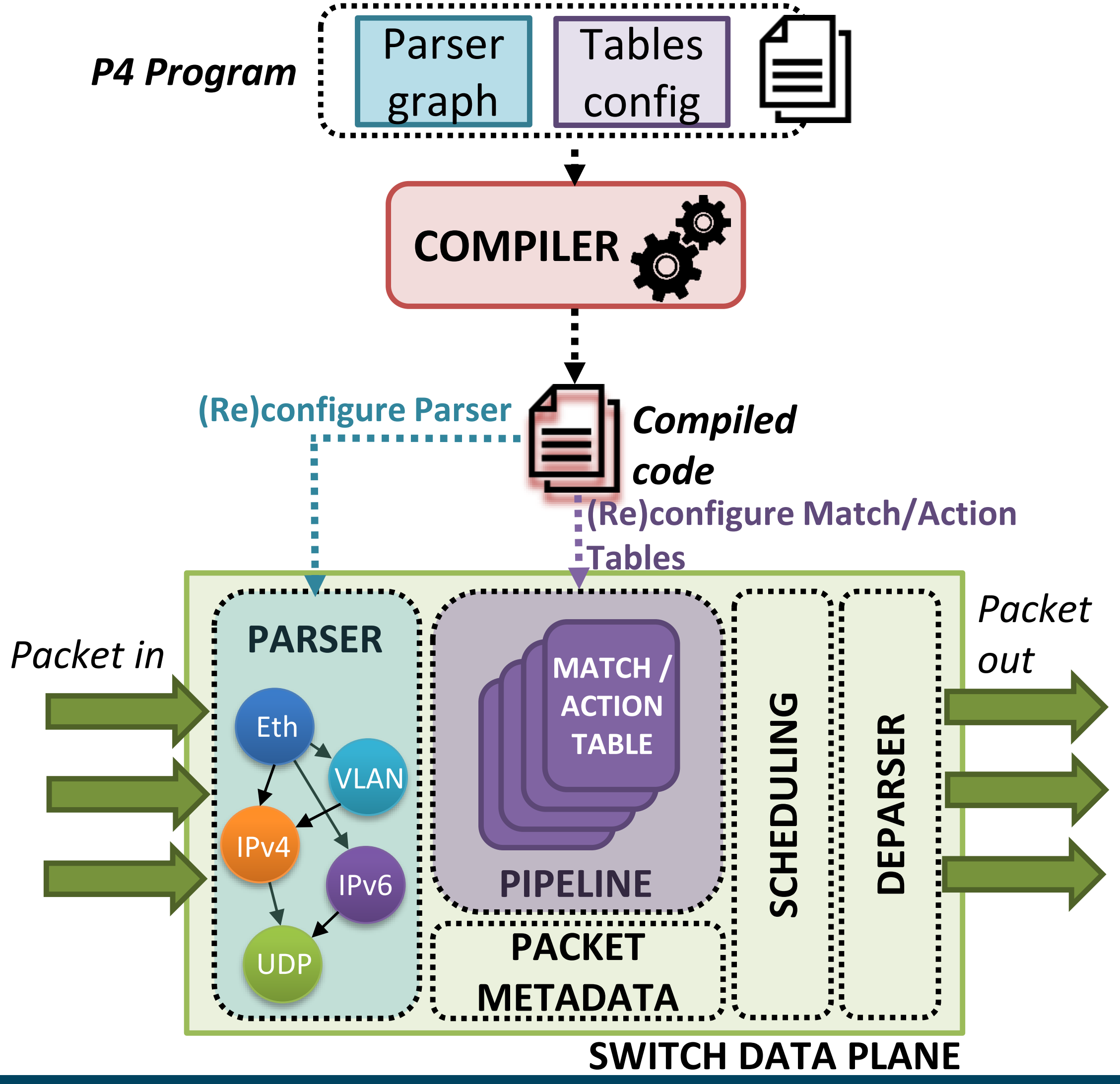
- Validate the novel programmability and monitoring concepts implementable directly in the data plane
- Usage of P4 language for FPGA and new chips (e.g.: Barefoot Tofino) for improved monitoring and network functions
- Implement prototypes for two use-case:
  - In-band Network Telemetry (INT)
  - DDoS Detection, Monitoring and Mitigation

Netcope NFB-100G2

Arista 7170-32c

Edgecore Wedge100BF-32X

P4 data plane programming

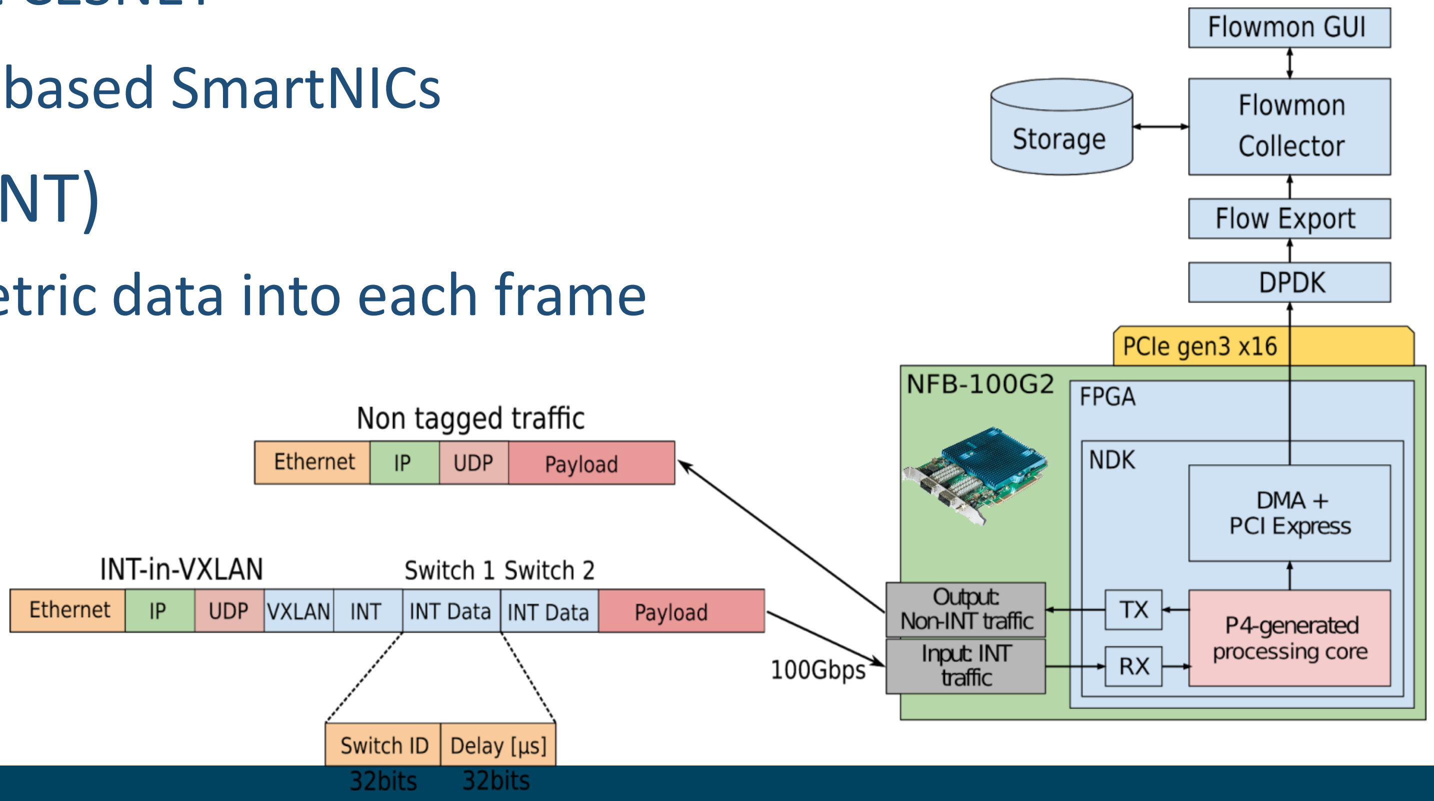



```
#include <v1model.p4>
header ethernet_t { bit<48> dst;  bit<48> src; bit<16>  etherType; }
struct headers { ethernet_t ethernet; }

parser MyParser(packet_in packet, out headers hdr) {
        state start { transition parse_ethernet; }
        state parse_ethernet {
                        packet.extract(hdr.ethernet);
                        transition accept;
}
control ForwardEgress(inout headers hdr, inout standard_metadata_t  meta) {
        table send_frame {
                        key = { hdr.ethernet.src: exact; meta.ingress_port: exact; }
                        actions = { rewrite_smac; NoAction; }
                        size = 256;
        }
        action rewrite_smac(bit<48> smac, bit<8> egress_port) {
                        hdr.ethernet.src = smac; meta.egress_port = egress_port; }
        apply { send_frame.apply(); }
}
```

Telemetry Use Case

- **Goal**: Framework for processing of telemetric data
  - Based on P4 implementation of In-band Network Telemetry specifications
  - P4-to-VHDL compiler developed at CESNET
  - VHDL code deployed on the FPGA-based SmartNICs
- P4 In-band Network Telemetry (INT)
  - Allows insertion/analysis of telemetric data into each frame
    - Taken path through the network
    - Buffer occupancy in network devices
    - Actual switch processing delays

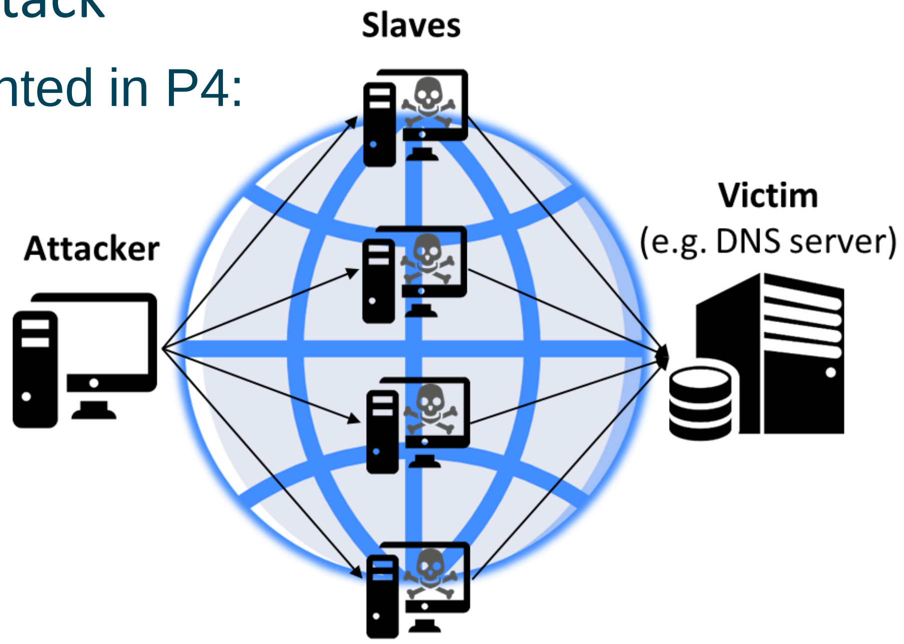


Telemetry Use Case [2]

- In-band Network Telemetry allows for detailed network debugging and performance analysis:
  - find flows causing congestion
  - find high end-to-end latency flows
  - detect events about flow path change
  - detect events about high e2e flow latency increase
  - observe high-resolution flow dynamics (how its characteristics changes, microbursts, etc)
  - observe high-resolution node queue dynamics

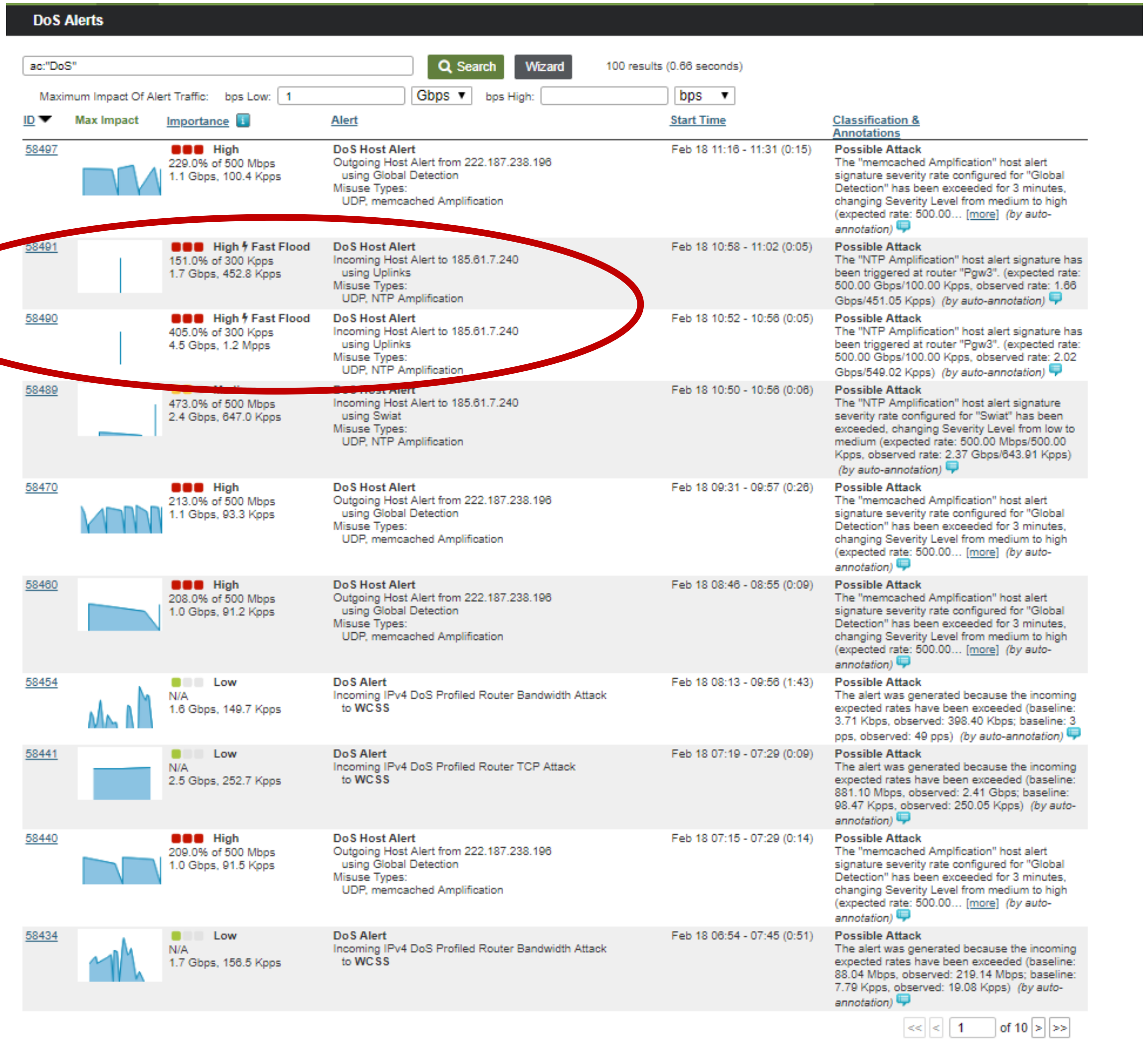
DDoS Use Case

- Goals of the DDoS use case:
  - **Very fast detection** of DDoS attacks on boundaries of NRENs/ GÉANT network
  - Providing detailed information about the **DDoS attack characteristics**
  - The possibility of almost **immediate blocking** of the attack
- Based on **Big Data streaming algorithms** (sketches) implemented in P4:
  - Memory-effective collection of summarized traffic statistics
- Usage of **P4 switches**:
  - P4 behavioral model - bvm2 (in the first phase)
  - Edgecore Wedge100BF-32X
  - Arista 7170-32c



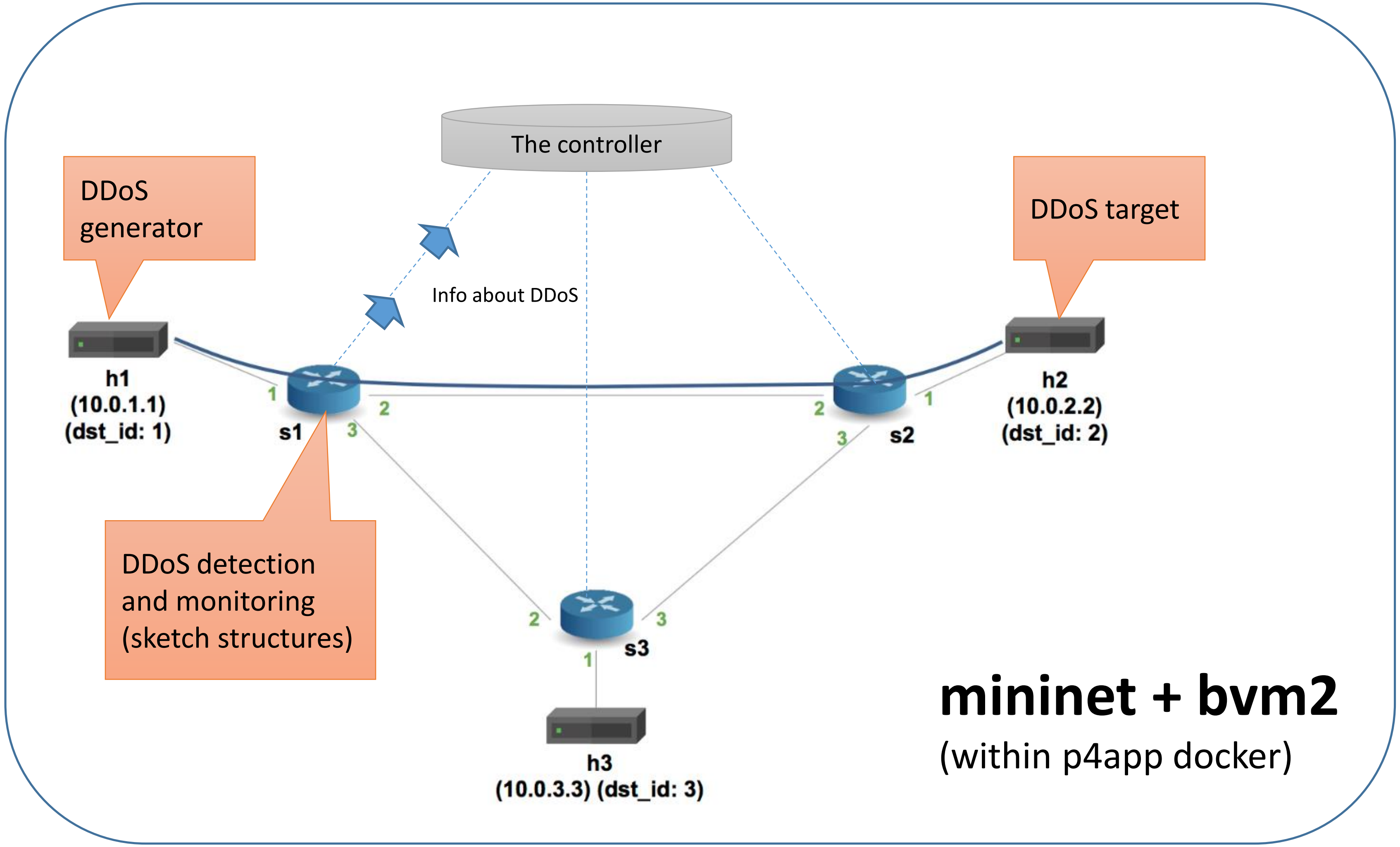
DDoS attacks are still a real problem

- **20-40 DDoS attacks per day** in PSNC
- Quite problematic are repeated short volumetric DDoS attacks **active for 1-5 minutes**
  - Too short time for existing mitigation techniques
- UDP **amplification flood** from **highly distributed IP addresses** around the world to a single IP address in PSNC network
- observed on the 10GE links **from big international network providers** and have an impact on our users and services because our **10GE links become overloaded**



First phase - virtual testing environment

- Each switch is a bvm2 instance (P4 switch emulator) started as part of **mininet virtual infrastructure**
- The **Python controller** is receiving short summaries about DDoS target and main DDoS characteristics
- **DDoS traffic** is generated by Python script
  - Random IP addresses from a set of network domains
  - Random set of src/dst ports



Agenda

Introduction on data plane programming in GN3-4

- Telemetry use case
- DDoS use case

Sketches and sketching algorithms

- General pros and cons
- Usage advantages in the network monitoring

DDoS detection and monitoring implementation

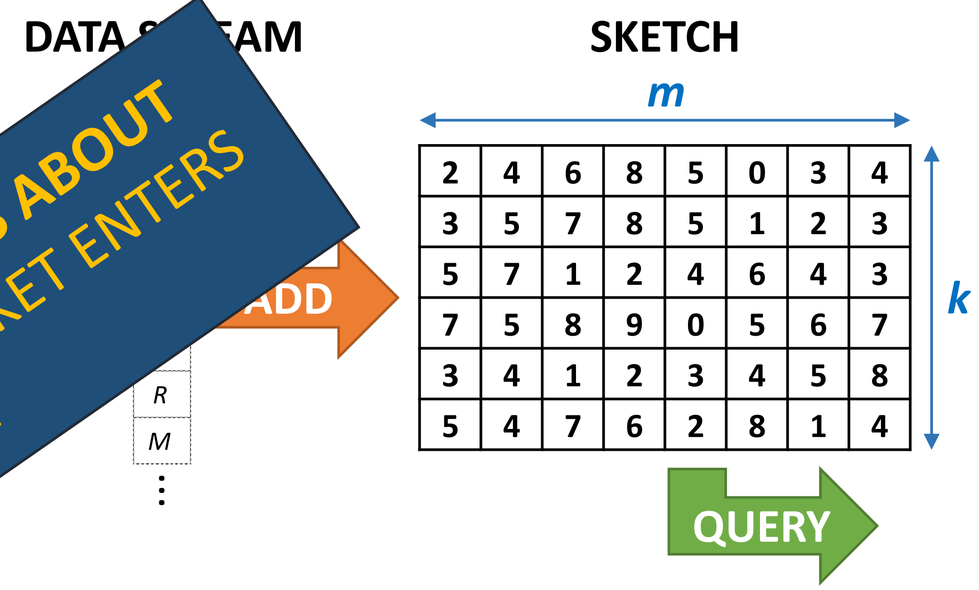
- Problem formalization
- Sketch-based algorithms
- P4 data plane and controller workflows
- Memory requirements
- Experienced problems

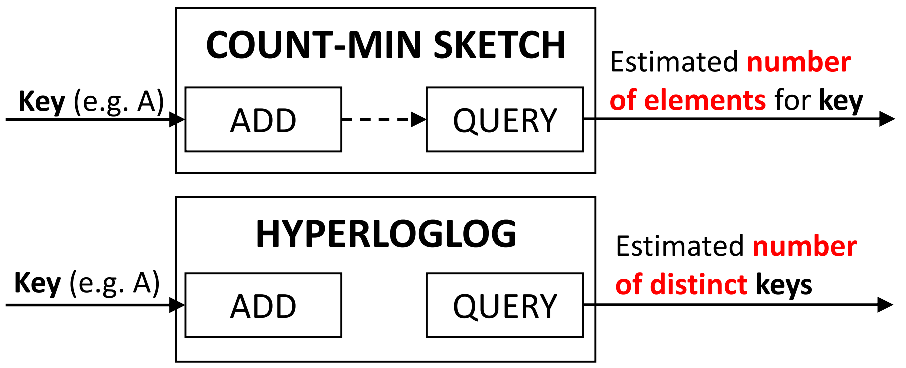
Demonstrations and conclusion

Algorithms that we want program in P4

- *Sketches*
  - Probabilistic data structures
  - Summarize statistics from a data stream (require
    instead of GB)
  - Provable memory/accuracy trade-off
- *Sketching algorithms*
  - Use specific *hashing* and *counti*
  - **Add**: update sketch with ne … eam
  - **Query**: get estimated d
- Example of sketching a
  - **Count-min sk** … mation)
  - **HyperLogL** … mation)
  - **Bloom filter** … p estimation)

POWERFUL TOOLS TO COLLECT **STATISTICS ABOUT PACKETS AND FLOWS** EVERY TIME A PACKET ENTERS THE SWITCH P4 PIPELINE!





Sketches and sketching algorithms for the network monitoring

- Plenty of monitoring tools already exist (e.g. NetFlow, SFlow...)
- Why sketches?

- Pros
  - Fast (in the *data plane*, at line rate)
  - Summarized data
    - Low memory consumpion
    - Light communication to network controller/monitoring system
  - All packets contribute to statistics (no *sampling*)
  - Good time granularity
    - Allows fast detection of anomalies/attacks

- Cons
  - Estimation of statistics (not deterministic)
  - Sketches need to be queried to get statistics

P4 is enabler for usage of a dedicated sketch memory structures/algorithms in switch chipset for very specific problems

Agenda

Introduction on data plane programming in GN3-4

Telemetry use case

DDoS use case

Sketches and sketching algorithms

General pros and cons

Usage advantages in the network monitoring

DDoS detection and monitoring implementation

Problem formalization

Sketch-based algorithms

P4 data plane and controller workflows

Memory requirements

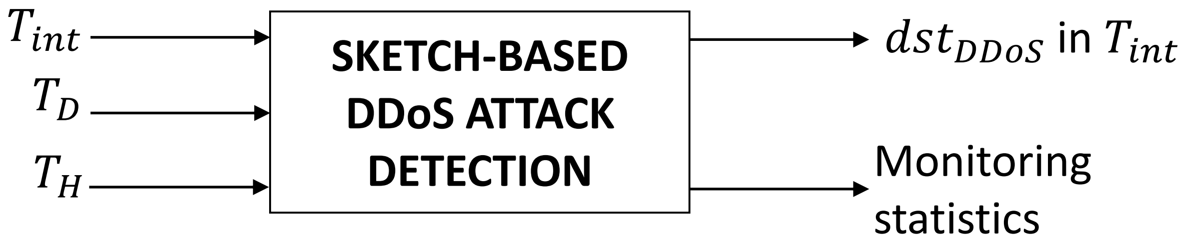
Experienced problems

Demonstrations and conclusion

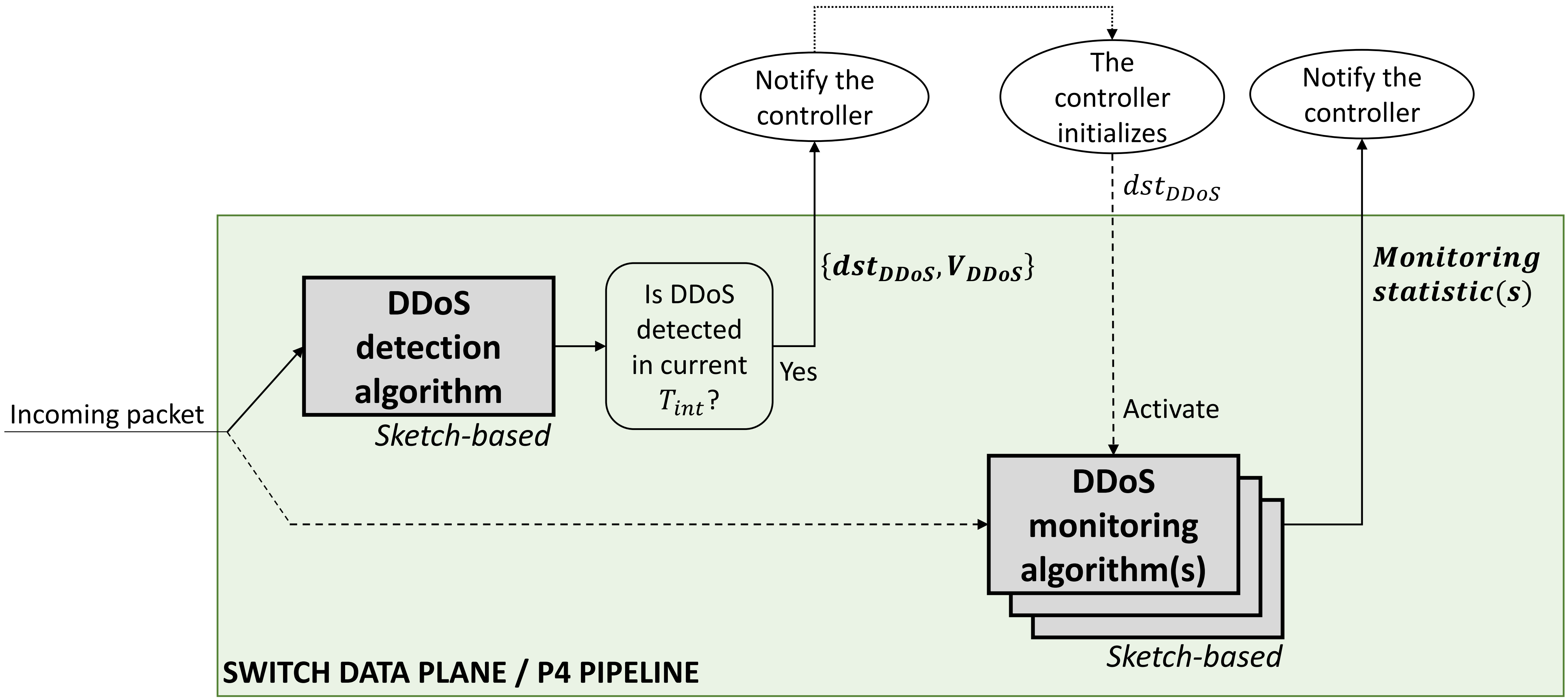
Formalization of the volumetric DDoS attack detection problem

- **Given**
  - A time interval T_{int}
  - A threshold on the number of *src* (cardinality threshold T_D)
  - A threshold on the number of packets routed towards *dst* (volume threshold T_H)

- **Identify** all the *dst* under DDoS attack in time interval T_{int} (i.e., dst_{DDoS})
  - *dst* that:
    1. Have been contacted by a number of $src > T_D$
    2. Have received a number of packets > T_H (i.e., V_{DDoS})
- **Collect relevant monitoring statistics** for dst_{DDoS} under attack

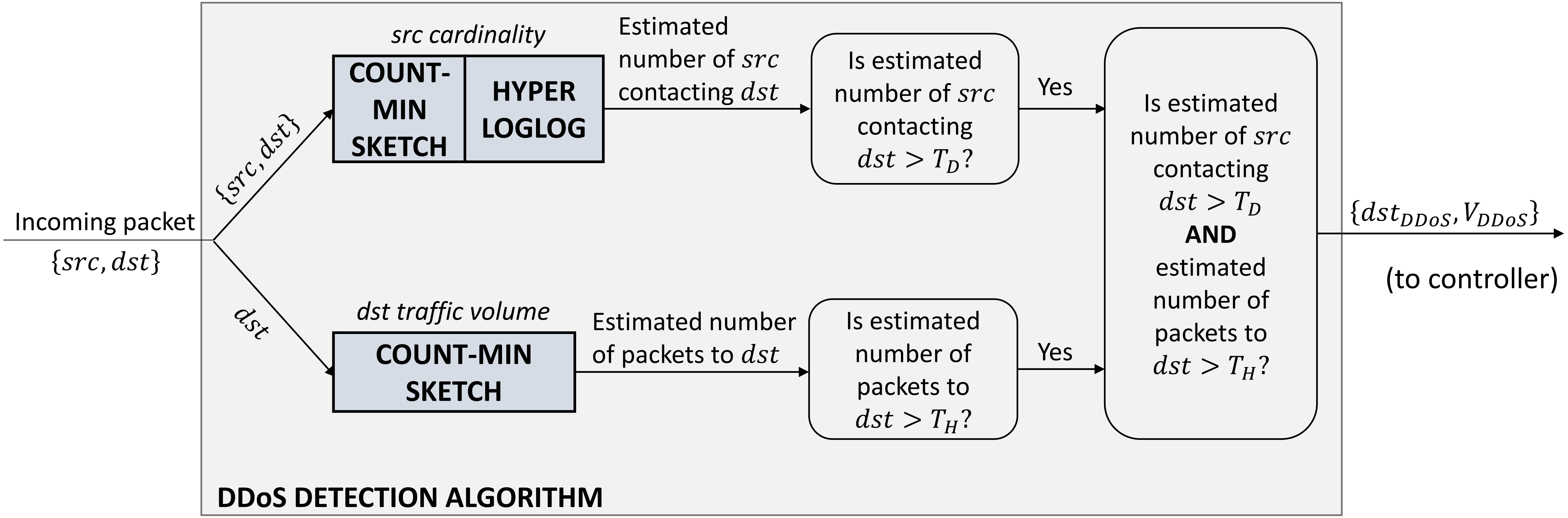


DDoS attack detection and monitoring workflow



Sketch-based DDoS attack detection

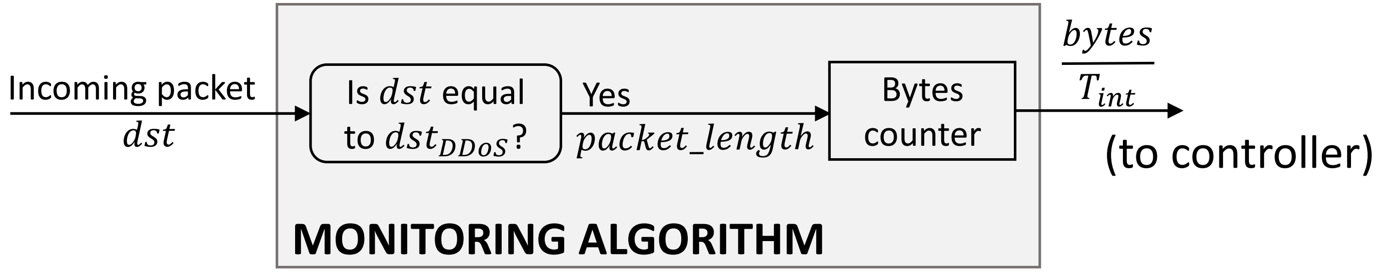
T_D: cardinality threshold
T_H: volume threshold



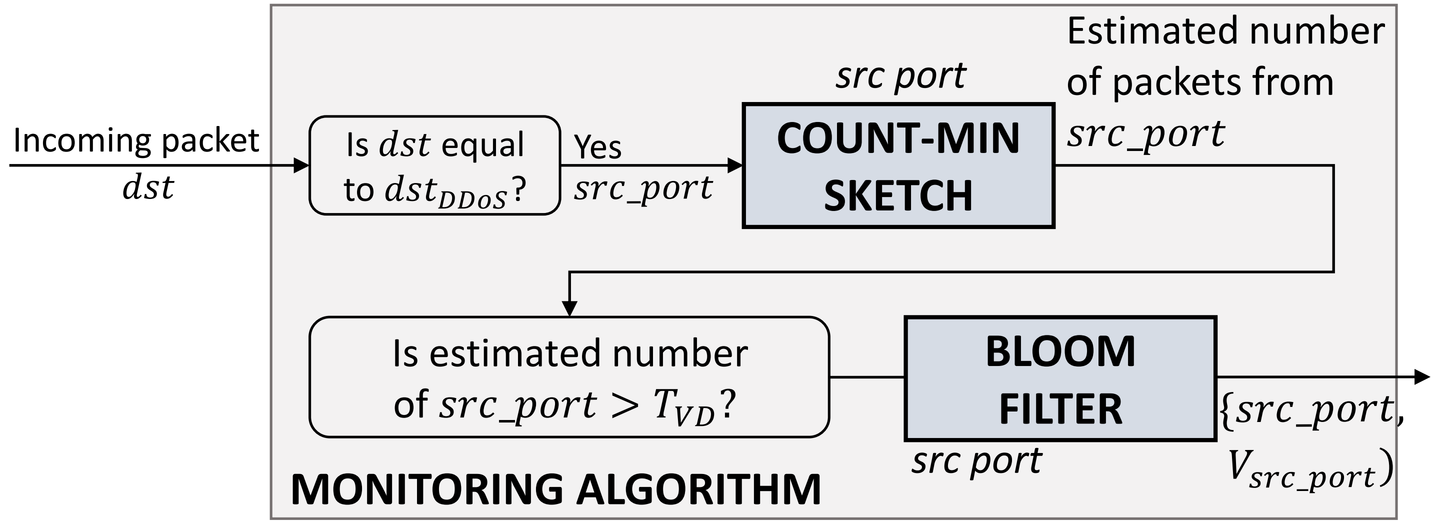
DDoS monitoring algorithms

- Can be activated only when a DDoS attack is detected
  - Monitor different traffic statistics
- Monitoring information that can be collected:
  1. Total traffic
  2. Frequent source subnets
  3. Frequent source UDP/TCP ports
  4. Frequent destination UDP/TCP ports
  5. Frequent IP protocol numbers

Total traffic



Frequent source UDP/TCP ports



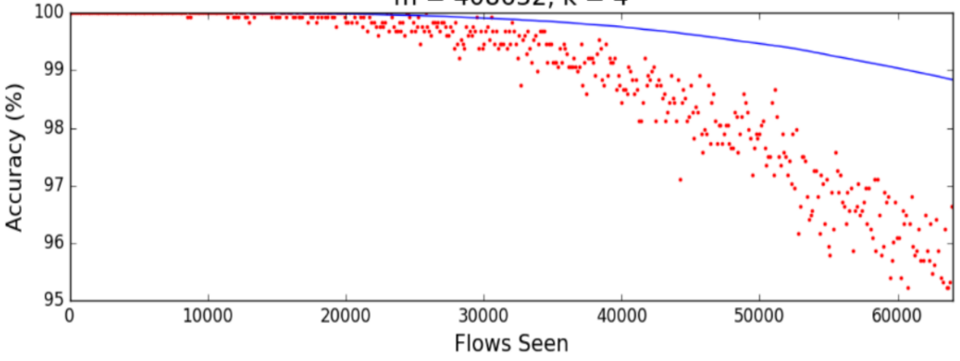
Big data streaming structures size optimalization

GÉANT
Networks • Services • People

	Min-count sketch	Bloom filter
Sketch width	$m = \left\lceil \frac{2 * \ln(items)}{error} \right\rceil$	$\mathrm{m} = \left\lceil -\frac{items * \ln(error)}{\ln(2)^2} \right\rceil$
Sketch depth	$\mathrm{k} = \left\lceil \frac{\ln(1-certainty)}{\ln(0.5)} \right\rceil$	$\mathrm{k} = \left\lceil \frac{m}{items} * \ln(2) \right\rceil$
False Positive Rate	$\mathrm{FPR} = \left[1 - \left(1 - \frac{1}{m}\right)^{items}\right]^k$	$\mathrm{FPR} = \left[1 - e^{\frac{-k*items}{m}}\right]^k$

bvm2 (v1model.p4 and psa.p4) contains the following **hash functions**:

- **crc32**, **crc32_custom**, **crc16**, **crc16_custom**
- crc32_cust and crc16_custom can **be configured with different set of parameters** – we must check if such hashes are pair-wise independent



G Cormode, S Muthukrishnan, „Approximating data with the count-min data structure”,
 - IEEE Software, 2012

J. Hill et al., „Tracking network flows with P4”, IEEE/ACM Innovating the Network for Data-Intensive Science, 2018

Current P4 prototype memory usage

- IP network prefixes and UDP/TCP ports are 16 bits
    - Count-min sketch:
        - $M = 2 \cdot \ln(65536)/0.01$ = 2219 cells
        - Memory = $4 \cdot$ 4bytes/cell $\cdot$ 2219 cells = 34.6KB
    - Bloom filter:
        - M = 2367 cells,
        - Memory= $4 \cdot$ 1bit/cell $\cdot$ 2219 cells = 1.08KB
- Total memory required:
    - $3 \cdot$ (34.6KB+1.08KB) = **107.2KB**

Memory usage reduction possible:

- Use more hash functions
- Use 2-bytes values instead of 4-bytes values
- Reuse sketches registers between different monitoring methods

Experienced problems when developing a prototype

- Initially code developed in the language version $P4_{14}$
  - But lack of asynchronous messaging from a switch to the controller
- Decided to **move to $P4_{16}$** because of availability of **Digest messages**
  - Almost complete **code rewrite was required**
- Still **many code repetitions** for each sketch structure
  - Cannot pass a reference or pointer to the Register structure to the control function
- **Lack of logarithm operation** makes HyperLogLog implementation much more difficult

Experienced problems when developing a prototype [2]

GÉANT
Networks • Services • People

- P4 behavioral model (bmv2) **virtual environment limitations**:
  - Heavy usage of Registers but Digests will be perfect for monitoring output
    - **Digests are not supported** in Apache Thrift protocol
    - **Registers cannot be read** using gRPC protocol
    - We choose to stay with Apache Thrift and just use `**simple_switch_CLI**`
  - Have to use 4 registers instead of one for Bloom filter implementation
  - `simple_switch_CLI` invocations **from the controller are very slow**:
    - **~0.1 sec** for reading value(s) from a very short register (e.g.: 10 values)
    - **~0.3 sec** for adding a table entry
    - **~0.1 sec** for resetting a memory structure (Register, Table, Counter, …)
  - **Problems in PSA architecture** usage (it is the architecture supported by Barefoot Tofino)
    - When **psa.p4** included than errors are raised (we stayed with **v1model.p4**)

Agenda

Introduction on data plane programming in GN3-4

Telemetry use case

DDoS use case

Sketches and sketching algorithms

General pros and cons

Usage advantages in the network monitoring

DDoS detection and monitoring implementation

Problem formalization

Sketch-based algorithms

P4 data plane and controller workflows

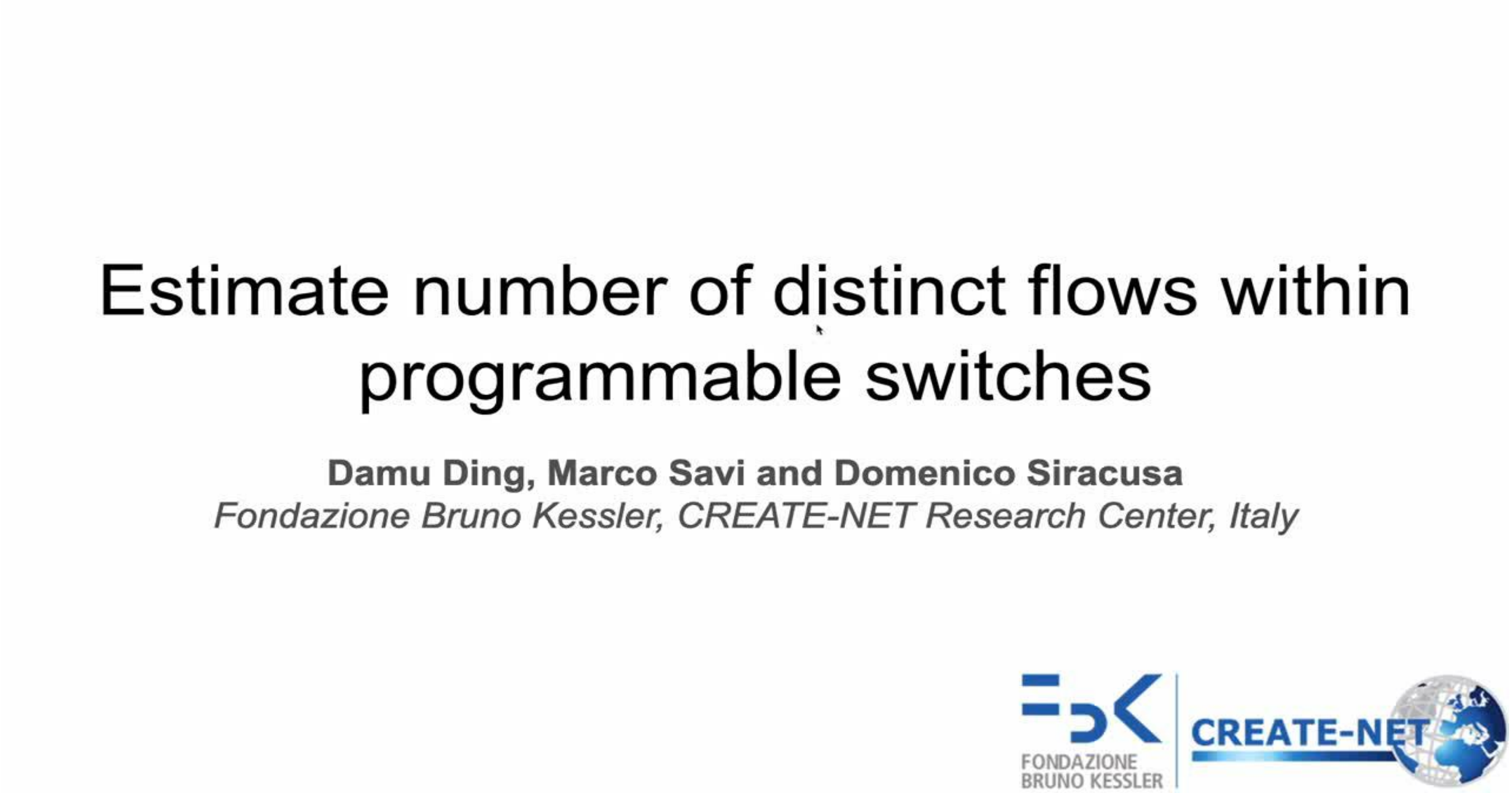
Memory requirements

Experienced problems

Demonstrations and conclusion

Demonstrations

- HyperLogLog demonstration (https://www.youtube.com/watch?v=4qr1OTLq8JQ&feature=youtu.be)



- Live demonstration of the DDoS traffic monitoring

Conclusion and open issues

GÉANT
Networks • Services • People

- Data plane programming improves **flexibility** in handling packets
- Such flexibility can be used to enhance **network monitoring**
- **Sketches** and **sketching algorithms** are a very promising solution to collect packet and flow estimated statistics directly in the data plane
- We presented as use case a sketch-based **DDoS detection** strategy, implementable in P4
- **Demonstrating** early prototype in a **P4 emulated environment**

- Open issues
  - P4 emulated environment -> Next step: performance evaluation in a **real testbed** (real hardware with **Tofino** chip)
  - We focused on detection → Next step: **mitigation**
  - We focused on a single switch → Next step: **network-wide strategies**

Thank you

Q&A

Email address:
damianp@man.poznan.pl

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

The scientific work is published for the realization of the international project co-financed by Polish Ministry of Science and Higher Education in the years 2019 - 2022 from financial resources of the programme entitled "PMW"; Agreement No. 5023/H2020/2019/2

Orchestrating DDoS monitoring algorithms in P4 switch

GÉANT
Networks • Services • People

The controller decides:

- What monitoring function(s) activate within observation interval
- How long a observation interval is

```
@name("ddos_destinations_monitored")
table ddos_destinations_monitored {
    key = {
        hdr.ipv4.dstAddr : lpm;
    }
    actions = {
        src_distribution_monitor;
        src_port_distribution_monitor;
        src_distribution_monitor;
        dst_port_distribution_monitor;
        total_traffic_monitor;
        NoAction;
    }
    default_action = NoAction();
}
```

Option 1) ALL MONITORING FUNCTIONS ACTIVATED PER TIME INTERVAL

Ingress packet

time interval T_1	time interval T_2
src_distribution_monitor	src_distribution_monitor
src_port_distribution_monitor	src_port_distribution_monitor
dst_port_distribution_monitor	dst_port_distribution_monitor
ip_proto_distribution_monitor	ip_proto_distribution_monitor
total_traffic_monitor	total_traffic_monitor

Egress packet

…

Packet delay to be checked

Option 2) A SINGLE MONITORING FUNCTION ACTIVATED PER TIME INTERVAL

Ingress packet

time interval T_1	time interval T_2	…	time interval T_n	time interval T_{n+1}	time interval T_{n+2}
src_distribution_monitor	src_port_distribution_monitor	…	total_traffic_monitor	src_distribution_monitor	src_port_distribution_monitor

Egress packet

DDoS monithoring algorithm workflows