Data Plane Programming and In Band Network Telemetry – GN4-3 WP6 T1

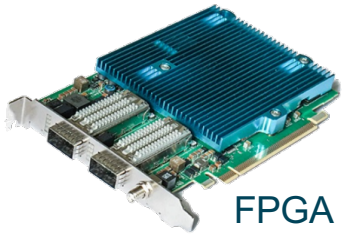
Mauro Campanella (GARR), Tomas Martinek (CESNET), Damian Parniewicz (PSNC), Federico Pederzolli (FBK),

Telemetry and Big Data Workshop
10 Nov 2020

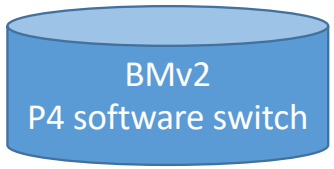
www.geant.org

Why Programming the Data Plane

- Validate the novel programmability functions and monitoring concepts implementable directly in the data plane
- Use of P4 language for FPGA and new chips (e.g. Barefoot Tofino) for improved monitoring and network functions
- Implementing prototypes for two use cases:
  - In-band Network Telemetry (P4 INT)
  - Simple DDoS Detection and Mitigation (reported by Damian Parniewicz)



FPGA



Edgecore Wedge100BF-32X

Arista 7170-32c

Tofino (Barefoot)

Network Telemetry : Streaming and In-Band

Telemetry is mostly push based in contrast to legacy pull based, e.g. SNMP

Streaming Telemetry streams (selected) data continuosly (average on finite observation windows of a few seconds) from devices and uses a standard data model usually based on YANG (see Pavle Vuletic' s presentation)

In Band Network Telemetry model exports information from the data plane directly in the data plane, by adding headers to packets.

- Instantaneous measure of queue depth (at time of packet traversal)
- Line speed (INT for all or selected packets/flows)
- Headers may be added by each node in the path, providing e2e per hop measurements
- INT is best used with data plane programming, we used P4
- Three main switch roles: INT source, transit and sink

DPP-INT achievements

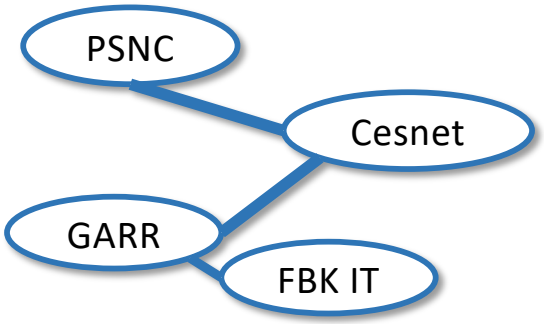
Develop P4 INT code for different platforms (BMv2, Tofino, FPGA)

Connect three labs (PSNC - Poland, CESNET - Czech Republic, FBK – Italy) over NRENs networks with public addressing

Develop basic Big Data collection and presentation environment

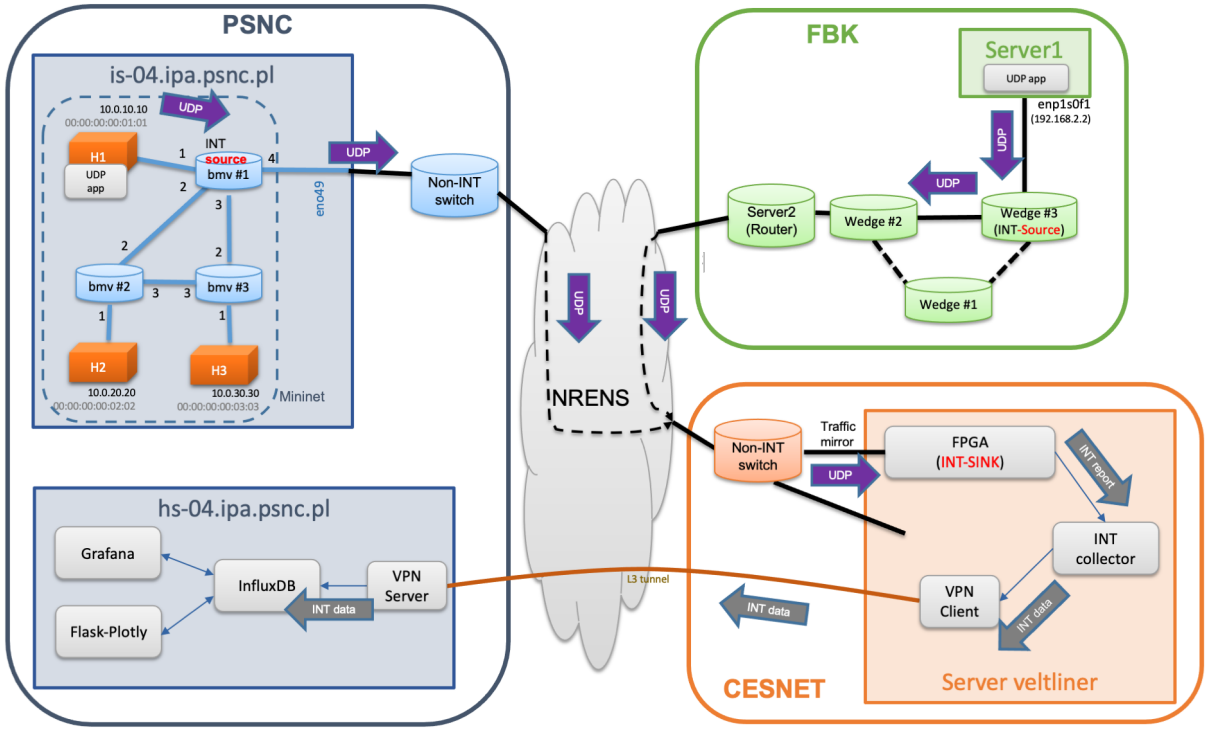
Test INT functionalities and measure UDP flow behaviour, later extend to TCP

Prepare a INT tool and knowledge-base to be used for flow and network debugging at this scale (*ongoing*)

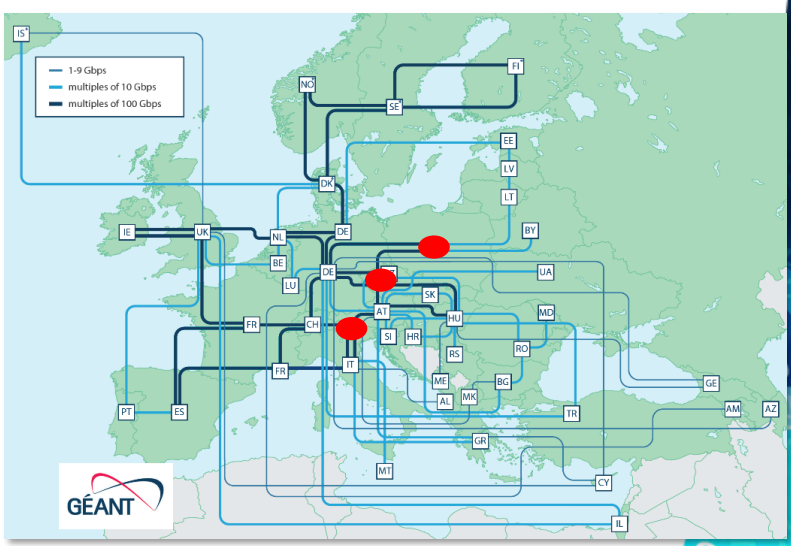


INT Testbed using production NRENs networks

INT : testing on live networks



DPP- INT distributed test bed: operational



- 3 switch types
- UDP packets flow on NRENs networks at few Kpackets/s
- Collected INT data in CESNET is sent back to PSNC for collection and presentation.

Steps, challenges and new knowledge:

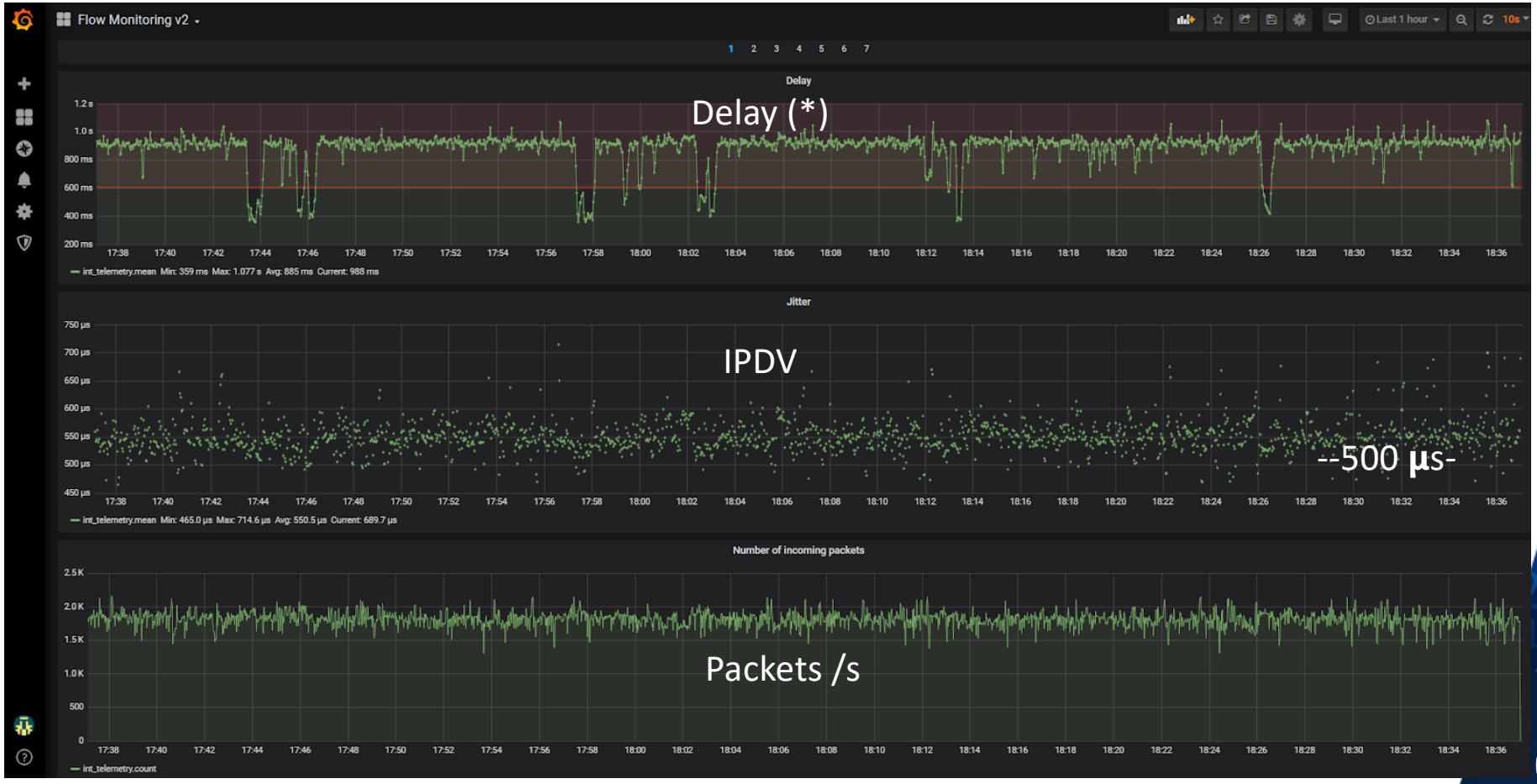
Define INT data and Header position	Using: Source and destination timestamp, seq. number, INT headers inserted before IP payload
Coding using P4	HW programming, limited cycles and memory, lack of some arithmetic operations, lower flexibility in use of registers
Collect INT data	Tuned InFluxDB (100Kpackets/s single thread)
Present	Used Grafana and added IPDV computation in data plane
Analyze	"Knowledge" under development now

Time synchronization to be tuned to sufficient precision (few microseconds?), improve node clock stability

Early Measurements – 5 seconds PSNC-CESNET, 2Kpacket/s

(*) delay is not the real value, clocks not yet synchronized

Early Measurements – 1 hour PSNC-CESNET



(*) delay is not the real value, clocks not yet synchronized

Next steps

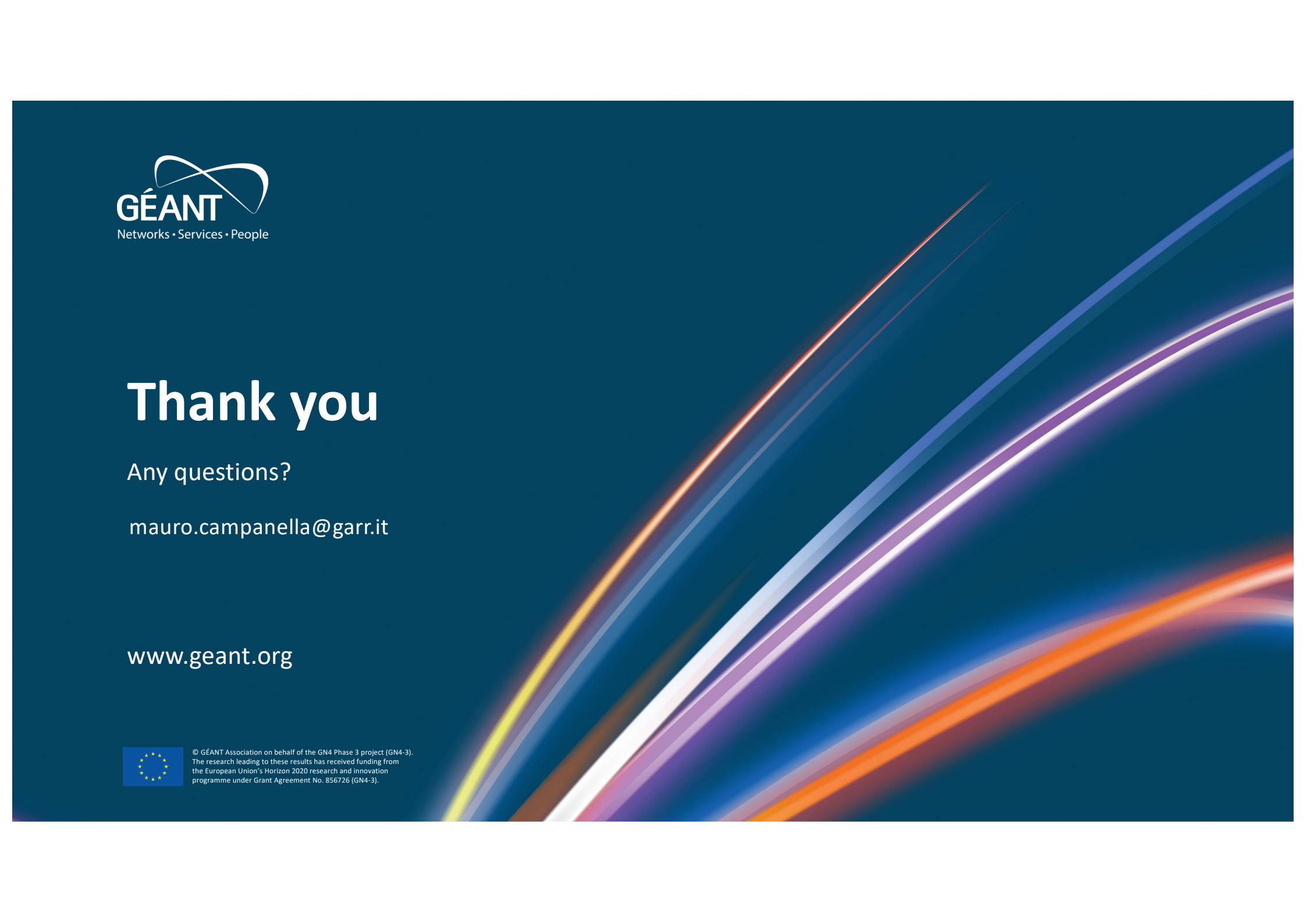
- Further test INT code for the "transit" function, code is now on project's GitLab
- Enlarge testing topology (up to transatlantic?)
- Apply INT to other use cases (debug traffic, also LAN or cloud)
- Offer INT on DPDK (almost ready) to provide an INT tool that does not require special hardware, runs in off-the-shelf computing equipment and that can have many Gigabit/s performance

And collaborate with other interested groups!

Summary

- Data Plane Programming (using P4) is not business-as-usual, requires specific expertise however is becoming available in various platforms. It works well, within chip hardware and software boundaries
- INT is a powerful magnifying glass on network behaviour
- The use of such tools requires handling of large amount of "raw" data, to be used for analytics and more. It need further insight, tools and equipment to scale

- For more information contact mauro.campanella at garr.it



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).

Learning steps - coding

P4 INT code developed for all platforms, however needs careful tuning as chip programming implies very detailed hardware specification and it is a function of HW platform

Chips have a limited number of cycles per packets, finite memory, not all high level instructions

The emulation environment (BMv2) works, however it is only the first step, program validation needs hardware testing

Network functions like threshold based DDoS can be implemented and work as expected

Code developed, now available in a GÉANT GitLab code base under Apache2 license