PROGRAMMABLE INTEGRATED TELEMETRY – RON20 RESULTS

Mauricio Solis, Paweł Maćkowiak

TNO innovation for life

MOTIVATION

- The state of today's internet:
  - The number of dependencies is growing
    - ISP with their infra
    - Part of operation is being outsourced
    - More and more services moving to cloud.

- GAIA-X [1] , SCION [2], Responsible Internet [3]

- This leaves users (both end-users and ISP in relation to each other) with:
  - Transparency (who processes the traffic, and how?)
  - Accountability (why are things performed in a certain way?)
  - Security aspects (users might wish not to use certain devices [4])

PROGRAMMABLE INTEGRATED TELEMETRY

- The first step to tackle this problem would be to give users more insight.
  - What to share?
  - How to share the data without compromising anyones interested?

- We wanted to explore the possibility of giving that insight.
  - There might be additional steps, to be taken before giving that insight such as modeling relations between users/ISP and providing that model to the user.

  - We worked with assumption that the relations between involved parties are known.

TELEMETRY TO THE RESCUE ? BUT WHAT'S NEW ? HOW?

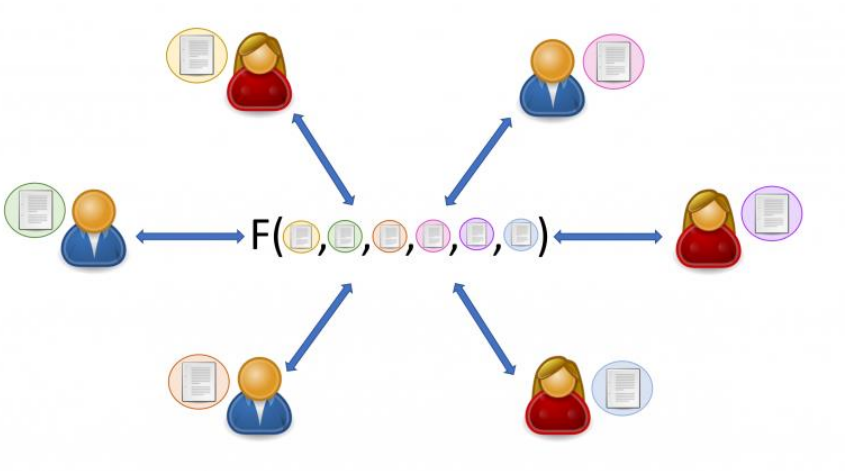
- Telemetry – obviously not new, possibility to measure at various points, lots of opensource

- Programmable Telemetry
  - Some solutions available: Barefoot INT, ONOS, previous RONs, OvS(OF/SDN), FD.io VPP
  - Measure on-demand, choose granularity, select app/traffic of interest

- Programmable Integrated Telemetry – collect app-related information from:
  - app container, VM, host NIC, switch ports ('vertical' integration) ← gathering data
  - various domains – consumer, network provider, cloud ('horizontal' integration) ← sharing data
    - certain solutions also available

PROBLEM SOLVING APPROACH

- Sharing all the data might result in unwanted effects e.g. security incidents
- P2P, Client-Server
- Proposed solution:
  - **Yao's Millionaires' problem**
  - Secure multi-party computation

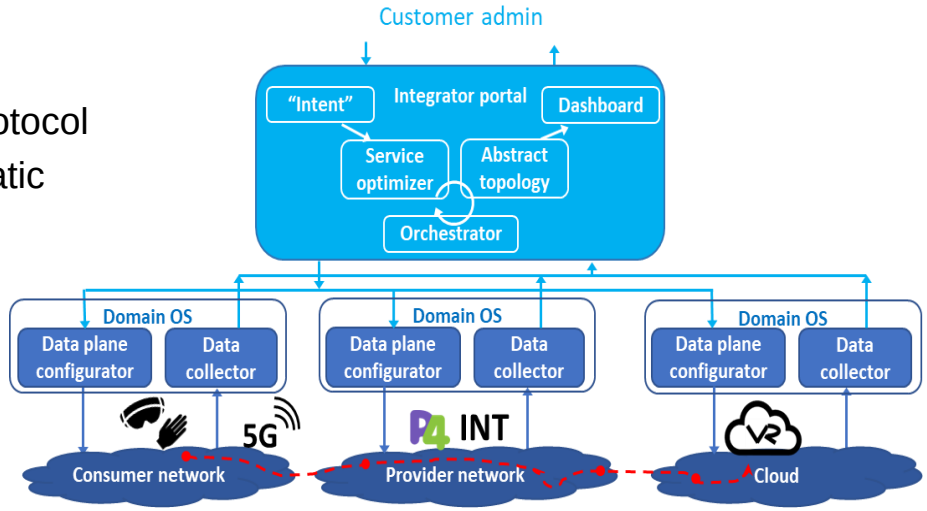


Secure Multi-Party Computation https://www.esat.kuleuven.be/cosic/blog/the-three-musketeers-of-secure-computation-mpc-fhe-and-fe/

FUTURE PROOF – TECH AGNOSTIC/MODULAR

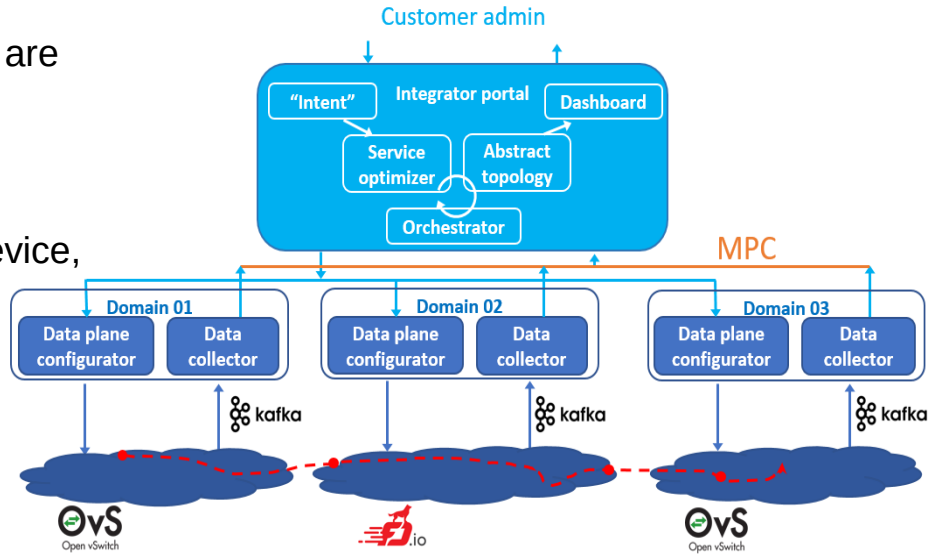
- MPC frameworks are not developed equal
  - ? General Purpose vs Specific Purpose
  - High level language describes the MPC protocol
  - ? Intent Definition Language used for automatic generation of the protocol code

- Providing the data for the protocol execution
  - Different sources, different technologies
  - Message broker
    - communication
    - minimizing the mutual awareness



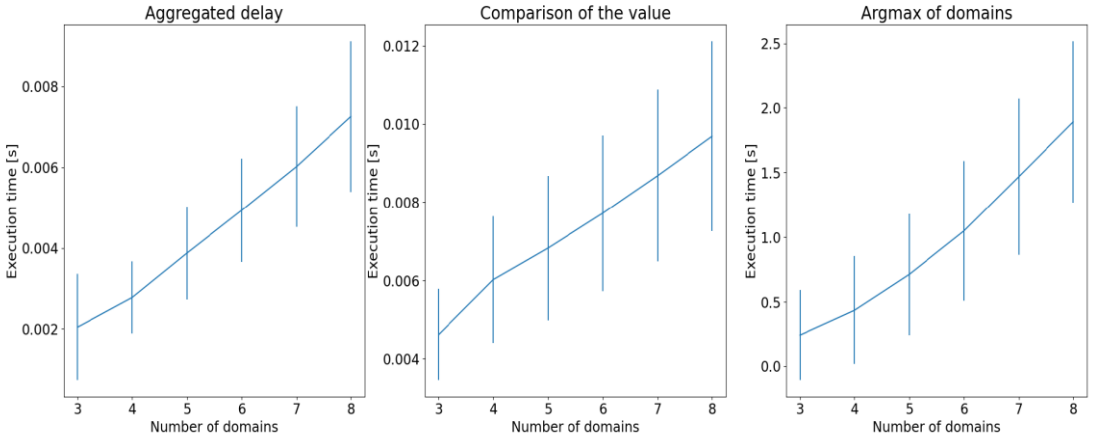
POC

- Possible functions that can be performed:
  - Check if requirements are met (what devices are used? how are they used?)
  - Share the resource usage with out reveling underlaying infra
    - Do that with different granularity (per device, per zone, per domain)
- Performed test on 3 functions:
  - Aggregated delay
  - Argmax of delays
  - Check if SLAs are met



RESULT

- Data from multiple sources
- For simple functions (aggregated delay) system can perform near real time calculations
- Protocol performance is dependent on the performance of the technology providing the data
- In PoC we used different data sources one of which was FD.io VPP

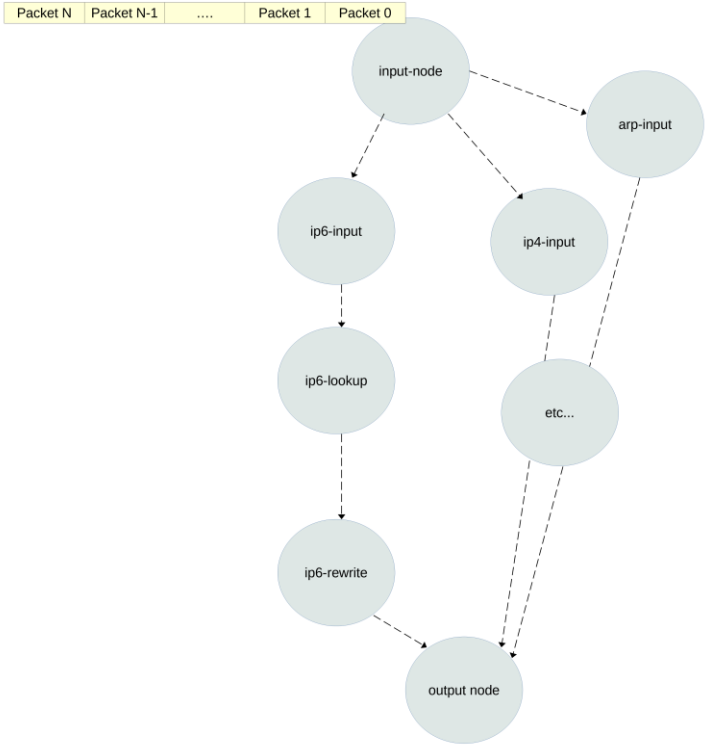


FAST DATAPLANE.IO VECTOR PACKET PROCESSOR (FD.IO VPP) OVERVIEW

- VPP:
  - Layer 2 – 4 multi-platform network stack that can run on x86, ARM, and Power architectures.
  - Several device drivers can be used: DPDK, Linux Socket, eXpress Path (XDP)
  - Processes packets, “vectors”, by performing same operations on all for faster processing.
- Telemetry
  - Added as a plugin to main core functionality
  - Internet Standard RFC 3954 for Netflow v9 and RFC 7011 for IPFIX
    - Flow based telemetry sampled from 1 second
  - In-situ Operations, Administration, and Maintenance (iOAM)
    - Modifies production packets by aggregating telemetry data onto each packet as header

FD.IO VPP OVERVIEW CONT.

- Directed acyclic graph
- “Nodes”, a.k.a, functions that perform packet processing on vector of packets.
- Idea is to perform same node operations to every packet in a vector allowing for better I-cache and D-cache performance compared to one-by-one in a stream.
- Node modualarity allows for new features to be implemented by re-directing packet to new node.



CURRENT IOAM IN VPP

- IETF Drafts
  - Data Fields for In-situ OAM
    - What, type of telemetry data
  - Encapsulations for In-situ OAM Data
    - How, to transport
  - In-situ OAM IPv6 Options
    - IPv6 Extension header used
    - Hop-by-hop header specifications
- Limitations
  - Telemetry data is not very useful
  - Uses outdated or expired drafts

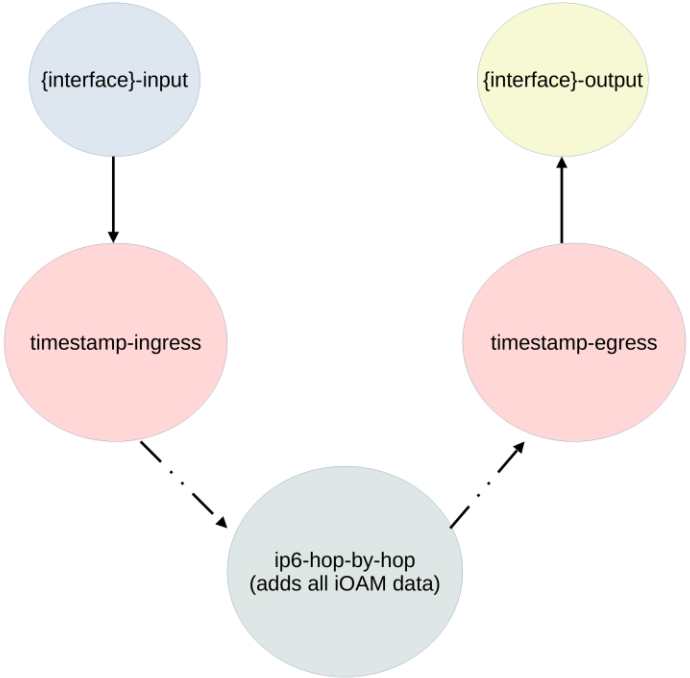
Octet Offset	0								1								2								3							
Bits:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Next Header								Header Extension Length (As 8-octets)								Option Type (59 for Telemetry Trace Data)								Option Length (As octets)							
4	Trace Type								Number of iOAM Nodes								Reserved															
8	Hop Limit (TTL)								Node Identifier																							
12	Ingress Interface Identifier																Egress Interface Identifier															
16	Timestamp																															
20	Application Specific Data																															

CONTRIBUTION TO IOAM IN VPP – UPDATED IOAM

Octet Offset	0								1								2								3							
Bits:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Next Header								Header Extension Length (As 8-octets)								Padding															
4	Option Type (59 for Telemetry Trace Data)								Option Length (As octets)								Reserved								Telemetry Type (Pre-allocated)							
8	Namespace Identifier																Node Length, Flags, Remaining Length															
12	Trace Type																															
16	Hop Limit (TTL)								Node Identifier Short																							
20	Ingress Interface Identifier – Short																Egress Interface Identifier – Short															
24	Timestamp (sec)																															
28	Timestamp (sub-sec)																															
32	Hop / Transit Delay																															
36	Application Specific Data – Short																															
40	Queue Depth																															
44	Checksum Complement																															
48	Hop Limit (TTL)								Node Identifier – Wide																							
52	Node Identifier cont. – Wide																															
56	Ingress Interface Identifier – Wide																															
60	Egress Interface Identifier – Wide																															
64	Application Specific Data – Wide																															
68	Application Specific Data cont. – Wide																															
72	Buffer Occupancy																															
76	Opaque Data Length								Opaque Data Scheme Identifier																							
… … … 332	Variable Length Opaque Data																															

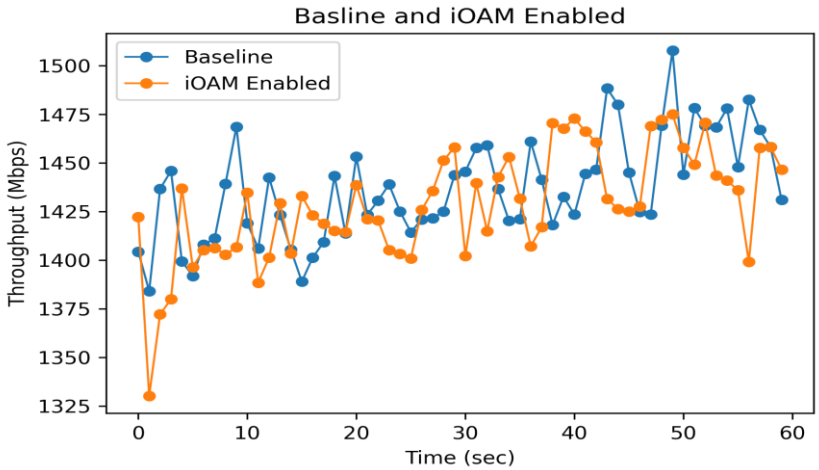
IMPLEMENTATION – HOP DELAY

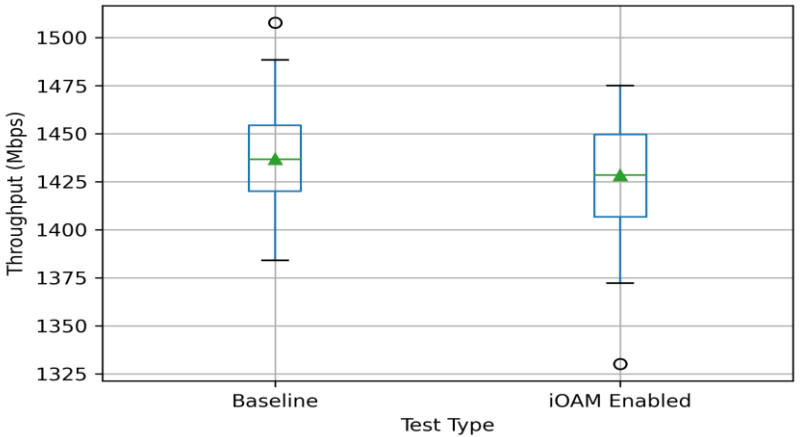
- Issue:
  - No current way for timestamping packets right after ingress and before egress
- Approach
  - New plugin feature required
    - Timestamp ingress using POSIX time
    - At egress, take another stamp, calculate the hop delay, insert into packet



VALIDATION

- iPer3
  - 1518 Byte Payloads
  - **Y-axis: Iteration of test**
  - **X-axis: Throughput[Mbps]**
- No significant changes in regards to throughput
  - Average 1.5Gbps





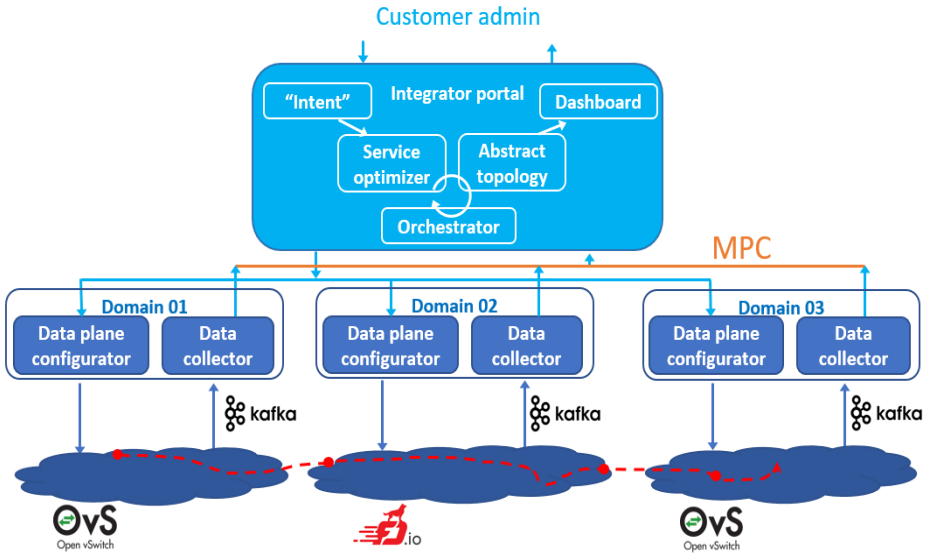
SUMMARY
NEXT STEPS

- General:
  - Further develop ‘vertical’ integration
  - Further develop on-demand capabilities
  - Further evaluate performance to gain more insight

- FD.io VPP
  - Bug fixes and VPP contribution to community

DEMO

- The initialization of the domains
  - SDN telemetry
  - FD.io VPP telemetry
- The data is exported to message broker
- MPC computation
  - The data is private
  - Only the output is revealed




THANK YOU FOR YOUR ATTENTION
Take a look:
TNO.NL/TNO-INSIGHTS
TNO innovation for life