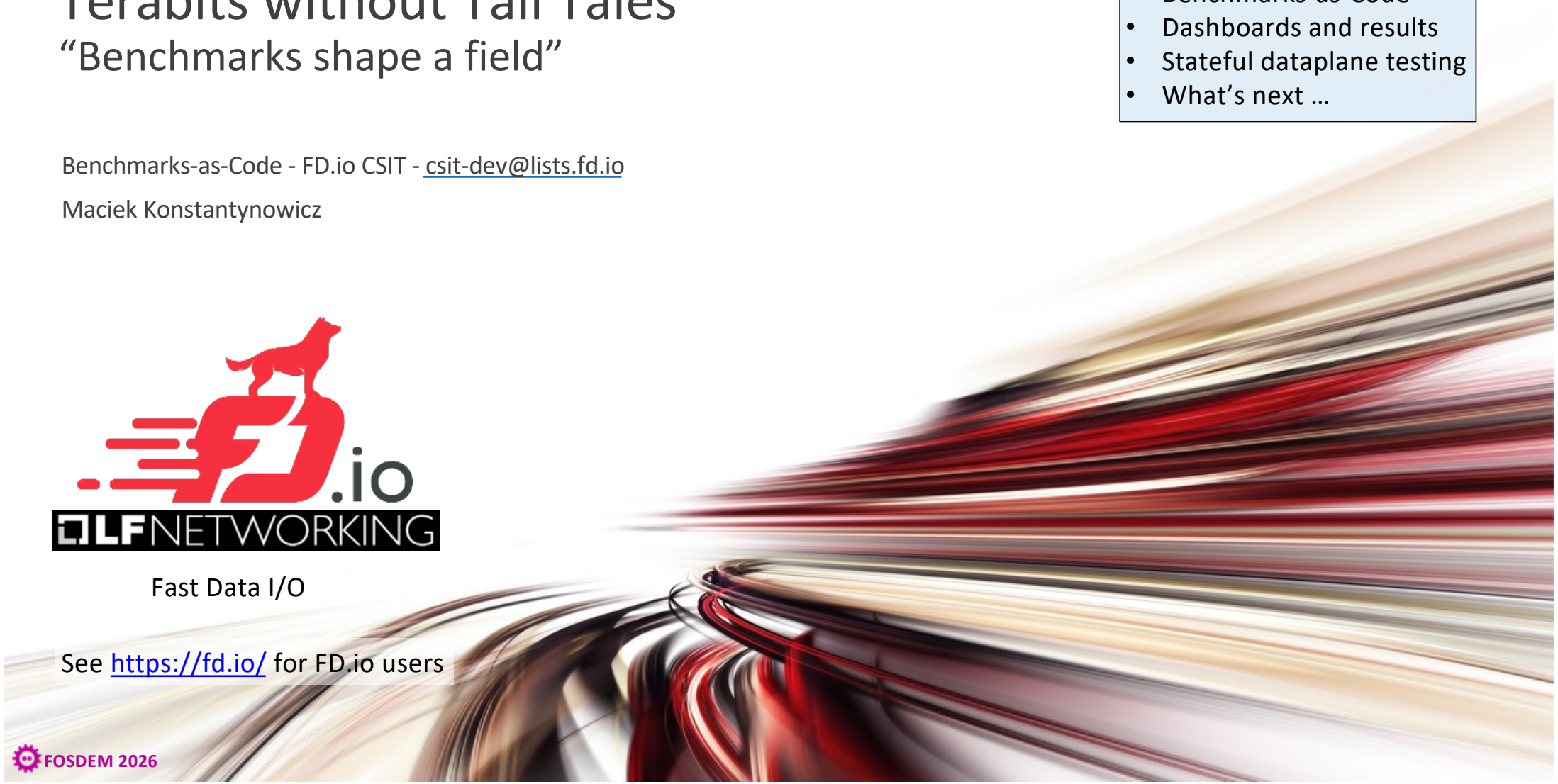
Terabits without Tall Tales

“Benchmarks shape a field”

- FD.io VPP and CSIT
- Benchmarks-as-Code
- Dashboards and results
- Stateful dataplane testing
- What’s next …

Benchmarks-as-Code - FD.io CSIT - csit-dev@lists.fd.io

Maciek Konstantynowicz



Fast Data I/O

See https://fd.io/ for FD.io users

FD.io Core Projects: VPP and CSIT

VPP
Vector Packet Processing

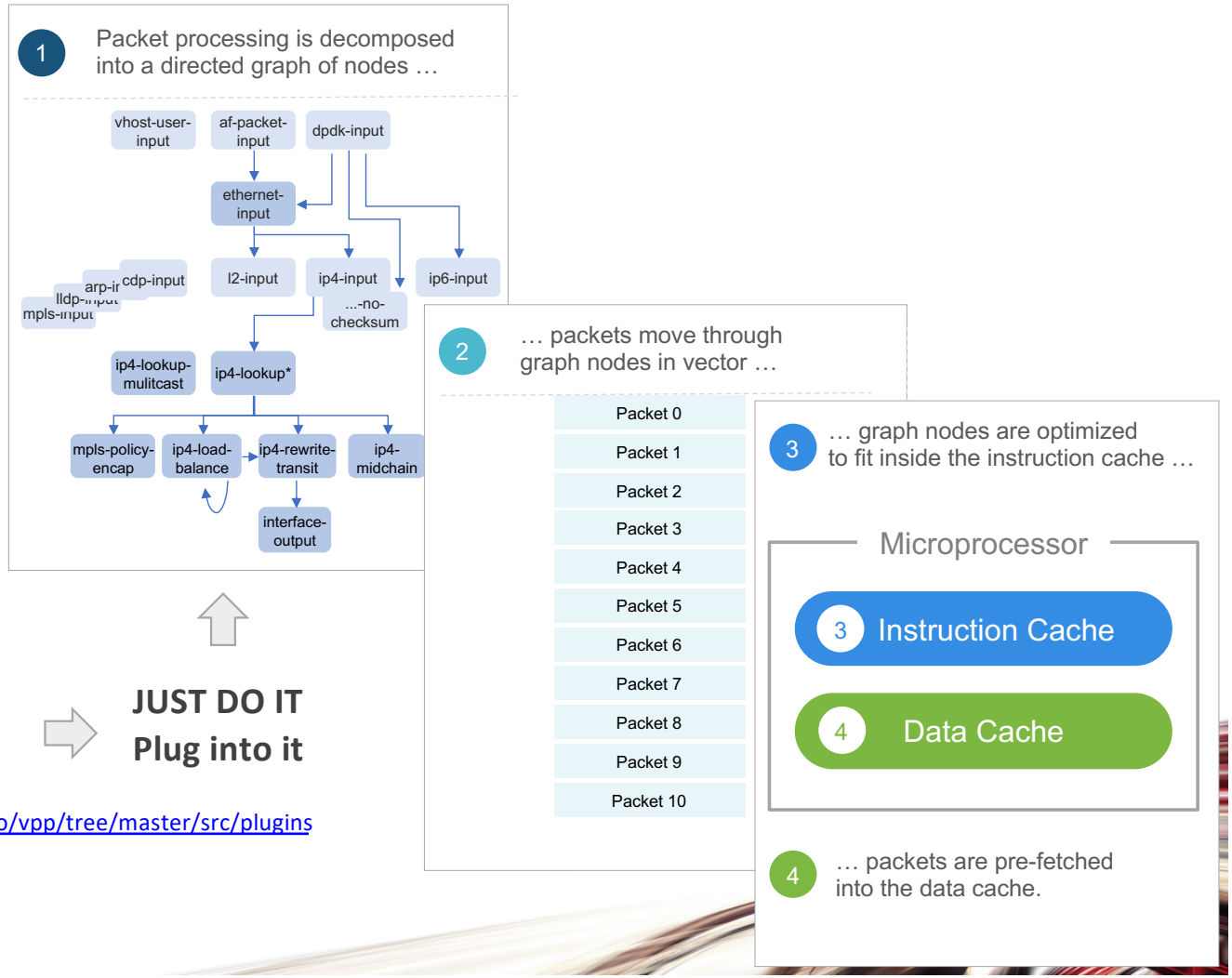
Vector Processing:
Graph-node engine processes packets in vectors to maximize CPU data and instruction cache locality and SIMD efficiency.

Terabit scale with linear scaling:
Designed for multi-core scale-out; throughput grows near-linearly with cores when I/O isn't the bottleneck.

Feature rich data-plane:
Broad L2-L4 feature set (routing/switching, NAT, ACLs, SRv6, IPsec, tunnels, TCP/UDP/QUIC) with high-performance plugins.

Hardware optimized:
High-speed offload options (e.g., crypto, mem-copy) and fast physical and memory I/O interfaces for x86/Arm platforms.

Modular, extensible design:
Plugin-first architecture; add/replace features as nodes in the graph without rewriting the core.



https://github.com/FDio/vpp/tree/master/src/plugins

FD.io Core Projects: VPP and CSIT

CSIT
Continuous System Integration & Testing

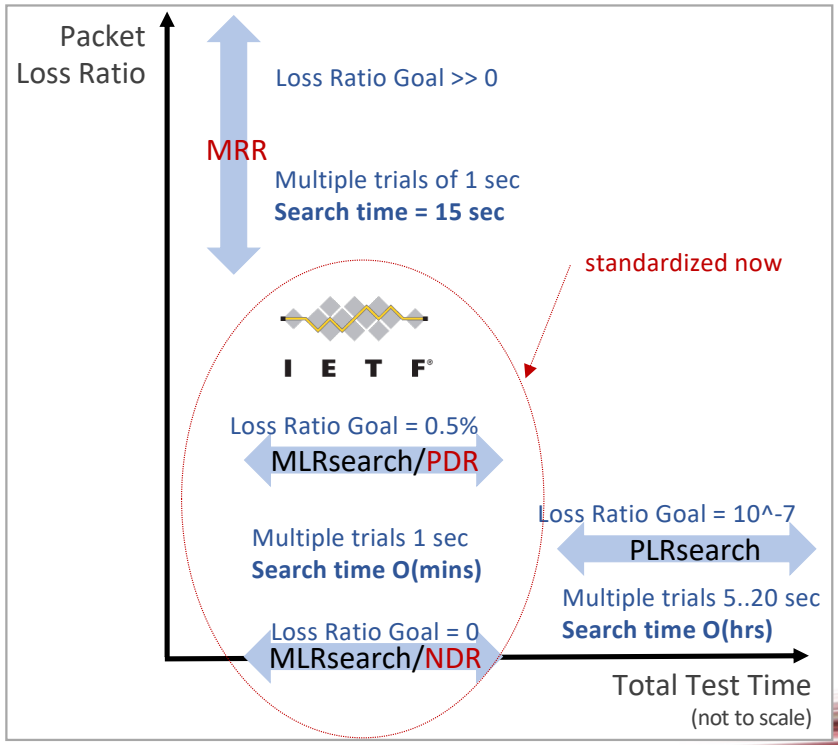
Benchmarking-as-code quality gates:
Automated pipeline with reproducible tests, configs, topologies, and artifacts; run continuously to guard VPP dataplane quality and prevent regressions.

CI-optimized methodology:
MLRsearch* (Multiple Loss Ratio search) to make throughput and loss benchmarks fast enough for CI - not hours-long RFC2544-style runs.

Meaningful SW networking metrics:
Discovers NDR (0% loss), PDR (e.g., 0.5% loss), MRR, plus latency and soak metrics aligned with real data-plane scenarios.

Data-driven trending & analytics:
Publishes structured artifacts and interactive dashboards that visualize history - making long-term trends, anomalies, and “what changed?” instantly visible.

Repeatable results at scale across labs:
Hundreds of test cases (VPP + DPDK) run on diverse physical testbeds (Xeon/Atom/Arm) with accuracy bounds and noise-awareness for reliable regression detection.



* https://datatracker.ietf.org/doc/draft-ietf-bmwg-mlrsearch/ - in RFC publication queue.

What's Benchmarked …

COTS CPUs and DPUs Systems

	Processor Family	NICs
Intel Xeons	Intel Xeon **EMR** 8568Y+ (on-chip HW accelerators: QAT, DSA) 2S Server	e810-2p100GE
	Intel Xeon **ICX** 8358 2S Server	xxv710-2p25GE, e810-2p100GE, cx7-2p200GE
	Intel Xeon **SPR** 8462Y+ (on-chip HW accelerators: QAT, DSA) 2S Server	e810-2p100GE, cx7-2p200GE
Modern Arm	Nvidia **Grace** C1 (Neoverse V2 cores) 1S Server	cx5-2p100GE, cx7-2p200GbE
	Marvell **Octeon10** CN106xx (Neoverse N2 cores) DPU	CN106 GSERM0 2p100GbE
	ALT Ampere Altra Q80-30 (Neoverse N1 cores) 1S Server	xl710-2p40GE, cx6-2p200GE
Older Systems	AMD EPYC **Zen2** 7532 2S Server	xxv710-2p25GE, cx5-2p100GE
	Intel Atom **SNR** P5362B (on-chip HW accelerators: QAT) 1S Server	e810-4p25GE

https://csit.fd.io/cdocs/infrastructure/fdio_dc_testbed_specifications/

Benchmark Tests

- L2 Ethernet Switching
- IPv4, IPv6 Routing
- IPsec, Wireguard with IPv4 Routing
- SRv6 Routing
- Features: ACLs, NAT44, policing, …
- IPv4, IPv6 Tunnels
- Hoststack: TCP, UDP, QUIC
- KVM VMs vHost-user
- Docker Container Memif
- Drivers: DPDK, AVF, RDMA, AF_XDP, TAP/GSO

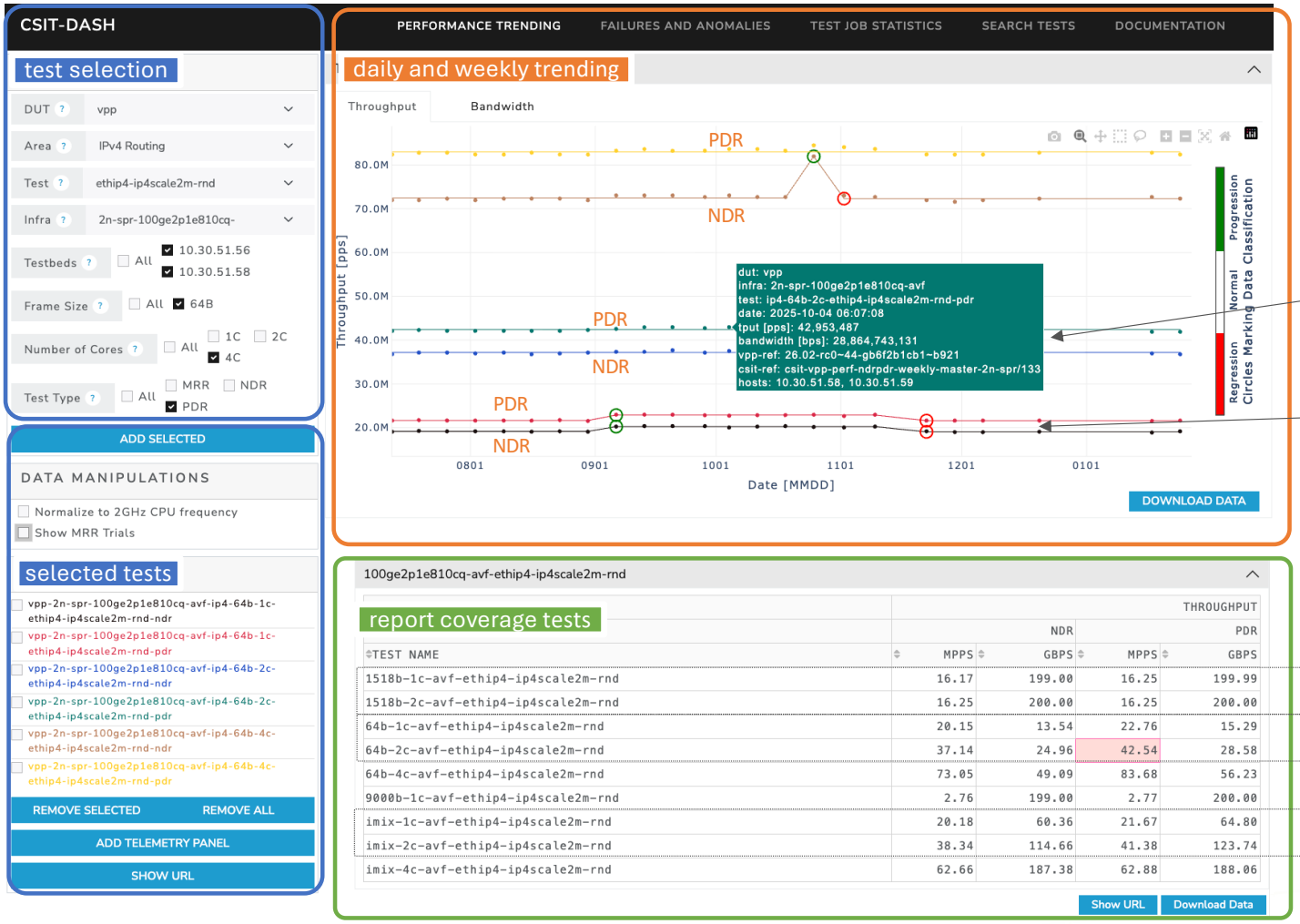
Test Methodologies

- Packet Throughput and Latency
- Stateful NAT44ed
- Stateful Host-stack
- Speedup Multi-Core
- Soak Tests
- Reconfiguration Tests

https://github.com/FDio/csit/tree/master/tests/vpp/perf

* Intel Xeon GNR (P-core) and SRF (E-core) servers – in the LFN FD.io racks already …

2n-spr – IPv4 scale2m-rnd – 1, 2, 4 core



PDR – discovered Partial Drop Rate with 0.5% loss.
NDR – discovered Non Drop Rate with zero loss.

@64B

2 hw-core
PDR **42** Mpps
NDR 37 Mpps

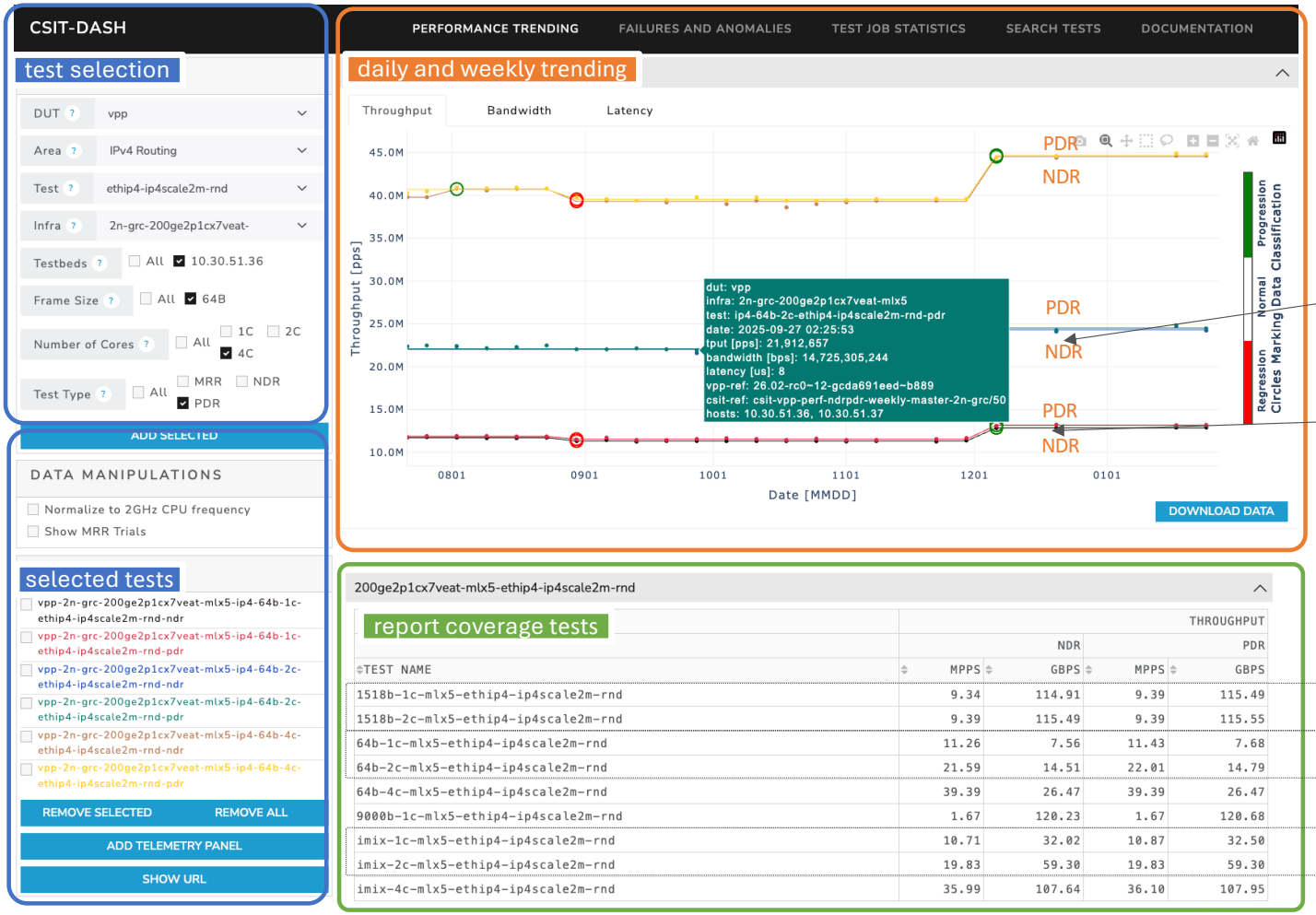
1 hw-core
PDR **22** Mpps
NDR 20 Mpps

	1 hw-core PDR		2 hw-core PDR	
	Mpps	Gbps	Mpps	Gbps
@1518B	16	200	16	200
@64B	22	15	42	28
@IMIX	21	64	41	123

https://csit.fd.io/trending/#eNrlVcuOwjAM_JruZWWp8YYtFw7L8h8opIZW6sMkoavu15MiJIMEEgd6yiHPsT3xaKT40DvaempW2WKdFesMi7qMU_b18xmXgRmwA88OVJ4fCFnRUuX2CGbYQ80avvUOlAUK1XSKw1vTELbguhK60k218HeqVZ7CXWFBuBoFeUon8caRkYTlKVAgf8Py8FUSvHemJV__k2TEdgS3URuBlL2nCSPfoNdWi8Ol4t1CcjpC8juExCQcifM7EpNwJM7vSJ2EI_X8jtRJOFK_6MjF5qPrXXv5vePeV_0fBFebxl-vztUmIv4

https://csit.fd.io/coverage/#eNpVjEEOwiAQRU-DG4MZUIKbLqy9hyE42iaEThnE9PaWuEA3k_z_3h_GgD5Pc-yE7YW2KfB2xfGy_wZtFNRCX2txf-VGC9EfKZga1OagQKbtu2NsGo3rjxMlU5IK4ImaFJ4V-EW68mi-S-jaYKJTDXbY8Ti_b8FljH7thOmFGT5sjTZM

2n-grc – IPv4 scale2m-rnd – 1, 2, 4 core



PDR – discovered Partial Drop Rate with 0.5% loss.
NDR – discovered Non Drop Rate with zero loss.

@64B

2 hw-core
PDR **22** Mpps
NDR 21 Mpps

1 hw-core
PDR **11** Mpps
NDR 11 Mpps

200ge2p1cx7veat-mlx5-ethip4-ip4scale2m-rnd

report coverage tests

TEST NAME	THROUGHPUT NDR MPPS	NDR GBPS	PDR MPPS	PDR GBPS
1518b-1c-mlx5-ethip4-ip4scale2m-rnd	9.34	114.91	9.39	115.49
1518b-2c-mlx5-ethip4-ip4scale2m-rnd	9.39	115.49	9.39	115.55
64b-1c-mlx5-ethip4-ip4scale2m-rnd	11.26	7.56	11.43	7.68
64b-2c-mlx5-ethip4-ip4scale2m-rnd	21.59	14.51	22.01	14.79
64b-4c-mlx5-ethip4-ip4scale2m-rnd	39.39	26.47	39.39	26.47
9000b-1c-mlx5-ethip4-ip4scale2m-rnd	1.67	120.23	1.67	120.68
imix-1c-mlx5-ethip4-ip4scale2m-rnd	10.71	32.02	10.87	32.50
imix-2c-mlx5-ethip4-ip4scale2m-rnd	19.83	59.30	19.83	59.30
imix-4c-mlx5-ethip4-ip4scale2m-rnd	35.99	107.64	36.10	107.95

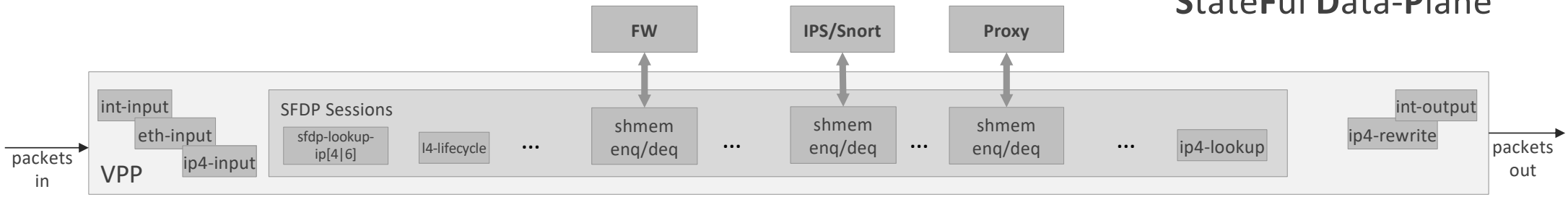
	1 hw-core PDR		2 hw-core PDR	
	Mpps	Gbps	Mpps	Gbps
@1518B	9	115	9	115
@64B	11	7	22	14
@IMIX	10	32	19	59

https://csit.fd.io/trending/#eNrlVslqwzAQ_Rr1UsbYYyk-9dDU_1FUeRobvAhJTZN8faRQmBhKT6EYdNA2b9bHO8iHxdG7p_FFqL1o9gKboYubqF-f43G0FnCGgzOAZXkgtJU5NUfSAabxpGCwEnbyAyoDFPr0issbPRJO4OYO5s6lbPiWsnVfYZWaEdufGfmjIEdoR5pDYlWGAvm7Or_2xc6fTk_khwtxRByIcRP5Yagy6zLhbO_Q1bD9wl2oxGtVFnVZqKqod8muWtG0N9_Hs25zYt3-M-uYidZxU1rHTLSOm9K6zETrclNal5loXT5a66p9mhc33f4y8e775RuCG_Tof0xXHltIjg

https://csit.fd.io/coverage/#eNpVjMEOwiAQRL8GLwYDq4RTD9b-hyG4aZsg3SyI7d9b4gG9TDLzZiZhQJ_nJXbC9gIsh7SrOF-PXwNGqxrArQaPV260EP2RgtwgmJNWkvd3l7DVaNp-OlGO7CUoNSKQ9qst6LJ8htW0hWN0bTLTpRo7HNK0vO_BZYx-64TphRk-_QE3bQ

FD.io VPP: Introducing Stateful Sessions => **SFDP**

State**F**ul **D**ata-**P**lane



VALUE

Simpler stateful design (multi-tenant by default)
- Easier config, ops, debugging
- More consistent behavior across services

Higher efficiency + scale for L4–L7 services
- Removes redundant packet/session processing
- Better CPU/memory use as services grow

Foundation for dynamic stateful service chaining
- Pluggable multi-service chaining per tenant
- Efficient metadata sharing + fast verdict execution
- Easier per-tenant session correlation/analytics

CAPABILITIES

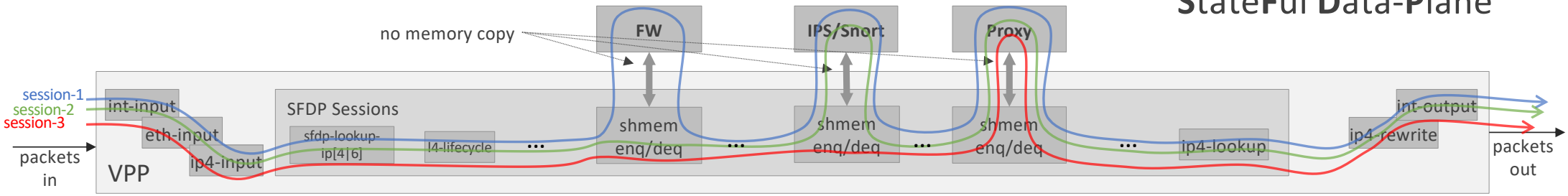
Shared, multi-tenant session table (platform primitive)
- Eliminates per-service duplicate state
- Session lifecycle: start/end timestamps
- Session-linked logging
- Rich stats + metadata

Coherent (session, service) metadata plane
- Chain-wide metadata distribution (no per-hop hacks)
- Fast, robust packet + session lookup
- Ultra-low latency path to services (zero-copy)

L4–L7 security verdict fast-path
- Permit / block / redirect / chain update
- Early drop + inline L3–L4 protections
- Plan: early SYN-flood handling

FD.io VPP: Introducing Stateful Sessions => **SFDP**

State**F**ul **D**ata-**P**lane



MAIN CONCEPTS

Multi-tenant (multi-context) sessions

- Every packet is mapped to a **session** (existing or new)
- **6-tuple key** = 5-tuple + **context-id** (alternative key formats possible)
- **Context-id** from encapsulation / attachment (VLAN, VXLAN/GNV VNI, Int-id, ...)

Hierarchical service chaining

- A service chain is an ordered list of services to traverse
- Two directions: **Forward** (initiator -> responder) and **Reverse** (responder -> initiator)
- **Per-context**: configure two chains (forward and reverse)
- **Per-session**: session inherits both session chains at creation
- **Per-packet**: packet selects the active chain (forward or reverse) once session is known

Dynamic service chaining

- Services can **modify the remaining chain** (insert / skip / redirect)
- Done via the **SFDP API** (optimized for fast-path updates)

Blog: https://medium.com/fd-io-vpp/a-stateful-data-plane-for-vpp-introducing-sfdp-8badd254640d

EXISTING SFDP SERVICES

```
acl
dot1q
geneve
reass
interface_input
l4-lifecycle
nat
sample
tcp-check
snort
```

source: https://github.com/FDio/vpp/tree/master/src/plugins/sfdp_services

docs: https://github.com/FDio/vpp/tree/master/docs/developer/corefeatures

FD.io VPP: SFDP Preliminary Results ...

Testing SFDP with stateful traffic generators:

- FD.io CSIT labs with TRex – Nx10GbE
- Intel labs executing CSIT tests with Ixia – Nx100GbE

First benchmarks indicate capability in the order of millions CPS on a single Xeon core (~3 Mcps on EMR 8568Y+).

test logs: https://logs.fd.io/vex-yul-rot-jenkins-1/csit-vpp-perf-verify-2n-emr/21360296627/log.html.gz

latest console output: https://github.com/FDio/csit/actions/runs/21511261884/job/61978615412

FD.io CSIT: Benchmarking areas we want to go after …

Continued development of stateless packet tests:

⇒ K8s calico-vpp integration

⇒ built-in CPU/DPU HW accelerators: crypto, mem-copy, TM/QoS, …

Adding stateful session routing and chaining:

⇒ baselining of SFDP

⇒ services over SFDP

⇒ IDS/IPS (e.g., with Snort) tests

Adding new HW, next-gen Xeons / AMDs / Arms / DPUs

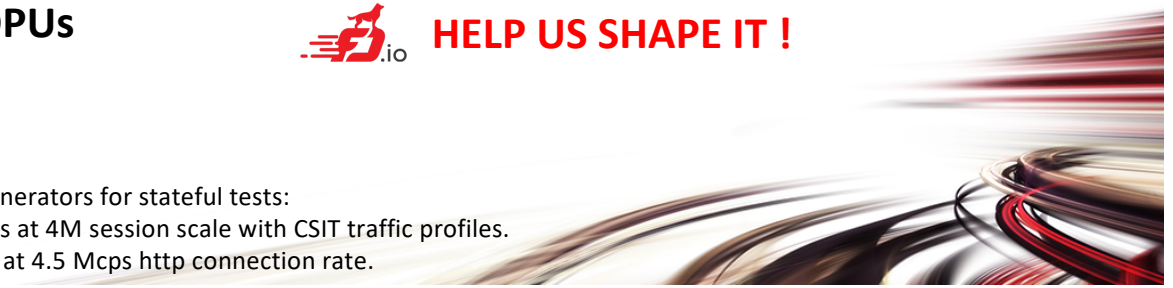
Looking for contributors interested in chipping in …

- Traffic generators – stateless, stateful*
- Deployment use cases
- Benchmarking design and implementation

And on the VPP SFDP front …

- Adding new services e.g., Suricata
- Extending the framework

HELP US SHAPE IT !



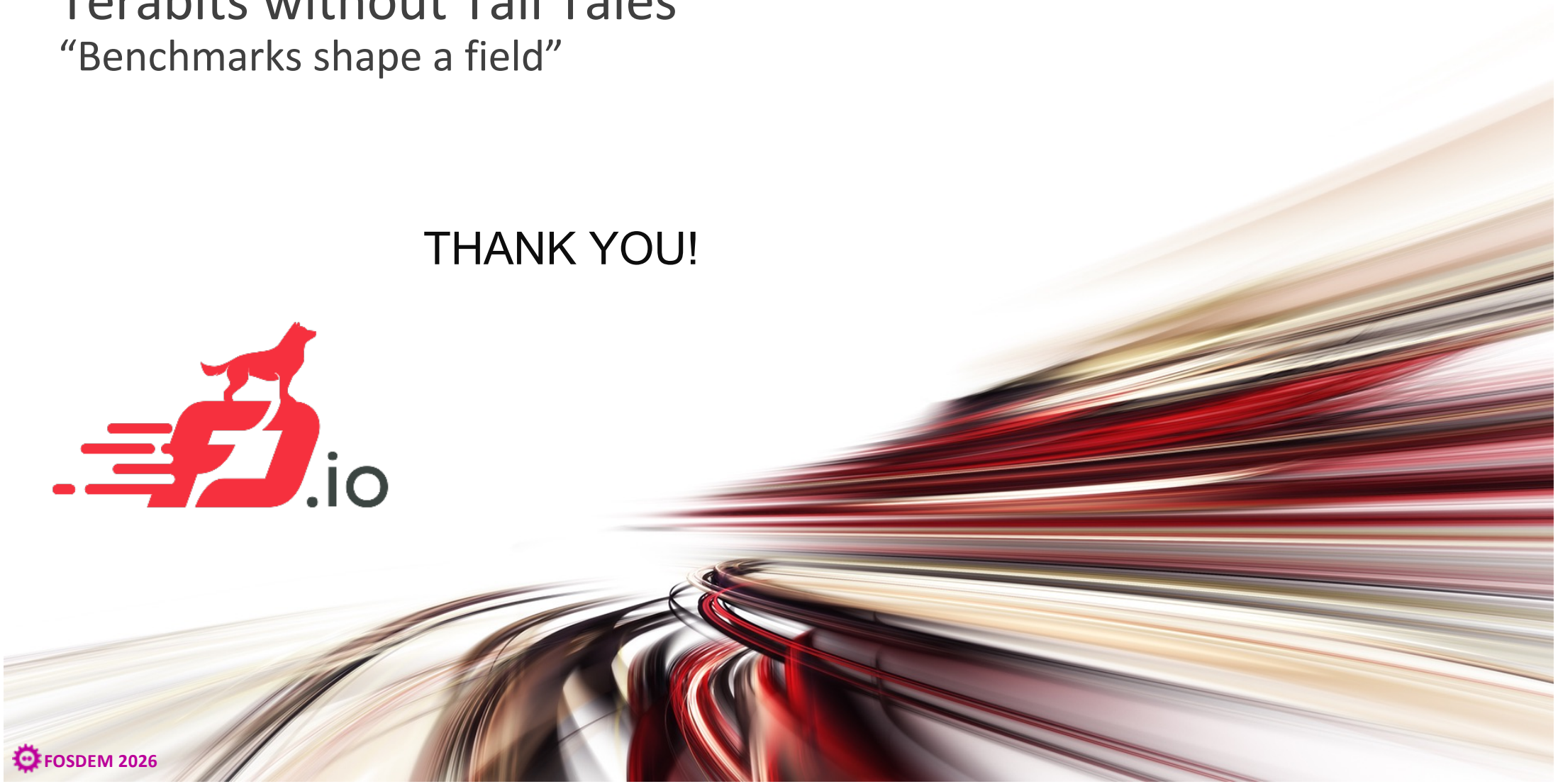
FOSDEM 2026

* Facing limitations of traffic generators for stateful tests:
  - TRex maxing out at 1.9 Mcps at 4M session scale with CSIT traffic profiles.
  - Ixia CloudStorm maxing out at 4.5 Mcps http connection rate.

Terabits without Tall Tales

"Benchmarks shape a field"

THANK YOU!



CSIT Resources

- **Project**
  - Wiki pages: https://wiki.fd.io/view/CSIT
  - Meetings: https://wiki.fd.io/view/CSIT/Meeting
  - Mailing list: csit-dev@lists.fd.io
- **CDash**
  - Dashboard: https://csit.fd.io
- **Source Code**
  - Git repo: https://git.fd.io/csit
  - Github mirror: https://github.com/FDio/csit
  - Gerrit reviews: https://gerrit.fd.io
- **Standalone libraries**
  - Speeding up binary search using shorter measurements: https://pypi.org/project/MLRsearch/
  - Locating changes in time series by grouping results: https://pypi.org/project/jumpavg/

CSIT Resources

- **Technical Papers**
  - SPR 2Tbps IPsec (2023)
    - https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-high-performance-ipsec-with-4th-gen-intel-xeon-scalable-processor-technology-guide
  - “Benchmarking Software Data Planes Intel® Xeon® Skylake vs. Broadwell” (2019)
    - https://www.lfnetworking.org/wp-content/uploads/sites/55/2019/03/benchmarking_sw_data_planes_skx_bdx_mar07_2019.pdf
  - “Benchmarking and Analysis of Software Data Planes” (2017)
    - https://fd.io/docs/whitepapers/performance_analysis_sw_data_planes_dec21_2017.pdf
- **Technology Demonstrator Video Clips**
  - “VPP: A Terabit Secure Network Data-plane” (Intel Xeon Icelake 07-APR-2021)
    - https://www.youtube.com/watch?v=ipQQmjzE_g0
  - “FD.io: A Universal Terabit Network Dataplane" (Intel Xeon Skylake, 11-JUL-2017)
    - https://www.youtube.com/watch?v=aLJ0XLeV3V4
- **FD.io Presentations**
  - https://wiki.fd.io/view/Presentations
- **Other FD.io Materials**
  - https://fd.io/
  - https://fd.io/news/whitepapers/