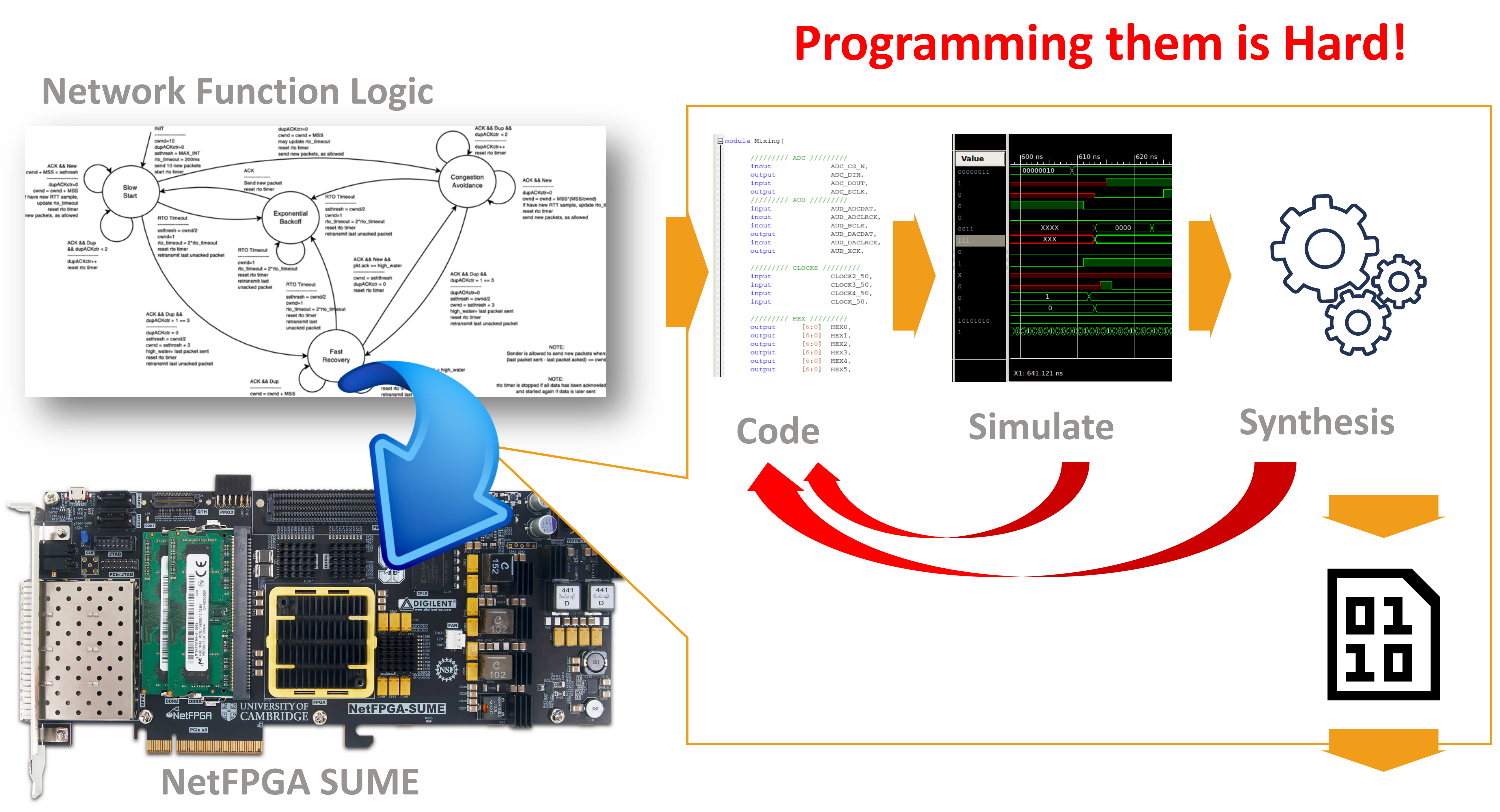
TOR VERGATA UNIVERSITY OF ROME

cnit consorzio nazionale interuniversitario per le telecomunicazioni

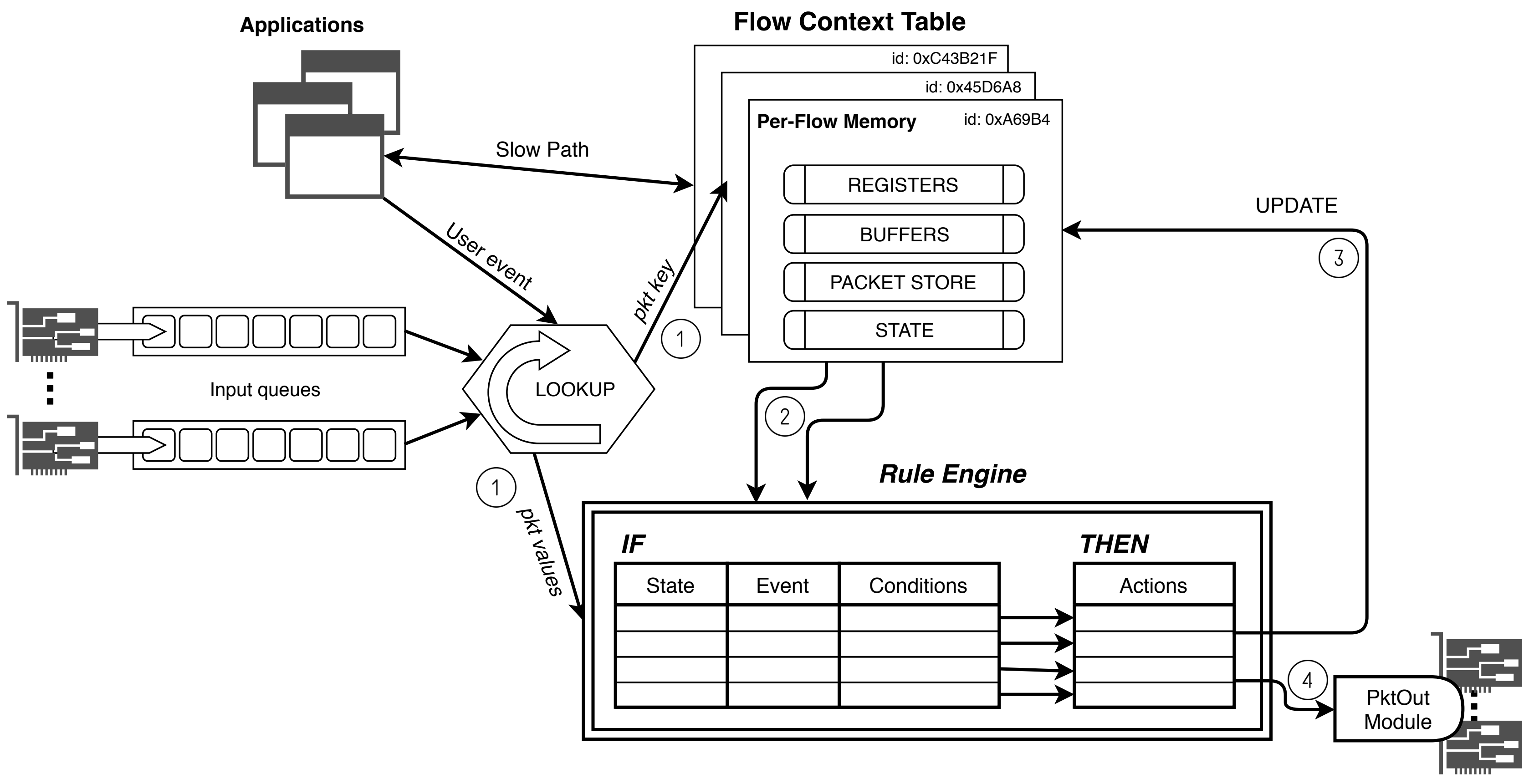
Offloading MapReduce tasks in Stateful Programmable Dataplanes: feasibility analysis and preliminary insights

Valerio Bruschi, Marco Faltelli, Angelo Tulumello, Salvatore Pontarelli, Giuseppe Bianchi

1. Offloading to the FPGA NIC improves performance



2. FlowBlaze abstraction [1]



1. From each packet FlowBlaze produces a **key-values pair**
2. The **pkt-key** identifies the flow-context associated to the packet *(and retrieves the partial results computed previously)*
3. The **pkt-values** and the partial results identifies a set of actions to perform
4. If needed, FlowBlaze can generate packets that act as triggers.

3. DSL for Network Function (…and beyond?)

if part

State == "wait"	**Event** == "pktRcvd"	**Conditions** == (A > 7) && (Reg1 == 2)

then part

Action(s):
TX(pktB)
setTimer(B, +10ms)
Pkt C -> HashTable
	NextState
	=
	"active"
		Update(s):
		Reg1 += 4
		Hash(C) ->
		Reg2

FlowBlaze works by implementing multiple (eXtended)FSMs in hardware.

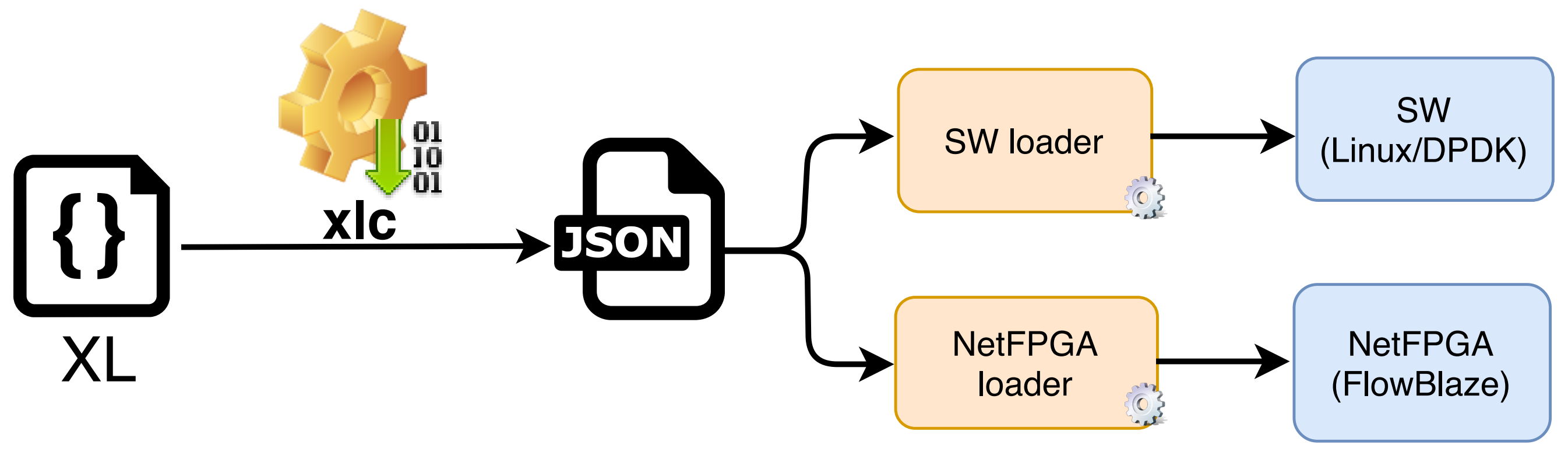
XFSM can be modeled a set of ***"if-then"* clauses.**

the **XFSM Lang (XL)** is a Domain Specific Language to describe XFSMs

However, FlowBlaze is not only an accelerator for network functions.

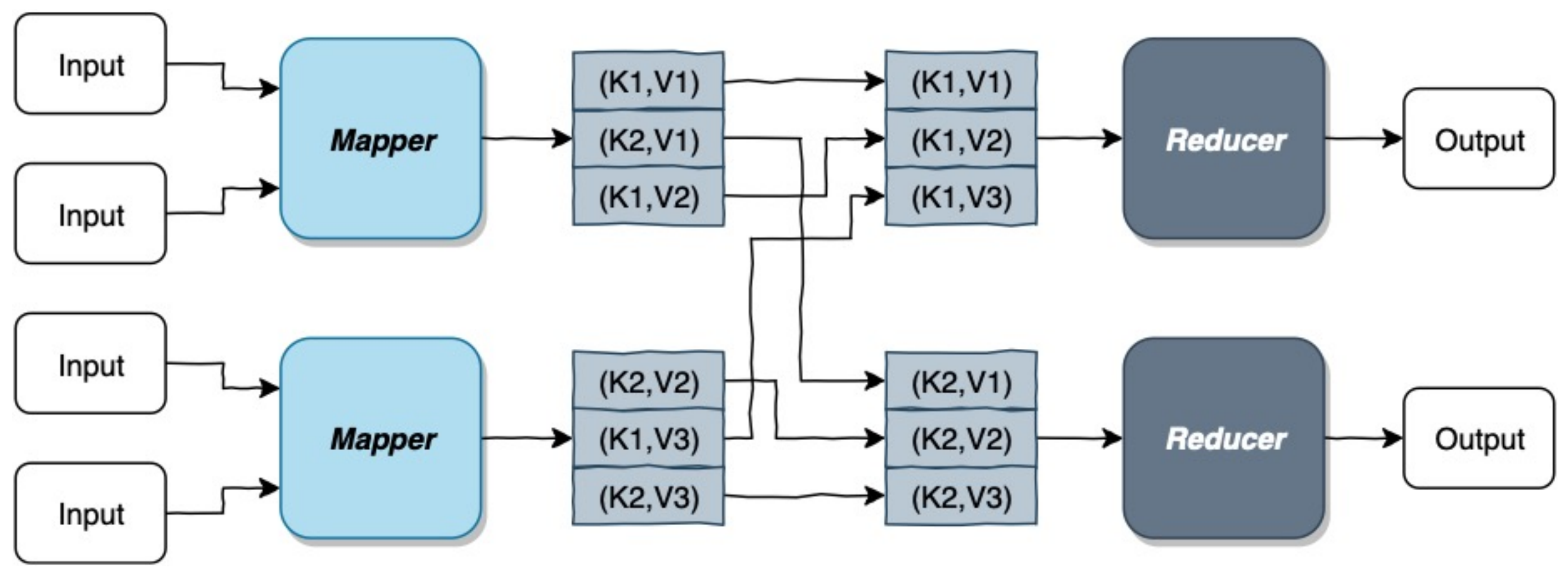
Because we discover that is more general than expected,
(it can offload general stateful applications, **if modeled by XFSMs**).

4. XFSM Lang portability



The main advantage of using XL is to have a **platform independent portable code**, which can be deployed on every platform capable of executing XFSMs

5. MapReduce



MapReduce is composed of two phases:

- a ***Map*** phase that performs **filtering** and **sorting**
- a ***Reduce*** phase that **groups** key and **performs aggregation**

These steps are separate and distinct!
they are distributed on several nodes in the cluster.

The overall data processing:

1. is performed in **parallel**
2. Is separated by **data transfers** between nodes.

6. Offloading online MapReduce jobs

The Velocity dimension (as one of the Vs used to define Big Data) brings **many new challenges** that are not addressed yet, such as **real-time processing** on fast and **continuously** generated data:

- The traditional MapReduce paradigm is not an appropriate solution for **low-latency processing**! (because typically it uses a batch-based approach that start and finish on contents that do not change during the processing)

Offloading MapReduce tasks on programmable network devices can alleviate and address these issues:

- Programmable dataplane are usually implemented on **hardware platforms** providing **flexible, line-rate** data processing capabilities.

FlowBlaze shares many similarities with the MapReduce paradigm and results as a good candidate to offload its tasks:

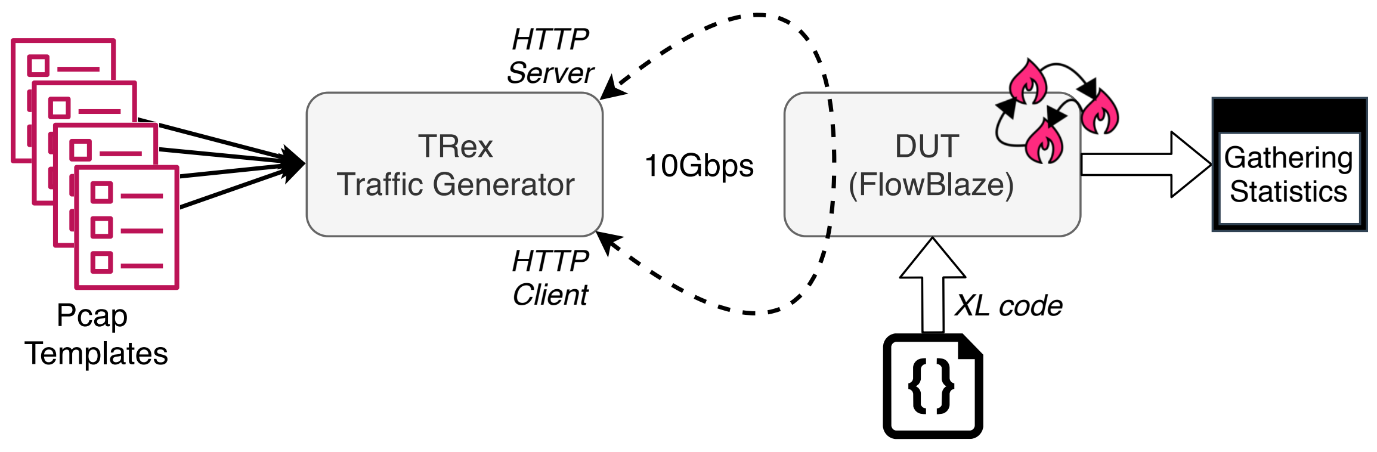
- the Map phase produces a set of intermediates key-values pairs from the input, as FlowBlaze does from the packet (①②).
- the Reduce phase aggregates all the intermediate values, as FlowBlaze does for each packet within the same flow (②③④).

7. Experiments (preliminary insights)

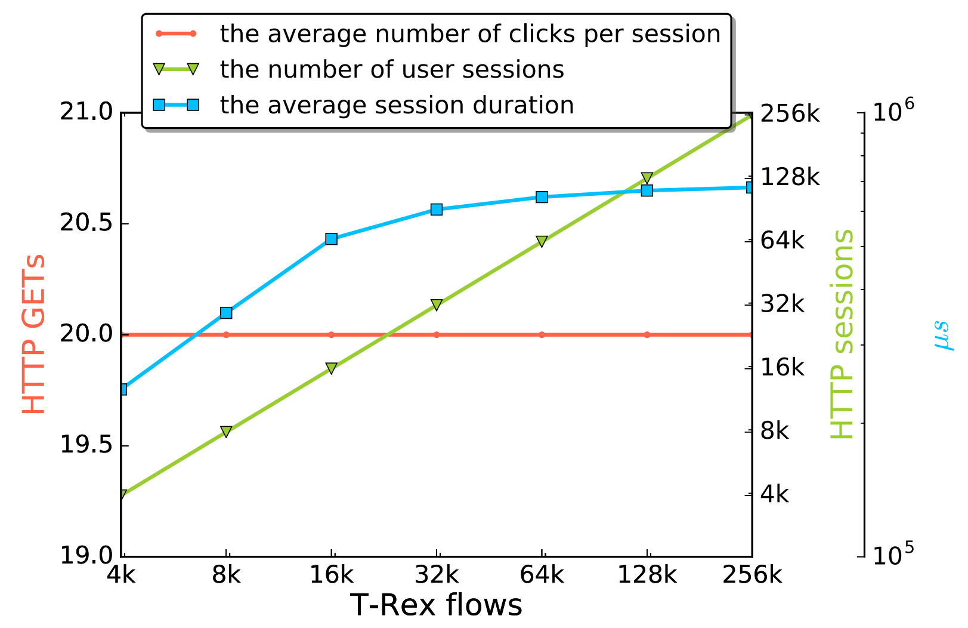
We developed a simple **click-stream analysis over HTTP traffic** from the literature[2] to validate the approach proposed.

The MapReduce task should compute three different metrics through the overall HTTP traffic:

- *the number of **user sessions**:*
  - *group by 5-tuple and count*
- *the average **number of clicks per session**:*
  - *for each session group by HTTP.GET and count*
- *the average **session duration**:*
  - *for each session group by Server.id and average session_time*



The result of our experiment campaign shows that the MapReduce task offloaded in the FlowBlaze abstraction is able to compute the targeted metrics!



- *the average number of clicks per session **remains constant** while the load varies*
- *the number of user sessions **increases as expected** with the number of flows generated by T-Rex*
- *the average session duration is only affected by Trex transmission buffers due to an increased number of flows per second.*

In our experiment we did not experience any packet losses by the DUT, even if the traffic generator saturated the link speed (It is worth noticing that with packet losses, the DUT would be not able to detect specific event and this would affect the final metric computation)

[1] Pontarelli, Salvatore, et al. "FlowBlaze: Stateful Packet Processing in Hardware." NSDI. 2019.
[2] Logothetis, Dionysios, et al. "In-situ MapReduce for Log Processing." 2011 USENIX Annual Technical Conference (USENIX ATC'11).