PROGRAMMABLE NETWORKS

The P4 Programming Language

Open-source language to specify packet processing logic

- Official website: https://p4.org

P4 goals

- Target independence
- Protocol independence

P4 supports multiple platforms

- Software platform
  - DPDK, t4p4s
- Network Processing Unit (NPU)
  - Netronome SmartNIC

Field-Programmable Gate Array

- Intel Tofino

Application-specific IC (ASIC)

- Intel Tofino

P4 Event-based P4 Architectures

- P4 event architecture for t4p4s [4], similar to [2] for SUME NetFPGA
- Two pulling-based queues are processed in every iteration
- DPDK timers are used to trigger timing-based events
- Performance variables to investigate:
  - Timer update frequency
  - Cost of creating and checking events
  - Interference between packets and events

State Management in P4

- State Keeping is essential for many applications, possible via:
  - Registers: limited functionality, may be fragmented
  - Tables: key matching, but only updatable by controller (before)

- Table Update Approaches

  - Digest: introduces a sleep of 1 s
  - Change method: close to original implementation, but avoid detour
  - Pointer method: directly changes entries using their pointers

- Integrated updatable tables entries using $\_ref$ annotation into t4p4s [3]

Dynamic Network Functions - Idea

- Trusted parties can deliver code for network functions
- These will be applied to all further traffic of the flow on-the-fly
- E.g., for In-Network-Computing

Libmoon/Lua Prototype

- Libmoon [1] is used for implementing a prototype
- LuaJIT uses just-in-time compilation for a high performance
- HasTable for function matching
- loadstring() to compile new functions

Median latencies: 322 $\mu$s, 26.5 $\mu$s, and 65.3 $\mu$s

- Updates are possible at line-rate, nearly as fast as lookups

Dynamic Network Functions - Evaluation

- Using one thread/task
- Using two threads/tasks


