Design, Implementation and Evaluation of IPv4/IPv6 Longest Prefix Match support in Multi-Architecture Programmable Dataplanes

Fabricio E Rodriguez Cesen , Christian Rodolfo Esteve Rothenberg

Department of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computer Engineering (FEEC)
University of Campinas (Unicamp)
CEP 13083-852 – Campinas, SP, Brazil

{frodri,chesteve}@dca.fee.unicamp.br

Abstract – With Programming Protocol-Independent Packet Processors (P4) gaining traction to define how the information is processed through the pipeline of programmable datapaths, and OpenDataPlane (ODP) to create an open-source, cross-platform set of Application Programming Interfaces (APIs) for the networking data plane. Multi-Architecture Compiler System for Abstract Dataplanes (MACSAD) is an approach that converges P4 and ODP in a conventional compilation process, archiving portability of dataplane applications without compromising the target performance capabilities. This work aims at adding IPv4/IPv6 Longest Prefix Match (LPM) support to MACSAD. The proposed LPM support combines a lookup algorithm and the ODP library with MACSAD table support to create a complete forwarding base. We develop a new ODP Helper library for IPv6 lookups based on the current IPv4 solution and we evaluate its performance for diverse workloads and target configurations.

Keywords – P4, Software Defined Networking, Performance analysis, Programmable networks

1. Introduction

With the increasing amount of data being transmitted through the Internet every second, the requirements to the network are increasing exponentially, and it is necessary to evaluate the actual capabilities and how to manage these traffic efficiently. A flexible, re-designable and configurable network is needed, which ensures the possibility to change the features of the dataplane.

Software Defined Networking (SDN) [3] is an emerging network architecture where the network splits into control and forwarding plane. The first communications standard interface defined between the control and forwarding layers was OpenFlow. However it has some limitations, for example, it needs to know the types of the headers of the packets, which makes it difficult to implement new protocols and headers. Therefore, to enable programming the forwarding chip to support new protocols, it is proposed a new protocol with independent programming abstractions, such Programming Protocol-Independent Packet Processors (P4) [7].

P4 is an open source language for expressing how are processed the packets by the pipeline of a network forwarding element. It is base on a Match+Action forwarding model, and it works together with SDN protocols. With SDN new protocols to program the routing devices have emerged, as well as new data plan hardware and languages, such P4. To provides an open-source, cross-platform set of Application Programming Interfaces (APIs) for the networking data plane technologies as OpenDataPlane (ODP) [5] appeared.

With P4 and ODP working together, it is possible to determine and program dataplanes beyond multiple targets with a common compiler system. Multi-Architecture Compiler System for Abstract Dataplanes (MACSAD) [8, 9, 10] is an approach to converge P4 and ODP through a common compilation process delivering portability of dataplane applications without compromising target performance improvements, translating P4-defined dataplanes into high-level ODP APIs.

The MACSAD implementation allows performing a basic Layer 2 forwarding. However, it limits the use-cases with the switch and the capabilities to performs a Layer 3 packet forwarding. The current limitations of IP lookup support in MACSAD are the following ones:

- **Limited support of IPv4 lookup**. There is no IPv4 lookup algorithm implementation featuring the ODP APIs.
- **No IPv6 support in ODP**. ODP helper library provides support for IPv4 lookup forwarding. However, there is not an actual implementation of IPv6 lookup provided by ODP.
- **Performance evaluation**. An evaluation of the Switch (MACSAD) with the complete IPv4/Ipv6 lookup process is indispensable.

To address the identified issues, the main objective of this work is to design, implement and evaluate the Longest Prefix Match (LPM)

IPv4/IPv6 support in MACSAD. To this end, the following specific objectives are proposed:

The rest of this text contains the following topics. Literature review, with background details and related works are covered in Section 2. Architecture proposal for the design and implementation of IPv4/IPv6 Longest Prefix Match (LPM) support for MACSAD is presented in Section 3. The performance evaluation results are covered in Section 4. Finally, conclusion and future works are discussed in Section 5.

2. Background and Related Work

This section defines three main concepts of our research work: P4 expressing how packets are processed ; ODP as the API for the networking dataplane; and finally MACSAD.

P4 is an innovation providing an abstract model suitable for programming the network data plane [7]. A P4-enabled device is protocol independent. It delineates the packet headers and specify the packet parsing and processing behaviors.

ODP project is a networking dataplane API specification. It allows application developers to write and implement dataplane applications. It can leverage portability and multi-platform support, besides of the use of specific hardware acceleration capabilities. ODP is at a higher abstraction level than Data Plane Development Kit (DPDK)[1] and Netmap[2] , and can use their user-space fast packet processing I/O to improve performance.

ODP provides a helper libraryas an extended library supporting table management such as hash table, and IP lookup (IPv4-only). On the contrary DPDK offers various fully optimized table management libraries similar to ODP including IPv6 support. An application developed using ODP APIs needs to be linked to the *ODP implementation* block (see Fig. 1) in ODP software stack of the target platform.

MACSAD is an approach to converge P4 and ODP. MACSAD architecture overview is shown in Figure 2. It has three main modules: (i) Auxiliary Frontend, (ii) Auxiliary Backend and (iii) Core Compiler.

[1] http://dpdk.org/

[2] http://info.iet.unipi.it/~luigi/netmap/

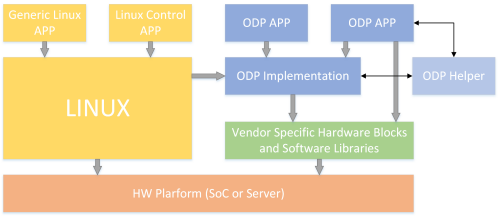

Figure 1. ODP Architecture. Source: [5]

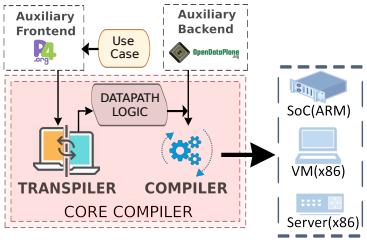

Figure 2. MACSAD Architecture. Source: [10]

- **Auxiliary Frontend** Creates the Intermediate Representation (IR) for the Core Compiler, based on the P4 code as an input. The p4-hlir project is used to translates the P4 programs into a High Level IR (HLIR).
- **Auxiliary Backend** Is used to give a common Software Development Kit (SDK) for the compiler incorporating the ODP API.
- **Core Compiler** Encompasses the Transpiler and Compiler internal modules. The *Transpiler* determines the lookup mechanism, the size, and type of tables that are going to be created. The *Compiler* takes in Transpiler output and compile along with ODP APIs provided by the Auxiliary Backend to create the MACSAD Switch (MACS) (MACSAD compiled binary code is referred to as MACS) for the desired target.

While there is potentially large amount of literature on IPv6 implementations, we briefly emphasize on related programmable dataplane activities similar to our proposed IPv4/IPv6 LPM solutions. OpenvSwitch (OVS) [6] is the traditional widely adopted OpenFlow based software switch with support for DPDK packet I/O for higher performance. PISCES [4] is developed by extending OVS bringing in high-level Domain Specific Language (DSL) such as P4 support to achieve programmability. But it is limited by the restricted OVS pipeline abstractions and only supports IPv4. Similarly T4P4S [2], the closest alternative to MACSAD, is also a software switch which maps P4 abstractions to DPDK with the help of a Hardware Abstraction Layer. However, it is not multi-platform, and DPDK's LPM library for IPv6 is not supported.

3. Layer-3 forwarding (IPv4/IPv6) Implementation

This section presents LPM prototype support in MACSAD in terms of use cases, namely, Layer-3 forwarding with IPv4 (L3-IPv4) and IPv6 (L3-IPv6). The implementation and partial results were presented at [12].

These use cases are implemented with support of ODP Helper library for LPM lookup mechanism where 32-bit and 128-bit keys are used for IPv4 and IPv6 address lookup. The P4 pipeline consists of two tables; IP lookup is performed on the first table along with corresponding actions of standard L3 packet processing (i.e., MAC re-writing, TTL/Hop Limit decrement). Then, the final packet update happens at egress section by the second table, which changes the source MAC address before sending out the packet.

- **L3-IPv4** IPv4 lookup algorithm in MACSAD uses a binary tree to perform the prefix lookup. We chose this root prefix to be 16-bit netmask. The binary tree has three levels (16-8-8) with a worst case scenario of 3 memory accesses for each IPv4 lookup.
- **L3-IPv6** Under this use case, we developed a new ODP Helper library module based on the available IPv4 solution. It is similar to DPDK[3] solution with 15 levels of tables. The 1^{st} level is of 16-bit followed by 14 additional levels of 8-bit each).

4. Performance Evaluation

We evaluate performance for the two LPM use cases using three different packet I/O engines (DPDK, Netmap, Socket_mmap). For each combination, we explore the scalability for different workloads (packet traces, table entries) and configuration options (e.g., CPU cores) using Network Function Performance Analyzer (NFPA) [1] as a benchmarking tool. To generate the traces we developed a packet crafter tool BB-Gen [11, 13] that will provide the necessary PCAP files to be used with NFPA.

4.1. Testbed and Methodology

Our testbed contains two Lenovo ThinkServer RD640 servers with Intel Xeon E5-2620v2, 6 Cores, Hyper-Threading disabled, running at 2.1GHz, 8*8GB DDR3, a dual-port NIC (10G), and run with Ubuntu Linux 16.04 LTS (kernel 4.4). The Tester server runs NFPA, and connected to the Device Under Test (DUT). The MACS is configured to forward packets received from one port to the other and eventually back towards NFPA, which in turn analyzes the packet throughput concerning packets per second (pps) and bits per second (bps).

For both L3-IPv4 and L3-IPv6, different number of cores (1, 2, 4, and 6) were allocated to the DUT, distinctive workloads were configured by setting different number of IP prefixes (100, 1K, 10K, 100K, 1M) in the lookup table and a matching number of L3 flows in the synthetic traces were used.

L3-IPv4. Figure 3 (Left Side) shows the performance of L3-IPv4 for different Forwarding Information Base (FIB) sizes and packet I/O drivers. The red y axes labels refer the line rate for different packet sizes. (i.e., 8.44 Mpps for 128 bytes and 4.52 Mpps for 256 bytes). The results for L3-IPv4 is grouped into three sectors indicating different packet sizes (i.e., 64, 128, 256). Each sector is further divided into five different points marking the complexity of the pipeline, i.e., the size of the FIB (100, 1K, etc.). It can be observed that MACS with DPDK reaches the line rate with packets sizes of 256 bytes regardless of the FIB table size. The performance of Netmap is comparatively lower but it reaches line rate with 512B packets. Also it is interesting to note that, the measured results for 1M FIB entries are better than for 100K FIB entries. From the results, it is clear that the Linux Socket_mmap driver never saturates the 10G interfaces due to the highly increased number of system calls, fundamental kernel scheduling, costly context switching, etc. imposed by the Linux kernel itself.

L3-IPv6. Results for the L3-IPv6 use case with 1 CPU core are shown on the right side of Fig. 3. The performance results lead to a conclusion similar to L3-IPv4 where DPDK reaches line rate with 256 byte packets for all FIB sizes. There are some performance differences in case of Netmap driver, a slight drop as the number of FIB entries grows, and what is more, when the FIB size reaches 1M the line rate is not achieved even with the biggest packet size. However, when comparing our results to L3-IPv4, we must point out that the peculiarity with 100K and 1M number of entries observed before also applies for L3-IPv6. From the IPv4 and

[3] http://dpdk.org/doc/guides-16.04/prog_guide/lpm6_lib.html

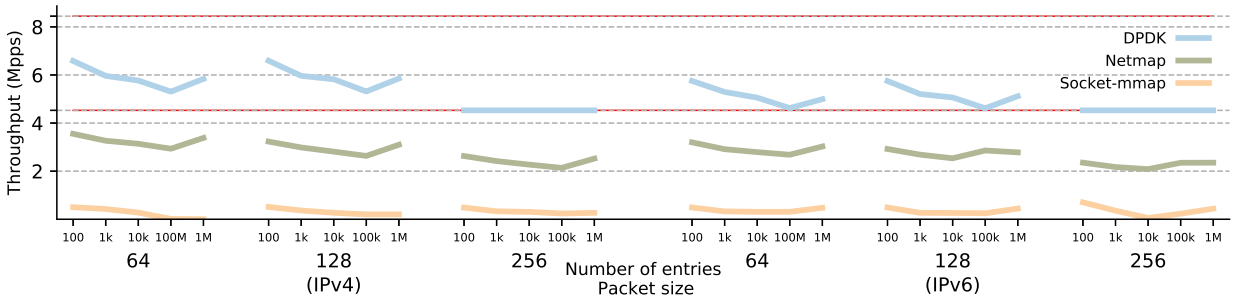

Figure 3. IPv4/IPv6 forwarding performance for different I/O drivers (1 CPU core). Source: [12]

IPv6 results, it is clear that with small packets (i.e., 64 and 128) when the FIB number of entries increases, the performance slightly reduce.

5. Conclusions and Future Work

We contributed with the addition of use cases to MACSAD, confirming the ability of MACSAD to offer (re-)configurable SDN dataplanes supporting different pipelines while confirming to portability and performance as well. We demonstrated the performance of our LPM implementations by running MACS over different packet I/O drivers (DPDK, Netmap, Socket_mmap). With this work, we accomplished some open source contributions. The developed IPv6 lookup library will be suggested for adoption by the ODP community. We developed a new packet crafter tool (BB-Gen) that natively creates packets for different standard and custom protocols. Furthermore, we added various trace files to the NFPA repository too.

We will continue to improve the IPv6 library by implementing support for different 1^{st} level sizes and varying number of subtree levels. We are also planning to analyze how the performance is being affected by the variation of the prefix length and investigate additional performance properties, e.g., packet loss, latency, CPU cycles, etc.

Acknowledgments

This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil under grant agreement UNI.61.

References

[1] Levente Csikor et al. Nfpa: Network function performance analyzer. *IEEE NFV-SDN*, 2015.

[2] Sándor Laki et al. High speed packet forwarding compiled from protocol independent data plane specifications. In *ACM SIGCOMM'16 Posters and Demos*, 2016.

[3] Diego Kreutz et al. Software-Defined Networking: A Comprehensive Survey. pages 1–61, 2014.

[4] M. Shahbaz et al. PISCES: A Programmable, Protocol-Independent Software Switch. In *ACM SIGCOMM*, 2016.

[5] OpenDataPlane. Opendataplane.org. `https://www.opendataplane.org`, 2013.

[6] OVN. Open vswitch. `http://openvswitch.org/`, 2009.

[7] P. Bosshart et al. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, July 2014.

[8] P Gyanesh Patra et al. MACSAD: Multi-Architecture Compiler System for Abstract Dataplanes (Aka Partnering P4 with ODP). In *ACM SIGCOMM'16 Demo and Poster Session*, 2016.

[9] P Gyanesh Patra et al. Macsad: High performance dataplane applications on the move. In *IEEE HPSR*, pages 1–6, June 2017.

[10] P Gyanesh Patra et al. Towards a Sweet Spot of Dataplane Programmability, Portability and Performance: On the Scalability of Multi-Architecture P4 Pipelines. *IEEE JSAC issue on Scalability Issues and Solutions for SDN*, September 2018.

[11] Fabricio Rodriguez Cesen et al. BB-Gen: A Packet Crafter for Data Plane Evaluation. In *SBRC 2018 - Salão de Ferramentas*, May 2018.

[12] Fabricio Rodriguez Cesen et al. Design, Implementation and Evaluation of IPv4/IPv6 Longest Prefix Match support in P4 Dataplanes. In *CSBC 2018 - 17º WPerformane*, July 2018.

[13] Fabricio Rodriguez et al. BB-Gen: A Packet Crafter for P4 Target Evaluation. In *ACM SIGCOMM'18 Demo and Poster Session*, 2018.