Design, Implementation and Evaluation of IPv4/IPv6 Longest Prefix Match support in P4 Dataplanes

Fabricio E Rodriguez Cesen[1], P Gyanesh Kumar Patra[1], Christian Esteve Rothenberg[1], Gergely Pongracz[2]

[1]University of Campinas (UNICAMP), Brazil

[2]Ericsson Research, Hungary.

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

{Gergely.Pongracz}@ericsson.com

Abstract. *New trends in dataplane programmability inside Software Defined Networking (SDN) paradigm are in an effort to bring multi-platform support with a high-level definition of the dataplane pipeline functions. The Multi-Architecture Compiler System for Abstract Dataplanes (MACSAD) can integrate the Protocol-Independent Packet Processors (P4) domain-specific language and the OpenDataPlane Project (ODP) APIs, to define a programmable dataplane across multiple targets in a unified compiler system. In this paper, we present and evaluate the IPv4/IPv6 Longest Prefix Match (LPM) support in MACSAD. We develop a new ODP Helper library implementing the IPv6 lookup mechanism based on the current IPv4 solution and evaluate its performance and scalability for diverse workloads and target platform configurations.*

1. Introduction

Ubiquity and deep proliferation of Internet services are driving the bandwidth requirements upward exponentially, and also the necessity to carryout the actual capability evaluation of the networks. A flexible, (re-)configurable network is seen as a solution to deal with the continuously evolving network features and ever-growing bandwidth requirements by re-defining the dataplane on demand.

Software Defined Networking (SDN) [Kreutz et al. 2015] is an emerging network architecture to advocate separation of control plane from data plane. The first communications standard interface defined between the control and data plane was OpenFlow [McKeown et al. 2008]. OpenFlow adoption was marred because of its protocol dependent nature and difficulty to define new custom protocols. Programming Protocol-Independent Packet Processors (P4) [Bosshart et al. 2014] is an open source domain specific language (DSL) for expressing how packets are processed by the pipeline of a network element. P4 is evolving as a de facto language to define dataplane in SDN infrastructures. Similarly OpenDataPlane (ODP) [OpenDataPlane 2013], an open-source project, has emerged as another candidate for describing programmable dataplanes providing a common set of Application Programming Interfaces (APIs) for multiple targets.

Multi-Architecture Compiler System for Abstract Dataplanes (MACSAD) [Patra et al. 2016, Patra et al. 2017] is an approach to converge P4 and ODP through a common compilation process delivering portability of dataplane applications without

compromising target performance. During it's development we observed the absence of IPv6 lookup mechanism in ODP (limited to IPv4 and no short-term plans related to this implementation by ODP team) to support more levels of addressing hierarchy, and a greater number of addressable nodes, and proposed an ODP Helper library with a lookup mechanism for IPv6 as a contribution to ODP open-source community and also extended the effort to add the support to MACSAD too. We lead a complete *performance* and *scalability* evaluation of existing IPv4 and new entrant IPv6 for diverse test workloads (packet traces, table sizes) and target platform configurations (e.g., I/O, CPU #cores).

The rest of this text contains the following topics. Background details and related works are covered in Section 2. Section 3 describes our architecture proposal for the design and implementation of IPv4/IPv6 Longest Prefix Match (LPM) support for MACSAD. Section 4 shows the performance evaluation results. 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 Protocol Independent Switch Architecture (PISA) [Gurevich 2015, McKeown 2016] allows custom definition of network protocols in switch design approach using match+action abstractions. Top-down analysis reveals the need of a Domain Specific Language (DSL) to describe packet processing in these devices. P4 as a high level DSL provides a high enough abstractions to define dataplane, invariably supporting (re-)configuration of PISA devices. P4 is designed to be protocol independent to be able to support custom protocols and target independent to write dataplane applications agnostic to target.

ODP project is a networking dataplane API specification. It defines a set of high-level APIs spanning a common set of standard features across multi targets including ARM, Power PC, and x86 making dataplane applications portable. It enables programmers to write dataplane applications without knowing the underlying hardware. It allows programmers to leverage any specific hardware acceleration capabilities too. 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. Netmap works with linux kernel and allow usage of all linux based tools. DPDK fastpath is separated from linux and it restricts access of the linux based tools to configure interfaces.

ODP provides a helper library[3] as 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 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/`
[3] `https://github.com/Linaro/odp/tree/master/helper`

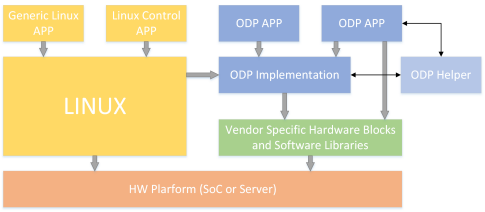

Figure 1. ODP Architecture. Source: [OpenDataPlane 2013]

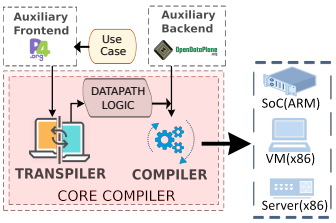

Figure 2. MACSAD Architecture. Source: [Patra et al. 2016]

- **Auxiliary Frontend** The 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 literatures on IPv6 implementations, we briefly emphasize on related programmable dataplane activities similar to our proposed IPv4/IPv6 LPM solutions. OpenvSwitch (OVS) [OpenvSwitch 2016] is the traditional widely adopted OpenFlow based software switch with support for DPDK packet I/O for higher performance. PISCES [Shahbaz et al. 2016] is developed by extending OVS bringing in high-level DSL such as P4 support to achieve programmability. But it is limited by the restricted OVS pipeline abstractions and only supports IPv4. Similarly T4P4S [Laki et al. 2016], 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).

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[4] solution with 15 levels of tables. The 1^{st} level is of 16-bit followed by 14 additional levels of 8-bit each).

[4] `http://dpdk.org/doc/guides-16.04/prog\_guide/lpm6\_lib.html`

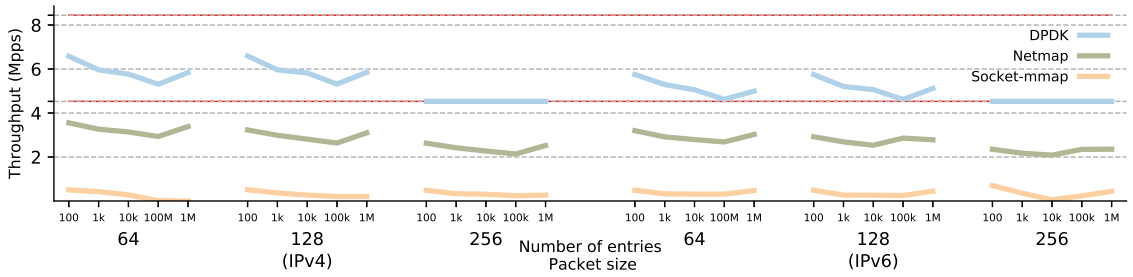

Figure 3. IPv4/IPv6 forwarding performance for different I/O drivers (1 CPU core).

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) [Csikor et al. 2015] as a benchmarking tool. To generate the traces we developed a packet crafter tool[5] that will provide the necessary PCAP files to be used with NFPA. The pipeline implementation and other informations for reproducibility purposes including the P4 programs[6] used by MACSAD and the traffic generator tool (BB-Gen) are available in our public repositories.

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 Intel X540-AT2 NIC (10G), and run with Ubuntu Linux 16.04 LTS (kernel 4.4). The Tester server runs NFPA with DPDK v17.08 and PktGen v3.4.5, and connected to the Device Under Test (DUT [Bradner and McQuaid 1999]). The DUT supports multiple packet I/Os to illustrate the ability to accommodate various different platform features, such as DPDK v17.08, ODP v1.16.0.0, Netmap v11.2, and the basic Linux Socket_mmap provided by the Linux kernel. 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

[5] https://github.com/intrig-unicamp/BB-Gen
[6] https://github.com/intrig-unicamp/macsad-usecases

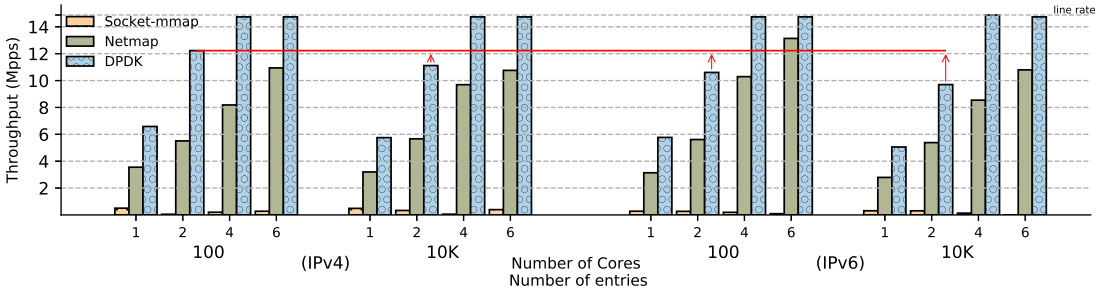

Figure 4. IPv4/IPv6 different cores performance (64 bytes packets).

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 even with the largest packets (1518 bytes) 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 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.

Figure 4 shows a throughput comparison as the number of cores increases from 1 to 6. When the number of CPU cores increases, MACSAD can process more packets resulting in higher throughput. It is notable (red line Fig. 4) that as the number of FIB entries increases, the throughput reduces slightly. Moreover, when table key size increase from 32 bytes (IPv4) to 128 bytes (IPv6) the performance also decreases. This is a significant finding in the understanding of how the complexity of the number of FIB entries and key sizes affect the throughput. This assumption might be addressed in future studies, analyzing the performance with different key sizes and including a variation of FIB size.

5. Conclusions and Future Work

We contributed with the addition of use cases to MACSAD, confirming 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). Confirming that Socket-mmap is slower as it is the linux default driver without any fastpath advantages from Netmap or DPDK. Netmap performance is lower than DPDK on ODP system because in case of Netmap packet copy operation becomes costlier as in case of DPDK support ODP implement zero-copy feature. With this work, we accomplished some open source contributions. The developed IPv6 lookup library will be suggested for adoption by ODP project. We developed a new packet crafter tool that natively creates packets for different standard and custom protocols, and able to generate PCAP files up to more than 1M entries with different headers distributions.

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

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. (2014). P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*.

Bradner, S. and McQuaid, J. (1999). Benchmarking methodology for network interconnect devices. RFC 2544, RFC Editor. `https://www.ietf.org/rfc/rfc2544.txt`.

Csikor, L., Szalay, M., Sonkoly, B., and Toka, L. (2015). Nfpa: Network function performance analyzer. *IEEE Conference on NFV and SDN Demo Track*.

Gurevich, V. (2015). P4 Tutorial. `https://p4.org/assets/Nov-2015-P4-Bootcamp-Labs-Guide.pdf`.

Kreutz, D., Ramos, F., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*.

Laki, S., Horpácsi, D., Vörös, P., Kitlei, R., Leskó, D., and Tejfel, M. (2016). High speed packet forwarding compiled from protocol independent data plane specifications. In *ACM SIGCOMM'16 Posters and Demos*.

McKeown, N. (2016). Programming the Forwarding Plane. `https://forum.stanford.edu/events/2016/slides/plenary/Nick.pdf`.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*

OpenDataPlane (2013). OpenDataPlane.org. `https://www.opendataplane.org`.

OpenvSwitch (2016). OpenvSwitch. `http://openvswitch.org/`.

Patra, P. G., Rothenberg, C. E., and Pongracz, G. (2016). Macsad: Multi-architecture compiler system for abstract dataplanes (aka partnering p4 with odp). *ACM SIGCOMM Demo and Poster Session*.

Patra, P. G., Rothenberg, C. E., and Pongracz, G. (2017). Macsad: High performance dataplane applications on the move. *IEEE HPSR High Performance Switching and Routing*.

Shahbaz, M., Choi, S., Pfaff, B., Kim, C., Feamster, N., McKeown, N., and Rexford, J. (2016). Pisces: A programmable, protocol-independent software switch. *ACM SIGCOMM Computer Communication Review*.