P4Pi: P4 on Raspberry Pi for Networking Education

Sándor Laki
Eötvös Loránd University
lakis@inf.elte.hu

Radostin Stoyanov
University of Oxford
radostin.stoyanov@eng.ox.ac.uk

Dávid Kis
Eötvös Loránd University
kidraai@inf.elte.hu

Robert Soulé
Yale University
robert.soule@yale.edu

Péter Vörös
Eötvös Loránd University
vopraai@inf.elte.hu

Noa Zilberman
University of Oxford
noa.zilberman@eng.ox.ac.uk

ABSTRACT

High level, network programming languages, like P4, enable students to gain hands-on experience in the structure of a switch or router. Students can implement the packet processing pipeline themselves, without prior knowledge of circuit design. However, when choosing a P4 programmable target for use in the classroom, instructors face a lack of options. On the one hand, software solutions, such as the behavioral model (BMv2) switch, are overly simplified and offer low performance. On the other hand, existing hardware solutions are closed source and expensive.

In this paper, we present P4Pi, a new, low-cost, open-source hardware platform intended for networking education. P4Pi allows students to design and deploy P4-based network devices using the Raspberry Pi board, which has a price tag of less than many academic textbooks. We describe the high-level design of the P4Pi platform, offer some suggestions for how P4Pi could be used in the classroom, and present some additional use-cases for applications and functionality that could be developed using P4Pi.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**;
• **Networks** → **Bridges and switches**;

KEYWORDS

Network Education, P4, Programmable Switches, Raspberry Pi

1 INTRODUCTION

The networking community has long sought better ways to teach networking concepts. The introduction of the P4 language [4] and programmable network devices has provided an exciting opportunity for pedagogy—students can gain hands-on experience with the structure of a switch or router by implementing the packet processing pipeline themselves.

The P4 language provides a high-level programming abstraction for programmable network devices. The language is small enough, and similar enough to C, that the main concepts can be covered in one or two lectures. P4 significantly lowers the barrier to entry for students without prior knowledge of circuit design, which was historically a requirement for studying network devices, such as switches, routers, and NICs. Moreover, as programmable network hardware has seen increased adoption in both academia and industry, learning P4 itself can be a practical skill. There are now a large number of research projects, open-source projects, and production systems written in P4.

Nevertheless, while using P4 in the classroom is appealing, there are a limited number of available options for P4-programmable target devices. At one end of the spectrum, students can use the actual programmable ASIC-based hardware deployed in production settings. However, the cost of these devices presents a significant financial challenge. Purchasing even one device can be expensive, and it is impractical to provide every student with a dedicated device for development. While students can time-share access to a few devices, it is not an ideal solution.

At the other end of the spectrum, students can use software solutions, such as the behavioral model (BMv2) switch [1] combined with a network emulator, such as Mininet [16]. While several universities have successfully adopted this approach, programming on simulators or emulators lacks the realism that engages students. Additionally, as others have argued [6], network devices must integrate with existing systems and infrastructure, and therefore an important part of understanding network hardware is coping with interoperability. While possible, it is difficult for students to test interoperability in a software-only test environment.

Other solutions have been proposed that try to balance these extremes, such as NetFPGA [6, 10], an open source hardware platform for rapid prototyping of network devices. However, with a cost of around $1500 USD per board, it is still expensive to run a large class with NetFPGA. Moreover, using NetFPGA requires additional tools, servers, and FPGA design knowledge. It can be very time consuming for TAs and faculty to support students using FPGA devices, when those students have no prior FPGA experience.

In this paper, we describe P4Pi, a new, low-cost, open-source platform for teaching and research. With P4Pi, students can design and deploy P4-based network devices using the Raspberry Pi hardware device. Because Raspberry Pi boards are relatively inexpensive, with a price tag of less than many textbooks (under $100), it is feasible for every student in the class to have their own device. This provides students with the opportunity to gain hands-on experience with developing network hardware that they can use in practice (e.g., develop their own WiFi access point). These devices also readily lend themselves to projects that require interoperability.

P4Pi is developed as part of the P4 Education Working Group activities. The Education Working Group aims to provide educators and practitioners the knowledge and tools required to use P4Pi in class and at home, including tutorials, sample code, tools and community support. Furthermore, as P4Pi is based on the popular Raspberry Pi board, it appeals to other communities, such as hobbyists, and does not depend on a single-source provider.

This paper describes the high-level design of the P4Pi platform, offers some suggestions for how P4Pi could be used in the classroom,

and proposes additional use-cases for applications and functionality that could be developed using P4Pi. Of course, P4Pi is really intended to be an enabling technology. So, all of the software related to P4Pi is publicly available under an open-source license, and we actively seek contributions from the community.

2 P4PI REQUIREMENTS AND OVERVIEW

Requirements. In designing P4Pi, we looked for a target platform that met the following requirements:

- **Low Cost.** The platform must be low cost, such that it would be feasible for a department to set up a lab with tens of platforms or for a student to purchase a platform as part of the required materials.
- **Availability.** The platform must be widely available for purchase, worldwide. This requirement rules out designing a custom board.
- **Open Source.** Because the goal is education and outreach, and not commercialization, both the hardware and software should be open-source.
- **Easy to Use.** The platform must be easy for students to learn, and ideally, have training resources available for educators and students to consult.
- **Wired/Wireless Connectivity.** There must be at least one Ethernet port for wired connectivity. Although not a strict requirement, we viewed wireless connectivity as a soft requirement, since it would be easier for students to connect to the device using their laptops.

Given our expected use and deployment, we also identified non-requirements, by which we mean features that are typically important for network hardware, but not relevant for our particular setting.

- **Performance.** While the design of commercial ASICs is driven by performance requirements, we did not view high-performance as necessary for this project. An acceptable level of performance would be one that supports typical home usage (e.g., browsing the Internet, streaming media, etc.).
- **Scale.** While commercial networks (data center networks, Internet service providers) may need to support tens of thousands of connected nodes and millions of user addresses, our goal is to support connectivity of a class-size or home-size network.

Overview of Main Components. With these requirements in mind, we considered a number of potential hardware targets. We eventually converged on the Raspberry Pi single-board computer. At the time of writing, a Raspberry Pi 4 Model B, with 4GB of RAM costs between $40-$80 USD, and is widely available for purchase from a number of vendors worldwide. It includes a 1GbE RJ-45 (Ethernet) port and on-board WiFi. The hardware is open-source, and was designed with the goal of supporting basic computer science education. Moreover, there is a large community of hobbyists who use Raspberry Pi for their projects, meaning that there are a wide selection of resources and tutorials available for students who seek help.

With the Raspberry Pi as the target hardware, the P4Pi platform includes the following components:

(1) **Compiler.** P4Pi uses the T4P4S [2, 20] compiler as an open-source, multi-target compiler for P4. The front end of the T4P4S compiler is based on P4.org's p4c reference design. The backend of the compiler generates a high-performance software switch from a P4 program.
(2) **Software Switch.** P4Pi uses a DPDK-based software switch that can fully utilize the CPU cores dedicated to the execution of the P4 packet processing pipeline.
(3) **Reference Designs.** As instructive examples, P4Pi includes a set of reference designs for L2/L3 forwarding, network telemetry, and access control lists (ACLs).
(4) **Supporting materials.** P4Pi is hosted on the P4 Language Consortium's repository (https://github.com/p4lang/p4pi), providing both the source code for the platform, and training materials.

Using this platform, students can quickly and easily develop packet processing pipeline themselves written in P4, and deploy them in practical settings (e.g., connect their laptop, phone, or tablet to a personal WiFi access point).

We describe the P4Pi components in more detail in Section 4. First, though, we describe some ways in which P4Pi can be used for networking education.

3 USING P4PI FOR EDUCATION

Possible Deployments. We envision that P4Pi will be used in a practical class or lab. Students may be asked to purchase a Raspberry Pi themselves, as part of the required materials for a course. Some institutions may purchase the boards for students. Alternatively, a department or a group of principal investigators (PIs) could purchase a number of P4Pi platforms once and set up a lab.

While setting up a lab is often expensive, we note that twenty P4Pi platforms can be purchased for about the same price as a single smart NIC. Thus, P4Pi is an affordable option for a department or multiple PIs working together. Moreover, the cost for equipping a lab can be amortized by using P4Pi platforms in multiple courses, spread vertically across the degree (i.e., in both introductory and advanced courses).

Setting up a lab of P4Pi platforms has practical benefits, familiar to everyone teaching hands-on courses. First, not all students may have computing platforms suitable to run practical exercises, either due to cost or because they use incompatible platforms (e.g., tablets). Second, there is often a significant overhead for supporting students with installing and running suitable environments on their machines. A similar overhead is associated with running lab machines with networking equipment (e.g., NIC, FPGA), which need to be maintained and updated. P4Pi alleviates these concerns by providing a common platform, with a prepared image that can be easily cloned or updated from year to year, creating the equivalent of a "plug and play" teaching environment.

Example Uses in Courses. In an introductory networking course, students might first use P4Pi to capture and view packets, and later to interact with other students, e.g., setting up a network, implementing a simple protocol, and communicating with other P4Pi platforms.

In a longer introductory or more advanced course, lecturers can introduce P4 and assign projects in which students implement programs on their programmable devices. A natural first project is basic forwarding. Basic forwarding can be made more complex

by extending the data plane with additional functionality, such as in-band network telemetry [11] or switch-assisted congestion control [3, 17].

At the postgraduate level, it is quite common for seminar courses on advanced networking to include a project in which students are asked to reimplement a system from a contemporary research paper, or implement some novel functionality of their choice. P4Pi is an ideal platform for students to explore such "cutting edge" topics in networking. In addition to implementing data plane functionality, students might also investigate more performance-driven topics, such as the use of kernel bypass (e.g., DPDK) and hardware/software co-design for acceleration.

P4Pi platforms can be further used in labs beyond traditional computer networks, such as in security and IoT related courses. Given the abundance of recent works on programmable devices in this area [13, 15, 19], this is likely to be an attractive solution.

4 P4PI DESIGN

Below, we discuss the main components of the P4Pi design in more technical detail. The initial P4Pi release uses Raspberry Pi 4 Model B as the base hardware platform. For a compiler and software switch, P4Pi extends the open-source T4P4S framework with Raspberry Pi-specific code hidden by an abstraction layer library.

The Raspberry Pi 4 Model B comes with a quad-core ARM64 processor and can be configured with 2GB, 4GB, or 8GB of RAM. Our prototype is tested with the 4GB configuration. The device also includes a wired Gigabit Ethernet port and on-board wireless networking (both 2.4 and 5GHz), enabling the device to act as a wireless access point or a simple WiFi router.

Figure 1 provides a high-level overview of the P4Pi architecture. As the figure shows, the architecture is divided into a fast path and a slow path. On the fast path, two CPU cores are isolated and dedicated for executing the packet processing pipeline. These two cores receive all traffic arriving from the two ports, the Gigabit interface and the wireless interface. The wireless interface is configured in access point mode and all the traffic received on this interface except the management traffic is bridged through the packet processing pipeline. On the slow path, the two remaining cores are used for running the operating system, the P4Runtime server, and the local control plane application, if needed.

4.1 Compiler

The initial release of P4Pi uses T4P4S [2, 20], an open-source, multi-target compiler that generates a high-performance software switch from a P4 program. T4P4S supports $P4_{16}$, and both v1model and PSA architectures. The front-end of T4P4S uses the p4c reference compiler. The back-end of the compiler generates target-agnostic switch code using the Data Plane Development Kit (DPDK) [9] and a hardware abstraction library, named Network Hardware Abstraction Library (NetHAL). In the generated code, all target-dependent operations are abstracted away by NetHAL. The NetHAL library must be ported to each specific hardware target.

We chose to use T4P4S for P4Pi because of its support for a DPDK-based software switch. It would also be possible to run the BMv2 software switch on the Raspberry Pi platform, but the performance is unacceptably slow for practical use. The p4c reference compiler

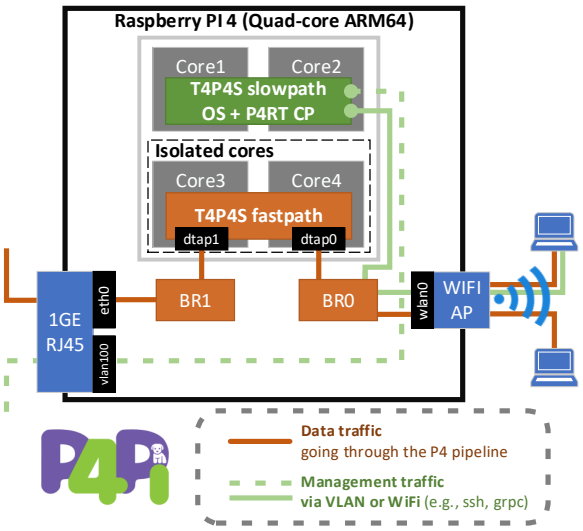

Figure 1: Overview of the P4Pi architecture

does offer an alternative DPDK backend. However, at the time of writing, T4P4S provides more complete language support. An added advantage of T4P4S is that the NetHAL library does allow for increased portability to other hardware platforms, if a user wants to deploy their P4 program on another device.

4.2 Switch

P4Pi uses T4P4S to generate a DPDK-based software switch that executes a given P4 program. The switch also includes a P4Runtime server that enables control plane applications, like ONOS [18], to connect to the switch via the standard GRPC-based API.

Using DPDK on Raspberry Pi has two key advantages. First, it fully utilizes the CPU cores dedicated to packet processing. Second, it provides advanced memory support (e.g., hugepages). The switch's fast path applies a run-to-completion execution model. Packets are processed on the same thread from parsing, through the application of the match-action pipeline, to the deparsing phase. Recently, DPDK has been extended with support for a Software Switch (SWX) pipeline aligned with the P4 language, and we foresee an increased use of P4 with DPDK.

In the current setup of P4Pi, the generated switch program creates two virtual interfaces (e.g., TAP, PCAP or KNI) that are bridged with either the wireless interface or the Gigabit Ethernet port of the Raspberry Pi as shown in Figure 1. For bridging the interfaces, we use Linux bridges with the MAC learning feature disabled. This ensures that all L2 frames are directed to the P4 switch and not the Linux networking stack. Note that other settings are also possible, depending on the use case.

4.3 Reference Designs

To demonstrate teaching scenarios, we consider four popular use cases supported on P4Pi as reference designs: simple switch, simple calculator, network telemetry, and access control.

Simple switch. The simple switch implements basic L3 forwarding. Upon receiving a packet, the switch updates the source and destination MAC addresses, decrements the time-to-live (TTL) in the IP header, and forwards the packet out of the appropriate port. The switch has a single forwarding table, which the control plane populates with static rules. In a classroom setting, students can use their laptop to connect to their P4Pi node via the WiFi access point. The generated switch forwards their traffic either to another laptop connected to the P4Pi node or to the next-hop router via the wired Ethernet interface. Note that the next-hop could also be a P4Pi node and students can use standard networking tools (e.g., ping, traceroute, iperf) to generate test traffic.

Calculator. This design implements a simple, two function calculator in P4. It is used to teach the basic operation of programmable network devices. A custom protocol header, P4Calc, includes header fields for two operands and one operator. The switch performs the calculation, saves the result into a result field, swaps the hardware addresses, and sends the packet back to the sender through the incoming port. In the classroom, students can send P4Calc messages to the P4Pi node via either wireless or wired interfaces. The packets can be generated by, e.g., a python script using Scapy to send P4Calc requests and capture the replies.

In-band Network Telemetry. This design extends basic L3 forwarding with a scaled-down version of In-Band Network Telemetry (INT). The telemetry functions allow users to track the path and the size of queues that packets travel through. The P4 program appends an ID and queue length to the header stack of every packet. At the destination, the sequence of switch IDs correspond to the path, and each ID is followed by the queue length of the switch's port. In a classroom setting, students can monitor the statistics of their P4Pi nodes, e.g., by connecting two laptops to the same P4Pi node and sending traffic between them. In a more complex classroom scenario, INT along multiple P4Pi nodes can be executed.

Access Control. This design implements a stateful firewall using a bloom filter. The firewall can be used to separate an internal network from the external network, and prohibit external-network hosts from establishing connections to hosts on the internal network. In classroom settings, the P4Pi node's wired interface is connected to a L3 router in the lab. The P4Pi node acts as a wireless access point, bridging all the local traffic towards the gateway router. The laptops connected to the P4Pi access point are considered as a private network domain, while the wired interface represents the untrusted public domain, emulating a realistic home-network scenario.

Although the P4 language was originally intended for forwarding plane configuration [5], researchers and developers quickly found creative applications of the language and hardware technology. Several projects explored using P4 to offload or accelerate services that traditionally live outside the network. Some examples include consensus protocols [7, 8], in-network caching [12], and conflict resolution for transaction processing [14]. We hope that P4Pi will inspire similar creativity, especially influenced by having a P4-programmable device that can be used for a home network. Some initial ideas for potential applications could be implementing parental controls, QoE/QoS, or telemetry for IoT devices.

5 CONCLUSION

P4Pi provides a platform for education that is low-cost, open-source, and runs on the widely available Raspberry Pi single-board computer. This paper provides a brief, technical overview of the main components of P4Pi, and describes a few ways in which we imagine P4Pi can be used for network education. Our hope is that the network community will find P4Pi useful and interesting, and will contribute to the project.

ACKNOWLEDGMENTS

P4Pi is developed as an activity of the P4 Education Working Group, one of five working groups within the Open Networking Foundation (ONF) P4 Infra Project. We thank the Network Programming Initiative (NPI) for their generous support, which allows us to purchase and distribute Raspberry Pi boards to early adopters.

REFERENCES

[1] 2021. Behavioral Model (bmv2). https://github.com/p4lang/behavioral-model.
[2] 2021. T4P4S source. https://github.com/P4ELTE/t4p4s.
[3] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman. 2008. Data Center Transport Mechanisms: Congestion Control Theory and IEEE standardization. In *Annual Allerton Conference on Communication, Control, and Computing*.
[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Computer Communication Review* 44, 3 (July 2014).
[5] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*.
[6] Martin Casado, Greg Watson, and Nick McKeown. 2005. Reconfigurable Networking Hardware: A Classroom Tool. In *Symposium on High Performance Interconnects*.
[7] Huynh Tu Dang, Marco Canini, Fernando Pedone, and Robert Soulé. 2016. Paxos Made Switch-y. *SIGCOMM Computer Communication Review* 44 (April 2016).
[8] Huynh Tu Dang, Daniele Sciascia, Marco Canini, Fernando Pedone, and Robert Soulé. 2015. NetPaxos: Consensus at Network Speed. In *ACM SIGCOMM Symposium on SDN Research*.
[9] DPDK 2021. DPDK. http://dpdk.org/.
[10] Glen Gibb, John W. Lockwood, Jad Naous, Paul Hartke, and Nick McKeown. 2008. NetFPGA—An Open Platform for Teaching How to Build Gigabit-Rate Network Switches and Routers. *IEEE Transactions on Education* 51, 3 (Aug. 2008).
[11] Inband Network Telemetry 2021. Inband Network Telemetry (INT). https://github.com/p4lang/p4factory/tree/master/apps/int.
[12] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. 2017. NetCache: Balancing Key-Value Stores with Fast In-Network Caching. In *ACM Symposium on Operating Systems Principles*.
[13] Mário Kuka, Kamil Vojanec, Jan Kučera, and Pavel Benáček. 2019. Accelerated DDoS Attacks Mitigation Using Programmable Data Plane. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*.
[14] Jialin Li, Ellis Michael, and Dan R. K. Ports. 2017. Eris: Coordination-Free Consistent Transactions Using In-Network Concurrency Control. In *ACM Symposium on Operating Systems Principles*.
[15] Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. 2021. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. In *USENIX Security*.
[16] Mininet 2019. Mininet. http://mininet.org.
[17] Peter Newman. 1993. Backward Explicit Congestion Notification for ATM Local Area Networks. In *IEEE Global Telecommunications Conference*.
[18] ONF. 2021. *Open Network Operating System*. https://opennetworking.org/onos/
[19] Qiaofeng Qin, Konstantinos Poularakis, and Leandros Tassiulas. 2020. A Learning Approach with Programmable Data Plane Towards IoT Security. In *IEEE International Conference on Distributed Computing Systems*.
[20] Péter Vörös, Dániel Horpácsi, Róbert Kitlei, Dániel Leskó, Máté Tejfel, and Sándor Laki. 2018. T4P4S: A Target-Independent Compiler for Protocol-Independent Packet Processors. In *IEEE International Conference on High Performance Switching and Routing*.