P4EDGE: P4-Programmability at the Edge for IoT Applications

Péter Vörös, Dávid Kis, Gergő Gombos, Dhulfiqar A. AlWahab, Sándor Laki
ELTE Eötvös Loránd University
Budapest, Hungary
{vopraai,kidraai,ggombos,dhulfi,lakis}@inf.elte.hu

ABSTRACT

With the advent of P4 and programmable data planes, various programmable networking hardware such as smartNICs, DPUs, IPUs, and switches have emerged. Due to various business reasons, all of them were designed for high-speed environments like data center networks. In addition to the high performance, their price tag also inhibits the widespread adoption of such devices in low-speed environments like IoT. Though P4PI project has been launched by ONF P4 Education WG to provide cheap P4-programmable hardware, it has been designed to support educational purposes and has far lower packet processing performance than recent home/IoT gateways. In this demo, we propose P4EDGE gateway that is built on an X86-based open hardware board and an open-source software stack, resulting in better packet processing performance than a Raspberry Pi. In this paper, we propose a demo implementing an IoT use case where different LEDs and sensors in a smart home environment can be assigned to each other in the data plane by adding an entry to a table through a P4EDGE UI. The data plane can detect changes in the sensor states and only notifies the corresponding LED if it is needed, triggering a message to turn it on or off. Unnecessary sensor traffic is filtered out by the data plane. In the demo, we also show other networking examples like routing/switching and firewall functions.

CCS CONCEPTS

• **Networks** → **Programmable networks**; **In-network processing**;

KEYWORDS

Deep network programmability, P4, IoT

1 INTRODUCTION

P4 [3] and programmable data planes transform networks into programmable platforms where the way how switches handle packets can be described at a high level. P4-devices including switches and smart NICs have emerged in the past few years to provide solutions for high-speed networks with low latency requirements. The cost, power consumption, and performance of these devices are both too high for usage at the edge (e.g., routers in a household, or at schools for education). However, these environments could also benefit from the innovation potential of deep network programmability needed for the large-scale deployment of radically new protocols and ideas. Currently, the only cheap alternative to these devices is the P4PI platform [4] that is built on top of Raspberry PI hardware and is maintained by ONF P4 Education WG to provide a cheap P4-programmable hardware for educational purposes. The packet processing performance of P4PI is good for prototyping, but it is far below what commercial gateways and home routers can provide.

In this paper, we introduce P4EDGE [2] as a P4-programmable platform built on an open X86-based router board (PCEngines APU4). This hardware is more expensive (200-300 USD) than the P4PI platform but it is equipped with a more powerful CPU (AMD Embedded G series GX-412TC) and four DPDK-compatible 1GbE NICs. With the P4EDGE platform, we aim to complement the portfolio of accessible P4-programmable platforms, introducing a second level after the P4PI platform for use cases where higher packet processing performance is required. We think that P4PI is an ideal platform for creating proof-of-concept prototypes, but real-world deployments require higher performance where P4EDGE can take over the role. The software stack of P4EDGE platform relies on our re-targetable open-source P4 compiler and packet processing software called T4P4S[5] and additional components that make the management of the device easier. We use T4P4S with the DPDK back-end to utilize the best performance of the four Intel NICs of the router board. The switch provides the standard P4Runtime API through which control plane applications can configure and manage the data plane in run-time. P4EDGE also offers a web UI for effortless application changes, runtime configuration (e.g., inserting table entries), and showing monitoring data and counter states.

2 DEMO SCENARIO

Our demo testbed is depicted in Fig. 1, containing two Raspberry Pi 4 Model Bs, named as Pi1 and Pi2. The P4EDGE switch runs on a PCEngines APU4 (AMD GX-412TC CPU, 4GB DDR3 RAM) with 4x1Gbps Ethernet ports used as switch ports in P4 programs and a 1Gbps management interface. Pi1

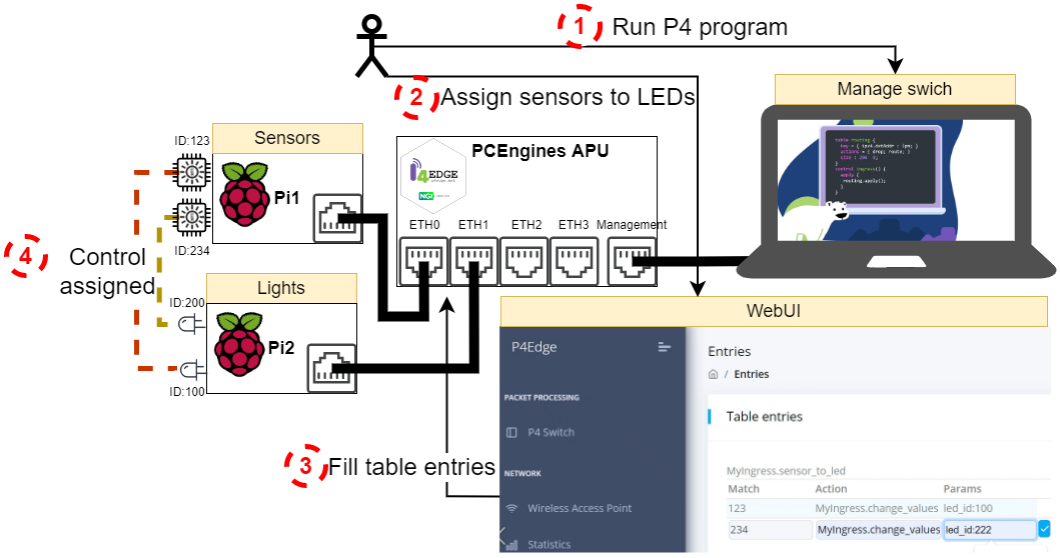

Figure 1: Demo setup.

and Pi2 are connected to two switch ports of the P4EDGE device. A video showing the following demo scenarios is available at [1].

2.1 Basic Smart Home

Pi1 acts as a sensor node, it has light sensors (with id: 123, and id: 234) attached to its GPIO pins, and it reports the sensor values every 100 ms. The sensor data is transmitted over UDP with an additional sensor header of 8 bits for sensor id and 8 bits for sensor value. Pi2 acts as a light switch, controlling two LEDs (with id: 100 and id: 200) with its GPIO pins. Pi2 receives UDP packets with sensor headers where the first 8 bits represent the LED ID, and the other part contains a boolean value to turn the LED on or off. As shown in fig 1 the two nodes are connected via the P4Edge switch. As a first step, the basic smart home P4 switch program has to be started (Step 1). When the switch receives a UDP packet with a sensor header it performs a table lookup to see if that sensor matches one of the LEDs. Unmatched sensor data is dropped. Therefore initially, while Pi1 reports its sensor data periodically, Pi2 does not receive any data. The next action is to assign the sensor IDs to LED IDs (Step 2). This can either be done manually from a P4Runtime terminal or from the P4Edge WebUI. This assignment triggers a table entry insertion (Step 3) in the switch. By inserting an exact match table entry into the sensor_id_to_led table with sensor_id=123 and action=change_values(led_id=100) we instruct the switch to assign the 123 sensor id to 100 LED (Step 4). To complete the setup, the same steps should be done for sensor:234 and led:200. After the insertion, the P4Edge will start to forward sensor packets from Pi1 to Pi2, but before the transmission, the sensor IDs are changed to the matching LED IDs. Because of this change in the packet payload, the UDP checksum must be recalculated. After the setup is complete, we can turn the LEDs on and off by covering the light sensor or exposing it to light.

2.2 Sensor Stream Filtering

In this IoT demo scenario, the data frequency of the sensors is much higher than necessary. Assume that the sensors are non-modifiable components of the network, but still, we want to minimize the redundant network traffic with some additional implementation steps. By extending the P4 program with registers, we can store the latest values for each sensor. This historical data can be used to determine if the condition of the sensor has changed or not. Unchanged sensor packages can be discarded, which does not affect the overall operation of the system, while the load on the network can be significantly reduced. The sensor ID is stored in 8 bits, so a maximum of 256 sensors is assumed. In the P4 code, we initialize a register array in which the sensor values are stored. For each incoming packet, first, read the register value and check if it differs from the value of the current packet. If the value does not match, we update the register and forward the packet as previously described, otherwise the packet is dropped.

2.3 Other examples

During the live demo, we will be able to show some of our additional examples. P4Edge supports applications such as an L2 switch, L3 router, and packet filtering based on MAC/IP addresses, port numbers, or protocols.

REFERENCES

[1] [n. d.]. P4Edge demo video. https://youtu.be/M1x9zTuZMq4. ([n. d.]). Accessed: 2022-06-01.

[2] [n. d.]. P4Edge Github page. https://github.com/P4EDGE. ([n. d.]). Accessed: 2022-06-01.

[3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.

[4] Sándor Laki, Radostin Stoyanov, Dávid Kis, Robert Soulé, Péter Vörös, and Noa Zilberman. 2021. P4Pi: P4 on Raspberry Pi for networking education. *ACM SIGCOMM Computer Communication Review* 51, 3 (2021), 17–21.

[5] 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 *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 1–8.

A EQUIPMENT TO BE USED FOR THE DEMO

We will use two Raspberry Pis equipped with appropriate sensors and LEDs and a PCEngines APU to act as the switch, also a notebook to access the management interface. One screen is sufficient for our demo purposes.

B SPACE NEEDED

Table with a space for 2 notebooks and the single-board computers. Space for one screen and the poster.

C RESOURCES NEEDED

One monitor/screen for the demo and a poster holder stand.

D SETUP TIME REQUIRED

To set up the demo, we need approximately 45 minutes.