Case study on extending P4 language with asynchronous compression functionality

Altangerel Gereltsetseg, Tejfel Máté
3in Research Group, Martonvásár, Eötvös Loránd University, Budapest

Abstract

Modern technologies like 5G, and Internet of Things (IoT) insists on specific deployment requirements from the communication network. The data plane programmability on software-defined networking (SDN) technology gives the possibility to develop or integrate on-demand network features and protocols in network devices (software or hardware) in a flexible, dynamic and efficient way. One of the successful programming languages for data plane programmability is P4. It is an open source domain-specific language. The functionality of the language can be improved by using external functions. The usual way of execution of these functions is synchronous. However, effective execution for some extensions needs an asynchronous approach. This study presents the possibilities to extend P4 language with asynchronous compression functionality in an effective way and a case study introducing one of the possibilities in the virtual environment.

Implementation

The study uses the following technologies.
-T4P4S – Translator for P4 code, developed at ELTE.
-V1model architecture in P4 data plane programming language.
-DPDK compression function as a case study.
-Ucontext(user thread context) module of the standard library.

General workflow

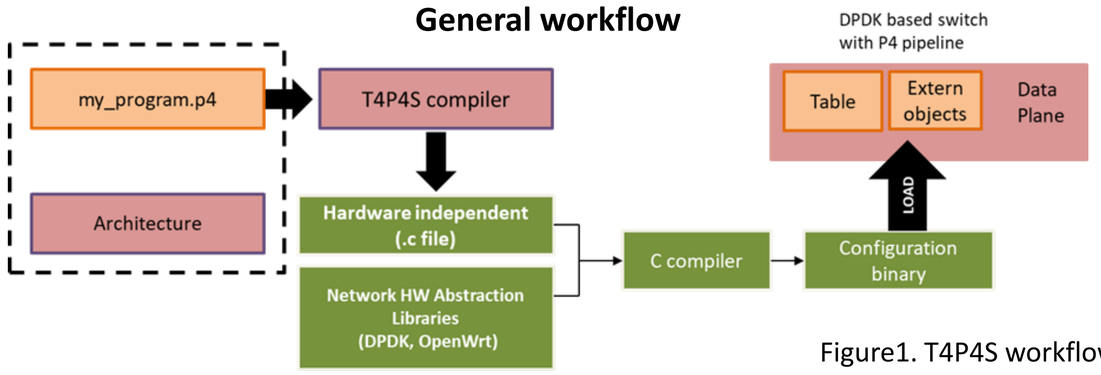


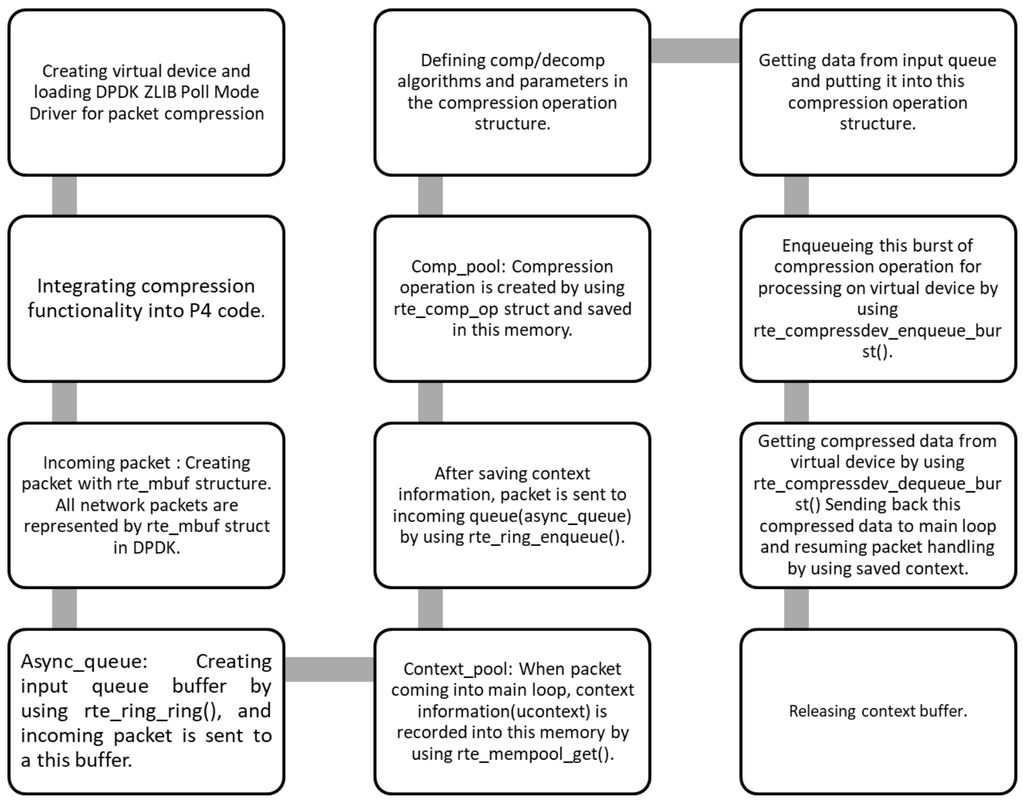
Figure1. T4P4S workflow

The T4P4S provides the architecture described above. The first task of the T4P4S is to generate hardware independent C codes from the P4 codes. Another design feature of the T4P4S is the creation of the Network Hardware abstract layer (NetHal). This layer allows integrating useful libraries and functions for packet processing. Nowadays, some libraries of DPDK and OpenWrt are somewhat implemented in this layer. In this case study, I have integrated DPDK compression functionality into this layer. This compression function is added into P4 code as an extern function. After compiling these codes using C compiler, it creates DPDK based software switch.

Packet processing overview on the DPDK based software switch

As mentioned above, the packet compression is performed in an asynchronous way. The main loop of the software switch parse incoming packet, then applies rules in match/action tables, and deparse packet. The compression function works on another thread. It means that DPDK virtual device which integrating compression functionality is created on another thread. If the packet comes into DPDK based software switch, the main loop of the switch examines the packet and saves the context of the packet. To save context, it uses ucontext model of C programming language. Then it sends this packet into compression device working in another thread. After the packet is compressed in the virtual device, it is sent back to the main loop. The main loop resumes the packet context and sends it to egress. During packet compression, other normal packets pass through the main loop of the switch.

Main steps in application



Packet handling step by step

Step 1: Incoming packet is sent to main loop of the switch.
Step 2. Main loop receives the packet and saves the whole context of the packet.
Step 3. Main loop sends packet with context into input queue (rte_ring_enqueue) for async operation.
Step 4, 5. Virtual compression device gets packet (rte_ring_dequeue_burst) from input queue, compresses it, and sends back to main loop.
Step 6. Main loop receives compressed packet and resumes the packet context and forwards it into egress.

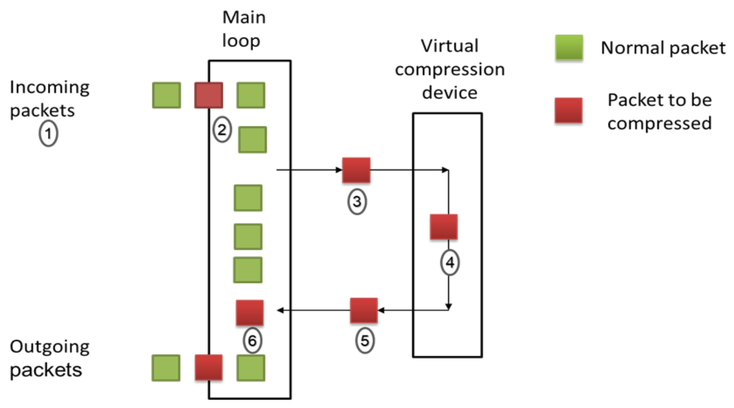


Figure 2. Packet forwarding abstract model in DPDK based virtual switch

Conclusion and future work

The problem of this implementation is that packet processing delay is high. To improve this implementation, we have proposed several ideas:
-Instead of saving whole context of packet, we can save some part of the context.
-We can attach additional information (header) into packet to resume packet handling after compression.
In the future, we will implement them and try to find some other efficient methods.

Acknowledgement

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).