Professorship for Computer Science
Communication Services, Telecommunication Systems and Computer Networks

Emulation of Multipath Transmissions in P4 Networks with Kathará

WueWoWAS2023

June 30, 2023

Marcel Großmann and Tobias Homeyer

Content

Motivation

- When a data packet is sent over the network, it can be lost
  - if lost, it is either retransmitted or dropped
- What if several packets sent to their destination contain critical data?
- For example, if a person has a remote surgery, the data must be transmitted quickly and reliably
- We propose a solution to duplicate important packages and send them to their destination via multiple routes
  - The receiving switch ensures that only a single copy of the traffic is further forwarded to its destination.

Kathará

Kathará is an open source container-based network emulation system [1].

General Information [1]:

- Used to test/develop networks in a sandbox environment
- Spiritual successor of the notorious Netkit
- Each device is emulated by a container (using Docker or Kubernetes)
- Each container can run on a different Docker image
- Uses the concept of network scenarios
- Can be installed on many operating systems such as Windows, Mac and Linux

P4 [3]

- P4 stands for Programming Protocol-independent Packet Processors
- P4 language is used for expressing how packets are proccessed by the data plane of a programmable switch
- P4 is designed to only specify data plane functionality of a programmable switch
- P4 was first introduced in the 2014 (original paper[1])

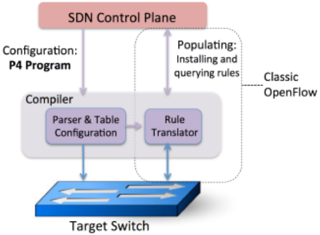

Figure 1: P4 is a language to configure switches[2, p.88]

[1] https://arxiv.org/pdf/1312.1719.pdf.

P4 switch vs traditional switch

- Data plane functionality on a P4 programmable switch is defined by the P4 program and is not fixed [3]
- Control plane communicates with the data plane using the same channels as in a fixed-function device, but the set of tables in the data plane are no longer fixed (defined by a P4 program) [3]

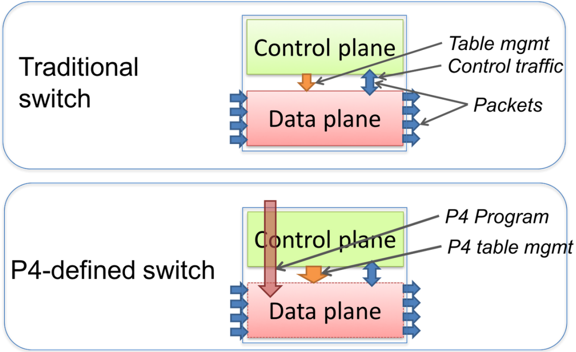

Figure 2: Traditional switches vs programmable switches by [3]

P4 advantages

P4 offers the following advantages [3]:

- **Flexibility**, many packet forwarding policies can be expressed in P4 programs
- **Expressiveness**, P4 is able to express hardware independent packet processing algorithm using only general purpose operation and table look ups
- **Software engineering**, P4 offers type checking, information hiding and software reuse
- **Component libraries**
- **Decoupling hardware and software evolution**, manufacturers are able to abstract architectures to further decouple the evolution of low-level architectural details from high-level processing
- **Debugging**, manufacturers are able to provide software models of their architecture to aid in development.

P4 VSS

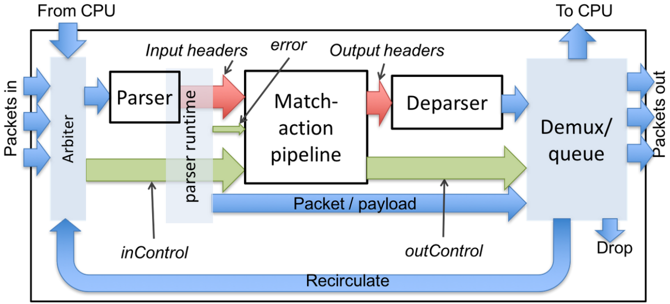

Figure 3: P4 Very Simple Switch [3]

Prototypes

Two Prototypes where created:

- Random Split
- Duplication

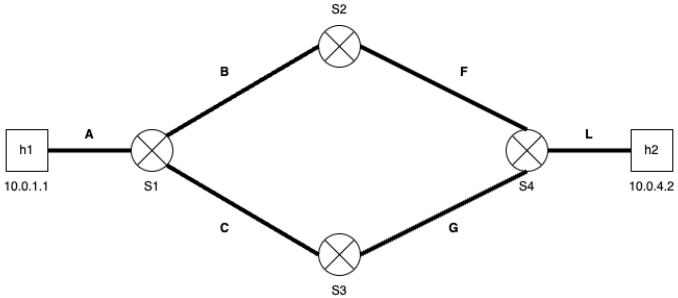

Figure 4: Topologie for all prototypes

Random Split

- Depending on a random value and a threshold, the packet is either forwarded over s2 or s3

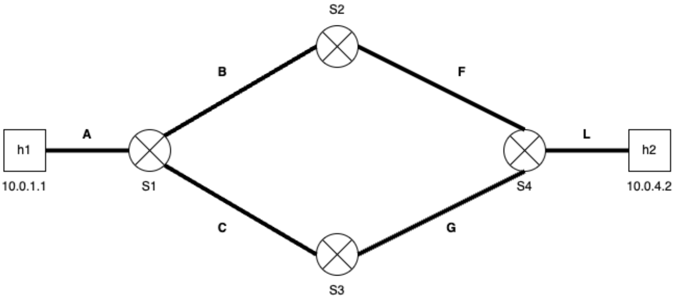

Figure 5: Topologie for all prototypes

Code example RS


```
action random_split_group(bit<14> random_split_group_id, bit<16> threshold, bit<16> maxNum){
  bit<16> randomVal;
  random(randomVal, (bit<16>) 0, (bit<16>)maxNum);
  if(randomVal >= (bit<16>) threshold) {
    meta.random_split_port = (bit<14>) 0;
  } else {
    meta.random_split_port = (bit<14>) 1;
  }
   meta.random_split_group_id = random_split_group_id;
}
```

Figure 6: Action random-split

Duplication

- Packets are duplicated at s1 (cloned packets are forwarded over s2)
- At s4, the copy of a packet that reaches it last is dropped (deduplication)
- A hash is created over each arriving packet and saved in the switch register (similar to bloom filters)

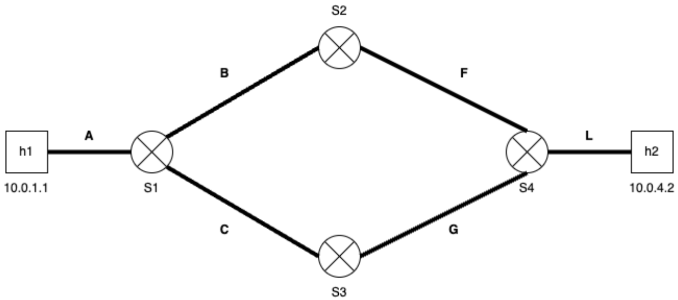

Figure 7: Topologie for all prototypes

Code example Duplication

```
action clone_packet() {
    const bit<32> REPORT_MIRROR_SESSION_ID = 500;
    //Clone from inggress to egress pipeline
    clone(CloneType.I2E, REPORT_MIRROR_SESSION_ID);
}
```

Figure 8: Action clone

Code example Duplication

```
hash(new_hash,
    HashAlgorithm.crc16,
    (bit<32>)0,
    {
        hdr.ipv4.srcAddr,
        hdr.ipv4.dstAddr,
        hdr.udp.srcPort,
        hdr.udp.dstPort,
        hdr.ipv4.protocol
    },
    (bit<32>)32767
);
    last_packet.read(hash_read, 0);
    last_packet.read(hash_read1, 1);
    last_packet.read(hash_read2, 2);
    last_packet.read(hash_read3, 3);
    last_packet.read(hash_read4, 4);
    log_msg("NEW_HASH= {}, SAVED_hash_0 = {}",{new_hash, hash_read});
    log_msg("NEW_HASH= {}, SAVED_hash_1 = {}",{new_hash, hash_read1});
    log_msg("NEW_HASH= {}, SAVED_hash_2 = {}",{new_hash, hash_read2});
    log_msg("NEW_HASH= {}, SAVED_hash_3 = {}",{new_hash, hash_read3});
    log_msg("NEW_HASH= {}, SAVED_hash_4 = {}",{new_hash, hash_read4});
    if(new_hash == hash_read) {
        log_msg("PACKET DROPPED");
        drop();
        last_packet.write(0, empty);
    } else if(new_hash == hash_read1) {
        log_msg("PACKET DROPPED");
        drop();
        last_packet.write(1, empty);
    } else if(new_hash == hash_read2) {
        log_msg("PACKET DROPPED");
        drop();
        last_packet.write(2, empty);
    } else if(new_hash == hash_read3) {
        log_msg("PACKET DROPPED");
        drop();
        last_packet.write(3, empty);
```

Figure 9: Remove duplicate packets

Code example Duplication

```
action shift_register(bit<32> value_hash) {
    bit<32> temp;
    bit<32> temp2;
    bit<32> temp3;
    bit<32> temp4;
    last_packet.read(temp,0);
    last_packet.read(temp2,1);
    last_packet.read(temp3,2);
    last_packet.read(temp4,3);
    last_packet.write(4, temp4);
    last_packet.write(3, temp3);
    last_packet.write(2, temp2);
    last_packet.write(1, temp);
    last_packet.write(0, value_hash);
}
```

Figure 10: Action shift-register

Evaluation

- Python script with sender and receiver were used to send UDP packet between h1 and h2
- Wireshark was used to capture all traffic on all CDs
- Log messenges were used to print the created and saved hash values
- Pcap was used on s4 switch in the Duplication Prototype

Live demonstration

- Live demonstration of the prototypes with Wireshark captures
- Code: `https://github.com/uniba-ktr/p4_multipath`

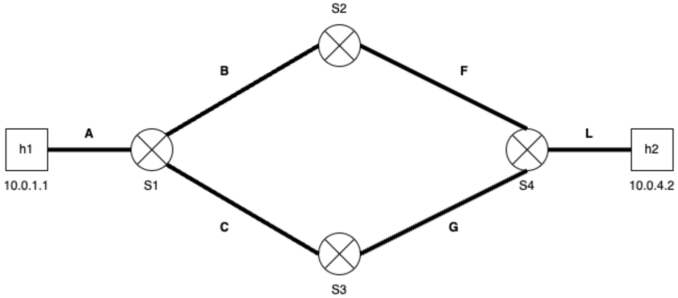

Figure 11: Topologie for all prototypes

Conclusion & Future Work

- implementation of the Duplication prototype was a success (with UDP and TCP)
- P4 runs well with Kathará
- Packet deduplication and cloning is easily adjustable with P4

Future Work:

- Onos can be added to enable more control over the control plane
- Network congestion
- Automatic reroute and establishing of routes (when switches fail or new switches are added)
- Testing Scalability

References I

[1] Kathará, “Kathara.” [Online]. Available: https://www.kathara.org/

[2] D. W. A. Vahdat, G. Varghese, “P4: Programming protocol-independent packet processors,” *ACM SIGCOMM*, vol. 44(3), 2014.

[3] T. P. L. Consortium, “P416 language specification.” [Online]. Available: https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html

[4] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 2012. [Online]. Available: http://www.amazon.com/Computer-Networking-Top-Down-Approach-Edition/dp/0132856204%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0132856204

[5] P. L. N. Hiroaki Motohashi and H. Senkiya, “Implementation of p4-based schedulers for multipath communication,” *IEE, Digital Object Identifier 10.1109/ACCESS.2022.3192539*, vol. 10, 2022.

[6] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

Questions ?

Tobias Homeyer
tobias.homeyer@stud.uni-bamberg.de