COIN
Internet-Draft
Intended status: Informational
Expires: April 3, 2021

H. Singh
MNK Labs and Consulting
M-J. Montpetit
Concordia Univeristy

September 30, 2020

Use of P4 Programs in IETF Specifications
draft-hsingh-coinrg-p4use-00

Abstract

The IETF specifies several algorithms operating in the data plane of a network node, including liveliness detection, congestion control, network measurement, security, and load balancing. Such algorithms are commonly specified using English or flow charts. As an alternative, this document proposes that P4 programs can be used to specify some data plane algorithms. P4 is a programming language created in 2014 to program the data plane of network nodes such as switches, routers, smartNICs, and generic compute targets.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 3, 2021.

Table of Contents

1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Introduction

Research papers for data plane algorithms already use P4 programs to specify algorithms. The MIT Domino compiler [MIT-Domino] which synthesizes hardware logic also outputs a P4 program. For example, the HULA [HULA] data plane congestion control algorithm includes P4 programming logic in the paper.

In another section, this document shows an example for how textual description and flow chart of an algorithm could be augmented or replaced by a P4 program. Lastly, This document presents a summary of the P4 programming language.

3. Example Use

An IETF document in [I-D.chen-nvo3-load-banlancing] discusses the flowlet algorithm for load balancing. The draft includes description of algorithm in section 4.1 and a state machine diagram in section 5. Further, if other tables are used in conjunction with the flowlet table, in what sequence does one invoke the tables? Specifying the algorithm of the draft as a P4 program is appropriate. Open source P4 compiler (p4c) already includes a P4 program which implements the flowlet algorithm. See

https://github.com/p4lang/p4c/blob/master/testdata/p4_16_samples/flowlet_switching-bmv2.p4

The program uses five tables and the ingress control block shows in what order are the tables invoked. The flowlet algorithm exists in the lookup_flow_map and update_flowlet_id P4 actions. The program uses P4 registers to maintain state. The IETF draft uses timers. The P4 code uses timestamps since P4 does not support timer yet. A rudimentary timer in a P4 program can use arithmetic to determine whether it is an even/odd minute based on data plane clock used for timestamping packets.

A portion of the P4 program listed above is shown below. The portion shows two P4 actions which implement the flowlet algorithm.

```
        action lookup_flowlet_map() {
            hash(meta.ingress_metadata.flowlet_map_index,
                HashAlgorithm.crc16,
                (bit<13>)0, { hdr.ipv4.srcAddr, hdr.ipv4.dstAddr,
                hdr.ipv4.protocol, hdr.tcp.srcPort,
                hdr.tcp.dstPort }, (bit<26>)13);
            flowlet_id.read(meta.ingress_metadata.flowlet_id,
            (bit<32>)meta.ingress_metadata.flowlet_map_index);
            meta.ingress_metadata.flow_ipg =
            (bit<32>)standard_metadata.ingress_global_timestamp;
            flowlet_lasttime.read(
                  meta.ingress_metadata.flowlet_lasttime,
                  (bit<32>)meta.ingress_metadata.flowlet_map_index);
            meta.ingress_metadata.flow_ipg =
                        meta.ingress_metadata.flow_ipg -
                        meta.ingress_metadata.flowlet_lasttime;
            flowlet_lasttime.write(
                   (bit<32>)meta.ingress_metadata.flowlet_map_index,
                (bit<32>)standard_metadata.ingress_global_timestamp);
        }
        action update_flowlet_id() {
            meta.ingress_metadata.flowlet_id =
               meta.ingress_metadata.flowlet_id + 16w1;
            flowlet_id.write(
               (bit<32>)meta.ingress_metadata.flowlet_map_index,
               (bit<16>)meta.ingress_metadata.flowlet_id);
        }
```

Figure 1: Two P4 actions implement the flowlet algorithm

The ingress control block invokes table lookup using 'table.apply()'.

```
        apply {
           @atomic {
           flowlet.apply();
           if (meta.ingress_metadata.flow_ipg > 32w50000)
               new_flowlet.apply();
           }
           ecmp_group.apply();
           ecmp_nhop.apply();
           forward.apply();
        }
```

Figure 2: Code shows order of invocation for table lookup

4. Summary of P4

First, https://p4.org is a great resource to start with. The website includes specifications, pointers to P4 tutorials, p4c, the P4 mailer, P4 Slack Channel, and other details to P4 events, blogs. etc. The README.md file at https://github.com/p4lan/p4c/ includes details on how to compile a P4 program. P4 started with a P4-14 version in 2014. Since, May 2017, a new version in P4-16 and compiler are available. A list of hardware targets to use for P4 programming is available here: https://github.com/hesingh/p4-info

5. Security Considerations

Use IPSec [RFC4301].

6. IANA Considerations

None.

7. Acknowledgements

Thanks (in alphabetical order by first name) to Nick McKeown for encouraging this work.

8. References

8.1. Normative References

[I-D.chen-nvo3-load-banlancing]
Chen, H., "Load balancing without packet reordering in NVO3", draft-chen-nvo3-load-banlancing-00 (work in progress), October 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

8.2. Informative References

[HULA] Katta, N., Hira, M., Kim, C., Sivaraman, A., and J. Rexford, "HULA: Scalable Load Balancing Using Programmable Data Planes", March 2016, <https://dl.acm.org/doi/pdf/10.1145/2890955.2890968>.

[MIT-Domino] Sivaraman, A., Cheung, A., Budiu, M., Kim, C., Alizadeh, M., Balakrishnan, H., Varghese, G., McKeown, N., and S. Licking, "Packet Transactions: High-level Programming for Line-Rate Switches", January 2016, <http://web.mit.edu/domino/>.

Authors' Addresses

Hemant Singh
MNK Labs and Consulting
7 Caldwell Drive
Westford, MA 01886
USA

Phone: +1 978 692 2340
Email: hemant@mnkcg.com
URI: https://mnkcg.com/

Marie-Jose Montpetit
Concordia Univeristy
1455 Boulevard de Maisonneuve O
Montreal, Quebec 01886
Canada

Email: marie@mjmontpetit.com