Programming
The Network Data Plane in P4

Changhoon Kim (chang@barefootnetworks.com)
Status quo

in English

“This is roughly how I process packets ...”

• Prone to bugs
• Very long and unpredictable lead time

Fixed-function ASIC
Protocols evolve ...
more rapidly than we wish they would

• Encapsulation protocols for network virtualization
  – 1\textsuperscript{st} gen (~2010): VXLAN, NVGRE
  – 2\textsuperscript{nd} gen (~2012): STT, VXLAN-GPE
  – 3\textsuperscript{rd} gen (~2014): NSH, Geneve

• OpenFlow
  – OF 1.0 (Dec 2009): 12 fields (Ethernet, TCP, IPv4)
  – OF 1.1 (Feb 2011): 15 fields (MPLS, inter-table metadata)
  – OF 1.2 (Dec 2011): 36 fields (ARP, ICMP, IPv6)

Everybody has opinions!
Can’t we just let them build their own protocols?
Programmable network devices

- Some devices are or will be more programmable than fixed-function ASICs
- CPUs: 10s of Gb/s
- FPGAs, NPUs: 100s of Gb/s
- Protocol-independent switch ASICs: a few Tb/s
  - RMT [SIGCOMM’13] and a few emerging solutions
  - Merchant silicon with fully programmable parser and generic match-action logic
  - In next few years this kind of silicon will dominate
Turning the tables

"This is precisely how you must process packets"

Switch OS
Run-time API
Driver

PISA device
(Protocol-Independent Switch Architecture)
What does this mean?

• To network device vendors
  – S/W programming practices and tools used in every phase
  – Extremely fast iteration and feature release
  – Differentiation in capabilities and performance
  – Can fix even data-plane bugs in the field

• To large on-line service providers
  – No more “black boxes” in the “white boxes”
  – Your devs can program, test, and debug your network devices all the way down
  – You keep your own ideas
Why we call it protocol-independent packet processing
Device does not understand any protocols until it gets programmed

Logical Data-plane View (your P4 program)

Switch Pipeline

Programmable Parser

Match Table

Action ALUs

ACL

IPv4

IPv6

L2

Queues

packet

packet

Packet

Match Table

Action ALUs

Match Table

Action ALUs

Match Table

Action ALUs

Match Table

Action ALUs

CLK
Mapping logical data-plane design to physical resources

Logical Data-plane View
(your P4 program)

Switch Pipeline

Programmable Parser

L2 Table
L2 Action Macro

IPv4 Table
V4 Action Macro

IPv6 Table
V6 Action Macro

ACL Table
ACL Action Macro

Queues

CLK
Re-configurability

Logical Data-plane View (your P4 program)

Switch Pipeline

Programmable Parser

L2 Table
L2 Action Macro

MyEncap
MyEncap

IPv4
IPv6

IPv4
IPv6

ACL
ACL Action Macro

Queues

CLK
What does a P4 program look like?

```
header_type ethernet_t {
  fields {
    dstAddr : 48;
    srcAddr : 48;
  }
  parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
      0x8100  : parse_vlan;
    }
  }
}

header_type my_encap_t {
  fields {
    foo : 12;
    bar : 8;
    baz : 4;
    qux : 4;
    next_protocol : 4;
  }
}
```
What does a P4 program look like?

```
header_type ipv4_t {
  fields {
    version : 4;
    ihl     : 4;
    diffserv: 8;
    totalLen: 16;
    identification: 16;
    flags  : 3;
    fragOffset : 13;
    ttl    : 8;
    protocol : 8;
    hdrChecksum : 16;
    srcAddr    : 32;
    dstAddr    : 32;
    options    : *
  }
  length     : (ihl << 2);
  max_length : 60;
}
```
What does a P4 program look like?

```
table ipv4_lpm
{
  reads {
    ipv4.dstAddr : lpm;
  }
  actions {
    set_next_hop;
  }
}

action set_next_hop(nhop_ipv4_addr, port)
{
  modify_field(metadata.nhop_ipv4_addr, nhop_ipv4_addr);
  modify_field(standard_metadata.egress_port, port);
  add_to_field(ipv4.ttl, -1);
}
```

```
control ingress
{
  apply(l2);
  apply(my_encap);
  if (valid(ipv4) {
    apply(ipv4_lpm);
  } else {
    apply(ipv6_lpm);
  }
  apply(acl);
}
MyNetVirtualization (your new protocol)

- Tenant Isolation Logic
- Forwarding Logic (P4)
- Table Population

Compiler

- Compiled Forwarding Logic
- Table Population
- Switch Runtime API
- Tenant Isolation Logic

Linux

- Switch Chip Driver

Program (boot or init time) → PISA chip

Add/delete (run time)
DEMO!

In-band Network Telemetry (INT)

Demo Setup

#include "includes/headers.p4"
#include "includes/parser.p4"

/********************************/
/* Device Personality Selection */
/********************************/

// #define P_MINIMAL
#define P_BASE
// #define P_MSDC_SPINE
// #define P_MSDC_TOR
// #define P_MSDC_BL
// #define P_INTERNET_ROUTER
...

// Flexible Tables
#define LPM_TABLE_SIZE 16384
#define IPV6_LPM_TABLE_SIZE 4096
#define HOST_TABLE_SIZE 131072
#define IPV6_HOST_TABLE_SIZE 32768
...
header_type routing_metadata_t {
  fields {
    bd : BD_BIT_WIDTH; /* bridge domain */
    vrf : VRF_BIT_WIDTH; /* routing domain */
    v6_vrf : VRF_BIT_WIDTH; /* routing domain */
    ...
  }
  table bridge_domain {
    reads {
      routing_metadata.bd : exact;
    }
    actions {
      nop; // Not used
    }
    bd_set;
  }
...
Scenario: Debugging Long Latency Tail

HTTP download (from H1 to H3)
In-band Network Telemetry in P4

```
action export_queue_latency (sw_id) {
  add_header(int_header);
  modify_field(int_header.kind, TCP_OPTION_INT);
  modify_field(int_header.len, TCP_OPTION_INT_LEN);
  modify_field(int_header.sw_id, sw_id);
  modify_field(int_header.q_latency, intrinsic_metadata.deq_timedelta);
  add_to_field(tcp.dataOffset, 2);
  add_to_field(ipv4.totalLen, 8);
  subtract_from_field(ingress_metadata.tcpLength, 12);
}

table int_table {
  reads {
    ip.protocol;
  }
  actions {
    export_queue_latency;
  }
}
```

Add TCP Options & copy switch ID and queue latency Into the options
INT Open-source and Spec


Improving Network Monitoring and Management with Programmable Data Planes

By Mukesh Hira & LJ Wobker

Quicklinks: The INT specification - - - INT GitHub repository - - - INT demo video

Compute virtualization and the widespread deployment of virtual machines has lead to an extension of the network into the hypervisor. Network virtualization solutions have emerged that enable rapid programmiong of network services — logical switches, virtual firewalls, virtual load balancers, and virtual multiplexers. The current virtual machines and network interfaces in the virtual network so as to enable deployment of virtual services over any physical network infrastructure, the only requirement from the physical network being IP connectivity between the hypervisors.

While the decoupling of physical and virtual topologies has advantages, it is important to have some interaction between the physical and virtual switches to allow for end-to-end monitoring of the entire physical + virtual network from a “single pane of glass” and to help in troubleshooting and fault isolation in complex physical + virtual topologies.

We propose methods for various network elements to collect and report their state in real-time, allowing for improved cooperation between the virtual and physical layers without requiring intermediate layers such as CPU driven control planes. The general term we have applied to these methods is INT: Inband Network Telemetry.
Scenario: Resolving Traffic Split Problem

<table>
<thead>
<tr>
<th>Traffic Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1 → H3</td>
</tr>
<tr>
<td>H2 → H4</td>
</tr>
</tbody>
</table>

Spine1 (dc.p4)  Spine2 (dc.p4)

Leaf1 (dc.p4)  Leaf2 (dc.p4)

H1  5Mbps  H2  5Mbps  

H1  5Mbps  H3  5Mbps  

5Mbps  H2  5Mbps  H4  5Mbps  5Mbps 

Monitor
Simple Flowlet Switching

**Flowlet Detection & Mgmt**

<table>
<thead>
<tr>
<th>last time</th>
<th>flowlet id</th>
</tr>
</thead>
<tbody>
<tr>
<td>82019445</td>
<td>4</td>
</tr>
<tr>
<td>82028039</td>
<td>13</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>81084924</td>
<td>29</td>
</tr>
<tr>
<td>82148703</td>
<td>7</td>
</tr>
</tbody>
</table>

Hash (e.g., 5-tuple)

- Packet hdr & metadata

**ECMP and LAG**

Next hop and port selection using hash of 6-tuple:

\[5\text{-tuple} + \text{flowlet id}\]

```c
if (current_time – last_time > \text{timeout}) {
    flowlet_id += 1;
}
last_time = current_time;
```

- **Hash collisions are infrequent**
  - Flowlets are very short, and hence # of active flowlets is only up to tens of thousands
- **Hash collisions are inconsequential**
  - Flowlets are very small; sub-optimality caused by collisions is negligible
- **Time out is configurable**
  - Range can be as low \(~1\) us to \(~100\) ms; a few hundred usec seems most suitable for DCs
Flowlet Switching in P4

field_list_calculation flowlet_map_hash {
  input {
    l3_hash_fields;
  }
  algorithm : crc16;
  output_width : FLOWLET_MAP_SIZE;
}

register flowlet_lasttime {
  width : 32;
  instance_count : 8192;
}

register flowlet_id {
  width : 16;
  instance_count : 8192;
}

action lookup_flowlet_map() {
  modify_field_with_hash_based_offset(ingress_metadata.flowlet_map_index, 0,
    flowlet_map_hash, FLOWLET_MAP_SIZE);

  register_read(ingress_metadata.flowlet_id,
    flowlet_id, ingress_metadata.flowlet_map_index);

  modify_field(ingress_metadata.flow_ipg, intrinsic_metadata.ingress_global_timestamp);
  register_read(ingress_metadata.flowlet_lasttime,
    flowlet_lasttime, ingress_metadata.flowlet_map_index);
  subtract_from_field(ingress_metadata.flow_ipg, ingress_metadata.flowlet_lasttime);
  register_write(flowlet_lasttime, ingress_metadata.flowlet_map_index,
    intrinsic_metadata.ingress_global_timestamp);
}

table flowlet {
  actions {
    lookup_flowlet_map;
  }
}

field_list flowlet_l3_hash_fields {
  ingress_metadata.lkp_ipv4_sa;
  ingress_metadata.lkp_ipv4_da;
  ingress_metadata.lkp_ip_proto;
  ingress_metadata.lkp_l4_sport;
  ingress_metadata.lkp_l4_dport;
  ingress_metadata.flowlet_id;
}

field_list_calculation flowlet_ecmp_hash {
  input {
    flowlet_l3_hash_fields;
  }
  algorithm : crc16;
  output_width : ECMP_BIT_WIDTH;
}

action set_ecmp_select(ecmp_base, ecmp_count) {
  modify_field_with_hash_based_offset(ingress_metadata.ecmp_offset, ecmp_base,
    ecmp_hash, ecmp_count);
}

action set_flowlet_ecmp_select(ecmp_base, ecmp_count) {
  modify_field_with_hash_based_offset(ingress_metadata.ecmp_offset, ecmp_base,
    flowlet_ecmp_hash, ecmp_count);
}

table ecmp_group {
  reads {
    ingress_metadata.ecmp_index : exact;
  }
  actions {
    set_ecmp_select;
    set_flowlet_ecmp_select;
  }
  size : ECMP_GROUP_TABLE_SIZE;
}
This will accelerate innovations

• Areas of innovation (just to name a few)
  – Reducing feature set → “biggest bang for buck”
  – Network monitoring, analysis, and diagnostics
  – Tunnel-splicing gateways
  – Load balancing
  – Attack detection and mitigation
  – Host-stack offloading

• Across various types of network devices
  – Hardware and software devices (OVS, eBPF, etc.)
  – Switches, NICs, middle-boxes, etc.
This will create lots of significant R&D opportunities

• Novel network and protocol designs
  – Take advantage of unprecedented amount of visibility and control into the network
  – Joint design and optimization between hosts and network

• Development tools
  – Compilers, debuggers, static analyzers, profilers, etc.
  – P4-programming assistance tools

• Network verification, test, and simulation

• Many more …
Maintains the language spec
P4.org – P4 Language Consortium

Maintains key dev tools under Apache license

- Reference P4 programs
- Compiler
- P4 software switch
- Test framework
P4.org – P4 Language Consortium

Open for participation by any individuals or organizations
Hosts P4 workshops and boot camps

- 2\textsuperscript{nd} P4 Workshop on Nov/18 at Stanford
- 1\textsuperscript{st} P4 Boot camp in the same week
Switch.p4: A typical L2/L3 switch in P4

- Feature set
  - Basic L2 switching: MAC learning, VLAN, flooding, and STP
  - Basic L3 routing: IPv4 and IPv4 routing with VRF
  - LAG and ECMP
  - Tunneling: VXLAN, NVGRE, Geneve, and GRE
  - Basic ACL: MAC and IP ACLs
  - Unicast RPF
  - MPLS: LER, LSR, IPVPN, VPLS, and L2VPN
  - Host interface
  - Mirroring: Ingress and egress mirroring with ERSPAN
  - Counters/Statistics

- More features coming soon
  - IP Multicast, NAT, QoS, Ingress policing, etc.
Summary

• In next few years **data-plane programmability** will quickly become commonplace

• This will accelerate innovations in networking

• No more “black boxes” in “white boxes”

• As **the common industry-wide forwarding language**, **P4** will play crucial roles

• Lots of R&D and business opportunities opening up – join and contribute!