Tracking P4 Program Execution in the Data Plane

SOSR ‘20

Suriya Kodeswaran
Mina Arashloo, Praveen Tammana, Jennifer Rexford
Background: P4 Compilation

• Compiler maps programs to hardware target
  • Optimizes to work around hardware restraints

• Compiler bugs
  • If-Else statements always taking one branch
  • Mapping multiple variables to the same registers

• These bugs change the way packets are processed
Software: P4 Control Flow

```c
apply {
    if (hdr.srcRoutes[0].isValid()) {
        if (hdr.srcRoutes[0].bos == 1) {
            srcRoute_finish();
        }
        srcRoute_nhops();
        if (hdr.ipv4.isValid()) {
            update_ttl();
        }
    } else {
        drop();
    }
}
```

- **Bug when packet takes wrong path**
Goal: Tracking Packet Execution Path

• Naïve solution: Reserve a bit for each decision

• Challenges:
  • Switch.p4: $10^{34}$ ways to process packets with longest path length of 44 tables
  • Limited per-packet state

• Need an efficient solution!
Expected vs Actual Behavior

- Series of match-action units placed by compiler
- Actions happen concurrently on a stage
  - Fast packet processing
  - Actions are dependent on each other
- Order in pipeline != order in control flow
How do you trace path execution?

• **Key observation:** P4 control flow is always a DAG

• Ball-Larus (BL) algorithm
  • Provides path encoding for DAGs
  • Update a variable on edge transitions
  • Paths are uniquely identified by sum of edges
Enabling Ball-Larus in P4

![Diagram showing the process of enabling Ball-Larus in P4]

- **Input program**
- **BL Encoding**
- **Encoding algorithm**
- **Encoded program**
- **P4 compiler**
- **p4c**
- **Hardware target**
Challenges of naïve BL encoding

- Single variable does not scale
- Extra dependencies during compiler mapping
- Don’t change the system we are tracking!
Example: Challenge of naïve BL encoding

<table>
<thead>
<tr>
<th>Path</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGH</td>
<td>0</td>
</tr>
<tr>
<td>ABDEH</td>
<td>1</td>
</tr>
<tr>
<td>ABDEFH</td>
<td>2</td>
</tr>
<tr>
<td>ABCDEH</td>
<td>3</td>
</tr>
<tr>
<td>ABCDEFH</td>
<td>4</td>
</tr>
</tbody>
</table>
Idea: DAG Partitioning

Path | Sum of V1
--- | ---
AGH | 0
ABDEH | 1
ABDEFH | 2
ABCDEH | 3
ABCDEFH | 4

Path | Sum of V2
--- | ---
EH | 0
EFH | 1
Our Solution: Multivariable Ball-Larus

• Partition the control flow DAG
  • One BL variable per partition

• Mutually exclusive actions can be in same partition
  • Actions from the same table
  • Actions on different stages

• Path is uniquely identified by tuple of BL variables: \( \{V_1, V_2, \ldots, V_n\} \)
Minimizing Hardware Overhead

• DAG partitioning introduces additional overhead

• Solution: Use Linear Program to optimize # bits required for encoding

• Constraints:
  • Hardware’s maximum bit size
  • Actions update one variable
  • Dependent actions update different variables
# Results

<table>
<thead>
<tr>
<th>Programs</th>
<th>Paths ($N$)</th>
<th>Tables</th>
<th>Stages</th>
<th>Actions</th>
<th>Path Vars ($K$)</th>
<th>Added Actions</th>
<th>Added Metadata (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tna-action-selector.p4</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>our approach: 3</td>
</tr>
<tr>
<td>source-routing.p4</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>optimal: 3</td>
</tr>
<tr>
<td>tna-multicast.p4</td>
<td>36</td>
<td>6</td>
<td>4</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>fabric-bng.p4</td>
<td>$1.01 \times 10^9$</td>
<td>67</td>
<td>11</td>
<td>73</td>
<td>20</td>
<td>35</td>
<td>46</td>
</tr>
<tr>
<td>simple switch.p4</td>
<td>$1.76 \times 10^9$</td>
<td>35</td>
<td>7</td>
<td>157</td>
<td>8</td>
<td>11</td>
<td>47</td>
</tr>
<tr>
<td>switch.p4</td>
<td>$1.75 \times 10^{34}$</td>
<td>157</td>
<td>12</td>
<td>307</td>
<td>21</td>
<td>71</td>
<td>178</td>
</tr>
</tbody>
</table>


Conclusion and Future Work

• Data-plane primitive for tracking every packet’s execution path

• Idea: Multivariable Ball-Larus encoding

• Implemented on Tofino

• Future work: Detection and localization of bugs in P4 compiler and control plane
Questions?

Contact: suriyak@princeton.edu