FlowBlaze.p4: a library for quick prototyping of stateful SDN applications in P4

Daniele Moro*, Davide Sanvito^, Antonio Capone*

* Politecnico di Milano, Italy
^ NEC Laboratories Europe, Germany

IEEE Conference on Network Function Virtualization and Software Defined Networks 2020
9-12 November 2020
Virtual Conference
Outline

• Introduction
• The FlowBlaze architecture
• FlowBlaze.p4 library
• Using FlowBlaze.p4: Packet Limiter
• Use Case: Multi-Class Rate Limiter
• Conclusions
5G and Mobile Edge Computing requires offloading of network functions to data plane

- **P4**: reference language for data plane programming
- **State Machine**: powerful abstraction to develop stateful packet processing
- **FlowBlaze**: EFSM-based stateful packet processing architecture
Introduction - 2

Issues with FlowBlaze utilization:
• Missing prototyping platform
• Manual (error-prone) mapping from EFSM to FlowBlaze table entries
• No FlowBlaze P4 implementation

FlowBlaze.p4:
• FlowBlaze library implementation in P4
• Open source library
• GUI to automatically translate EFSM into table entries
• Exploit all the tools from the P4 Community
The FlowBlaze* architecture - 1

- EFSM based stateful packet processing
- Multi-stage: stateless (OpenFlow like) + stateful (EFSM-based) stages

The FlowBlaze architecture - 2

Stateful Stage

• Arbitrary Flow Definition with associated context (State and Flow Data Variable - FDV)
• Conditions evaluated on the FDVs (<, >, <=, >=, ==, !=)
• Transitions in EFSM Table (if <conditions> and <state> then <new_state, actions>)
• Update FDVs (+, -, *, <<, >>) and packet action (e.g., forward, drop...)
FlowBlaze.p4 library
Stateful FlowBlaze Stage to P4
Communication between blocks via P4 Packet Metadata
Target: BMv2 software switch

<table>
<thead>
<tr>
<th>Flow Context</th>
<th>Registers*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*Indexing with hash function on the Flow Definition</td>
</tr>
</tbody>
</table>

| Conditions    | Series of if on the Flow Data Variables |

<table>
<thead>
<tr>
<th>EFSM Table</th>
<th>Match Action Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match:</td>
<td>&lt;&lt;conditions results, state, arbitrary fields&gt;&gt;</td>
</tr>
<tr>
<td>Action:</td>
<td>&lt;&lt;Set new state, Set Packet Action ID, Set Update Functions&gt;&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packet Action</th>
<th>Match Action Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match:</td>
<td>Packet Action ID</td>
</tr>
<tr>
<td>Actions:</td>
<td>User Defined P4 Actions</td>
</tr>
</tbody>
</table>

| Update Functions | Series of if to map the set update function to the actual action on the FDV |
Using FlowBlaze.p4

Packet Limiter: EFSM

Flow Definition

- IPv4 source address

- State **0**
  All the flows start from this state.
  Transition to the **Count** state.

- State **Count**
  Counts the number of packets.
  Auto-transition that counts the packets when below the threshold
  Transition to the **Drop** state when reached the threshold (10) packets.

- State **Drop**
  Drop all the packets.
  “Black-hole” state.
Using FlowBlaze.p4
Packet Limiter: Compile-time Configuration

1. Add FlowBlaze into your P4 application

2. `#define`:
   - **Flow Scope**: source IP address
   - **Packet actions**:
     - `forward()`
     - `drop()`
   - **EFSM header match**
   - **Condition header**

3. Compile the program
Using FlowBlaze.p4 (continued)

Packet Limiter: Run-time Configuration

4. Run the GUI:
   - Add states as in the EFSM
   - Build the transitions as in the drawn EFSM:
     - Match + Condition
     - Update Function + Packet Action

5. Run in Mininet with the provided Docker infrastructure
Use Cases

Multi-Class Rate Limiter

Flow Definition

• IPv4 source address

• State 0
  All the flows starts from this state.
  “Classify” the traffic by the source IP address setting the `max_bytes` FDV

• State Allow
  Let traffic through, counting the bytes
  Transition to the Drop state if in the predefined time-slot more than `max_bytes` passed.

• State Drop
  Drop all the packets.
  Transition back to the Allow state when time-slot ends.
Conclusions

• FlowBlaze: abstraction for stateful packet processing based on EFSM
• Missing: prototyping platform and P4 implementation

We provide:
• FlowBlaze.p4: a library for prototyping with FlowBlaze and P4
• GUI: automatic translation of EFSM into runtime configuration
• Docker-based environment
• Open source: available on GitHub

Future works:
• ONOS Integration
• DC-style fabric integration (e.g., Trellis)
Demo later today!
“Demonstrating FlowBlaze.p4: fast prototyping for EFSM-based data plane applications”

https://github.com/ANTLab-polimi/flowblaze.p4