FlowBlaze
Stateful Packet Processing in Hardware

Salvatore Pontarelli, Roberto Bifulco, Marco Bonola, Carmelo Cascone, Marco Spaziani, Valerio Bruschi, Davide Sanvito, Giuseppe Siracusano, Antonio Capone, Michio Honda, Felipe Huici, Giuseppe Bianchi
This work has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements No 761493 (“5GTANGO”) and No.762057 (“5G-PICTURE”)

*Now at ONF
Roberto

State Machines
State Machines

State machines to the rescue of complex forms

Tackling UI State Machines

Kinetic, NSDI’15

Figure 1: Intrusion detection FSM.

Figure 2: Stateful firewall FSM.

Figure 3: Data usage-based rate limiter FSM.

packets (e.g., all packets from the same host, in the case of the previous example). Each group of packets has a separate FSM instance; packets in the same group will always be in the same state. We call such a group of packets a located packet equivalence class (LPEC).
Programmable NICs

NICs with programmable ASICs, SoC, FPGAs...

E.g., Microsoft [AccelNet NSDI ‘17, NSDI ‘18]

Programming them is Hard!
Making programming easier

NF Logic

Code

Simulation

Synthesis

High-level Synthesis

Faster programming

Expressive

Hardware expertise

ClickNP [Sigcomm ’16], Emu [ATC ’17]

Match-Action Abstraction

Faster programming

NF Logic focused

Limited support for state

P4 [CCR ’14], Domino [Sigcomm ’16]
Match-Action Abstraction Limitations

Match-Action pipeline

State in tables
- large
- read only
  (wr from cplane)

State in registers
- small
- read/write
Extending Match-Action abstractions

Match-Action pipeline

FlowBlaze
Match-Action vs Finite State Machine (FSM)

if match then action

a table

if (match, state) then action

represented by a table

OpenState CCR’14
FAST HotSDN’14
Multiple state machines?

Example: Drop a flow after its 10th packet

Any pkt, c=10 \rightarrow \text{drop}

Any pkt \rightarrow c=c+1, \text{fwd}

Any pkt \rightarrow \text{drop}

<table>
<thead>
<tr>
<th>Flow ID</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPdst = 192.168.0.1</td>
<td>S</td>
</tr>
<tr>
<td>IPdst = 192.168.0.2</td>
<td>S</td>
</tr>
<tr>
<td>IPdst = 192.168.0.3</td>
<td>B</td>
</tr>
</tbody>
</table>

Each flow’s FSM evolves on its own

Per-flow state is common in network functions
Introducing per-flow state

<table>
<thead>
<tr>
<th>Flow ID (IP dst)</th>
<th>State</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.1</td>
<td>S</td>
<td>c=7</td>
</tr>
<tr>
<td>192.168.0.2</td>
<td>S</td>
<td>c=4</td>
</tr>
<tr>
<td>192.168.0.3</td>
<td>B</td>
<td>c=10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pkt Header</th>
<th>State</th>
<th>Cond.</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>B</td>
<td>*</td>
<td>drop</td>
</tr>
<tr>
<td>*</td>
<td>S</td>
<td>c=10</td>
<td>State=B, drop</td>
</tr>
<tr>
<td>*</td>
<td>S</td>
<td>c&lt;10</td>
<td>fwd</td>
</tr>
</tbody>
</table>

Flow State

Flow ID (IP dst)

Pkt headers, metadata

Flow Ctx Table

Table

if match

if state

ALUs

Table

Any pkt → c=c+1, fwd

Any pkt → drop

Global State

Any pkt, c=10 → drop

S

B

© NEC Corporation 2019 Images from multiple sources © of their owners NEC Public
Implementation issues

Insertion in the flow context table

Insertion = new flow

Variable insertion time

Cuckoo hash table
Implementation issues

Insertion in the flow context table

Insertion = new flow

Variable insertion time
Handling variable insertion time

Flow table: Cuckoo hash

Efficient
Constant lookup-time
Variable insertion-time

Waiting for Insertion!!

Throughput reduction

Latency increase
Flow context insertion handling

lookup time scales with pkt arrival rate

insertion time scales with flow arrival rate

Cuckoo hash table

Variable Insertion speed

Fast insertion

Fast lookup

read logic

Flow Ctx Table

Flow State

Stash

Variable Insertion speed

Fast insertion

lookup

insertion

P P P P
Implementation issues

Insertion in the flow context table

Insertion = new flow

Variable insertion time

State update latency

Feedback loop

Race conditions
Avoiding race conditions

Flow Ctx Table
Flow State

Table
if match if state

ALUs
Global State

P

P

P
P

c=10

 Any pkt, c=10 → drop

S B

Any pkt → c=c+1, fwd
Any pkt → drop

inconsistent state

c=10 c=9 c=9
Avoiding race conditions

Throughput reduction

Latency increase

Any pkt, c=10 → drop

Any pkt → c=c+1, fwd

Any pkt → drop
Avoiding race conditions

Lock pipeline for packets from the same flow

Performance degradation only in unlikely cases
Implementation issues

Insertion in the flow context table

- Flow Ctx Table
- Table
- ALUs
- Global State

Variable insertion time

State update latency

- Feedback loop
- Race conditions
Does it work?

FlowBlaze provides the same performance for all use cases

<table>
<thead>
<tr>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Load Balancer</td>
</tr>
<tr>
<td>UDP Stateful Firewall</td>
</tr>
<tr>
<td>Port Knocking Firewall</td>
</tr>
<tr>
<td>Flowlet load balancer</td>
</tr>
<tr>
<td>Traffic Policer</td>
</tr>
<tr>
<td>Big Flow Detector</td>
</tr>
<tr>
<td>SYN flood Detection and Mitigation</td>
</tr>
<tr>
<td>TCP optimistic ACK detection</td>
</tr>
<tr>
<td>TCP super spreader detection</td>
</tr>
<tr>
<td>Dynamic NAT</td>
</tr>
<tr>
<td>vEPC subscriber’s quota verification</td>
</tr>
<tr>
<td>Switch Paxos Coordinator</td>
</tr>
<tr>
<td>Switch Paxos Acceptor</td>
</tr>
<tr>
<td>In-network KVS cache</td>
</tr>
</tbody>
</table>

Test: 10Gb/s@64B flow definition: 5-tuple

FlowBlaze: NetFPGA@156.25MHz
Compared to: DPDK-VPP on Xeon X3470@2.93GHz, Intel 82599 10GbE NIC

<table>
<thead>
<tr>
<th>Tput (Mpps)</th>
<th>Stateless</th>
<th>UDP Stateful Firewall</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlowBlaze</td>
<td>14,8</td>
<td>14,8</td>
</tr>
<tr>
<td>VPP (1 core)</td>
<td>14,8</td>
<td>6,2</td>
</tr>
<tr>
<td>VPP (2 core)</td>
<td>14,8</td>
<td>12</td>
</tr>
<tr>
<td>VPP (3 core)</td>
<td>14,8</td>
<td>14,2</td>
</tr>
</tbody>
</table>
Stress test

Test:
40Gb/s@64B (NetFPGA line rate)

Flow distributions

<table>
<thead>
<tr>
<th>Trace</th>
<th>IP s</th>
<th>IP s,d</th>
<th>5 tpl</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNI1</td>
<td>575</td>
<td>997</td>
<td>4k</td>
</tr>
<tr>
<td>UNI2</td>
<td>948</td>
<td>3k</td>
<td>7k</td>
</tr>
<tr>
<td>MW15</td>
<td>12k</td>
<td>130k</td>
<td>152k</td>
</tr>
<tr>
<td>CHI15</td>
<td>92k</td>
<td>147k</td>
<td>178k</td>
</tr>
</tbody>
</table>

Max # active flows

<table>
<thead>
<tr>
<th>Trace</th>
<th>IP s</th>
<th>IP s,d</th>
<th>5 tpl</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNI1</td>
<td>13</td>
<td>19</td>
<td>39</td>
</tr>
<tr>
<td>UNI2</td>
<td>20</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>MW15</td>
<td>38</td>
<td>112</td>
<td>114</td>
</tr>
<tr>
<td>CHI15</td>
<td>135</td>
<td>144</td>
<td>144</td>
</tr>
</tbody>
</table>

Max # new flows/ms

Drop rate

156.25MHz

Stash

stash entries

UNI1 UNI2 MAWI CHI15

Global IP s IP s/d 5 tuple
Conclusion

FlowBlaze
• FSM Abstraction for packet processing
• Efficient FPGA implementation

Benefits
• Can keep state for 100Ks flows in flow tables
• Save several CPU cores for stateful NFs
• Power efficient (check the paper!)
• Low latency (check the paper!)

Check the paper, there’s a lot more!
FlowBlaze is open
Both software and hardware implementations
maintained by

Axbryd

https://github.com/axbryd/FlowBlaze

Thank you!
visit us and check our demo
at the poster session