ForCES-implemented Openflow switch over 100Gig switch

LTIR Laboratory, University of Quebec At Montreal

We believe in running code
Demonstrate ForCES Library implementability: Draft under discussion...

Show ForCES capability over 100 Gig Openflow Switch implementation

Demonstrate some flexibility of each approach: ForCES and Openflow under multiple considerations: Forwarding model, Applications

Evaluate the performance of different control plane strategies (ForCES vs Openflow).
Why we need to map Openflow forwarding table over a set of LFBs?

Answer: It is the main interface to change the behaviour of FE

- Openflow is based on tables of set of entries: Matchfields and actions
- Forces is based on LFBs: more generic approach based on state and object oriented interface
- Enable an openflow-compliant switch to be controlled by both an Openflow controller as well as a ForCES controller
Over which forwarding model we want to compare?

Answers: Multiple forwarding patterns can be suggested

- Forwarding patterns using parallelism and/or pipeline processing
- Forwarding pattern tables exploiting different search data structure types (TCAM, LPM, Hash, ...)

We believe in running code
Which Application we want to evaluate?

**Load balancer**

Create additional paths to support additional load

1. CE create paths to connect two servers
2. Rate reaches a threshold on a server interface
3. CE Application send request to create additional path

**Link Recovery**

Recover a down path

1. CE instantiate and configure LFBs in each FE of the network topology.
2. An active path is down
3. CE reconfigure LFBs to establish a new path

We believe in running code
Over which equipments?

- **EZchip pizzabox**
  - 100 Gig NP-4 network processor with integrated traffic management.
  - Task Optimized Processors dedicated for different network operations: parsing, searching, resolving, modifying and learning.
  - Freescale control CPU
  - 5 module bays with SGMII and XAUI interface supports.
  - Variety of hardware based search structures: NetLogic TCAM, LPM support, tree-based lookup structure, ...  

- **NFP PCIe card**
  - 20 Gig Network Flow processor
  - 40 programmable micro-engines for different networking operations based on IXP networking cores technology.
  - ARM control CPU
  - PCIe gen2 with I/O virtualization support

- **Cavium Octeon: Under consideration**
**Pipeline for a 802.1d Bridge, Router**

**Openflow Forwarding Model for a 802.1d Bridge, Router**

1. **InPort Table**
   - Matchfield: Port ID
   - Instructions: go-to MAC table/Drop

2. **MAC Table**
   - Matchfield: MAC dst
   - Instructions: ApplyAction(Learn Match)/WriteAction(Bridge Match)/Drop

3. **Rt Table**
   - Matchfield: DIP
   - Instructions: ApplyAction(uRPF Match, DBR Match)/Drop

4. **outPort Table**
   - Matchfield: Out Port ID
   - Instructions: Action(Queue)/Drop

**Controller Application for a 802.1d Bridge, Router**

<table>
<thead>
<tr>
<th>Table</th>
<th>Matchfield</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>InPort</td>
<td>Port ID</td>
<td>go-to MAC table/Drop</td>
</tr>
<tr>
<td>MAC</td>
<td>MAC dst</td>
<td>ApplyAction(Learn Match)/WriteAction(Bridge Match)/Drop</td>
</tr>
<tr>
<td>Routing</td>
<td>DIP</td>
<td>ApplyAction(uRPF Match, DBR Match)/Drop</td>
</tr>
<tr>
<td>out Port</td>
<td>Out port ID</td>
<td>Action(Queue)/Drop</td>
</tr>
</tbody>
</table>

**ForCES Forwarding Model for a 802.1d Bridge, Router**

**Tables Pipeline LFB (OFFlowTables)**

- Parse LFB
- InPort Table
  - Matchfield: Port ID Type: Direct Table
- MAC Table
  - Matchfield: MAC dst Type: Exact match
- Rt Table
  - Matchfield: DIP Type: LPM
- outPort Table
  - Matchfield: Out Port ID Type: Direct Table
Demo: Link recovery application

Tesbed using EZchip pizzabox based on 100 Gig EZchip’s NP-4 network processor

1-CE application instantiate and configure Path0 and Path1

2-FE1/FE2 link is down → Path0 is down

3-CE application reconfigure FE LFBs to establish new Path0
Numerical results

- FE instantiation time: 1.35ms
- LFB instantiation time: 0.17us
- ForCES-implemented Openflow switch Applications execution time

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Failover</td>
<td>234ms</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>319ms</td>
</tr>
</tbody>
</table>

- Performance tests lead to approximately 80 million packets per second for 64bytes packet size.
Lessons learned

What did we learn from the ForCES and Openflow implementations:

▶ There is no difficulty to map LFBs over Openflow components.
▶ ForCES doesn’t use capabilities of hardware, e.g. paralellism, hardware lookup structures (LPM, TCAM, ...)
▶ We can make our implementation without Action Type LFB, so both FlowTable LFB and GroupTable LFB are sufficient.
▶ Forwarding model of multiple tables needs to be mapped to one or more LFBs, i.e. all the tables are mapped to one LFB or subsets of tables are mapped to separate LFBs. This is useful when it comes to control different LFBs through separate control plane applications.

We believe in running code