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).

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-complient switch to be controlled by both an Openflow controller as well as a ForCES controller

Over which forwarding model we want to compare?

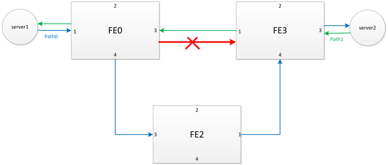
Answers: Multiple forwarding patterns can be suggested

- 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?

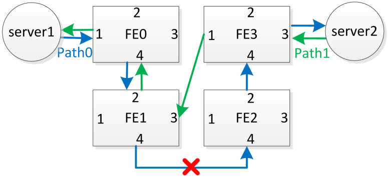
Load balancer



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

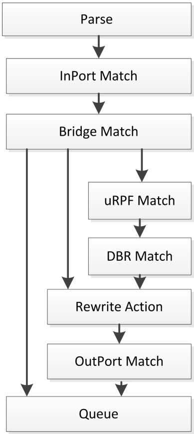
We believe in running code

Over which equipments?

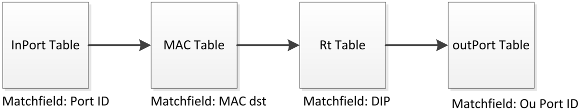
- EZchip pizzabox
  - 100 Gig NP-4 network processor with integrated traffic management.
  - Task Optimized Processors dedicated for different network operations: parsing, searching, resolving, modifying and learning.
  - Freescale control CPU
  - 5 module bays with SGMII and XAUI interface supports.
  - 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

We believe in running code

Pipeline for a 802.1d Bridge, Router



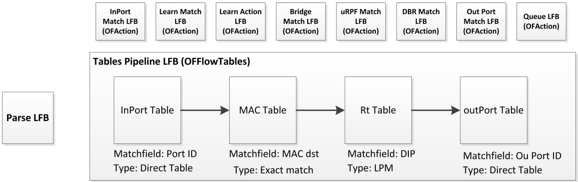
Openflow Forwarding Model for a 802.1d Bridge, Router



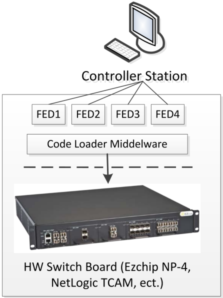
Controller Application for a 802.1d Bridge, Router

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

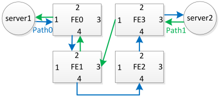
ForCES Forwarding Model for a 802.1d Bridge, Router



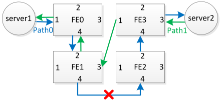
Demo: Link recovery application



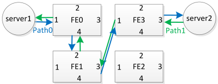
Tesbed using EZchip pizzabox based on 100 Gig EZchip's NP-4 network processor



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

We believe in running code

Numerical results

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

Failover	234ms
Load Balancing	319ms

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

We believe in running code

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