SDN, OpenFlow and the ONF

OpenFlow/Software-Defined Networking (SDN)

"OpenFlow/SDN is emerging as one of the most promising and disruptive networking technologies of recent years. It has the potential to enable network innovation and create choice, and thus help realize new capabilities and address persistent problems with networking. It also promises to give network operators more control of their infrastructure, allowing customization and optimization, therefore reducing overall capital and operational costs. "

Source: http://opennetsummit.org/why.html

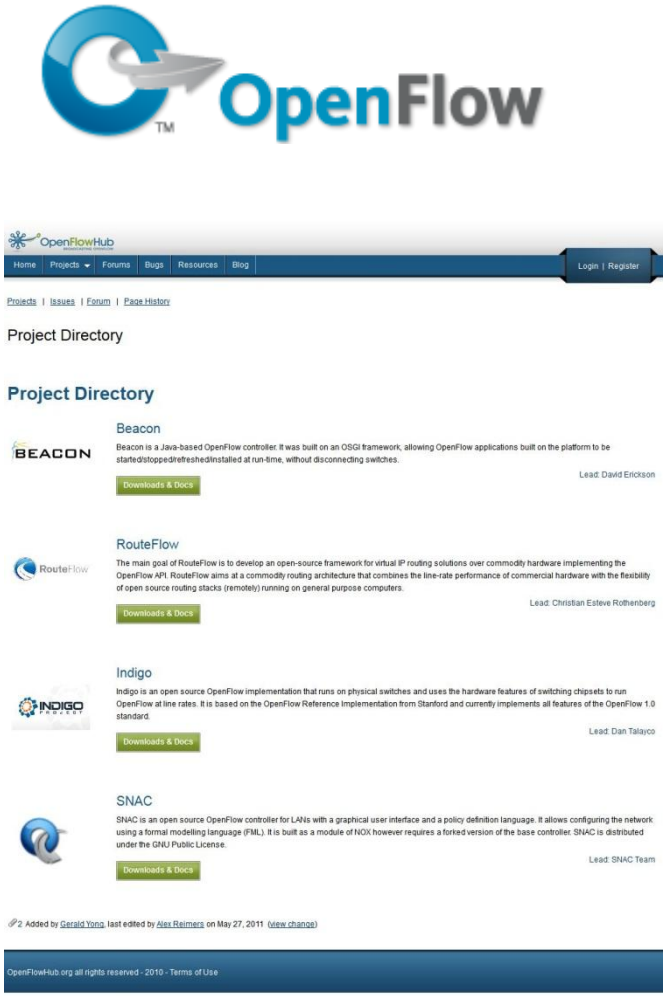
OpenFlow – Started 2008 at Standford

- Enabling innovation on campus
- Standard way to control flow-tables in commercial switches and routers
- Being deployed at Stanford
- Consider deploying it at your campus too

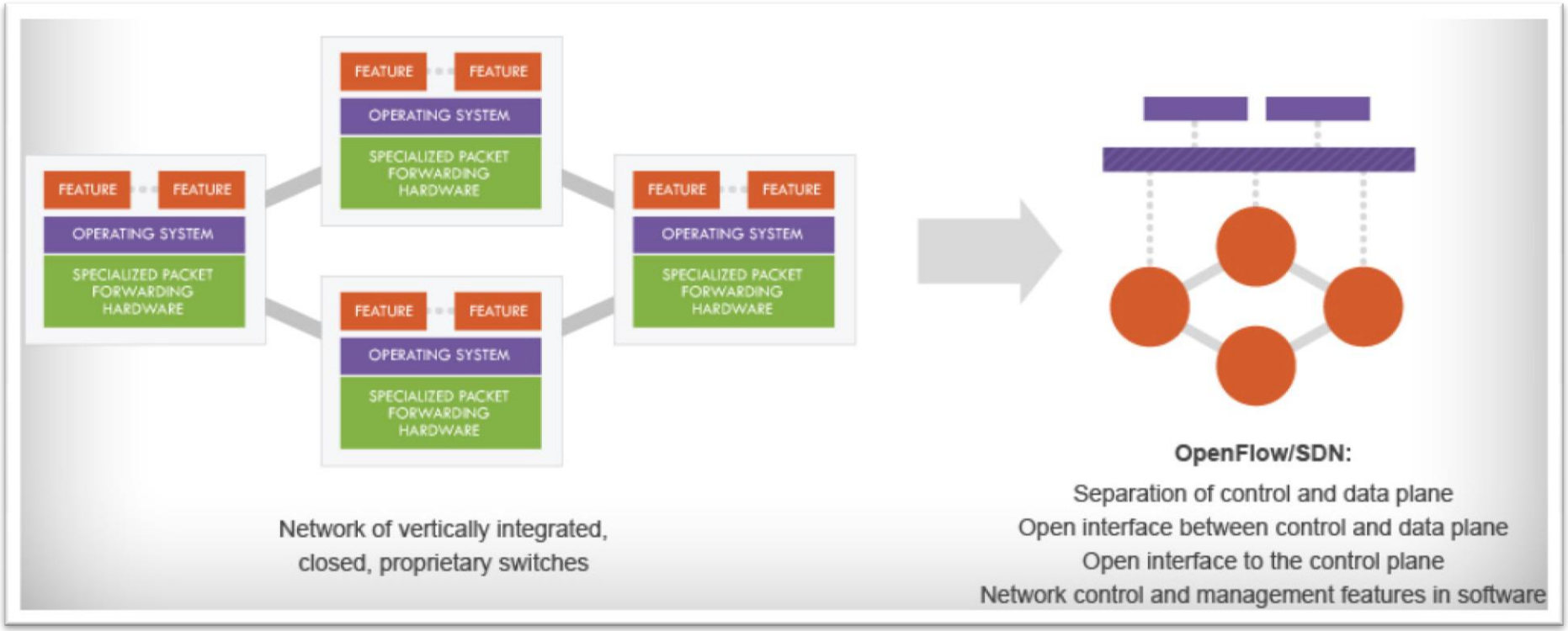
OpenFlow

(Or: *"Why can't I innovate in my wiring closet?"*)

ONF and Community grows fast!



What is the Difference?



Separation of control- and data plane!

How to configure your network?

- Static – Ask you Network Admin
- Automatic – Write some scripts
- Automatic – via Netconf, SNMP Protocols
- Dynamic – via Network Protocols
- Dynamic – via RADIUS, DIAMETER Protocols

Whats next?

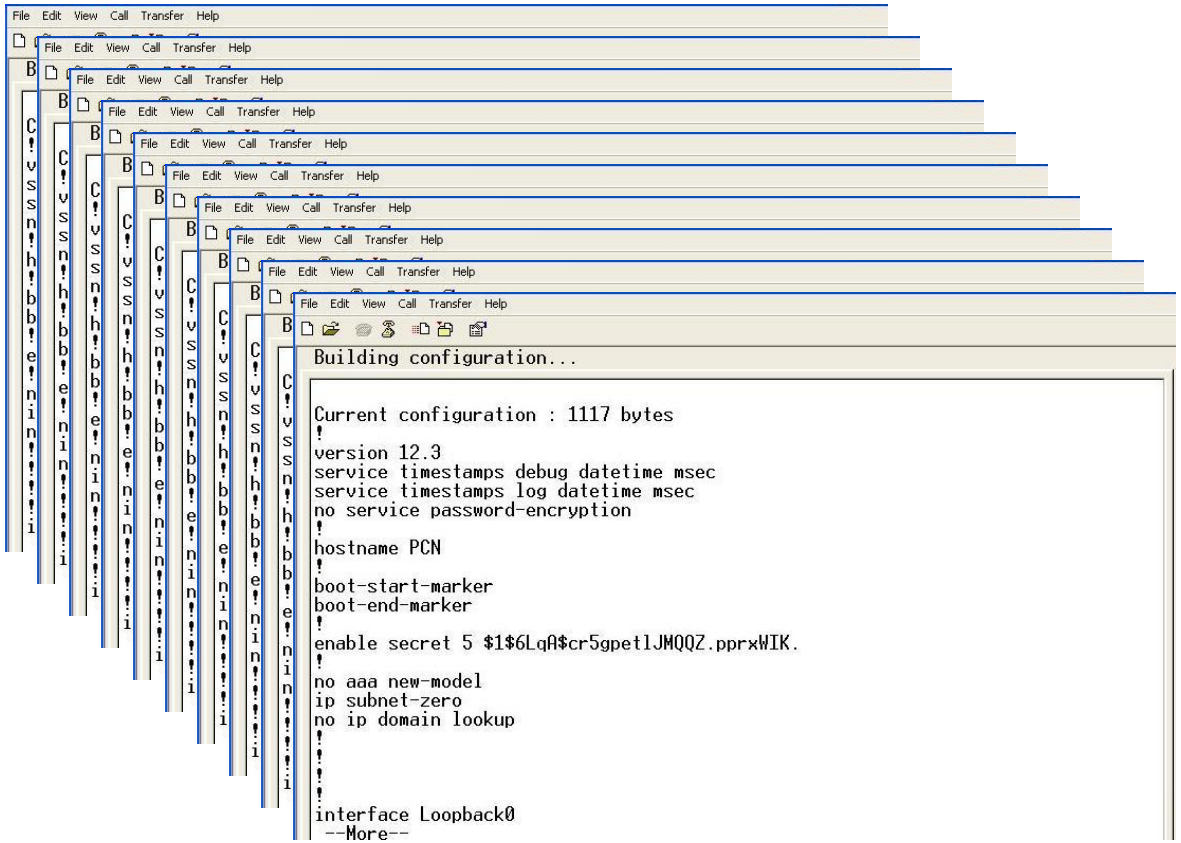
#1 Ask your Network Admin

File Edit View Call Transfer Help

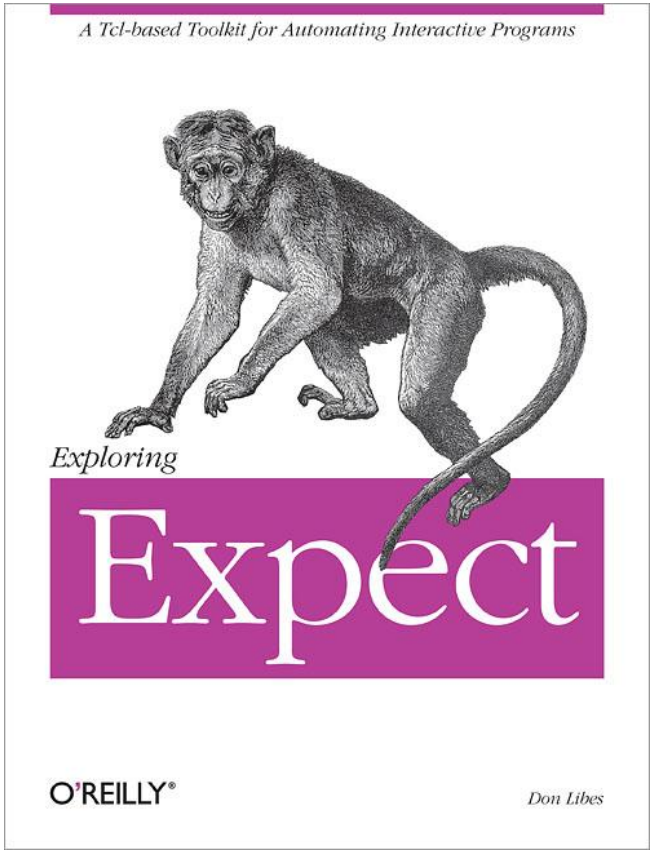
Building configuration...

```
Current configuration : 1117 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname PCN
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$6LqA$cr5gpetlJMQQZ.pprxWIK.
!
no aaa new-model
ip subnet-zero
no ip domain lookup
!
!
!
!
interface Loopback0
 --More--
```

#1 Ask your Network Admin

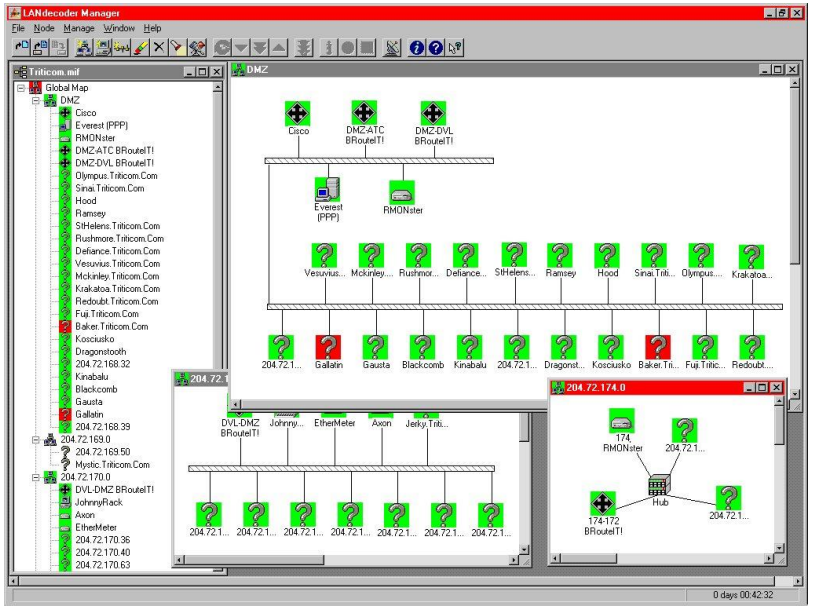


#1 CLI Automation



"Expect is quickly becoming a part of every UNIX user's toolbox. It allows you to automate Telnet, FTP, passwd, rlogin, and hundreds of other applications that normally require human interaction. Using Expect to automate these applications will allow you to speed up tasks and, in many cases, solve new problems that you never would have even considered before."

#3 Netconf / SNMP



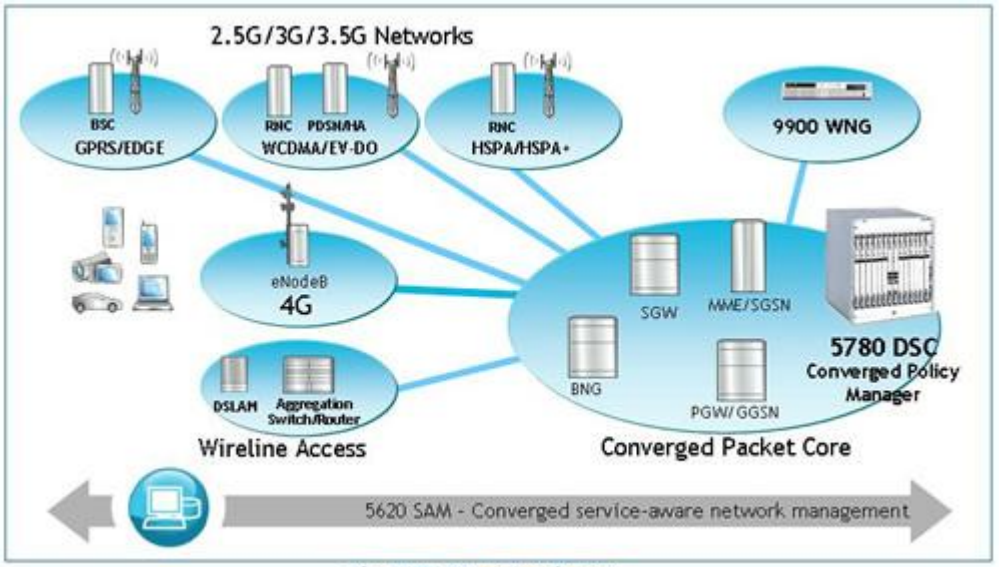
Formal protocols to configure and monitor network devices.

- Often read only!
- Back to CLI for configuration!

#3 Dynamic Configuration

Complex Standards and Protocols to adapt Network Behavior for requested Services.

- Carrier Centric approach
- Based on AAA Protocols (RADIUS, DIAMETER=)



The 5780 DSC network view

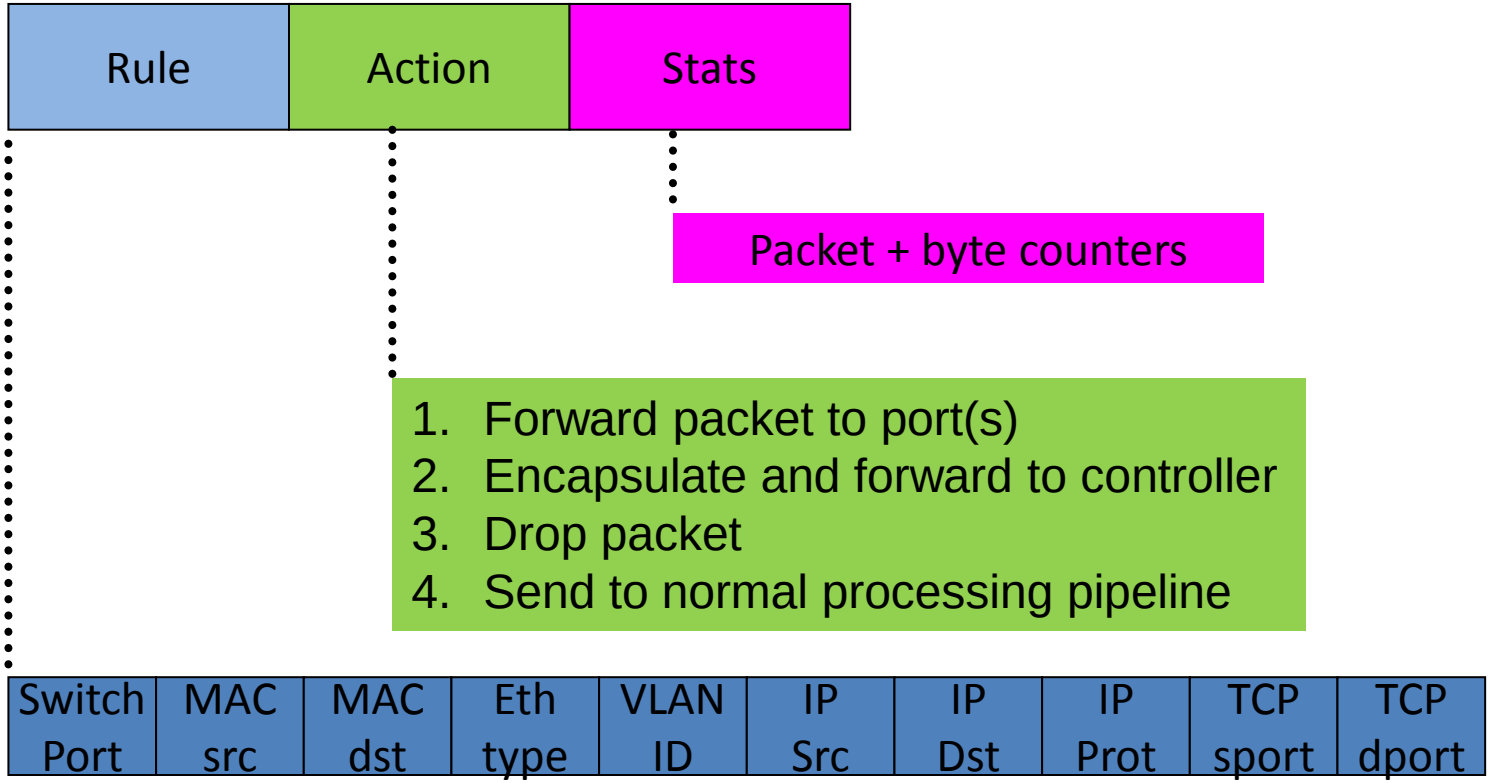
Conclusion!

You always **Configure** existing Functionality!

There is no common API to **Program** the network functionality!

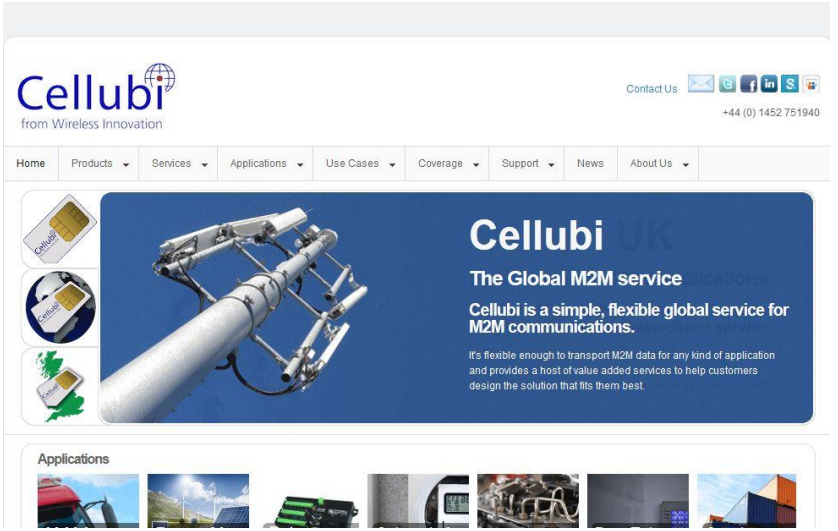
Program your Network!

With SDN/OpenFLow you get an **API** to **program** the **network behavior** with your language of choice!



#1 Cellubi m2m Network

- Production Network for m2m Communication
- Overlay Network over four 3G provider across the globe
- Fine granular policy control and 1:1 NAT funtionality



#2 Distributed BRAS

- Proof of Concept implementation
- Distributed PPPoE Termination close to the Edge of a Carrier FTTH Network
- Eliminates huge, central, costly BRAS installations
- Multicast Replication point is pushed to the edge for IPTV optimisation

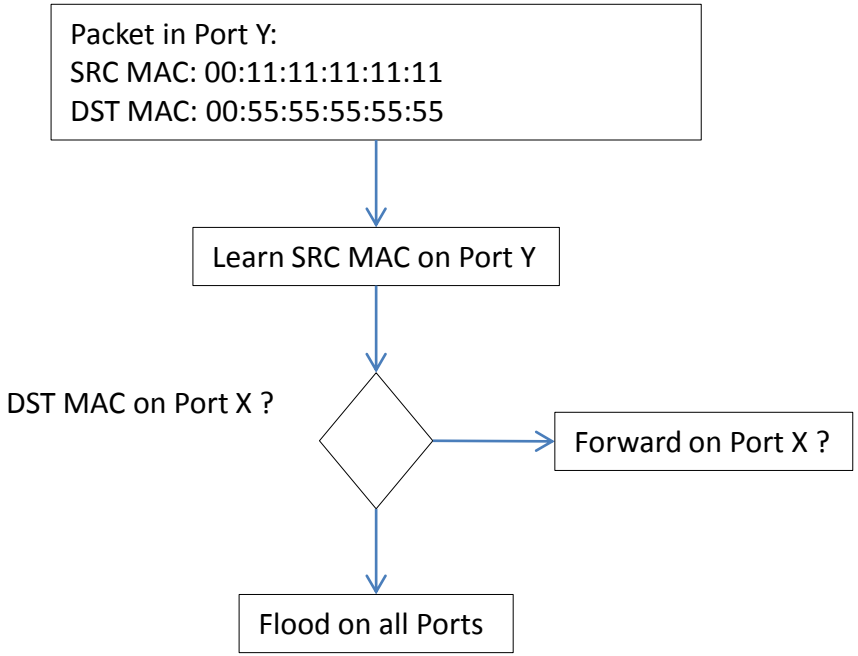
FlowER

Openflow Controller in Erlang

Classic Switch
vs.
Software Defined Networking (SDN)

Classic Network Switch

- Static control path (static configuration) VLANs, AAA, Filter, L3 Forwarding specified in configuration
- Limited matching in forwarding decision, mostly only things like MAC, VLAN and/or QoS tags
- Once a forwarding decision has been made, it can‘t be revised until it expires

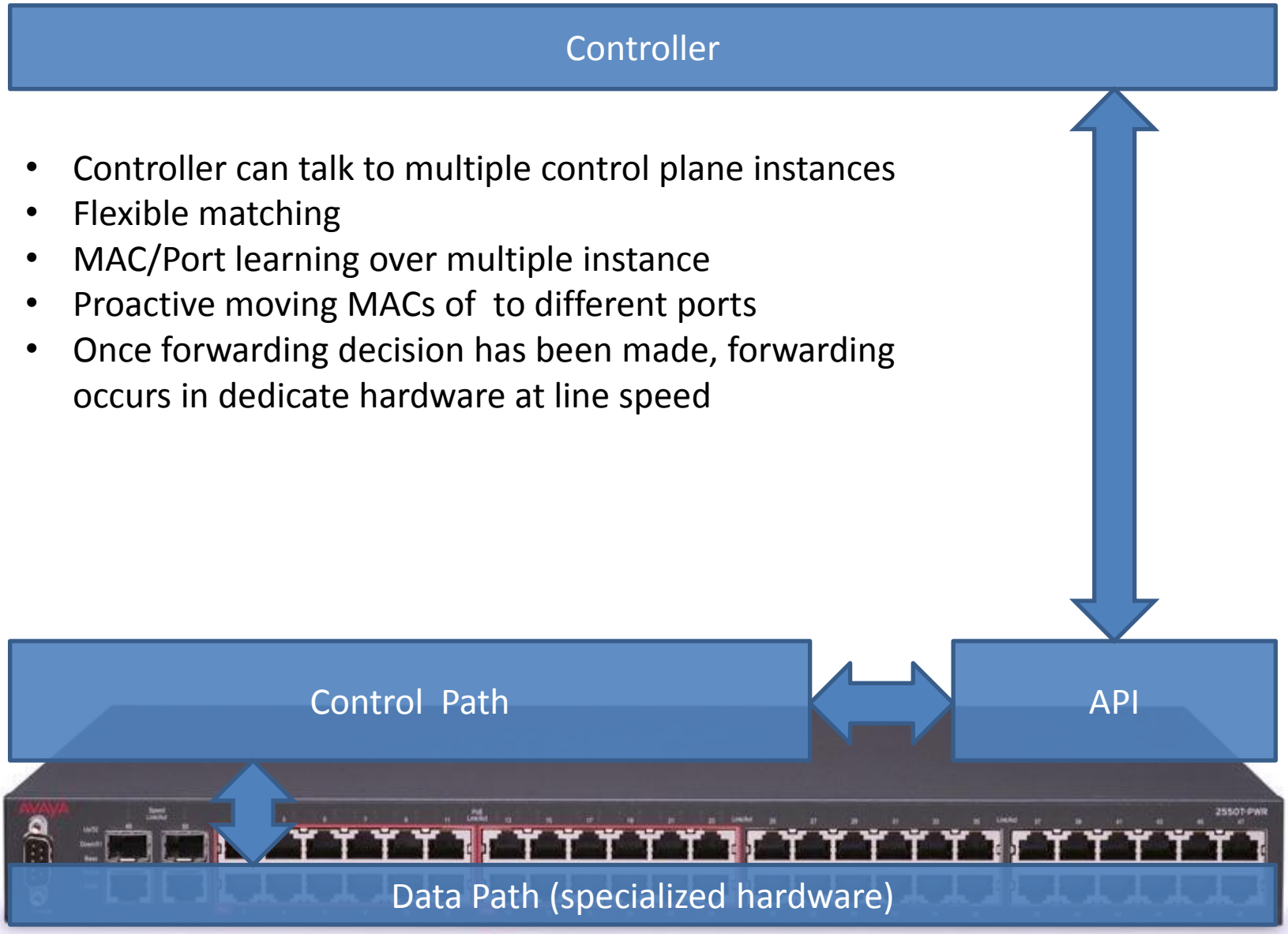


Interessting Questions:

- How can a MAC be moved to different port (e.g a VM migrating to a new host)
- Can a switched be sliced into different network partitions? E.g. Spanning Tree with multiple VLANs
- How can ports isolated from each other while still forming a single L2 domain? e.g. Ethernet-to-the-Home and Fibre-to-the-Home (FTTH) deployments

Software Defined Network (SDN)

- Control Plane decoupled from Forwarding Plane
- Network Control Plane accessible through API
- One Control instance can control multiple forwarding instances
- Can match on everything in the packet and in any combination (e.g. MAC+VLAN+IP+Port)
- Can alter packet during forwarding

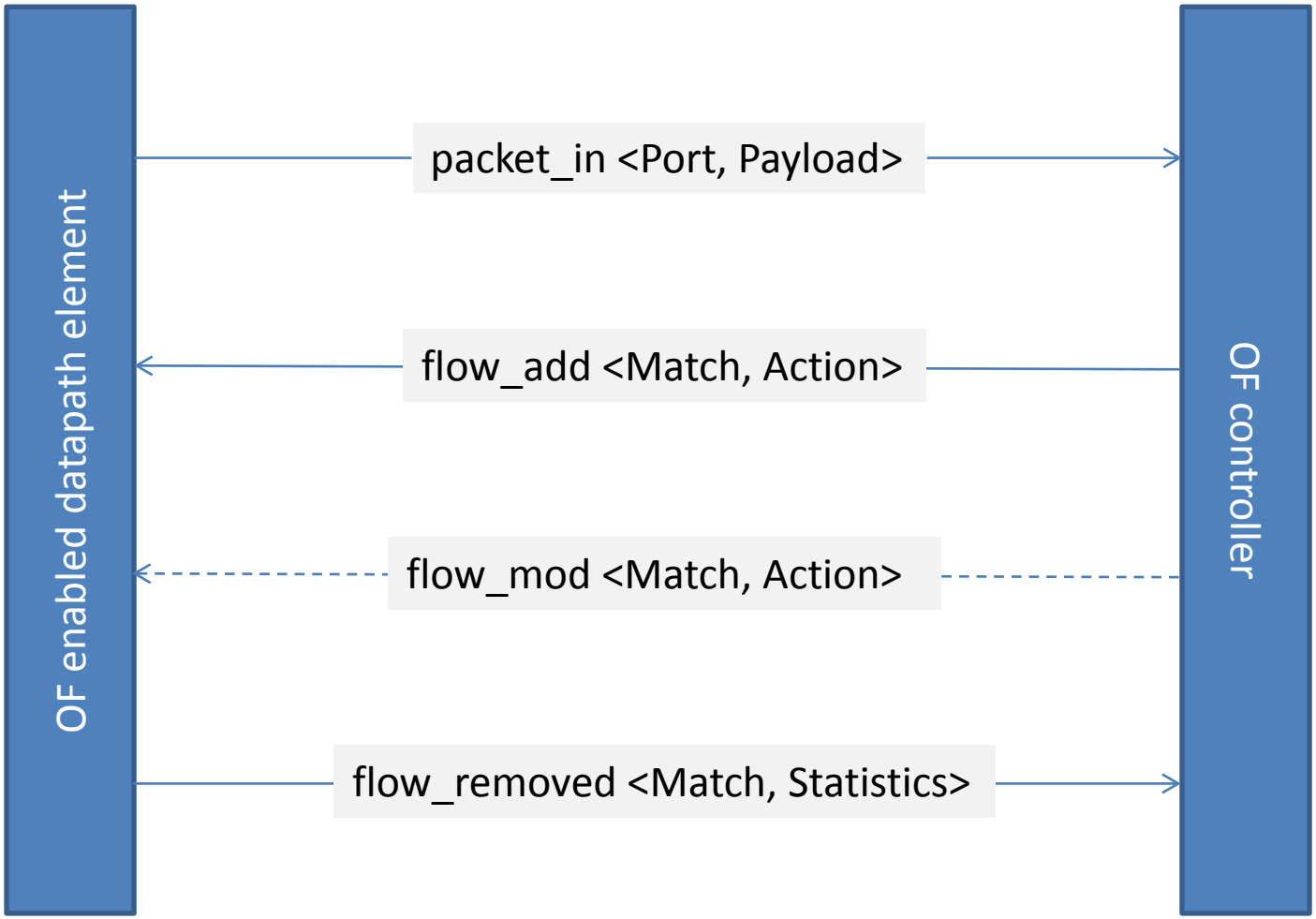


- Controller can talk to multiple control plane instances
- Flexible matching
- MAC/Port learning over multiple instance
- Proactive moving MACs of to different ports
- Once forwarding decision has been made, forwarding occurs in dedicate hardware at line speed

What is OpenFlow (OF)?

- Protocol specification and reference implementation of Software Defined Network (SDN)
- OF Datapath imlementation for Linux Kernel (replaces bridging) and FPGA board
- Used as basis in several commercial openflow enabled switches

Typical OF message flow



OpenVSwitch

- Linux Kernel Datapath, Controller and Configuration Database
- Can control OF enable hardware switch
- Used as software switch in Xen
- Supports for many standard management interfaces and protocols (e.g. NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag)
- Extensions over OpenFlow 1.0 Protocol

FlowER

- Modular platform for build OpenFlow switches in Erlang
- Concentrate on the Switch and Flow logic, Flower does the rest.

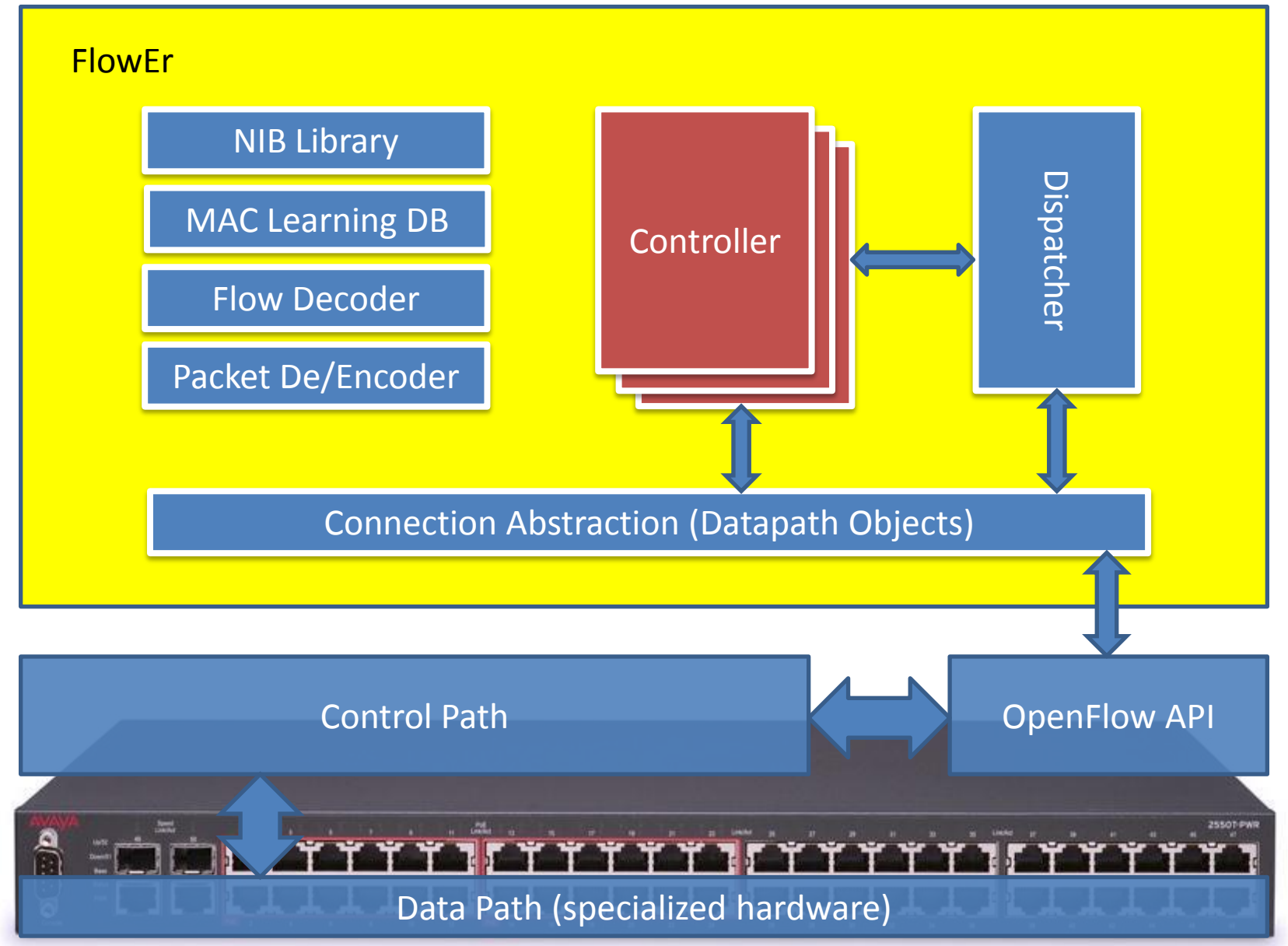
Provides:

- OpenFlow protocol, connection and data abstraction
- Tools and algorithms typicaly needed for a controller implementation

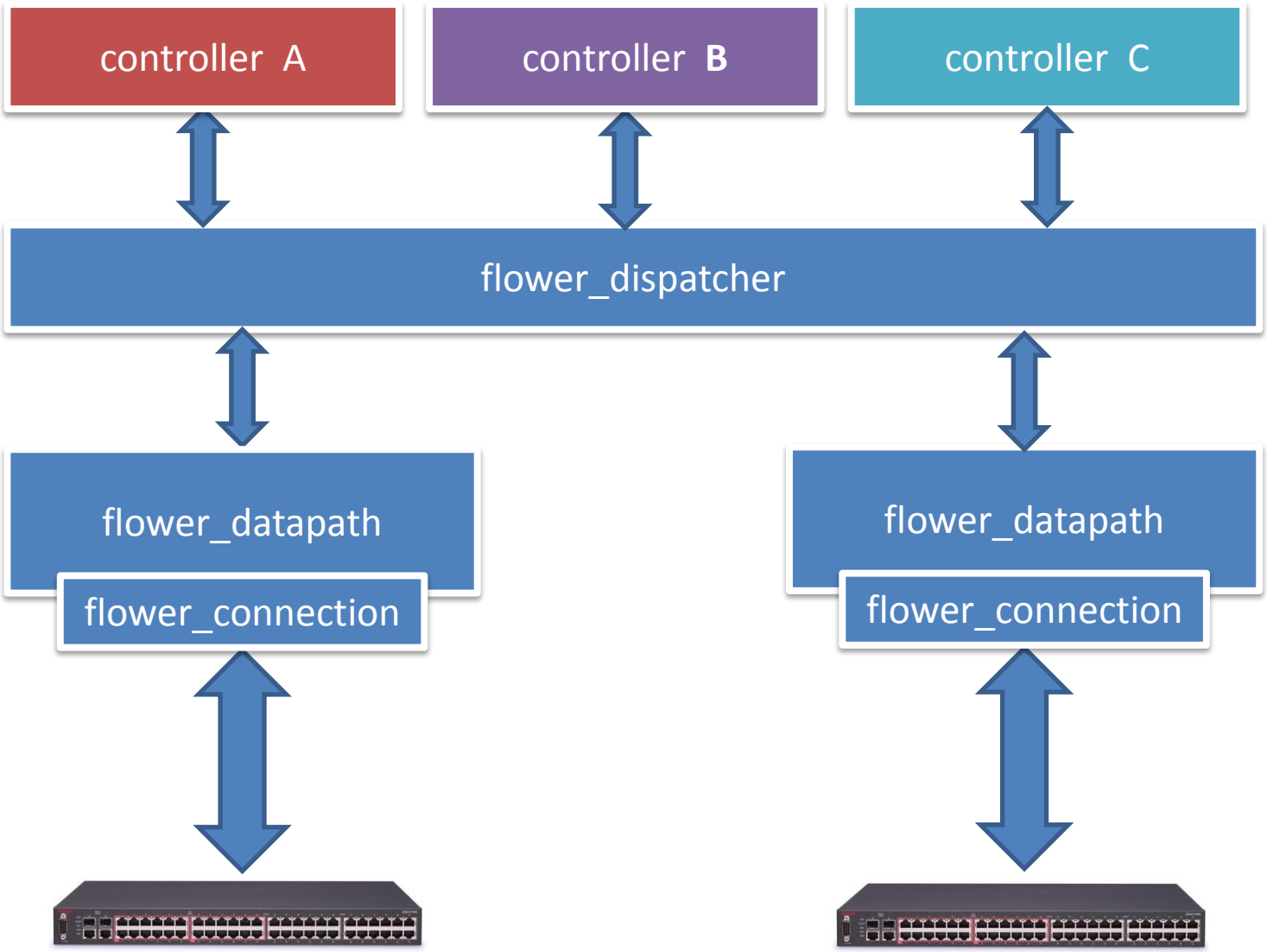
Why Erlang?

You probably already know?!?

- Flexible binary matching
- Fault isolation
- Concurrency
- ….


FlowEr
NIB Library
MAC Learning DB
Flow Decoder
Packet De/Encoder
Controller
Dispatcher
Connection Abstraction (Datapath Objects)
Control Path
OpenFlow API
Data Path (specialized hardware)

Flower Process Achitecture



Theory of Operation

- flower_datapath module reads events from network element
- flower_dispatcher forwards events to interested parties (controller)
- controllers are chainable through events
- controllers implement full switch or parts of it, generates new event processed by other controllers or sends answer to datapath

flower_datapath

- Abstracts the ‚real‘ OF enabled network element
- Manages connection setup and keep-alive
- Controllers register for events from datapath elements such as join, leave, packet-in
- Controllers messages to datapath elements such as packet-out or flow-mod

OpenFlow Packet Decoder

```
{[#ovs_msg{version = 1,type = features_reply,xid = 3,
       msg = #ofp_switch_features{datapath_id = 150876804345,
                      n_buffers = 256,n_tables = 2,
                      capabilities = [arp_match_ip,port_stats,table_stats,
                              flow_stats],
                      actions = [enqueue,set_tp_dst,set_tp_src,set_nw_tos,
                         set_nw_dst,set_nw_src,set_dl_dst,set_dl_src,strip_vlan,
                         set_vlan_pcp,set_vlan_vid,output],
                      ports = [#ofp_phy_port{port_no = 2,
                                 hw_addr = <<0,80,86,174,0,20>>,
                                 name = <<"eth2">>,config = [],
                                 state = [link_down],
                                 curr = [autoneg,copper],
                                 advertised = [autoneg,copper,'1gb_fd','100mb_fd','100mb_hd',
                                       '10mb_fd','10mb_hd'],
                                 supported = [autoneg,copper,'1gb_fd','100mb_fd','100mb_hd',
                                      '10mb_fd','10mb_hd'],
                                 peer = []},
                                           ....
                                          ]}}],
 <<>>}
```

Network Information Base (NIB)

NIB Library

Library for implementing:

- Network and network range based lookups
- Routing Tables

```
Nib = flower_nib4:new(),
Nib1 = flower_nib4:add({<<10,0,0,0>>, 8},
                       priv1, Nib),
{value, priv1} =
      flower_nib4:lookup(<<10,10,10,10>>, Nib).
```

MAC Learning DB

MAC Learning DB

- MAC address learning and lookup table
- Addres expiry
- Filter for special MAC addresses
- MAC to string formater

Flow Decoder

Flow Decoder

- Raw packet decoded into all matchable fields:
  - Src and Dst MAC Address
  - VLAN Id
  - L2 Protocol Type: IP, IPv6, ARP, LACP, ...
  - IP Protocol Type: UDP, TCP, ...
  - Src and Dst IP Address
  - Src and Dst Port
  - .....

De/encoder for network protocols

Packet De/Encoder

Decodes and build packet fragments for:

- IP, TCP and UDP header
- ICMP
- ARP
- more to come...

```
make_icmp({dest_unreach, pkt_filtered}, VLAN,
          DstMAC, SrcMAC, IPSrc, IPDst, Payload).
```

OpenFlow Matches

- OpenFlow matches specified as Erlang Term

```
#ofp_match{wildcards = 4178159,
           in_port = none,
           dl_src = <<0,0,0,0,0,0>>,
           dl_dst = <<0,0,0,0,0,0>>,
           dl_vlan = 0,dl_vlan_pcp = 0,
           dl_type = ip,nw_tos = 0,
           nw_proto = 0,
           nw_src = <<127,0,0,1>>,
           nw_dst = <<0,0,0,0>>,
           tp_src = 0,tp_dst = 0}
```

Given a decoded flow construct a match on IP Protocol and Src IP:

```
IP = <<127,0,0,1>>,
Flow = #flow{dl_type = ip,
             nw_src = IP, nw_dst = IP},
encode_ofp_matchflow([{nw_src_mask,32}, dl_type],
                     Flow).
```

Controller Implementations

flower_simple_switch:

- Sample learning Layer 2 switch
- Implements MAC learning only
- Less than 50 LOC!

flsc (not included):

- production ready Layer 3 switch and router in about 1100 LOC

Further Work

- IPv6 support
- Replace the MAC learning database with something **much** faster
- Improve and extend NIB Library
- Add load distribution in dispatcher
- …
- Extend Documentation ;-)

Available on Github

Repo: https://github.com/travelping/flower