Erlang SOLUTIONS

The OpenFlow Soft Switch

Krzysztof Rutka
@rptkr

History of OpenFlow

Stanford University

- Run experimental protocols in the campus networks,
- Exploit a common set of flow-table functions that run in many switches and routers,
- Provide an open protocol to control different switches and routers in a unified way,
- *OpenFlow: Enabling Innovation in Campus Networks* whitepaper.

http://www.openflow.org/documents/openflow-wp-latest.pdf

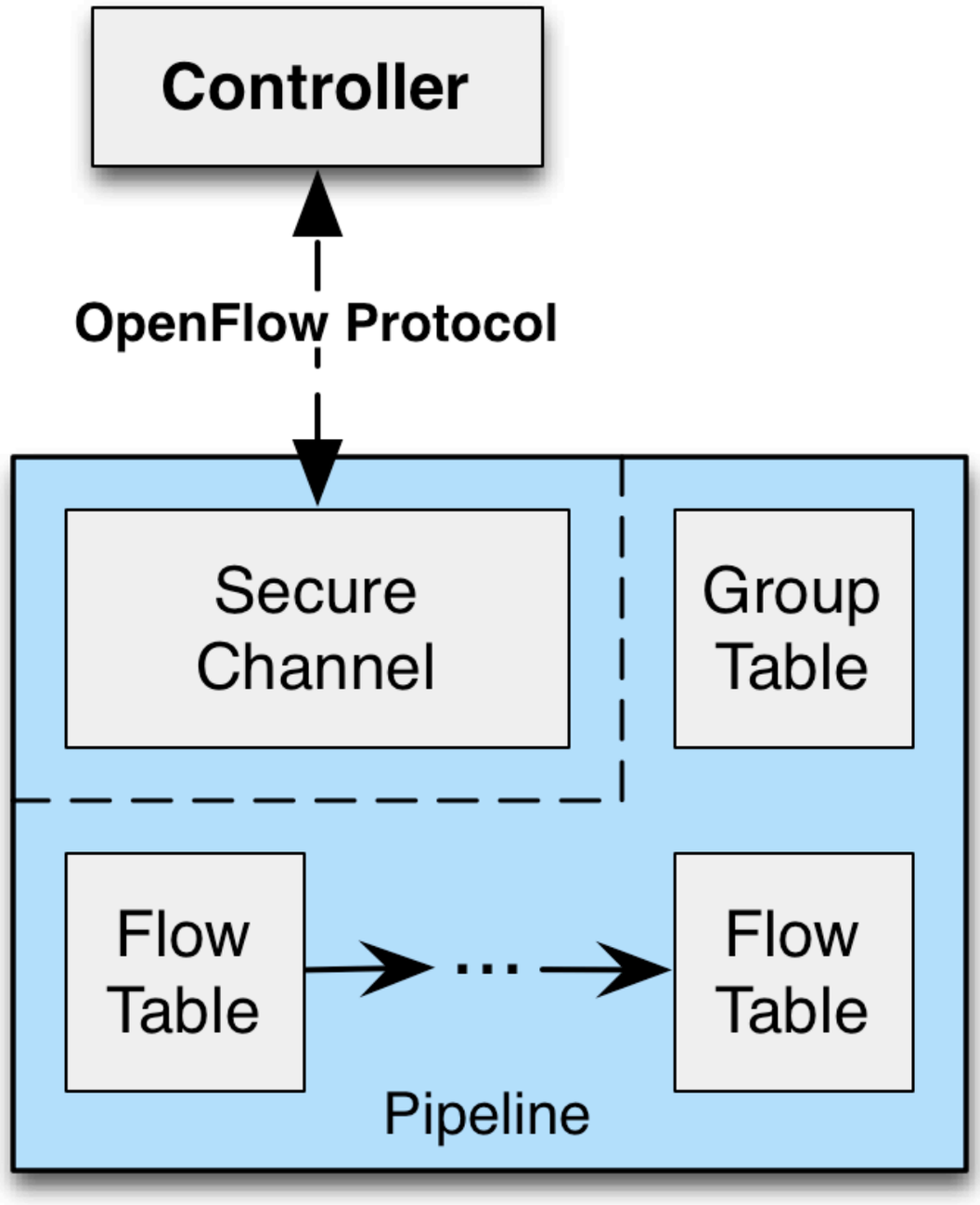
What is OpenFlow?

OpenFlow is the leading standard for Software Defined Networking.

Traditional networks	*OpenFlow/SDN*
Lots of protocols; STP, RIP, OSPF, BGP...	All computation and logic handled by software; OpenFlow Controller
Vendor specific interfaces	Common API; OpenFlow Protocol
Switches for L2 switching; Routers for L3 routing	One device; OpenFlow Switch; Flow Forwarding; L2 - L4

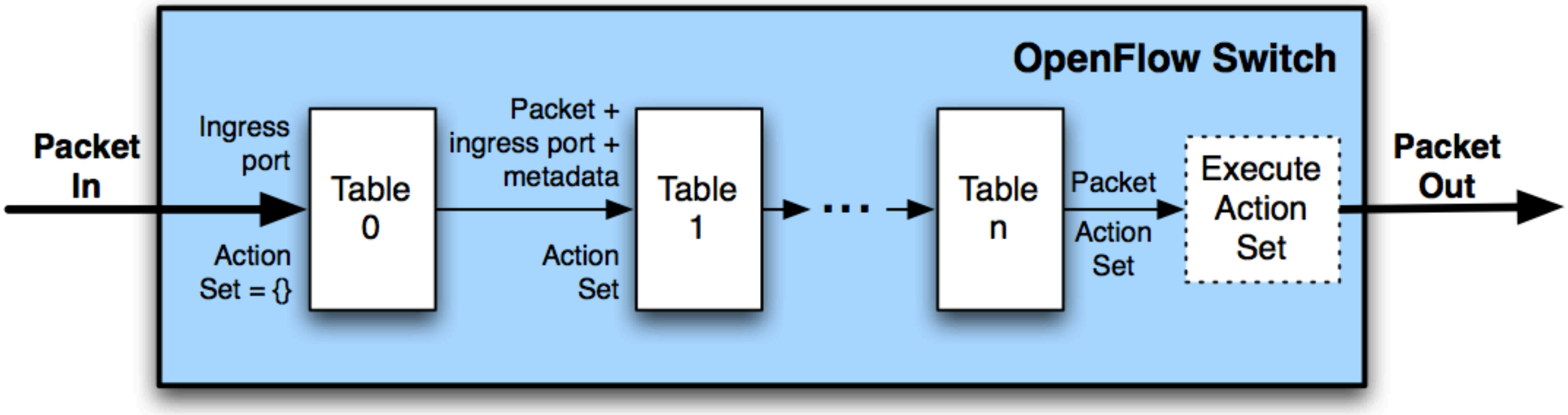
Open Network Foundation

How it works?



OpenFlow Switch

How it works?



OpenFlow pipeline

- Actions
- Instructions

OpenFlow specifications

OpenFlow 1.0 December 2009

https://www.opennetworking.org/images/stories/downloads/openflow/openflow-spec-v1.0.0.pdf

OpenFlow 1.1 February 2011

https://www.opennetworking.org/images/stories/downloads/openflow/openflow-spec-v1.1.0.pdf

OpenFlow 1.2 December 2011

https://www.opennetworking.org/images/stories/downloads/openflow/openflow-spec-v1.2.pdf

OpenFlow 1.3 TBA

OpenFlow ecosystem

Switches:

- OpenVSwitch http://openvswitch.org/
- of12softswitch (TrafficLab) https://github.com/CPqD/of12softswitch
- Hardware switches from HP, NEC...

Controllers (Frameworks):

- Floodlight http://floodlight.openflowhub.org/
- FlowER https://github.com/travelping/flower
- Beacon, Maestro, NOX/POX, Trema...

Others:

- OFTest http://oftest.openflowhub.org/

OpenFlow 1.0

LINC - pure OpenFlow soft switch

Full-featured

Full support for all OpenFlow 1.2 features

Reference implementation

Easily extensible

Pure Erlang implementation

Forwarding backend API

Infoblox

http://www.infoblox.com

Why Erlang? ;)

- Excellent development speed,
- Easy to read, modify and extend,
- Great for implementing binary protocols like the OpenFlow Protocol,
- Behaviours and callbacks as a way to make things modular.

OpenFlow Protocol 1.2 Library

- Erlang representation of OpenFlow Protocol structures and enumerations,
- Encoding/decoding of OpenFlow Protocol messages,
- OpenFlow Protocol parser,
- Support for older versions of the protocol.

OpenFlow Protocol 1.2 Library

```
-record(ofp_message, {
          experimental = false :: boolean(),
          version = 3 :: integer(),
          xid :: integer(),
          body :: ofp_message_body()
         }).

-record(ofp_hello, {}).

-record(ofp_echo_request, {
          data = <<>> :: binary()
         }).

-record(ofp_flow_mod, {
          cookie = <<0:64>> :: binary(),
          cookie_mask = <<0:64>> :: binary(),
          table_id = all :: ofp_table_id(),
          command :: ofp_flow_mod_command(),
          idle_timeout = 0 :: integer(),
          ...
```

OpenFlow Protocol 1.2 Library

OpenFlow Protocol **1.2** is not compatible with **1.0**.

- Message enumeration,
- Flat `ofp_match` structure,
- One field for tcp/udp,
- No instructions,
- Lots of other small incompatibilities.

OpenFlow Protocol 1.2 Library

gen_protocol

- Erlang behaviour,
- Simple encode/1, decode/1 callbacks,
- Convert structures from all versions to one common representation,
- OpenFlow Protocol 1.2 as a base.

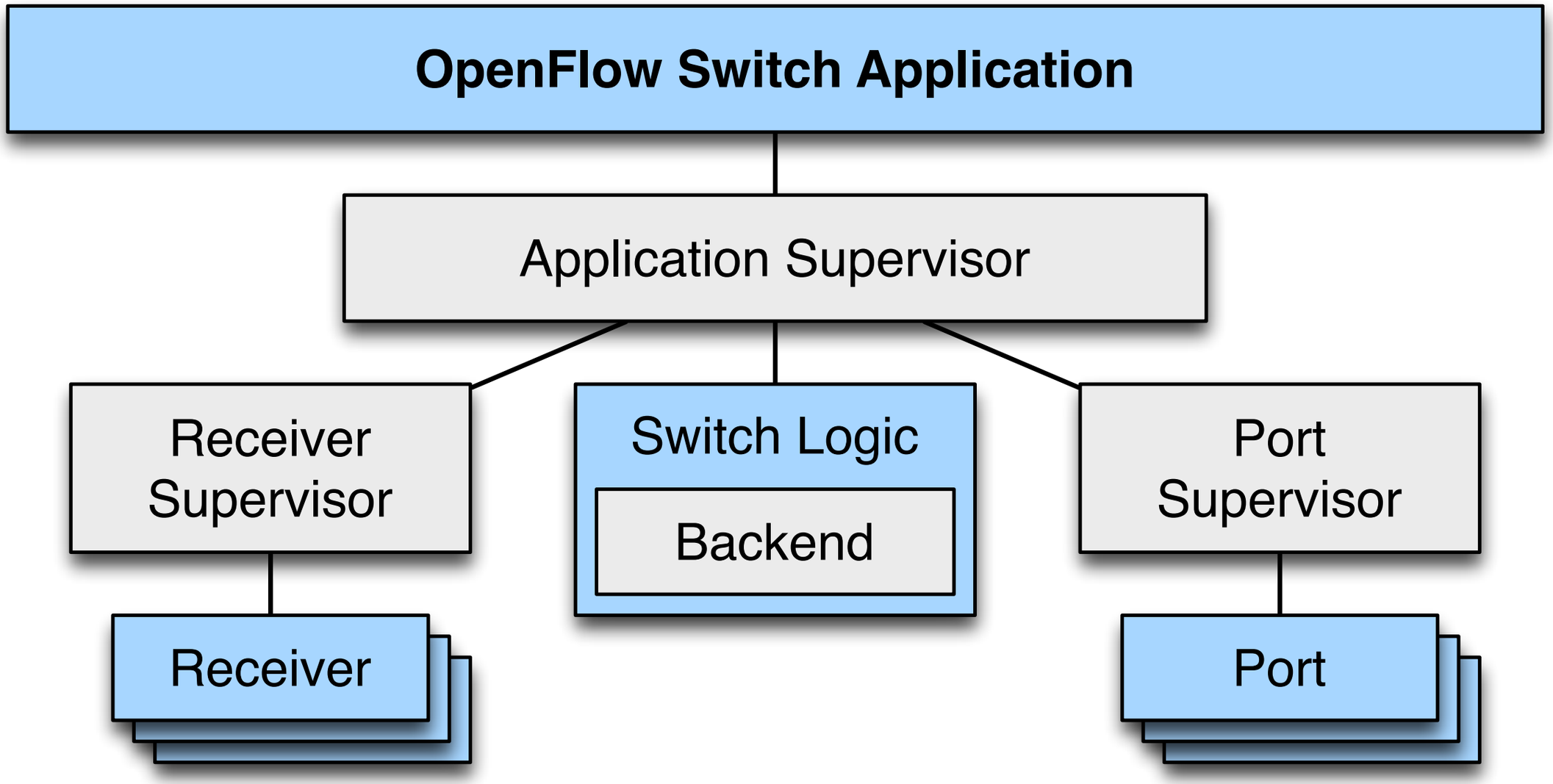
OpenFlow Protocol 1.2 Library

```
-module(gen_protocol).

%% Encode OpenFlow Protocol message
%% from Erlang representation to binary.
-callback encode(Message :: ofp_message()) ->
    {ok, Binary :: binary()} |
    {error, Reason :: any()}.

%% Decode OpenFlow Protocol message
%% from binary to Erlang representation.
-callback decode(Binary :: binary()) ->
    {ok, Message :: ofp_message()} |
    {error, Reason :: any()}.
```

Erlang implementation



Supervision tree

Common switch logic

- Communication with the OpenFlow Controllers,
- Managing controller roles,
- Managing the switch configuration,
- Handling simple messages independent of the actual forwarding logic.

Forwarding backend API

gen_switch

- Erlang behaviour,
- Separates common switch logic from the actual forwarding engine,
- Implements flow tables, group table, packet matching engine, port abstraction, etc.
- Callbacks to handle OpenFlow Protocol messages,
- Userspace, kernel, hardware...

Userspace implementation

ofs_userspace

- Implements `gen_switch` behaviour,
- Pure userspace Erlang implementation.

Ease of use
Extensibility **>** **Performance**

How to test a switch?

Quviq's QuickCheck

Simulate a real-life use cases, by generating:

- different switch configurations,
- different port configurations,
- random flows,
- random ethernet frames,
- OF Protocol messages with random content.

Current status

Where we are at the moment:

- implemented OF Protocol Library with support for versions 1.2 and 1.0,
- implemented OF Logical Switch in pure Erlang:
  - matching engine for all fields from 1.2,
  - support for (almost) all instructions, actions, reserved ports, groups, etc.

Where we are going:

- implemented support for OpenFlow 1.1,
- work on different forwarding backends.

Where to go next?

- Read the OpenFlow White Paper,

  http://www.openflow.org/documents/openflow-wp-latest.pdf
- Look at the OpenFlow Specification,

  https://www.opennetworking.org/images/stories/downloads/openflow/openflow-spec-v1.2.pdf
- Get involved.

FlowForwarding.org community and LINC will launch around June 11th!