AIR/IRI OVERVIEW

October 30, 2014

AIR/IRI Overview

AIR: A meta-language for specifying switch configurations

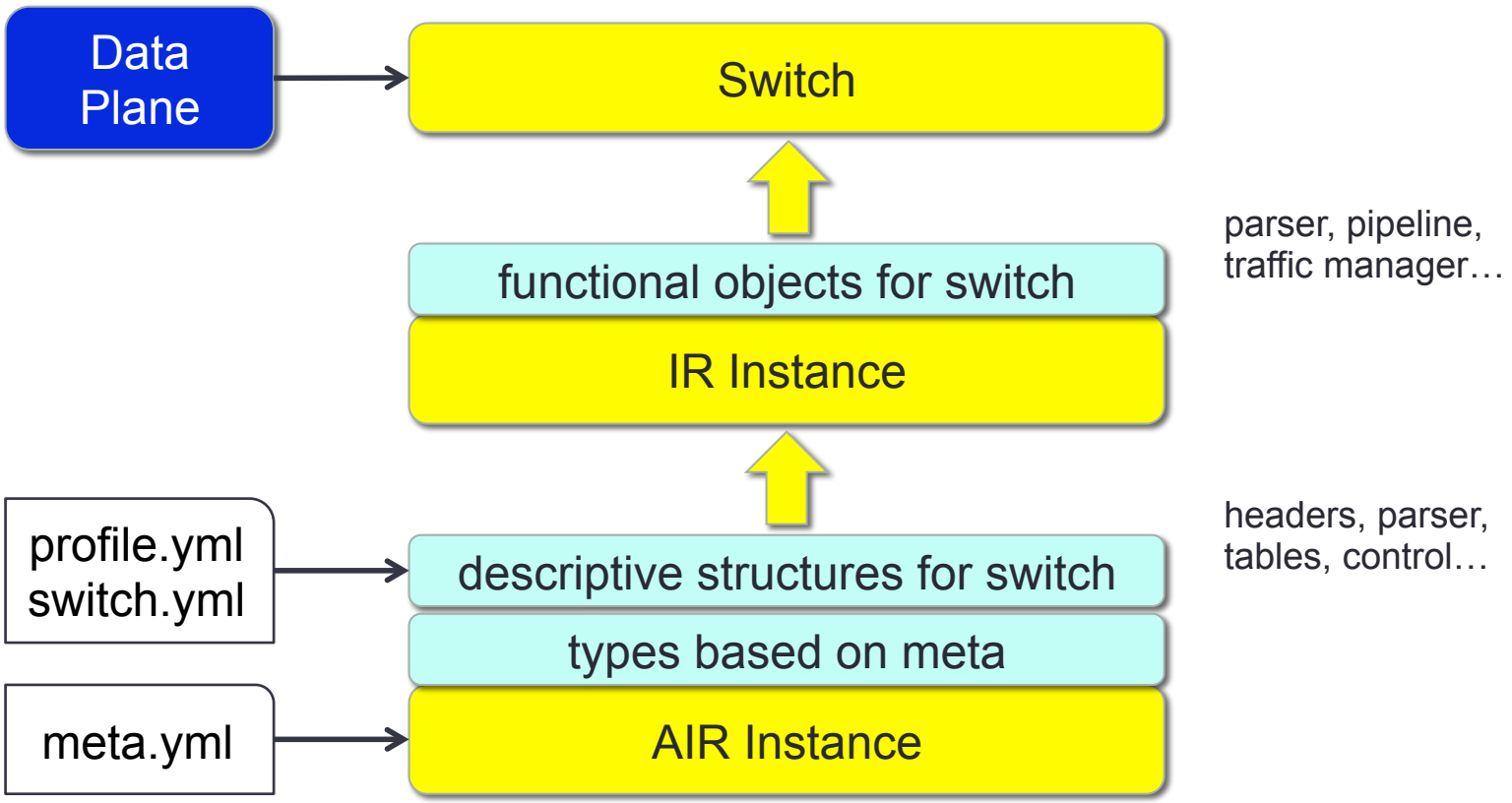
- Accepts a specification of types and their attributes
- AIR Python module converts YAML input into valid language structs

IRI: An “AIR interpreter”. AIR spec ➔ Running Switch

- Inherits from AIR and extends to functional components
  - Parser, pipeline (control + tables), traffic manager, packet
- Defines behavior for each type of object
- Instantiates data plane object and ties together

Tied together by a *Profile*: Defines Processors and Layout

AIR and IRI Architecture



Metalanguage for AIR: Types, Attrs

- `air_types` and `air_attributes` reserved identifiers

```
air_types :
  - table
  - header
  - metadata
  - action
  - parse_state
  - parser
  - control_flow
  - traffic_manager
  - processor_layout
```

```
air_attributes :
  table :
    - match_on
  header :
    - fields
    - max_depth # For header stack
  metadata :
    - fields
    - initial_values
...
```

Profiles: Processors and Layouts

Processors do something with packets

- Examples: Parser, Pipeline (with match+action), TM

Processor Layout: How are processors connected

- Currently: Profiles 0, 1 are static and linear

```
layout:
  type : processor_layout
  doc  : "The layout specification for the switch instance"
  port_count : 4 # TBD
  format : list  # Indicates static, linear connections
  implementation :
    - parser
    - ingress_flow
    - tm_queues
    - egress_flow
```

An IR Instance: A set of typed objects

- “Key” is the instance name.
- Use a graphical description for parsers, control flows, etc

```
ethernet :
  type : header
  doc : "The L2 header"
  fields :
    - dst_mac : 48
    - src_mac : 48
    - ethertype : 16
```

```
parser :
  type : parser
  doc : "Implementation of primary parser"
  format : dot
  start_state : ethernet_p
  implementation : >-
    digraph {
      ethernet_p -> ip4_p [value="0x0800"]
      ip4_p -> udp_p [value=6]
      ip4_p -> tcp_p [value=17]
    }
```

AIR/IRI Status

Code is checked in to ONF git:

- https://github.com/OpenNetworkingFoundation/air_iri

Switch running and passing packets

- Uses veth ports for data plane; uses OFTest dataplane
- Support for switch initialization for testing, experimenting present
- Documentation generated and available (doxygen; make doc)

Two basic profiles defined

- 0: parser => ingress => traffic manager
- 1: parser => ingress m+a => traffic manager => egress m+a

Next Steps

Implementation improvements

- E.g., exact match is linear search right now

More testing coverage

- Integrate with OFTest cases, exercise coverage tool

Interface to control plane

- OpenFlow interface
- Thrift interface for table update interfaces

Integrate with OVS data plane and configuration protocols

Develop alternative Profiles