AIR-IRI WALK THRU

January 16, 2015

Dan Talayco
ONF

Agenda

- Intro/Review
- Diagram with Layers
- The Layers Described
- Layers by Example
- Specifying Action Semantics (External Specifications)
- The Simulator
- Stretch goal: Output with Templates

Intro/Review, 1

- Goal is to provide a flexible means of describing packet processing logic

- Shooting for an Intermediate Representation
    - Compile to it from a high level language: Given a high level program, produce an instance of an IR description
    - Compile from it to real targets: Given an IR description, produce a configuration for a real target

Intro/Review, 2

- To provide flexibility, use multiple layers

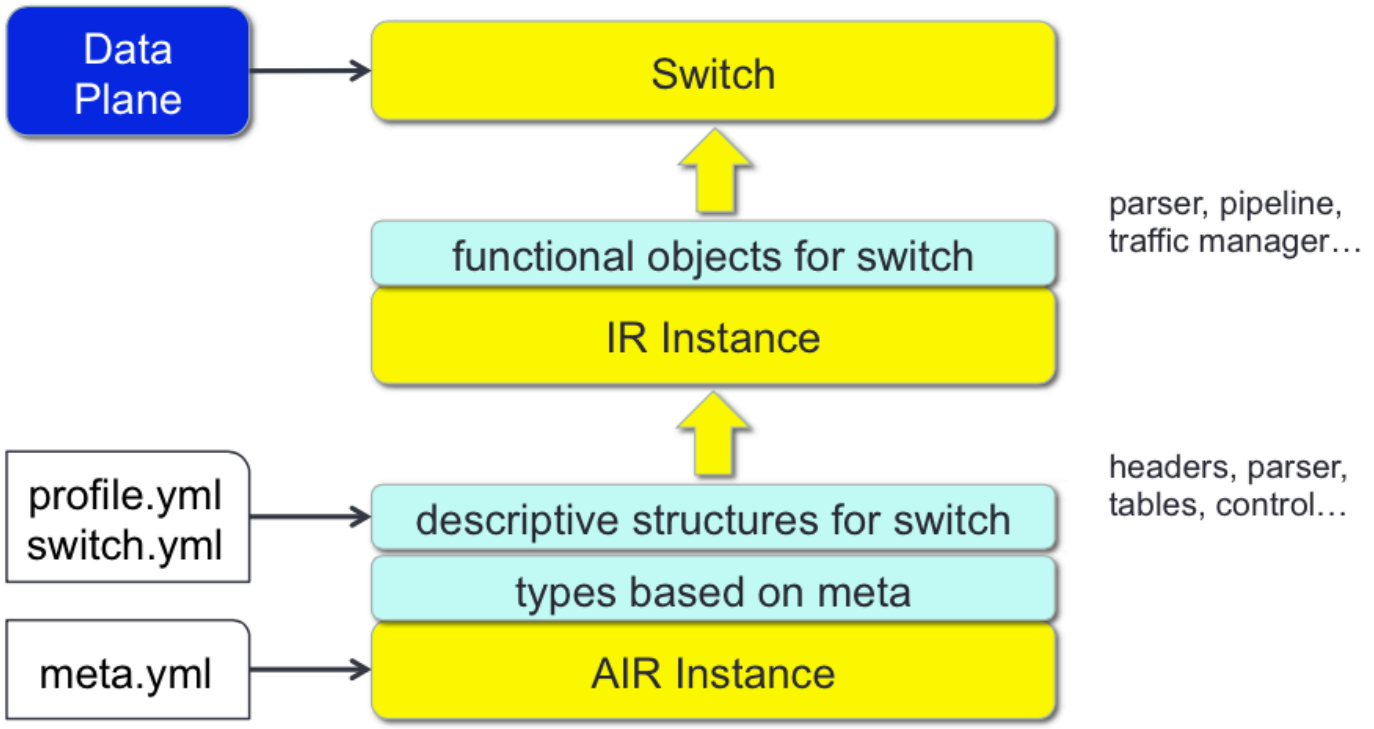
- At the bottom, define the domain of objects
- In the middle, define an instance in the domain
- At the top, provide tools for the forwarding instance; eg
  - A simulator for the instance
  - Templates making it easy to generate code from the instance
  - Wireshark dissectors

Layers: AIR, Instance, Sim

Simulation and generated assets

IR Instance: Defines the behavior

AIR: Defines the domain


Data Plane
Switch
parser, pipeline, traffic manager…
functional objects for switch
IR Instance
profile.yml
switch.yml
headers, parser, tables, control…
descriptive structures for switch
types based on meta
meta.yml
AIR Instance

The first layer: AIR

- AIR is a meta language: Use it to identify the objects that you want to manipulate
  - Input: meta.yml.
    - A YAML file describing the objects and their attributes you care about for programming your switch.
  - Output: air_instance class
    - A class which knows how to process a file conforming to meta.yml
    - Maintains sets of objects according to the types declared in meta.yml
    - Examples of objects are: Parser, Header, Field, Table, TM
- The result does not know about behavior of objects

The second layer: IR Instance

- IR: Given AIR + meta.yml, you can now specify an IR instance conforming to meta.yml
- First, define the behavior of the classes in meta.yml
- See iri/ directory for this:
  - Headers, primitive actions, traffic manager
  - iri_instance: uses air_instance to process an instance description (switch.yml) and then instantiates iri level objects.
- The instance
  - What headers, the parser, the tables, etc
  - Input 1: switch.yml. Instances of the Domain Classes from AIR
  - Input 2: profile.yml. How the instances are connected (layout)

The third layer: Something Useful

- See top level start.py
- Instantiates python objects with behavior defined in iri/
- Data plane borrowed from OFTest (submodule)
- Binds interfaces of data plane to IRI through iri/switch.py

- Another example: Generate code from templates

Example: Part 1. AIR meta types

- A list of types
- A subset of types which are “processors”; (more later)

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

```
# These objects must implement
# a process method
air_processors:
  - control_flow
  - parser
  - traffic_manager
```

Example: Part 2. AIR meta attributes

```
# All support type and doc
air_attributes :
  table :
    - match_on
  header :
    - fields
    - max_depth # hdr stack if > 1
  metadata :
    - fields
    - initial_values
  action :
    - format
    - parameter_list
    - implementation
  parse_state :
    - extracts
    - select_value # Optional
```

```
# CONTINUED
  control_flow : *graph_attributes
  parser :
    - format
    - implementation
    - start_state
  traffic_manager : # Experimental
    - queues_per_port
    - dequeue_discipline
    - egress_spec_map
  processor_layout:
    - format
    - implementation
    - port_count
```

The AIR module gives AirInstance

- Not usually seen unless working on a new meta.yml
- Base classes for the instance
- See http://sdnrocks.com/air_iri/html/classair_1_1air__instance_1_1AirInstance.html

Example Part 3: Declare Objects for IR Instance

These are in simple.yml

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

```
# Metadata object
pkt_md : # General metadata
  type : metadata
  doc : "General metadata for the packet"
  fields :
    # Virtual network instance identifier
    - vni : 16
```

```
# Action object
set_vni_a :
  type : action
  doc : "Set the VNI in metadata"
  format : action_set
  parameter_list :
    - vni_id
  implementation : >-
    modify_field(pkt_md.vni, vni_id);
```

Example Part 3: (continued) Declare Objects for IR Instance

Here’s the full parser

```
ethernet_p :
  type : parse_state
  doc : "Parse state for ethernet"
  extracts :
    - ethernet
  select_value :
    - ethernet.ethertype

vlan_p :
  type : parse_state
  doc : "Parse state for vlan tag"
  extracts :
    - vlan_tag
```

```
parser :
  type : parser
  doc : "Implementation of primary parser"
  format : dot
  start_state : ethernet_p
  implementation : >-
    digraph {
      ethernet_p -> vlan_p [value="0x8100"]
      ethernet_p -> vlan_p [value="0x9100"]
    }
```

The IRI developer implements behavior

- The IR classes inherit from the syntactic AIR classes
- The IR definer extends these classes with behavior
- Table class: http://sdnrocks.com/air_iri/html/classiri_1_1table_1_1Table.html
- Header class: http://sdnrocks.com/air_iri/html/classiri_1_1header_1_1HeaderInstance.html
- May define new classes to interact with AIR objects
  - Example ParsedPacket: http://sdnrocks.com/air_iri/html/classiri_1_1parsed__packet_1_1ParsedPacket.html
- Processors must implement “processor” method

Actions

- This treatment of actions is specific to the IR example being discussed
- Could implement completely different action semantics (with a different meta, or just a different representation)
- Primitive actions derived from references in action objects
- The IR implementer defines the behavior in subclasses
- See http://sdnrocks.com/air_iri/html/annotated.html

Layouts: Putting Processors Together

- The file profile_1.yml has a profile object
- This references objects from simple.yml (instance yaml)
- Currently just supports linear layout of objects (list format)

```
layout:
  type : processor_layout
  doc  : "The layout specification for the switch instance"
  port_count : 4
  format : list
  implementation :
    - parser
    - ingress_flow
    - tm_queues
    - egress_flow
```

The Simulator

- For a more complicated example, checkout l3.yml.
- Uses the same meta.yml
- Invoked simply by referencing the YAML file with start.py

- `sudo ${PYPATH} ./start.py -v profile_0.yml l3.yml`

  (Need PYPATH to find oftest properly)

Stretch Goal: Templating code

- Not quite ready, but almost.
- Example code pushed to templates branch
- Instantiate an instance of the IR (no dataplane needed)
- Pass instance.air_object_map to the templating engine