OpenFlow rules interactions: Definition and detection

Roberto Bifulco, Fabian Schneider
NEC Laboratories Europe, NEC Europe Ltd.
roberto.bifulco@neclab.eu
fabian.schneider@neclab.eu
Legacy Networks

- Today's networks are defined by the boxes that compose them
- A single box includes hardware, operating system, applications
- Boxes are interconnected to provide specific network functions

Software Defined Networks

- Networks are defined by software programs
- Boxes are dummy but programmable
- The network is general purpose, network functions are implemented as applications on top of a Network OS
Programmable Network

Tools
- NICE
- ndb
- FlowChecker
- HSA
- Mininet

Abstractions and Frameworks
- Onix
- FlowVisor
- Trema

Programming Languages
- Frenetic
- NetCore

Operating Systems
- ProgrammableFlow
- nox
- FloodLight
- BEACON

Machine Languages
- OpenFlow

DataPlane
(Hardware/Software)

Legacy interfaces
OpenFlow-enabled Switches (OFS) behavior is configured through OpenFlow Rules (OFR).

**OpenFlow Rule** is composed by:
- **Match Set**: identifies a flow;
- **Action Set**: defines the actions executed on each packet of the flow;
- **Priority**: Is used to relatively order rules in a switch.

**Rule representation**: \{ [Match set], [Action set], priority \}

OFRs **timeouts** are not considered: we are interested in rules installed in a switch in a **fixed point in time**.
How Rules are Applied to Flows

When two rules match the same flow, only the one with highest priority is applied.

- Rule 1: { [NW_DST: IP-Z], [out: 12], 2 }
- Rule 2: { [DL_DST: MAC-A], [out: 10], 1 }

Packet: {DL_DST: MAC-A, NW_DST: IP-Z}
- Only **Rule 1** is applied;

If priorities are equal, the action is **explicitly undefined** (i.e., it is an error!!)
Switch Behavior

The switch behavior can be defined only looking at the whole set of installed rules: Interaction of rules is important!

Development of OF applications is the process of defining when, where and what OF rules have to be installed at switches;

- No methodology to check rules interactions
- No tools to automate rules interactions checking
Relations between Two Rules

**Match set**
- Disjoint
- Exactly matching
- Subset/Superset:
  - [nw_dst: A] *is superset of* [nw_dst: A, t4_dst: 80]
- Correlated:
  - [nw_dst: A]
  - [nw_src: B]

**Action set**
- Disjoint
- Equal
- Subset/Superset:
  - [out: 12] *is subset of* [set-nw-dst: IP-A, out: 12]
- Related:
  - [out: 12] *is related to*
  - [out: 10]

Match sets and Action sets impact the detection of interacting rules in *slightly* different way.
Categorization of Rule Interactions

An interaction is defined looking at:

- Match sets relations;
- Action sets relations;
- Rules Priorities.

Interactions are categorized into:

- Duplication (trivial)
- Redundancy
- Generalization
- Shadowing
- Correlation (effect depends)
- Inclusion (special case of Shadowing)
- Extension (special case of Generalization)
Examples (1): Shadowing and Redundancy

<table>
<thead>
<tr>
<th>Rule No</th>
<th>Match NW_SRC</th>
<th>Match NW_DST</th>
<th>Action Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.1.2.3</td>
<td>*</td>
<td>Out: 10</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>192.168.0.1</td>
<td>Out: 12</td>
</tr>
<tr>
<td>3</td>
<td>10.1.2.3</td>
<td>192.168.0.1</td>
<td>Set NW_DST=2.2.2.2; Out: 11</td>
</tr>
<tr>
<td>4</td>
<td>10.1.2.3</td>
<td>*</td>
<td>Out: 11</td>
</tr>
<tr>
<td>5</td>
<td>*</td>
<td>*</td>
<td>Out: 11</td>
</tr>
</tbody>
</table>

- Shadowing can lead to **network errors**
- Redundancy and shadowing lead to **wasted resources**
  - TCAM, switch tables
  - Control channel interactions
### Examples (2): Generalization

Generalization rules might be needed when flows need “special” treatment to ensure proper operation of remaining flows.

<table>
<thead>
<tr>
<th>Rule No (ordered by priority)</th>
<th>Match NW_SRC</th>
<th>Match NW_DST</th>
<th>Action Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.1.2.3</td>
<td>*</td>
<td>Out: 10</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>192.168.0.1</td>
<td>Out: 12</td>
</tr>
<tr>
<td>3</td>
<td>10.1.2.3</td>
<td>192.168.0.1</td>
<td>Set NW_DST=2.2.2.2; Out: 11</td>
</tr>
<tr>
<td>4</td>
<td>10.1.2.3</td>
<td>*</td>
<td>Out: 11</td>
</tr>
<tr>
<td>5</td>
<td>*</td>
<td>*</td>
<td>Out: 11</td>
</tr>
</tbody>
</table>
Implemented in Python (v. 2.7) and integrated in the NOX controller. Tested on a single core of an Intel CPU E7600 @ 3.06GHz

Number of defined fields (i.e., non wildcard)
Developing an OpenFlow control application can be an hard task

We provided a definition for the possible OpenFlow rules interactions and a tool to check them in an OpenFlow switch:

We foresee several possible applications:
- Network debugging
- OF Rule-related network “invariants” to support programming paradigms
- Rules set optimizations for flow table space and control messages savings
- ...

We already applied this tool to ease the development of an advanced OpenFlow controller for the support of a Follow-Me Cloud scenario

Our implementation runs in a reasonable time, which makes it suitable for use during design/debug phase. We expect an optimized implementation to be usable also in production phase.
Empowered by Innovation