Classbench-ng: recasting Classbench after a decade of network evolution

Gianni Antichi
gianni.antichi@cl.cam.ac.uk
http://www.cl.cam.ac.uk/~ga288/
Packet Classification is the fine art of: Matching header fields of incoming packets against a set of rules and performing the corresponding action
Packet Classification is the fine art of: Matching header fields of incoming packets against a set of rules and performing the corresponding action

• packet forwarding
Packet Classification is the fine art of:

Matching header fields of incoming packets against a set of rules and performing the corresponding action

- packet forwarding
- application of security policies
Packet Classification is the fine art of: Matching header fields of incoming packets against a set of rules and performing the corresponding action

• packet forwarding
• application of security policies
• application-specific processing
Packet Classification is the fine art of:

Matching header fields of incoming packets against a set of rules and performing the corresponding action

- packet forwarding
- application of security policies
- application-specific processing
- application of QoS guarantees
Is the packet classification mechanic still a problem in 2017?
Is the packet classification mechanic still a problem in 2017?

- Increasing transfer rate $\Rightarrow$ faster classification
Is the packet classification mechanic still a problem in 2017?

- Increasing transfer rate => **faster** classification
- Increasing number of rules => **larger** data structures
Is the packet classification mechanic still a problem in 2017?

- Increasing transfer rate => faster classification
- Increasing number of rules => larger data structures
- Growing adoption of IPv6 => longer IP prefixes
Is the packet classification mechanic still a problem in 2017?

- Increasing transfer rate => faster classification
- Increasing number of rules => larger data structures
- Growing adoption of IPv6 => longer IP prefixes
- Adoption of OpenFlow/SDN => more header fields
Is the packet classification mechanic still a problem in 2017?

- Increasing transfer rate => faster
- Increasing number of rules => larger data structures
- Growing adoption of IPv6 => longer IP prefixes
- Adoption of OpenFlow/SDN => more header fields

NEW CHALLENGES FOR HW DESIGNERS
lot of research effort in the past identified better packet classification techniques leveraging the **characteristics of real rule sets** for faster searches.
lot of research effort in the past identified better packet classification techniques leveraging the characteristics of real rule sets for faster searches.

the capacity and efficiency of Ternary Content Addressable Memories (TCAMs) are also subject to the characteristics of rule sets.
lot of research effort in the past identified better packet classification techniques leveraging the characteristics of real rule sets for faster searches.

the capacity and efficiency of Ternary Content Addressable Memories (TCAMs) are also subject to the characteristics of rule sets.

new algorithms needs to be benchmarked
...characteristics of real rule sets....

Dear DC/network/cloud operator, can you please send me a snapshot of your forwarding tables so I can use them?

WE WANT YOU!
Not sure it is going to be so easy....
Not sure it is going to be so easy....

OR

Create a tool for automatic *generation of synthetic rule sets* with the same characteristic of real ones.
In a nutshell....

input parameters

Classification rules
In a nutshell....

seed

input parameters

Classification rules
Available tools use as an input either statistic distributions of real sets [1] or user-defined characteristics [2]

Available tools use as an input either statistic distributions of real sets [1] or user-defined characteristics [2].

[1] is good if we are seeking for an output as close as possible to a real set, but can become easily obsolete if the seeds are not updated accordingly.

Available tools use as an input either statistic distributions of real sets [1] or user-defined characteristics [2].

[1] is good if we are seeking for an output as close as possible to a real set, but can become easily obsolete if the seeds are not updated accordingly.

[2] is more flexible in the long term, but does not guarantee output characteristics similar to real sets.


What we want is the best of both worlds:
What we want is the best of both worlds:

• Fidelity
What we want is the best of both worlds:

- Fidelity
- Longevity
What we want is the best of both worlds:

- Fidelity
- Longevity

Classbench-<ng is the tool for you!!!
OpenFlow rule set

OpenFlow Analysis

OpenFlow seed

OpenFlow Generation

IPv4 seed

Improved ClassBench

IPv6 seed

IPv6 Generation

IPv4 5-tuples

IPv6 5-tuples
In contrast to Classbench, Classbench-ng can successfully generate IPv4, IPv6 and OpenFlow rules.
Classbench-ng, as Classbench, relies on seeds as input for the rule generation.
Classbench-ng is based on Classbench, but improves its IPv4 generation fidelity.
Classbench-ng provides modules for IPv6 and OpenFlow rules generation
Classbench-ng includes an analysis module, which is able to extract seeds from input rule sets.
The main idea behind **Classbench-ng** is to create a repository where researchers can upload just the **seeds** of the rule sets they might have/use.

This will foster **reproducibility**, but also will help researchers that do not have access to real rule sets to create synthetic ones.
To start with, Classbench-ng already provide some initial seeds, created after we analysed the following rule sets:
<table>
<thead>
<tr>
<th>Name</th>
<th>Prefixes or Rules</th>
<th>Source</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 Prefix Sets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eqix_2005</td>
<td>164 455</td>
<td></td>
<td>2005-07-02</td>
</tr>
<tr>
<td>rrc00_2015</td>
<td>571 351</td>
<td><a href="http://data.ris.ripe.net/">http://data.ris.ripe.net/</a></td>
<td>2015-07-02</td>
</tr>
<tr>
<td>rrc00_2005</td>
<td>168 525</td>
<td></td>
<td>2005-07-02</td>
</tr>
<tr>
<td>IPv6 Prefix Sets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eqix_2015</td>
<td>23 866</td>
<td></td>
<td>2015-07-02</td>
</tr>
<tr>
<td>eqix_2005</td>
<td>658</td>
<td></td>
<td>2005-07-02</td>
</tr>
<tr>
<td>rrc00_2015</td>
<td>24 162</td>
<td></td>
<td>2015-07-02</td>
</tr>
<tr>
<td>rrc00_2013</td>
<td>14 374</td>
<td><a href="http://data.ris.ripe.net/">http://data.ris.ripe.net/</a></td>
<td>2013-07-02</td>
</tr>
<tr>
<td>rrc00_2005</td>
<td>499</td>
<td></td>
<td>2005-07-02</td>
</tr>
<tr>
<td>Rule Sets From University Network</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uni_2010</td>
<td>96</td>
<td>university ACL</td>
<td>2010-08-30</td>
</tr>
<tr>
<td>uni_2015</td>
<td>122</td>
<td></td>
<td>2015-01-14</td>
</tr>
<tr>
<td>OpenFlow Rule Sets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of1</td>
<td>16 889</td>
<td></td>
<td>2015-05-29</td>
</tr>
<tr>
<td>of2</td>
<td>20 250</td>
<td></td>
<td>2015-05-29</td>
</tr>
<tr>
<td>of3</td>
<td>1 757</td>
<td>Open vSwitch in a cloud</td>
<td>2015-06-18</td>
</tr>
<tr>
<td></td>
<td>to 7 456</td>
<td></td>
<td>2015-07-14</td>
</tr>
</tbody>
</table>
some of the results from our analysis of IPv6 datasets
• 36 times more prefixes after 10 years of evolution

• the most common prefix length shifted from 32 (RIRs/ISPs) to 48 (end users/organizations)
some of the results from our analysis of OF rules deployed in a cloud datacenter
Destination MAC based forwarding.

Not much interest on the application side: l4 ports and protocols are specified less than 30% of the times.
Forwarding is based on either the exact destination or in a big subnet.
Rule type is a template that indicate which header fields are specified.

rule type number 796 refers to rules where **mac dst**, **eth type**, **ip proto**, **ip src**, and **ip dst** present specified values.
Evaluation of generation fidelity is based on the Root Mean Square Error (RMSE)

\[ RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\bar{y} - y_i)^2} \]
Evaluation of generation fidelity is based on the Root Mean Square Error (RMSE)

\[ RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\bar{y} - y_i)^2} \]

Target Value
Evaluation of generation fidelity is based on the Root Mean Square Error (RMSE)

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\bar{y} - y_i)^2}
\]
The beta-version of the code is released OpenSource.

We invite everyone from the community to contribute with new seeds taken from different scenarios

https://classbench-ng.github.io/
The beta version of the code is released OpenSource.

We invite everyone from the community to contribute with new seeds taken from different scenarios https://classbench-ng.github.io

ClassBench-ng
Synthetic classification rule sets generator.

Download .zip  Download .tar.gz  View on GitHub

About  Team  Links

ClassBench-ng is a tool for generation of synthetic classification rule sets for benchmarking, which is based on well-known (but longer maintained) ClassBench. The main features of ClassBench-ng are the following:

- improves IPv4 prefix sets generation accuracy (compared to original ClassBench)
- supports IPv6 prefix sets generation
- supports OpenFlow 1.0 analysis and generation

Usage
Thank you.