ClassBench-ng: Recasting ClassBench After a Decade of Network Evolution

Jiří Matoušek[1], Gianni Antichi[2], Adam Lučanský[3]
Andrew W. Moore[2], Jan Kořenek[1]

[1]Brno University of Technology [2]University of Cambridge [3]CESNET

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Agenda

Motivation

Analysis of Real Rule Sets
- IP Prefixes
- Ports and Protocol
- OpenFlow

ClassBench-ng

ClassBench-ng Evaluation
- IP Prefixes Generation
- OpenFlow Rules Generation

Summary

Packet Classification

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

- the basic operation of each networking device
- examples of use
  - packet forwarding
  - application of security policies
  - application-specific processing
  - application of quality-of-service guarantees
- the most common classification considers an IPv4 5-tuple

ip_src	source IPv4 prefix
ip_dst	destination IPv4 prefix
l4_src	source port
l4_dst	destination port
ip_proto	protocol

- a lot of existing research on packet classification

- many trends that influence packet classification
  - increasing transfer rates
    $\Longrightarrow$ faster classification
  - increasing number of classification rules
    $\Longrightarrow$ larger data structures
  - growing deployment of IPv6
    $\Longrightarrow$ longer IP prefixes
  - adoption of SDN with OpenFlow protocol
    $\Longrightarrow$ more header fields
- Internet evolution stimulates development of new packet classification algorithms
- new algorithms need to be benchmarked

- lack of real and publicly available benchmarking data
- benchmarking using synthetically generated rule sets

ClassBench[1]

- IPv4 5-tuples
- input parameters from real rule sets
- more precise output (w. r. t. parameters)

FRuG[2]

- IPv4 5-tuples, OF rules
- user-defined input parameters
- more flexible in the long term

- a precise and flexible benchmarking tool must be able to perform the analysis of real rule sets

---

[1] D. E. Taylor and J. S. Turner. ClassBench: A Packet Classification Benchmark. *Transactions on Networking*, 15(3):499–511, June 2007.

[2] T. Ganedegara, W. Jiang, and V. Prasanna. FRuG: A benchmark for packet forwarding in future networks. In *IPCCC*, pp. 231–238. IEEE, December 2010.

Recasting ClassBench

- today's Internet is no more the one of a decade ago
- questions with respect to ClassBench
  - Are the ideas behind ClassBench still valid after the decade of Internet evolution?
  - What are the characteristics of current real rule sets based on IPv4/IPv6 5-tuples and OpenFlow-specific fields?
  - What parameters should be extracted from different types of real rule sets?
  - How to extend ClassBench with respect to IPv6 and OpenFlow?

Agenda

Motivation

Analysis of Real Rule Sets

- IP Prefixes
- Ports and Protocol
- OpenFlow

ClassBench-ng

ClassBench-ng Evaluation

- IP Prefixes Generation
- OpenFlow Rules Generation

Summary

Analyzed Real Rule Sets

Name	Prefixes or rules	Source	Date
IPv4 prefix sets			
eqix_2015	550 511	Route Views	2015-07-02
eqix_2005	164 455	Route Views	2005-07-02
IPv6 prefix sets			
eqix_2015	23 866	Route Views	2015-07-02
eqix_2013	13 444	Route Views	2013-07-02
eqix_2005	658	Route Views	2005-07-02
Rule Sets From University Network			
uni_2010	96	ACLs from a university network	2010-08-30
uni_2015	122	ACLs from a university network	2015-01-14
OpenFlow rule sets			
of1	16 889	OpenFlow switch in a datacenter	2015-05-29
of2	20 250	OpenFlow switch in a datacenter	2015-05-29
of3	1 757 to 7 456	OpenFlow switch in a datacenter	2015-06-18 to 2015-07-14

- desired properties of a rule set representation
  - anonymity
  - completeness
  - scalability

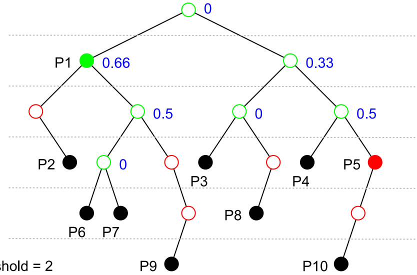
- representation of a prefix set using a trie (binary prefix tree)
- the same trie description as in ClassBench
  - prefix length distribution
  - branching probability distributions (1-child, 2-children)
  - average skew distribution

$$skew = 1 - \frac{weight(lighter)}{weight(heavier)}$$

  - prefix nesting threshold

Prefixes

P1 = 0*
P2 = 001*
P3 = 100*
P4 = 110*
P5 = 111*
P6 = 0100*
P7 = 0101*
P8 = 1010*
P9 = 01110*
P10 = 11100*

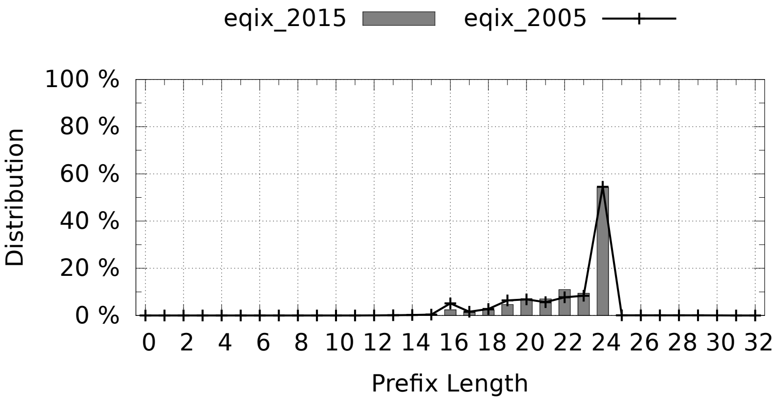


Prefix nesting threshold = 2

lengths	1-child	2-children	skew
0	0	1	0
0.1	0	1	0.5
0	0.25	0.75	0.33
0.4	0.75	0.25	0
0.3	1	0	
0.2			

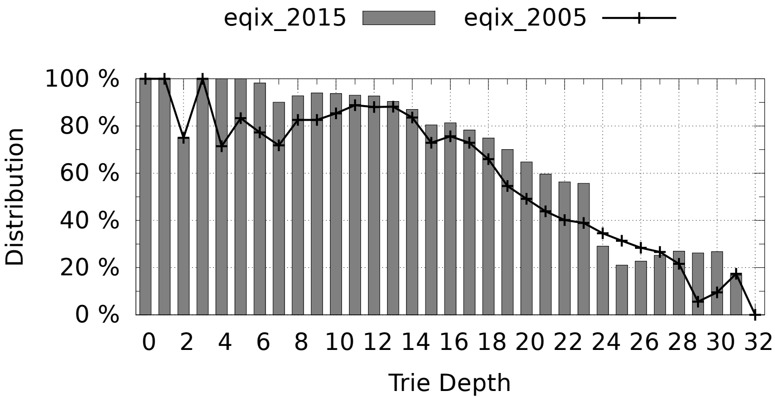
- 3 times more prefixes after 10 years of evolution

Prefix Length Distribution



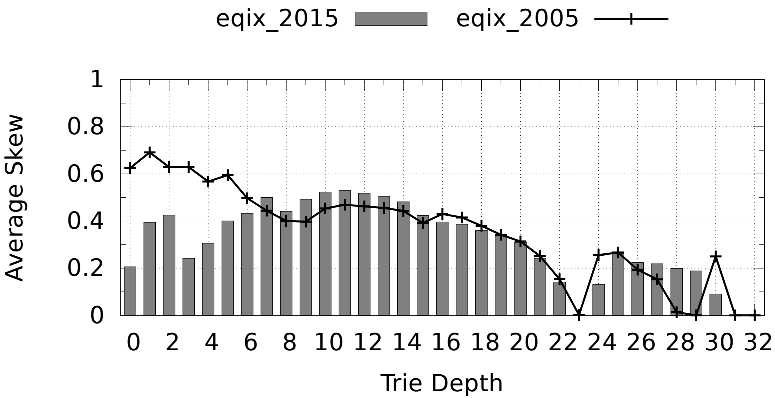
- 3 times more prefixes after 10 years of evolution

2-children Probability Distribution



- 3 times more prefixes after 10 years of evolution

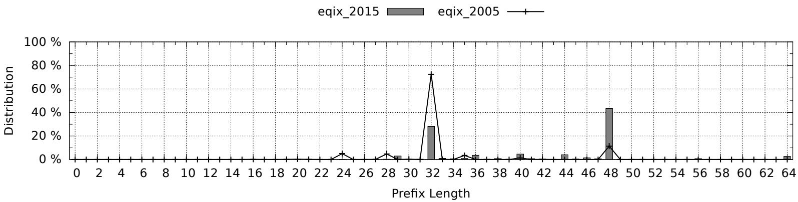
Average Skew Distribution



- 36 times more prefixes after 10 years of evolution

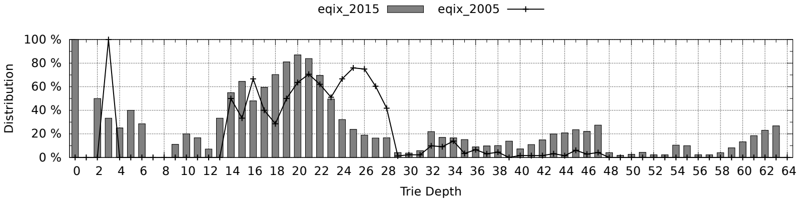
Prefix Length Distribution

- the most common prefix length shifted from 32 (RIRs/ISPs) to 48 (end users/organizations)

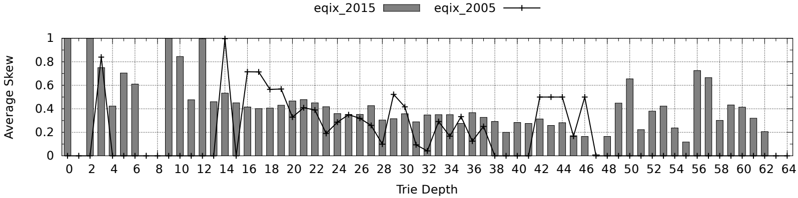


- 36 times more prefixes after 10 years of evolution

2-children Probability Distribution



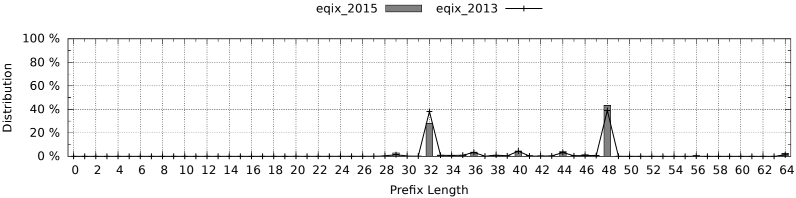
Average Skew Distribution



- 2 times more prefixes after 2 years of evolution

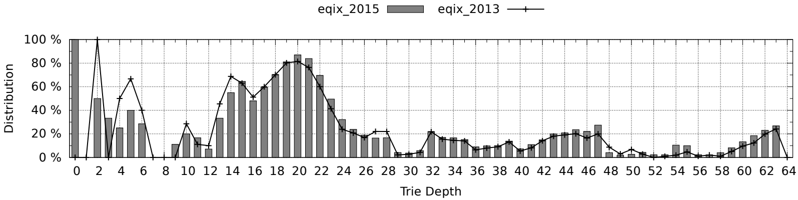
Prefix Length Distribution

- only minor changes in prefix length distribution

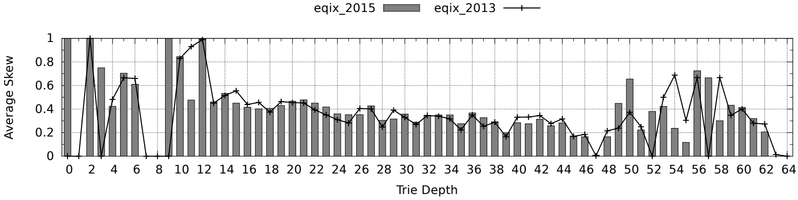


- 2 times more prefixes after 2 years of evolution

2-children Probability Distribution



Average Skew Distribution



Ports Representation

- 5 port classes are distinguished within the analysis

WC	wildcard
HI	user port range (1024 : 65535)
LO	well-known system port range (0 : 1023)
AR	arbitrary range
EM	exact match

Transport Layer Protocol

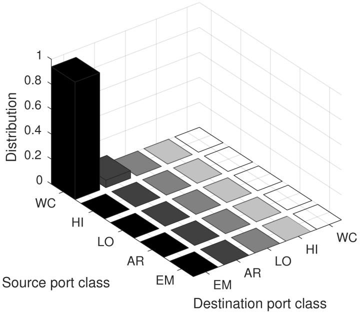
Data Set	Protocol Values		
	wildcard	TCP	UDP
uni_2010	26.0 %	71.9 %	2.1 %
uni_2015	38.5 %	54.9 %	6.6 %

Source and Destination TCP/UDP Port

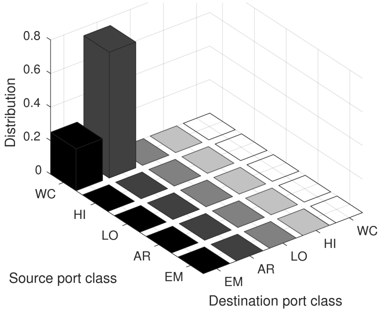
Data Set	Port Classes				
	WC	HI	LO	AR	EM
Source Port					
uni_2010	100.0 %	0.0 %	0.0 %	0.0 %	0.0 %
uni_2015	100.0 %	0.0 %	0.0 %	0.0 %	0.0 %
Destination Port					
uni_2010	26.0 %	0.0 %	0.0 %	5.2 %	68.8 %
uni_2015	38.5 %	0.0 %	0.0 %	8.2 %	53.3 %

- port pair class (PPC) helps to understand interdependencies between source and destination port classes
- PPCs in uni_2015 for TCP and UDP protocols

TCP



UDP



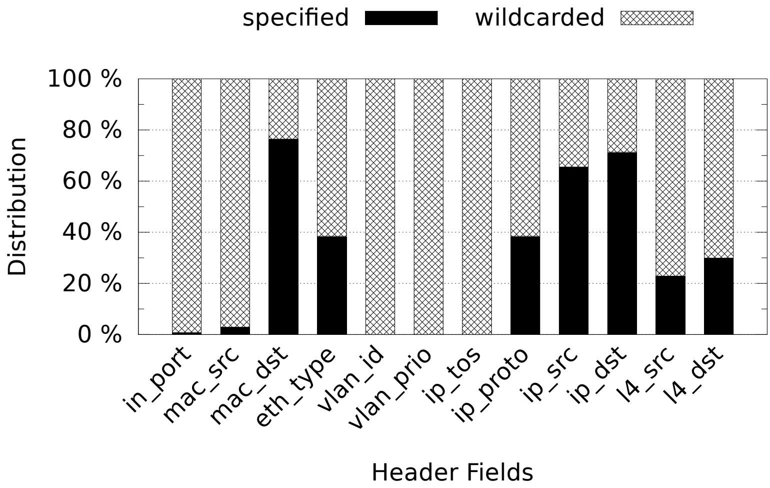
OpenFlow 1.0 Rules

- OpenFlow 1.0 extends the IPv4 5-tuple with 7 header fields

in_port	ingress port
mac_src	source MAC address
mac_dst	destination MAC address
eth_type	EtherType
vlan_id	VLAN ID
vlan_prio	VLAN priority
ip_tos	DSCP (former IP ToS)

Wildcarded-Specified Distribution

- header fields specification in rules from the of1+of2 rule set
- only 2 OF-specific fields specified in more than 20 % of rules



Unique Values Count and Uniqueness Factor

- only the rules specifying the particular field are considered
- uniqueness factor (in parenthesis) expressed in percentage

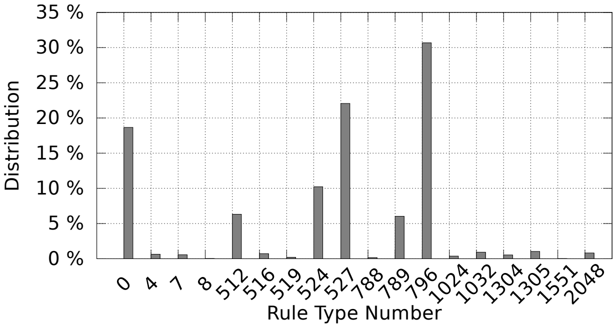
$$uniqueness\_factor_{field} = \frac{unique\_values_{field}}{rules\_specified_{field}}$$

Rule Set	in_port	mac_src	mac_dst	eth_type	ip_src	ip_dst	l4_dst
of1	123 (86.6)	27 (3.2)	593 (4.7)	1 (<0.1)	478 (4.6)	109 (0.9)	48 (2.2)
of2	140 (86.4)	19 (8.1)	791 (5.0)	1 (<0.1)	390 (2.8)	97 (0.7)	8227 (92.7)
of1+of2	182 (59.9)	45 (4.2)	1176 (4.1)	1 (<0.1)	498 (2.0)	119 (0.4)	8237 (74.2)

OpenFlow Rule Type

Describes which header fields are wildcarded/specified in rules of this type.

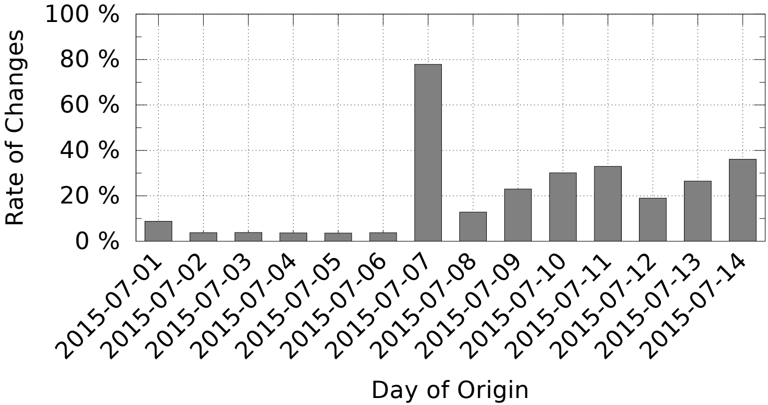
- a rule type can be represented as a 12-bit binary number
    - theoretically 4096 different rule types
    - practically only 18 utilized rule types in the of1+of2 rule set



OpenFlow Rule Set Dynamics

- dynamics of of3 expressed with the help of symmetric difference

$$A \Delta B = (A \setminus B) \cup (B \setminus A)$$



Motivation

Analysis of Real Rule Sets
- IP Prefixes
- Ports and Protocol
- OpenFlow

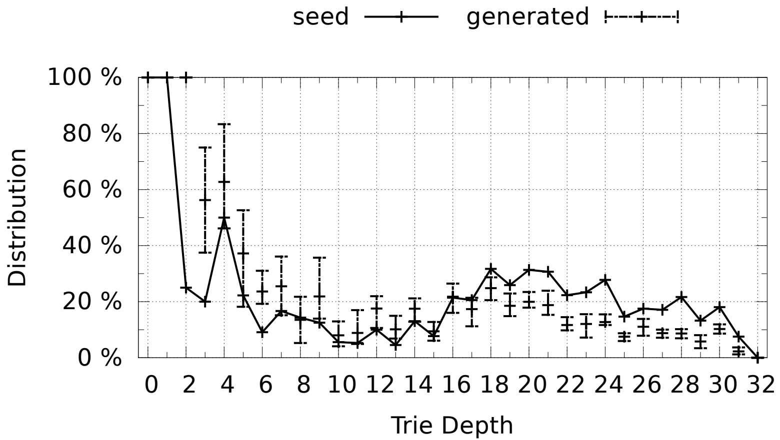
ClassBench-ng

ClassBench-ng Evaluation
- IP Prefixes Generation
- OpenFlow Rules Generation

Summary

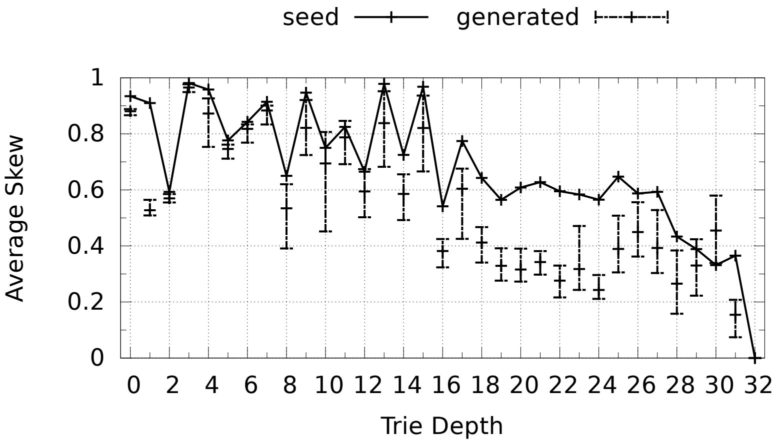
- comparison of 10 runs against original values from the acl4 seed

2-children Probability Distribution

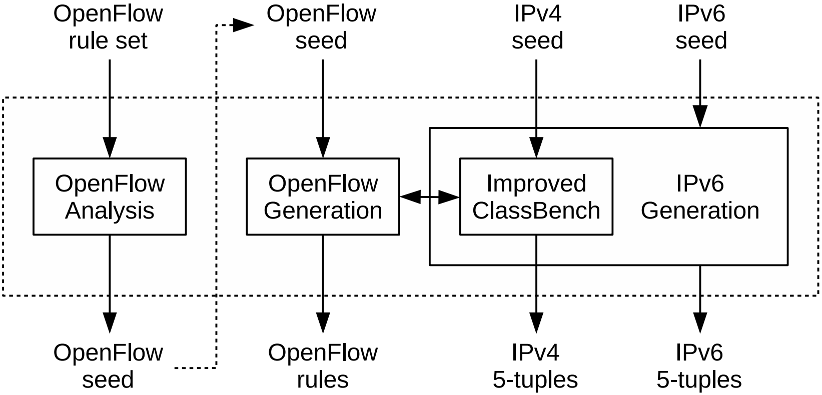


- comparison of 10 runs against original values from the acl4 seed

Average Skew Distribution



- built upon original ClassBench
- improves IPv4 prefixes generation accuracy
- supports IPv6 prefixes generation
- supports OpenFlow analysis and generation



Improved ClassBench

- IPv4 prefixes generation is improved using a trie pruning algorithm
    - starts from 100 times bigger prefix set
    - removes individual prefixes to adjust prefix set parameters to the given values
- 3 steps of the trie pruning algorithm
    1. branching probabilities adjustment (↓)
    2. average skew distribution adjustment (↑)
    3. prefixes length distribution adjustment (↓)
- steps 1 and 2 try to remove as less prefixes as possible
- each step aims to not alter the already ajusted characteristics

OpenFlow Analysis

- generates an OpenFlow seed from an OpenFlow rule set (in the `ovs-ofctl` format)
- 3 parts of the OpenFlow seed
  - rule type distribution
  - 5-tuple seed (compatible with ClassBench)
  - OpenFlow-specific seed
- 4 types of representation within the OpenFlow-specific seed
  - values (in_port, eth_type)
  - parts (mac_src, mac_dst)
  - size (vlan_id)
  - null (vlan_prio, ip_tos)

- consists of 3 steps
  1. uses Improved ClassBench to generate the given number of IPv4 5-tuples
  2. removes IPv4 5-tuple fields that are not part of the given OpenFlow rule type
  3. adds OpenFlow-specific header fields that are part of the given OpenFlow rule type
- does not allow to generate inconsistent rules (e. g., a rule specifying VLAN ID and EtherType `0x0800`)

Agenda

Motivation

Analysis of Real Rule Sets
- IP Prefixes
- Ports and Protocol
- OpenFlow

ClassBench-ng

ClassBench-ng Evaluation
- IP Prefixes Generation
- OpenFlow Rules Generation

Summary

- comparison on IPv4 prefixes generation with
  - ClassBench
  - FRuG
- comparison on IPv6 prefixes generation with
  - Non-random Generator[3]
- comparison on OpenFlow rules generation with
  - FRuG
- tools are compared using RMSE

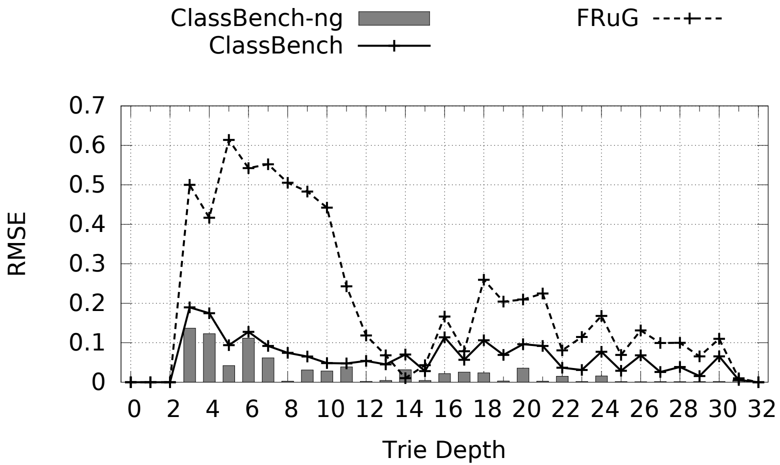
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{y} - y_i)^2}$$

  - tool-specific seeds extracted from a common original rule set
  - 10 individual runs of each tool ($n = 10$)
  - comparison of generated values (y_i) against the target value from the seed ($\bar{y}$)

[3] M. Wang, S. Deering, T. Hain, and L. Dunn. Non-random Generator for IPv6 Tables. In *HOTI*. IEEE, 2004.

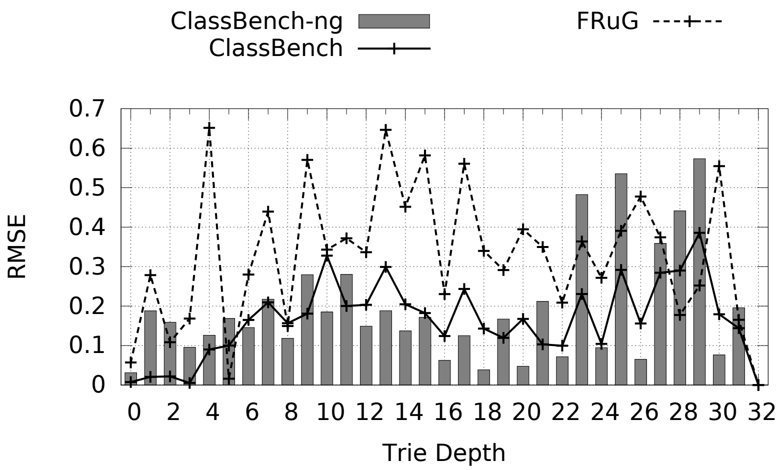
- the original rule set generated by ClassBench using the acl4 seed

2-children Probability Distribution



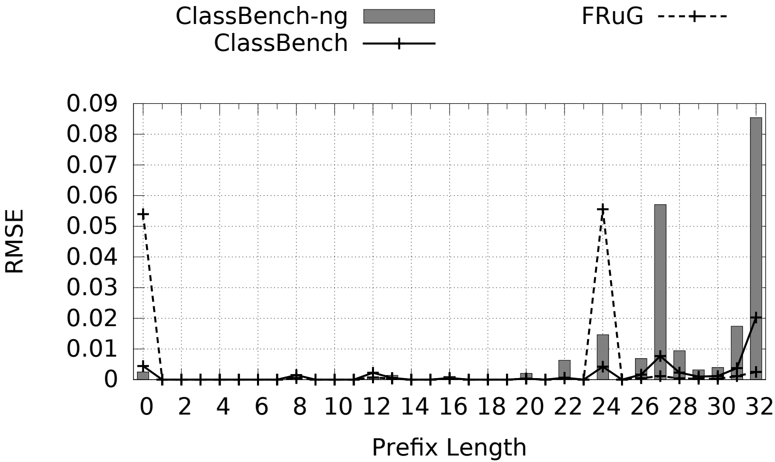
- the original rule set generated by ClassBench using the acl4 seed

Average Skew Distribution



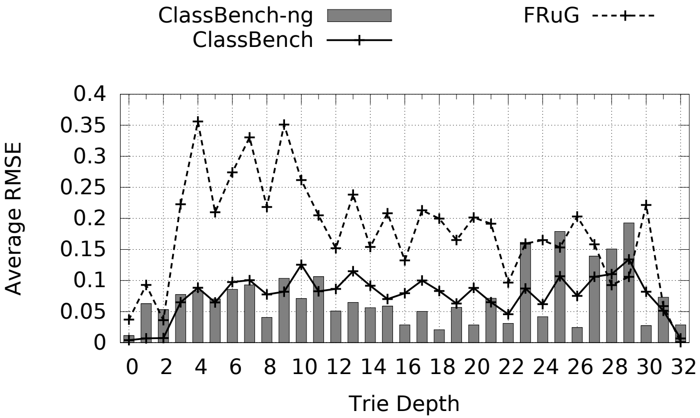
- the original rule set generated by ClassBench using the acl4 seed

Prefix Length Distribution



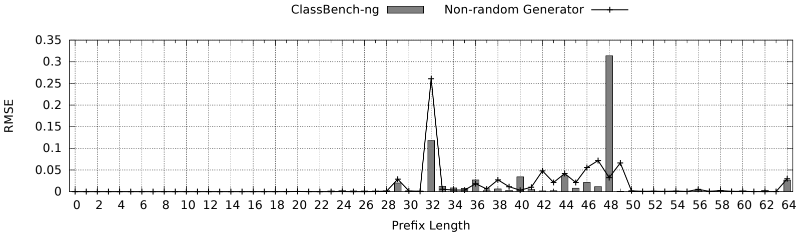
- the original rule set generated by ClassBench using the acl4 seed

$$RMSE^i_{avg} = \frac{RMSE^i_{prefixes} + RMSE^i_{branching} + RMSE^i_{skew}}{3}$$



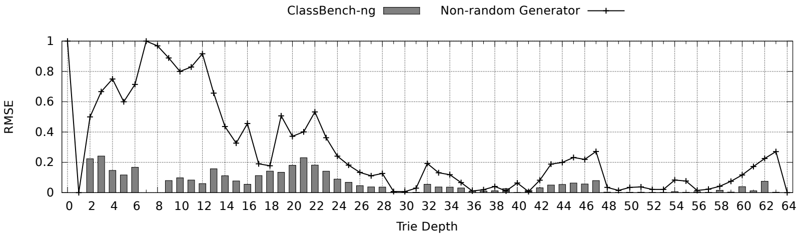
- two original rule sets from the rrc00_2015 source
- not entirely fair comparison because of different inputs
  - an IPv6 prefix set for ClassBench-ng
  - an IPv4 prefix set for Non-random Generator

Prefix Length Distribution

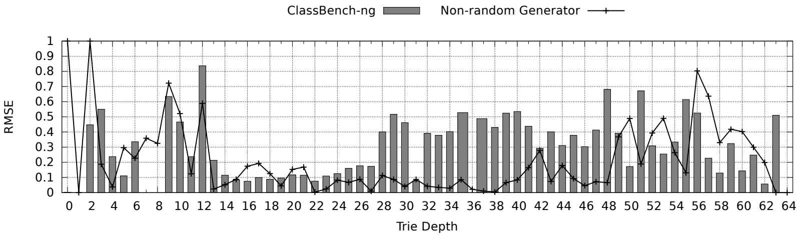


- two original rule sets from the rrc00_2015 source

2-children Probability Distribution

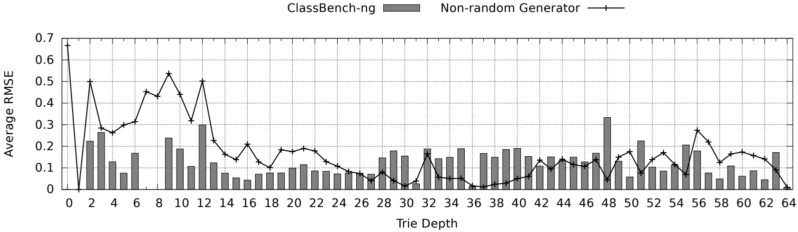


Average Skew Distribution



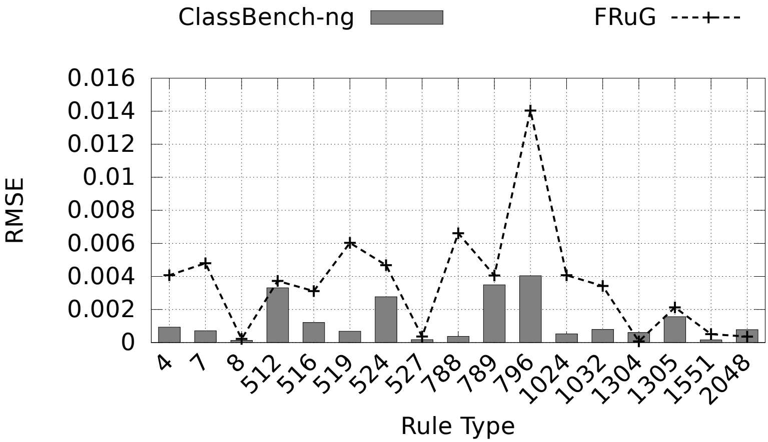
- two original rule sets from the rrc00_2015 source

$$RMSE^i_{avg} = \frac{RMSE^i_{prefixes} + RMSE^i_{branching} + RMSE^i_{skew}}{3}$$



- the original rule set is of1

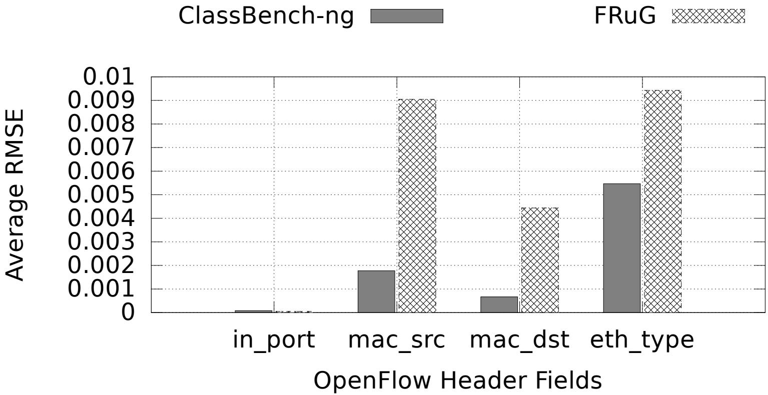
OpenFlow Rule Types



- the original rule set is of1

OpenFlow-Specific Header Fields

$$RMSE^{avg}_{field} = \frac{1}{N} \sum_{i=1}^{N} RMSE^{i}_{field}$$



Agenda

Motivation

Analysis of Real Rule Sets
- IP Prefixes
- Ports and Protocol
- OpenFlow

ClassBench-ng

ClassBench-ng Evaluation
- IP Prefixes Generation
- OpenFlow Rules Generation

Summary

Summary

- the detailed analysis of real classification rule sets
  - IPv4/IPv6 prefixes from core routers
  - ACL rules from a university network
  - OpenFlow 1.0 rules from a datacenter
- ClassBench-ng tool that is able to
  - accurately generate IPv4/IPv6 5-tuples
  - analyze real OpenFlow rule sets
  - accurately generate OpenFlow rules
- ClassBench-ng page at
  `https://classbench-ng.github.io`
  - link to to the ClassBench-ng repository
  - links to related tools/papers

Thank you for your attention