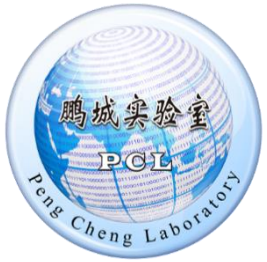


KickTree: A Recursive Algorithmic Scheme for Packet Classification with Bounded Worst-Case Performance

Yao Xin[1], Yuxi Liu[3], Wenjun Li[1,4], Ruyi Yao[2], Yang Xu[2], and Yi Wang[1,3]

[1]Peng Cheng Laboratory, [2]Fudan University, China
[3]Southern University of Science and Technology, China
[4]Harvard University, USA

ACM/IEEE ANCS 2021
December 13–16, 2021, Layfette, IN, USA

CONTENTS

01 Background

02 Motivation

03 Proposed Approach

04 Preliminary Evaluation

05 Conclusion and Future Work

CONTENTS

01 Background

02 Motivation

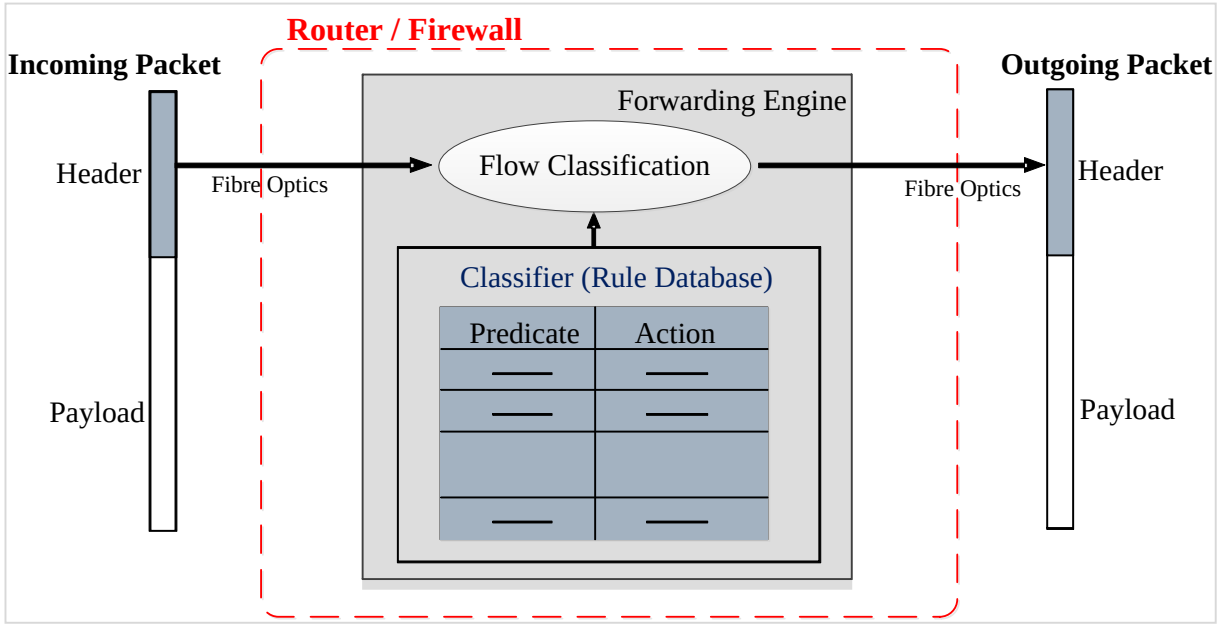
03 Proposed Approach

04 Preliminary Evaluation

05 Conclusion and Future Work

Packet Classification

- **Key for policy enforcement in packet forwarding**
  Firewall, QoS, OpenFlow, P4, etc.



exact match

match range

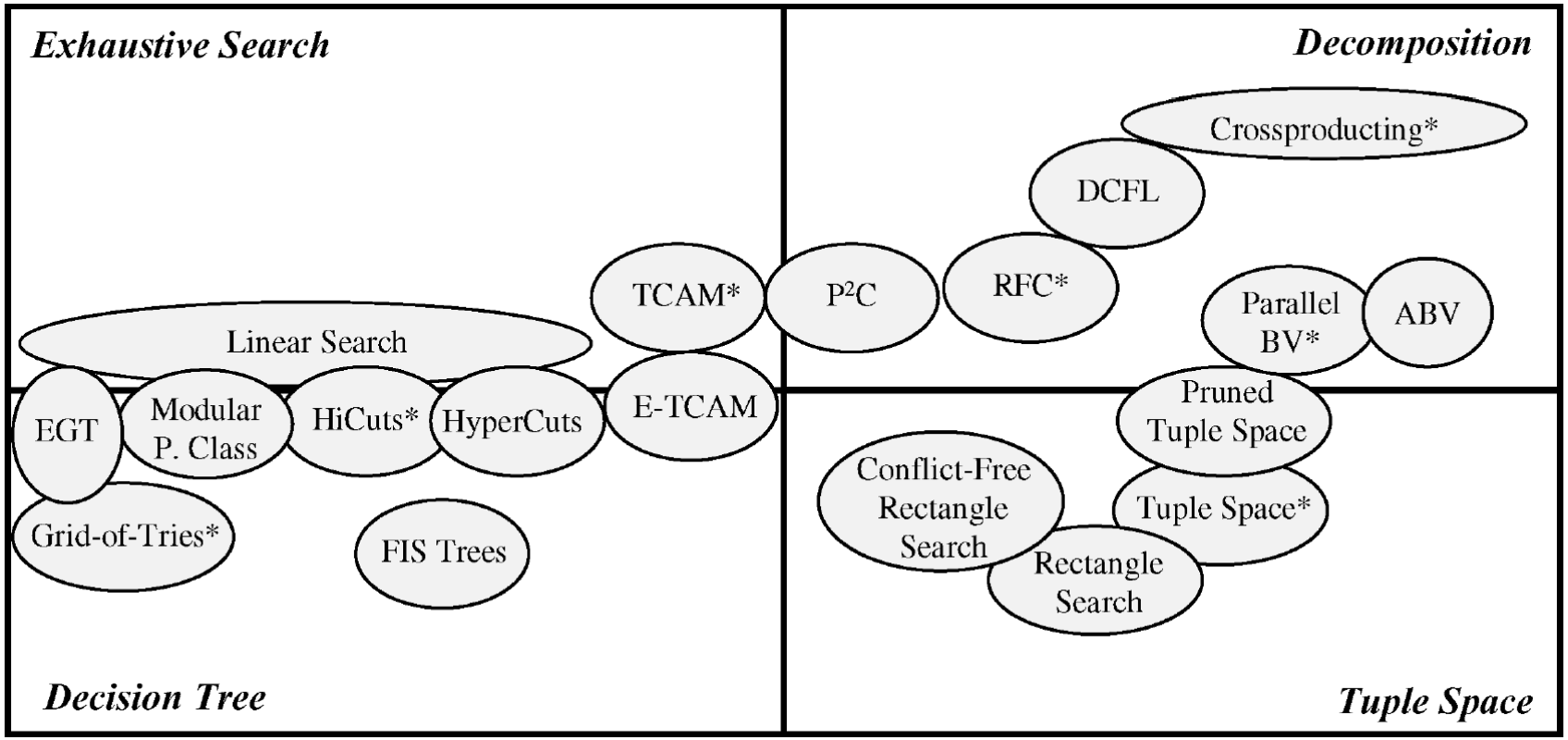
match any

#	An example OpenFlow 1.0 classifier/flow table (12-tuple)						Action
r_1	Ingress Port	Ether src	Ether dst	Ether type	VLAN id	VLAN priority	$Action_1$
	3	*	*	*2048*	*	*	
	IP src	IP dst	IP proto	IP ToS bits	TCP/UDP Src Port	TCP/UDP Dst Port	
	15.25.70.8/30	*18.15.125.3/28*	*0x11/0xff*	*1*	*1024 : 65535*	*80*	

...

Existing Solutions

- **Well-known taxonomy from David E. Taylor[CSUR 2005]**

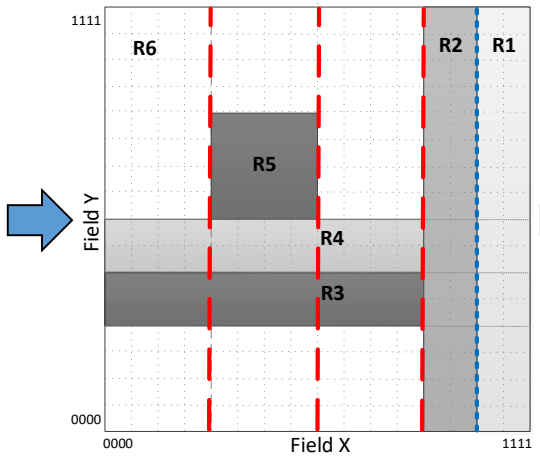


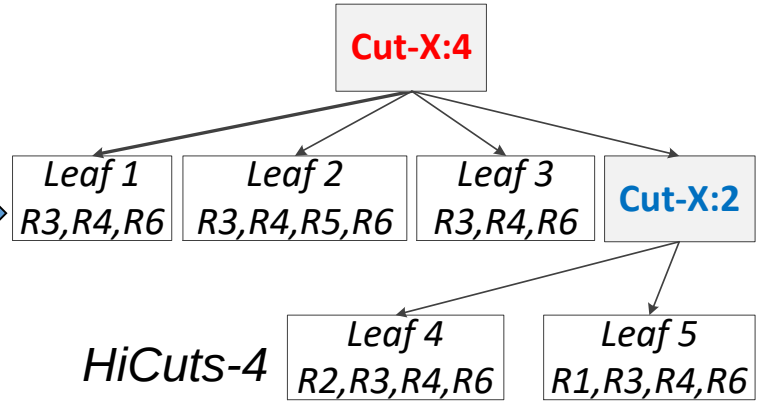
Notes: adjacent techniques are related;
hybrid techniques overlap quadrant boundaries;
∗ denotes a seminal technique

Review on Decision Tree

- **Decision-tree construction in packet classification**
  - 1. Rule table matching ↔ Point location in geometric space
  - 2. Partition the searching space into sub-spaces recursively
    - ✓ Root node: Whole searching space containing all rules
    - ✓ Internal node: #rule covered by sub-space > a predefined number of rules (*binth*)
    - ✓ Leaf node: #rule covered by sub-space <= a predefined number of rules

Rule #	Field X	Field Y	*Action*
R1	111*	*	*A1*
R2	110*	*	*A2*
R3	*	010*	*A3*
R4	*	011*	*A4*
R5	01**	10**	*A5*
R6	*	*	*A6*





- **Two major threads of building decision-trees**
  - Equal-sized cutting (HiCuts, TabTree, Kicktree)
  - Equal-dense splitting (HyperSplit, ParaSplit)

CONTENTS

01 Background

02 Motivation

03 Proposed Approach

04 Preliminary Evaluation

05 Conclusion and Future Work

Dilemma in Packet Classification on FPGA

Current hardware architecture on FPGA

Demands in OpenFlow switches

- Multi-field matching
- Large-scale rule set supporting
- Fast dynamic rule update

Main current FPGA implementations	Advantage	Challenges
Decision Tree	Fast classification	Rule replications hinder rule update
	Support large scale set	Unbalance and unbounded depth
Bit Vector (BV) Decomposition	Good performance	Restricted scale of Vector
	Support rule update	Only applicable to small-scale rule sets

TabTree: decision tree based method dedicated to FPGA

- **Partition rules into subsets based on *small fields***

  For a *W*-bit wide field F_i with the threshold value of 2^K, F_i is a *small field* if and only if there are no wildcard (*) at its most significant *W-K* bits, we call these *W-K* bits as *selectable bits*.

TABLE I
EXAMPLE RULE SET WITH TWO IPV4 ADDRESS FIELDS

rule id	priority	src_addr field	dst_addr field	*action*
R_1	14	228.128.0.0/9	0.0.0.0/0	*action1*
R_2	13	223.0.0.0/9	0.0.0.0/0	*action2*
R_3	12	0.0.0.0/1	175.0.0.0/8	*action3*
R_4	11	0.0.0.0/1	225.0.0.0/8	*action4*
R_5	10	0.0.0.0/2	225.0.0.0/8	*action5*
R_6	9	128.0.0.0/1	123.0.0.0/8	*action6*
R_7	8	128.0.0.0/1	37.0.0.0/8	*action7*
R_8	7	0.0.0.0/0	123.0.0.0/8	*action8*
R_9	6	178.0.0.0/7	0.0.0.0/1	*action9*
R_{10}	5	0.0.0.0/1	172.0.0.0/7	*action10*
R_{11}	4	0.0.0.0/1	226.0.0.0/7	*action11*
R_{12}	3	128.0.0.0/1	120.0.0.0/7	*action12*
R_{13}	2	128.0.0.0/2	120.0.0.0/7	*action13*
R_{14}	1	128.0.0.0/1	38.0.0.0/7	*action14*

Partition

TABLE II
PARTITIONED RULES WITH SMALL DST_ADDR FIELD

rule id	src_addr ($T_{src\_addr}= 2^{25}$)	dst_addr ($T_{dst\_addr}= 2^{25}$)	
	1-32th bits	**33-39th**	40-64th bits
R_3	0*********************************	**1010111**	1*********************
R_4	0*********************************	**1110000**	1*********************
R_5	00********************************	**1110000**	1*********************
R_6	1*********************************	**0111101**	1*********************
R_7	1*********************************	**0010010**	1*********************
R_8	*********************************	**0111101**	1*********************
R_{10}	0*********************************	**1010110**	*********************
R_{11}	0*********************************	**1110001**	*********************
R_{12}	1*********************************	**0111100**	*********************
R_{13}	10********************************	**0111100**	*********************
R_{14}	1*********************************	**0010011**	*********************

Each *selectable bit* can map rules into at most two rule subsets without any rule replications

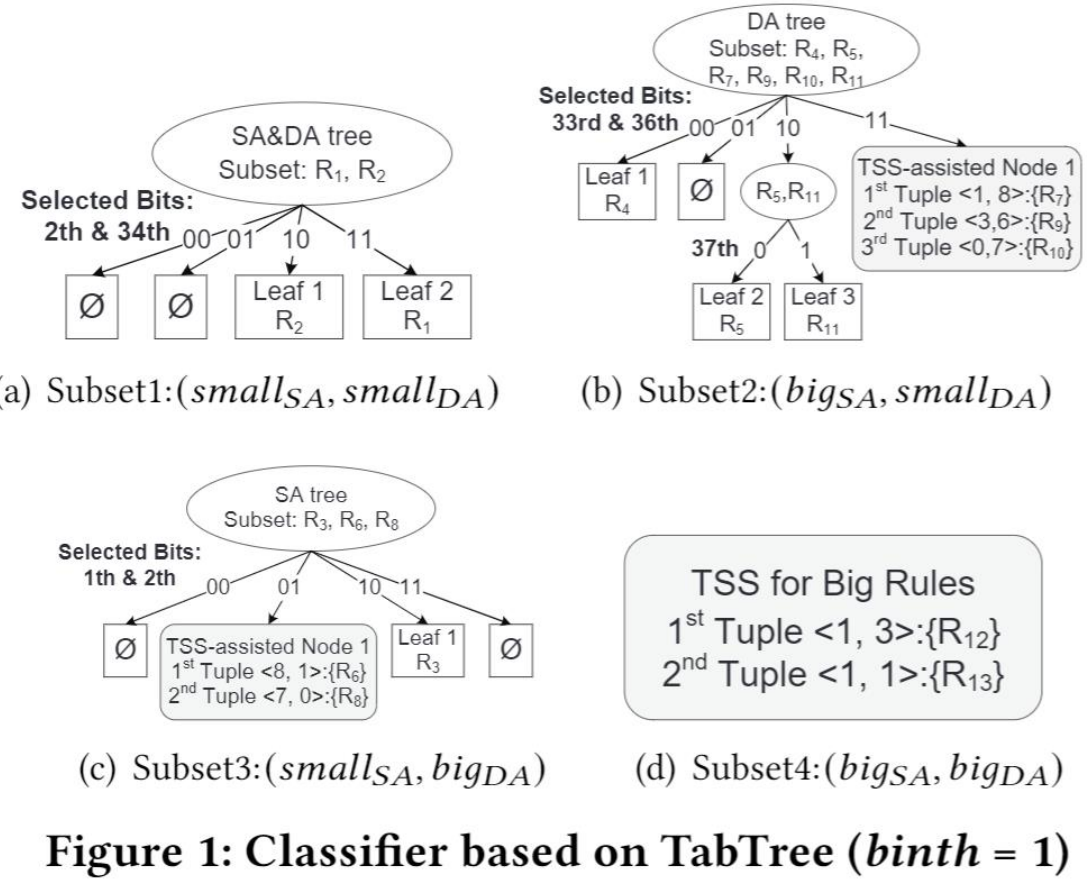
- **Build search trees by bit-selecting for each small field**
- **Tuple Space Search (TSS) or linear search assistant for leaf node**

Problems in FPGA implementation for TabTree

- ***Small field selection*** **relies on empirical characteristics of rules, and the number of subsets is the exponential size of the *small fields***
  - Eg. if *K* small fields are selected, 2^K subsets need to be generated.

- **The distribution of rule subset is uneven, so that the depth of each decision tree is very different, which is not conducive to the convergence of concurrent results by FPGA**

- **A large number of TSS leaf nodes in decision trees. Each TSS structure contains multiple hash tables, and the number of TSS is unpredictable for each rule set, which is not friendly to hardware implementation.**



(a) Subset1:$(small_{SA}, small_{DA})$ (b) Subset2:$(big_{SA}, small_{DA})$

(c) Subset3:$(small_{SA}, big_{DA})$ (d) Subset4:(big_{SA}, big_{DA})

Figure 1: Classifier based on TabTree ($binth = 1$)

CONTENTS

01 Background

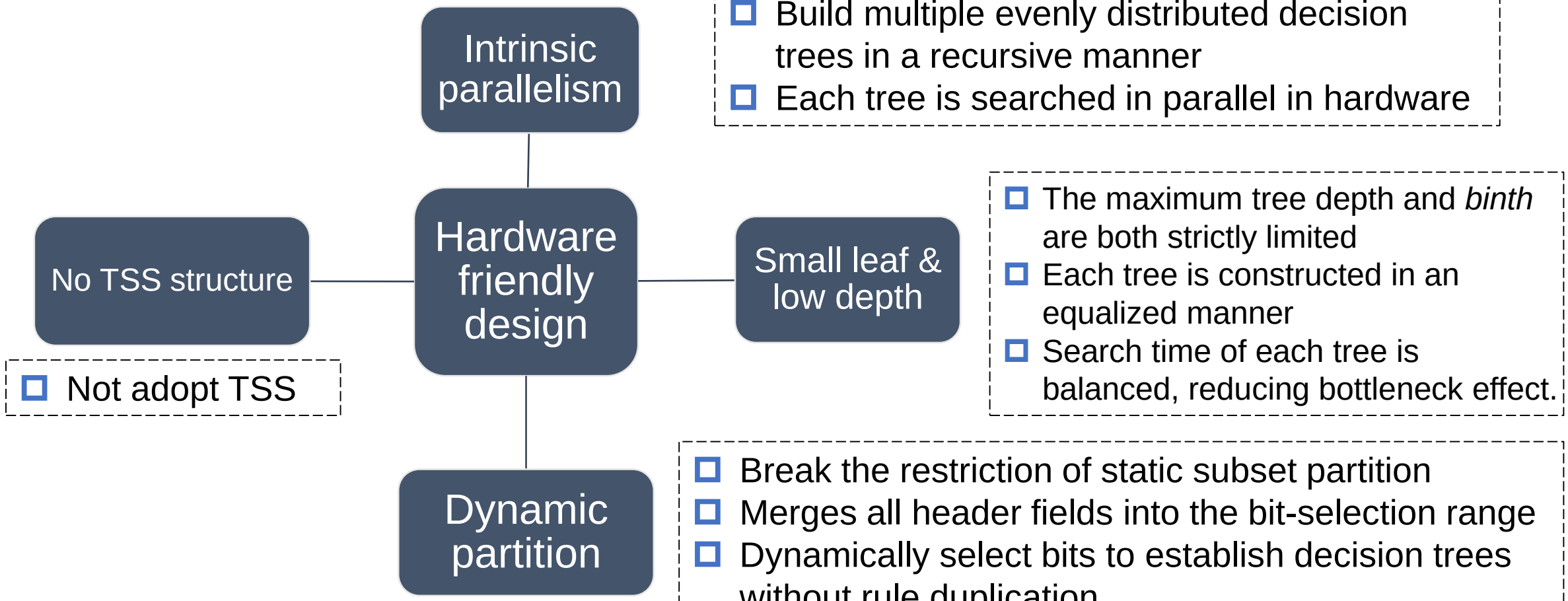
02 Motivation

03 Proposed Approach

04 Preliminary Evaluation

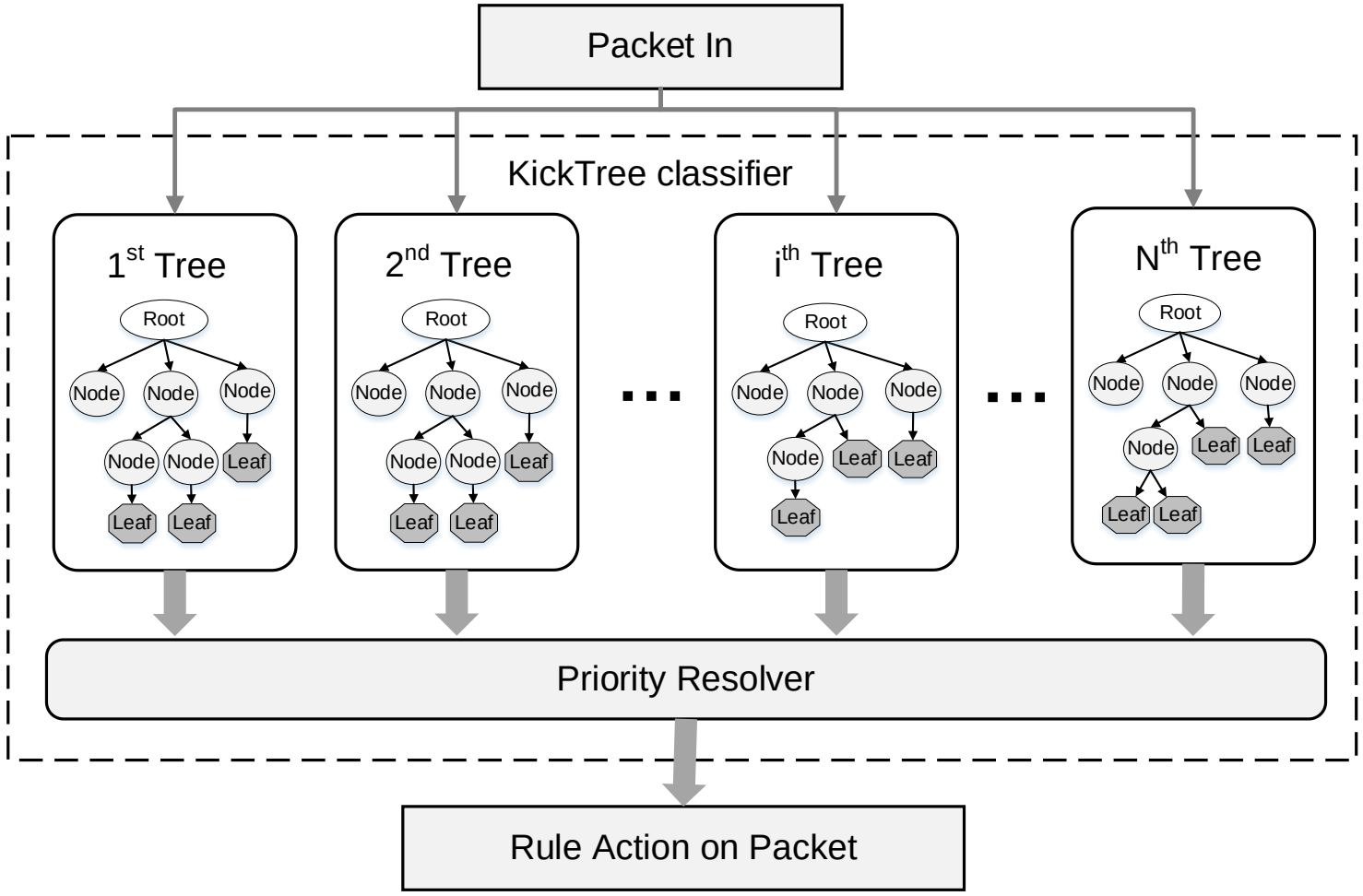
05 Conclusion and Future Work

KickTree: Ideas



The Framework of KickTree

- **Key features:**
  - **Subtrees are dynamically and recursively constructed without pre-partition**
  - **multiple evenly distributed decision trees**
  - **Limited tree depth and *binth* (worst-case bounded)**



Balanced Bit-selecting

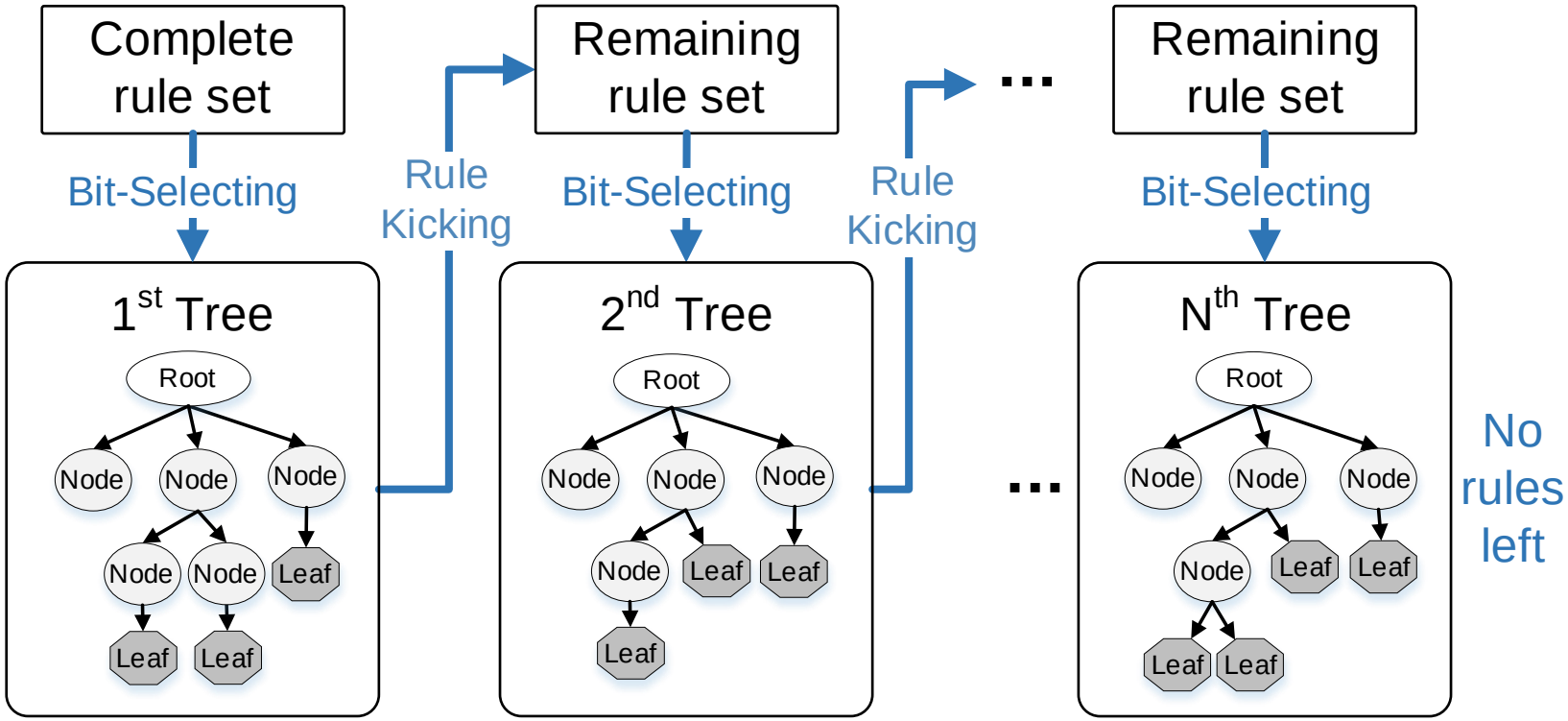
- **Local Optimal Strategy: select the "good“ bits one by one and tries to find the most balance for each bit**

$$imbalance(bit\ v) = |\#L\_Child - \#R\_Child|$$

- **Stop bit-selecting progress in one of the following cases**
  - tree depth achieves the predefined maximum value
  - number of rules in the tree node is less than *binth*
  - remaining unselected rule bits share same values and cannot separate rules from each other

KickTree Construction

- **Steps:**
  - Starts from building the first tree with the complete rule set
  - "kick" rules out of current tree in two cases:
    1) value of rules is wildcard in the selecting bit;
    2) current node is indivisible (leaf), and the rule number > *binth*
  - Recursive process continues until no rules left



A Working Example

- **An example rule set with four IPv4 address fields**

rule id	*priority*	SA	DA	SP	DP	*action*
R_1	13	228.128.0.0/9	124.0.0.0/7	119:119	0:65535	*action1*
R_2	12	223.0.0.0/9	38.0.0.0/7	20:20	1024:65535	*action2*
R_3	11	175.0.0.0/8	0.0.0.0/1	53:53	0:65535	*action3*
R_4	10	128.0.0.0/1	37.0.0.0/8	53:53	1024:65535	*action4*
R_5	9	0.0.0.0/2	225.0.0.0/8	123:123	0:65535	*action5*
R_6	8	107.0.0.0/8	128.0.0.0/1	59:59	0:65535	*action6*
R_7	7	0.0.0.0/1	255.0.0.0/8	25:25	0:65535	*action7*
R_8	6	106.0.0.0/7	0.0.0.0/0	0:65535	53:53	*action8*
R_9	5	160.0.0.0/3	252.0.0.0/6	0:65535	0:65535	*action9*
R_{10}	4	0.0.0.0/0	254.0.0.0/7	0:65535	124:124	*action10*
R_{11}	3	128.0.0.0/2	236.0.0.0/7	0:65535	0:65535	*action11*
R_{12}	2	0.0.0.0/1	224.0.0.0/3	0:65535	23:23	*action12*
R_{13}	1	128.0.0.0/1	128.0.0.0/1	0:65535	0:65535	*action13*

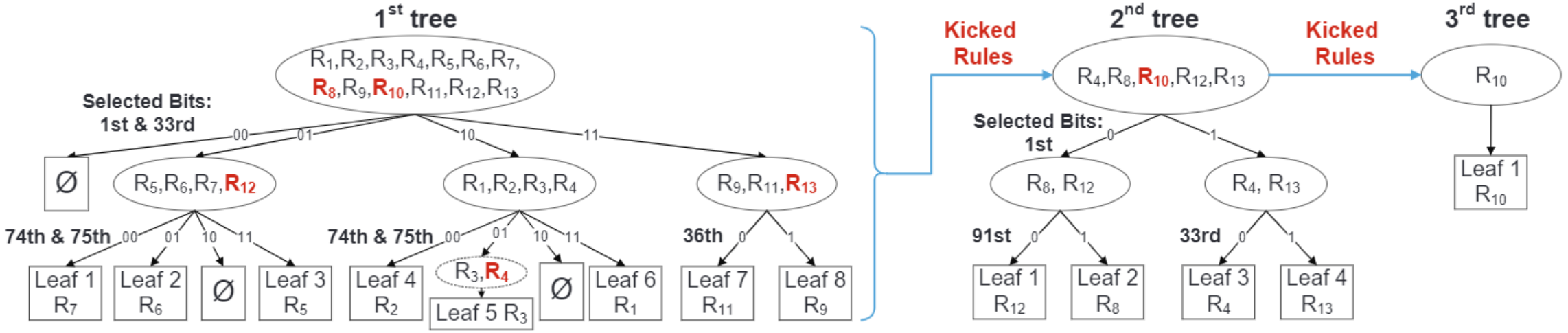
maximum number of bits to cut a node: **2**

binth: **1**

A Working Example

Selectable bit for example rule set

rule id	src_addr (SA) 1-32th bits	dst_addr (DA) 33-64th bits	src_port (SP) LCP 65-80th bits	dest_port (DP) LCP 81-96th bits
R_1	11100100**********************	011110**********************	0000000001110111	****************
R_2	11011110**********************	001001**********************	0000000000010100	****************
R_3	10101111**********************	0**********************	0000000000110101	****************
R_4	1**********************	00100101**********************	0000000000110101	****************
R_5	00**********************	11100001**********************	0000000001111011	****************
R_6	01101011**********************	1**********************	0000000000111011	****************
R_7	0**********************	11111111**********************	0000000000011001	****************
R_8	011010**********************	**********************	**********	0000000000110101
R_9	101**********************	111111**********************	**********	**********
R_{10}	**********************	1111111**********************	**********	0000000001111100
R_{11}	10**********************	1110110**********************	**********	**********
R_{12}	0**********************	111**********************	**********	0000000000010111
R_{13}	1**********************	1**********************	**********	**********



CONTENTS

01 Background

02 Motivation

03 Proposed Approach

04 Preliminary Evaluation

05 Conclusion and Future Work

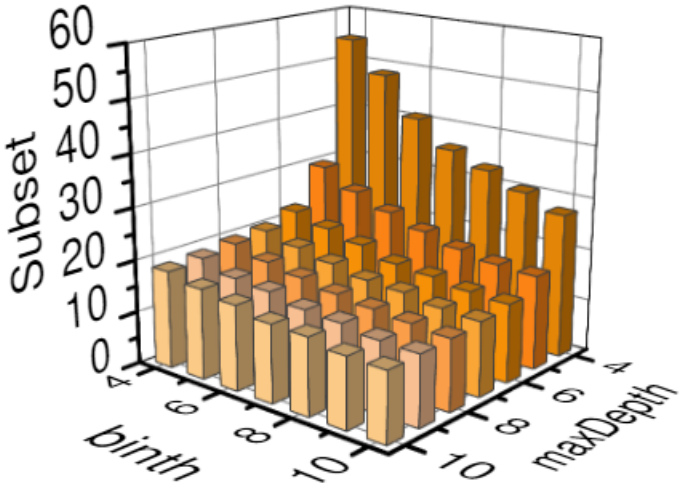
Experimental Setup

Compare Objects	**CutSplit: the latest cutting based decision tree** **TabTree: the latest decision tree based method targeting FPGA**	
Primary metrics	Number of subsets	Memory footprint Memory access Update performance
Rule sets	ACL, FW, IPC: 1k, 10k,100k generated by ClassBench 12 rule sets based on 12 seed files	
Selection bit length	Fixed to **4**	
Max tree depth	Not fixed	Fixed to **10**
Binth	Not fixed	Fixed to **8**

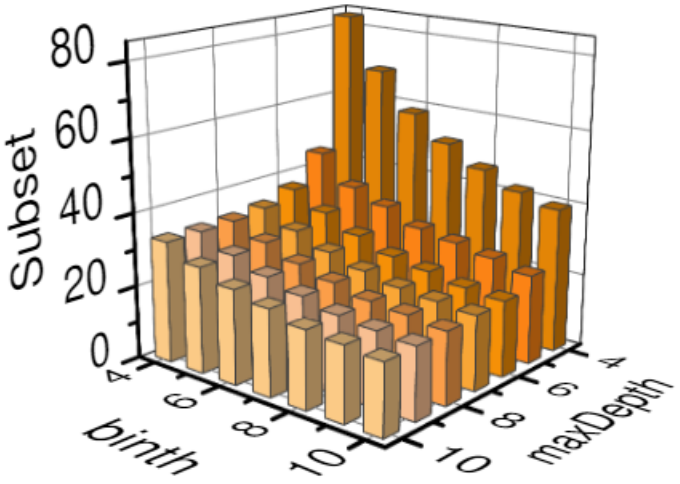
Our implementation of KickTree is available on https://github.com/wenjunpaper/KickTree

as well as http://www.wenjunli.com/KickTree

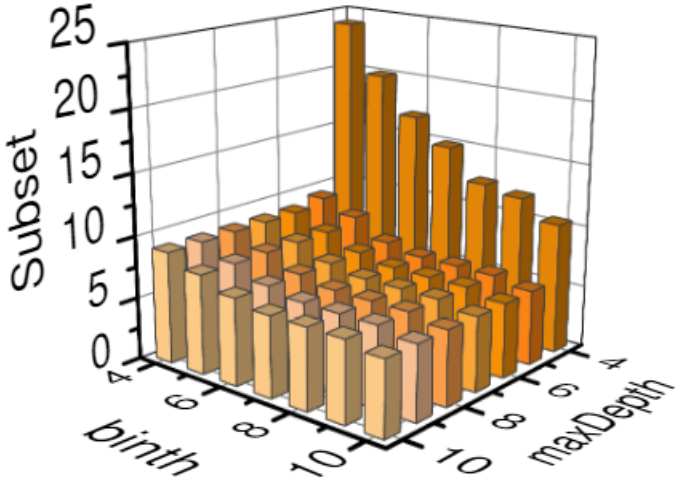
Number of Subsets



(a) ACL_100k

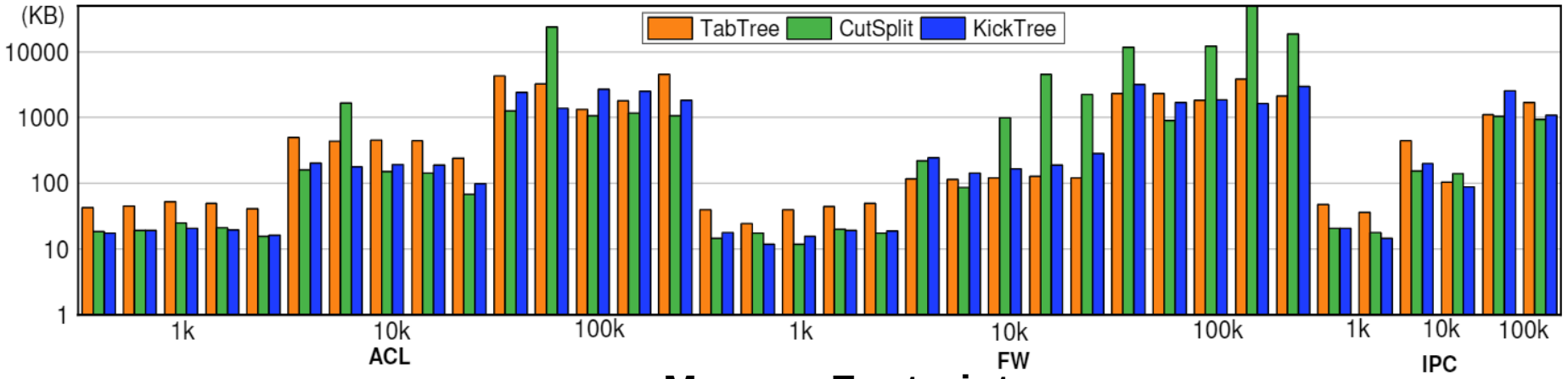


(b) FW_100k

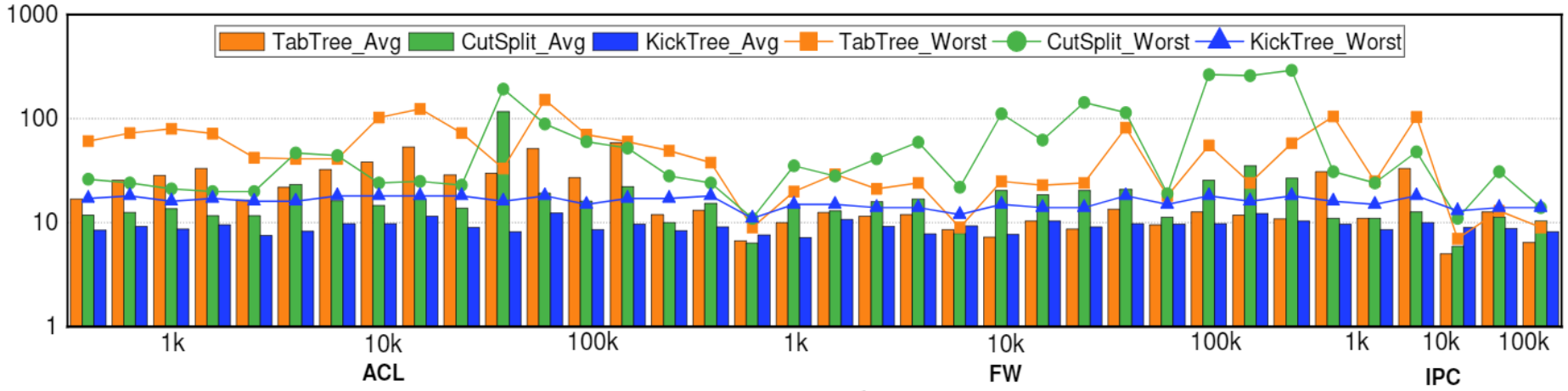


(c) IPC_100k

Memory Footprint & Memory Access



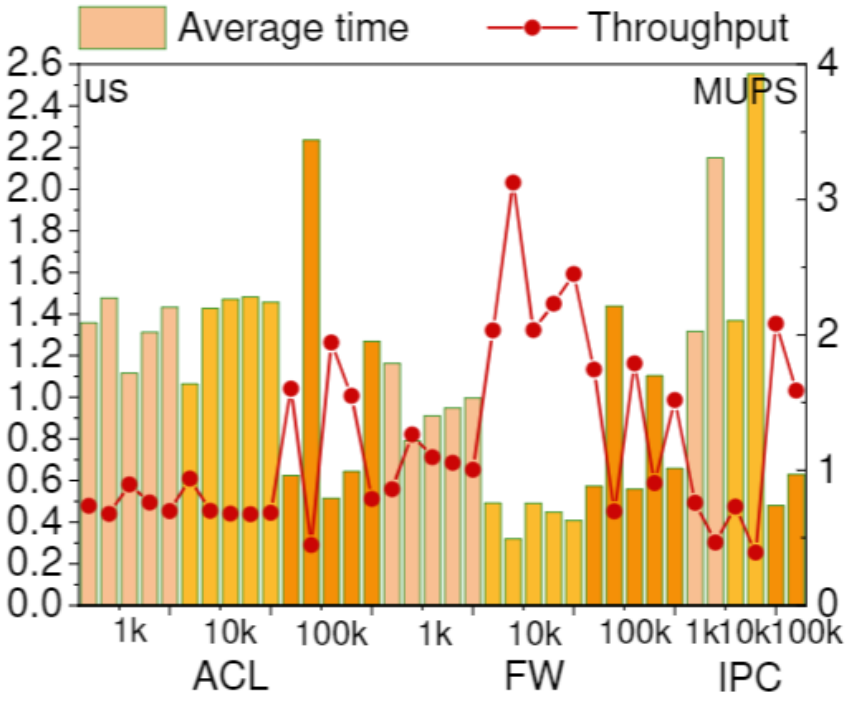
Memory Footprint



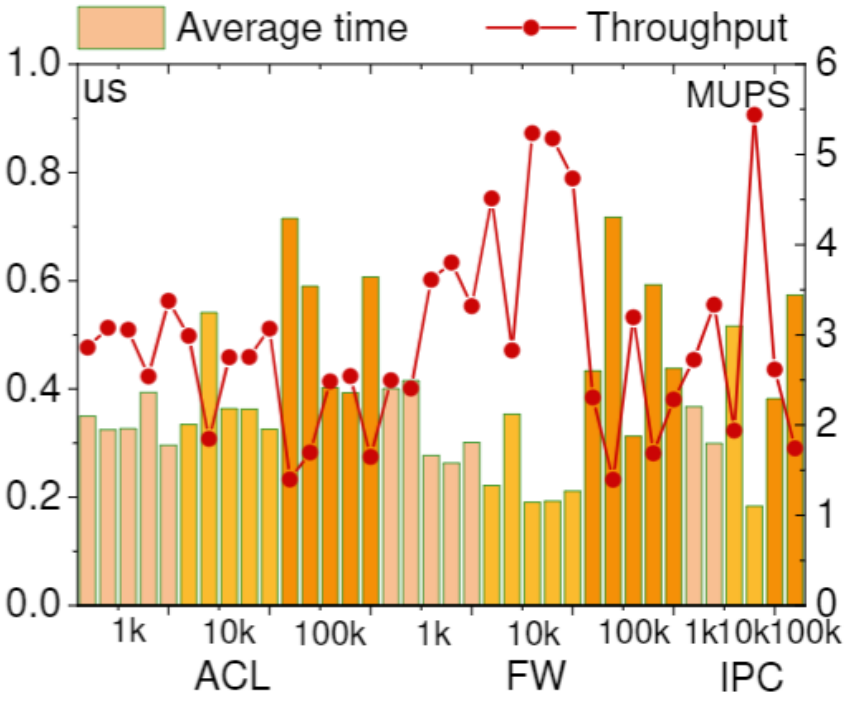
Average memory access and worst-case memory access

Performance comparison		
	TabTree	CutSplit
Average access reduction	1.2x	0.99x
Worst-case access improvement	2.14x	3.09x

Incremental Update Performance



(a) KickTree



(b) TabTree

CONTENTS

01 Background

02 Motivation

03 Proposed Approach

04 Preliminary Evaluation

05 Conclusion and Future Work

Conclusion and Future Work

Conclusion

- **KickTree**
  - an FPGA-friendly updatable packet classifier
  - more flexible bit selection scheme
  - an evenly distributed and worst-case bounded decision tree scheme

Future work

- improve with more balanced rule mapping
- a smaller number of subtrees
- hardware architecture designed and implemented on FPGA

Thanks!

Q&A