ADA: Arithmetic Operations with Adaptive TCAM Population in Programmable Switches

Mojtaba Malekpourshahraki[1]

Brent E. Stephens[2]

Balajee Vamanan[1]

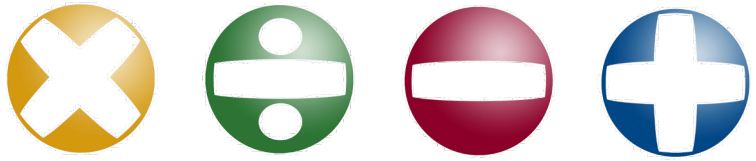
1: University of Illinois at Chicago

2: The University of Utah

Email: mmalek3@uic.edu

Arithmetic operations are ubiquitous

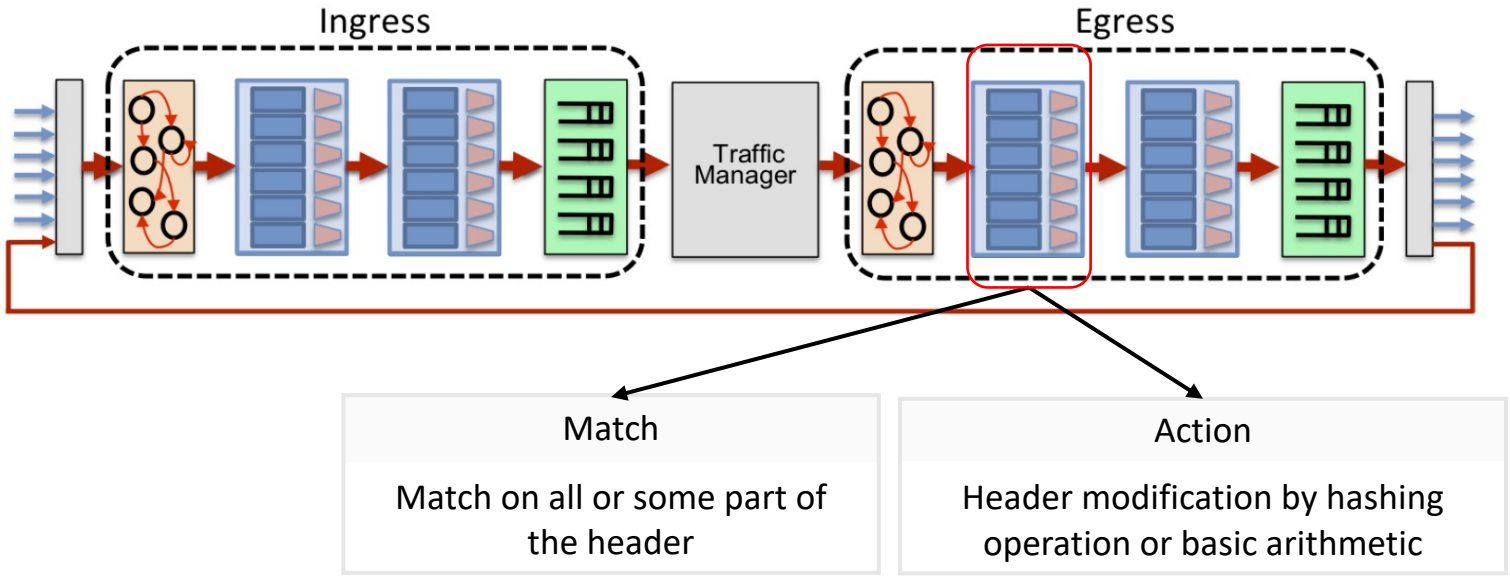
- Most real-world applications involve arithmetic operations
- Programmers assume arithmetic operations are universally supported in all devices
- Also true for most network applications
  - RCP
  - XCP
  - Conga



Programmable switches do NOT support many arithmetic operations

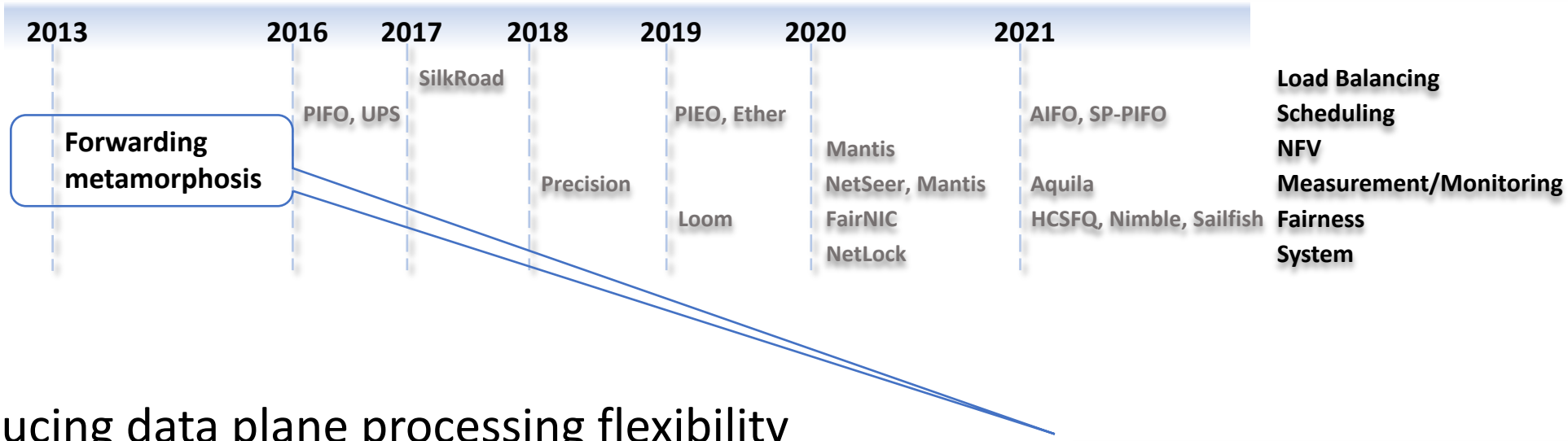
PISA architecture limits arithmetic operations

- Programable switches
  - Support packet header modifications at high rate
  - Match on headers and modify headers

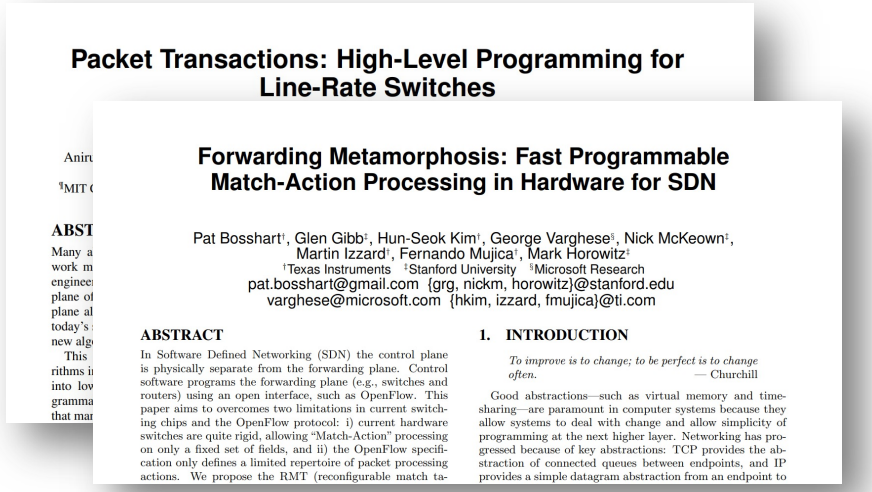


The architecture of programable switches is not amenable to support complex arithmetic operations such as multiplication

Programable networks are hot



- Introducing data plane processing flexibility
  - Use reconfigurable match-action table (RMT)



Packet Transactions: High-Level Programming for Line-Rate Switches

Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN

Pat Bosshart[†], Glen Gibb[‡], Hun-Seok Kim[†], George Varghese[§], Nick McKeown[‡], Martin Izzard[†], Fernando Mujica[†], Mark Horowitz[‡]
[†]Texas Instruments [‡]Stanford University [§]Microsoft Research
pat.bosshart@gmail.com {grg, nickm, horowitz}@stanford.edu
varghese@microsoft.com {hkim, izzard, fmujica}@ti.com

ABSTRACT

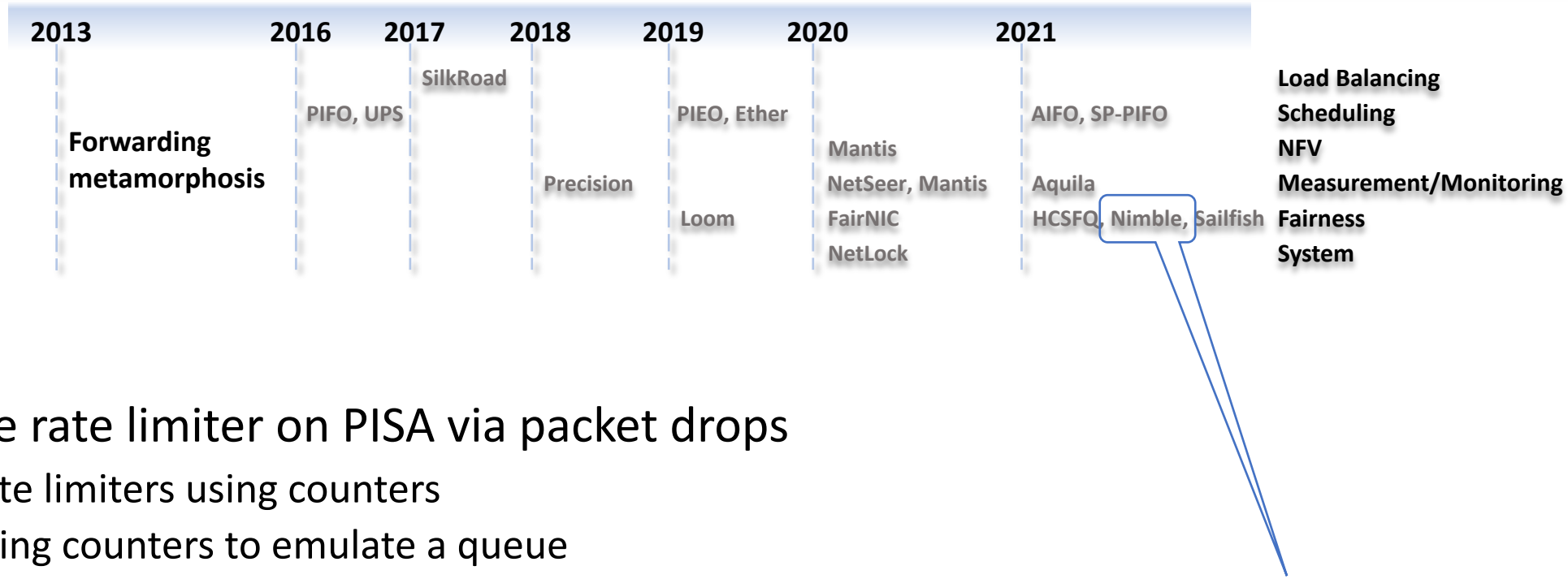
In Software Defined Networking (SDN) the control plane is physically separate from the forwarding plane. Control software programs the forwarding plane (e.g., switches and routers) using an open interface, such as OpenFlow. This paper aims to overcomes two limitations in current switching chips and the OpenFlow protocol: i) current hardware switches are quite rigid, allowing "Match-Action" processing on only a fixed set of fields, and ii) the OpenFlow specification only defines a limited repertoire of packet processing actions. We propose the RMT (reconfigurable match ta-

1. INTRODUCTION

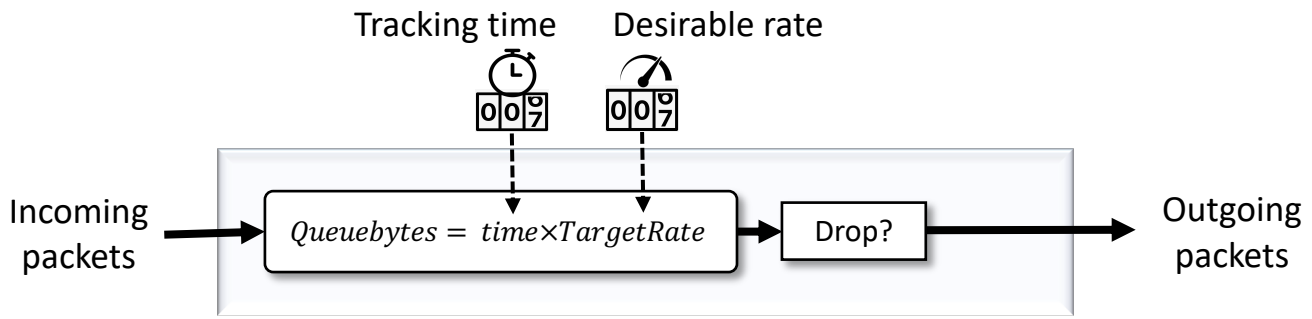
To improve is to change; to be perfect is to change often. — Churchill

Good abstractions—such as virtual memory and time-sharing—are paramount in computer systems because they allow systems to deal with change and allow simplicity of programming at the next higher layer. Networking has progressed because of key abstractions: TCP provides the abstraction of connected queues between endpoints, and IP provides a simple datagram abstraction from an endpoint to

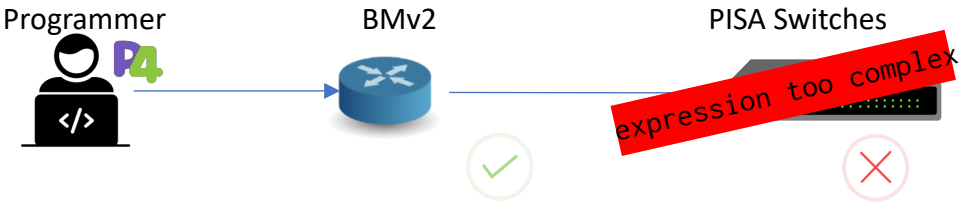
Our experience with programable networks …

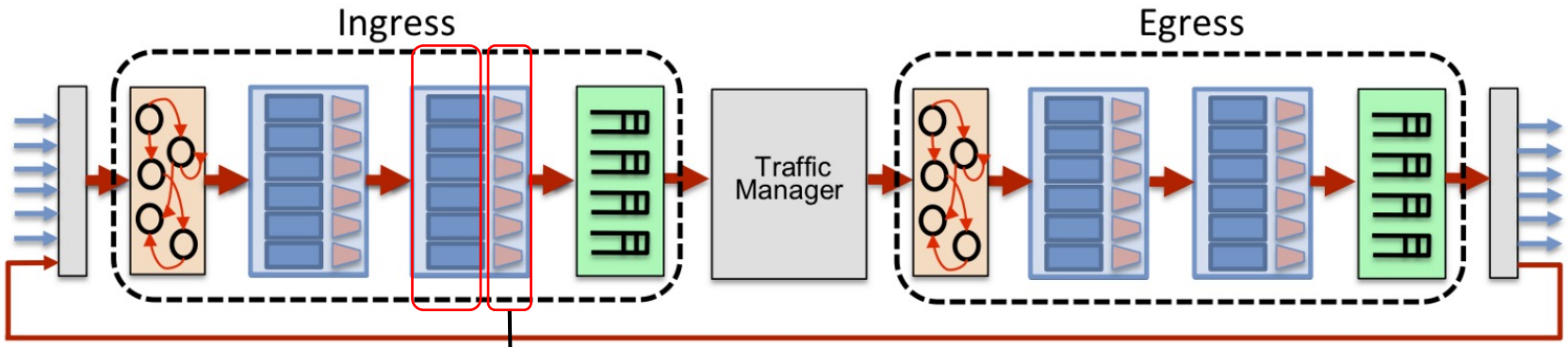


- Simple rate limiter on PISA via packet drops
  - Rate limiters using counters
  - Using counters to emulate a queue
  - Drop the packet if queue size exceeds drop threshold



And heartburns!





ALU is limited

Process standard arithmetic operations, shift bit, header modification, hashing operation

Supported operations

- Arithmetic operations: addition, subtraction
- Logical operations: left and right bit-shifts
- Header modification
- Hashing operation

Unsupported operations

- Multiplication
- Division

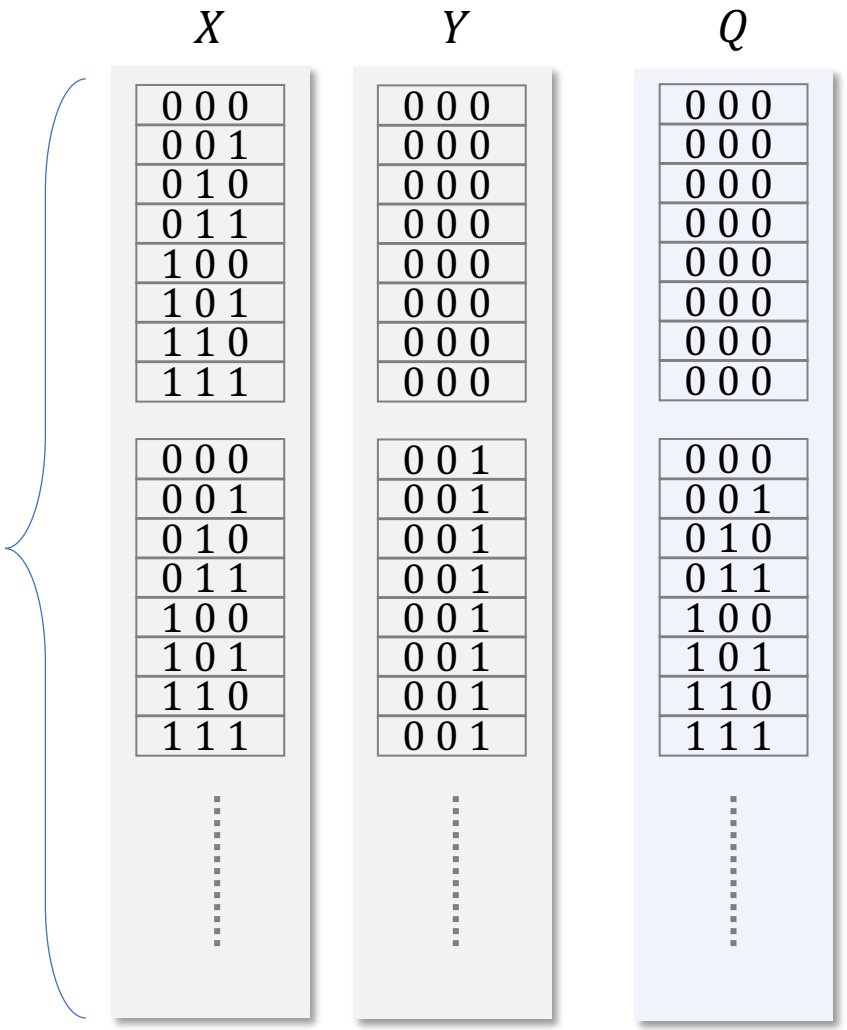
Agenda

- Problem statement
- Background
- Motivation
- Design
  - Control plane
  - Data plane
- Evaluation

Naive solution: Lookup tables

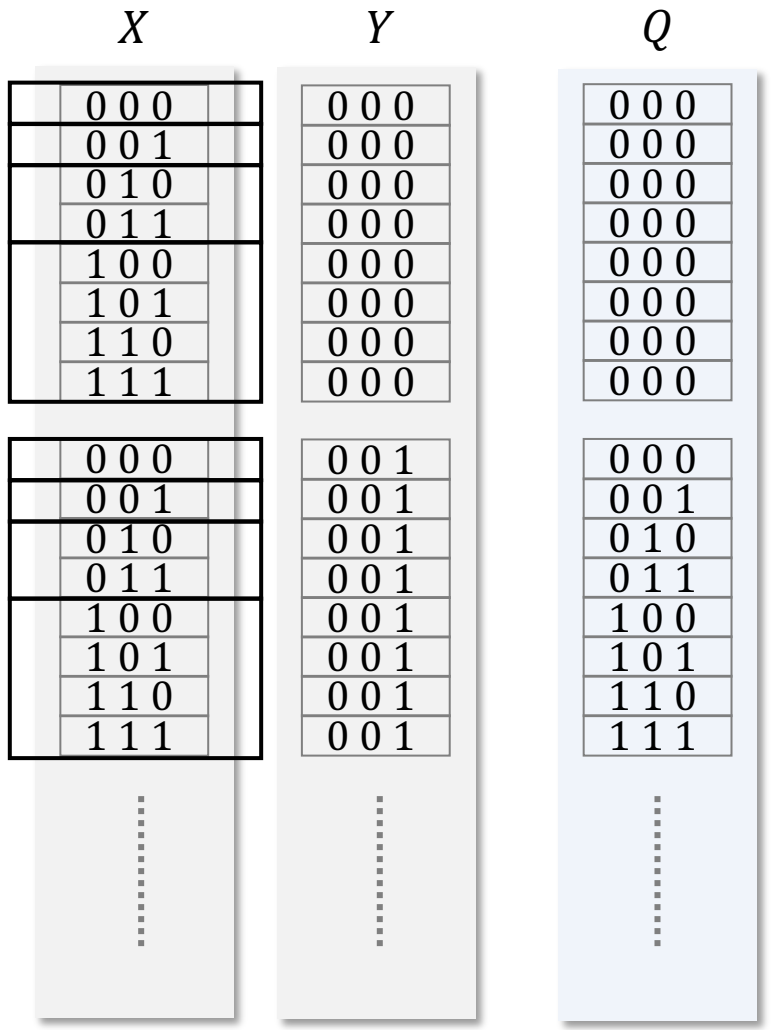
- Using TCAM to lookup the result
  - Requires too many entries

$$Entries = 2^3 \times 2^3 = 64$$



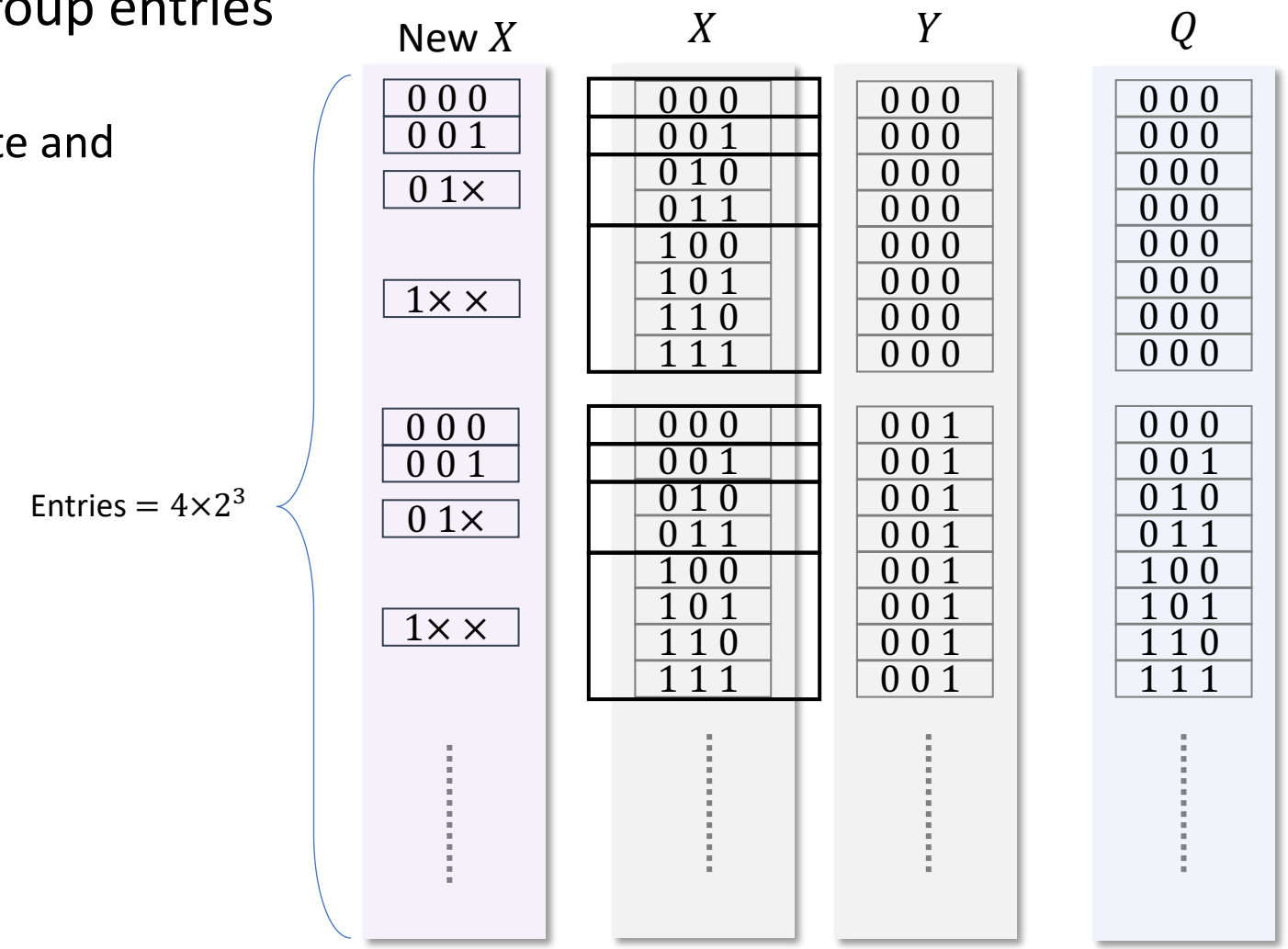
Slightly improved solution: longest prefix match (LPM)

- Grouping numbers such that
  - Numbers in the group are close
  - Each group has a *group head*
  - Using group head to calculate the product

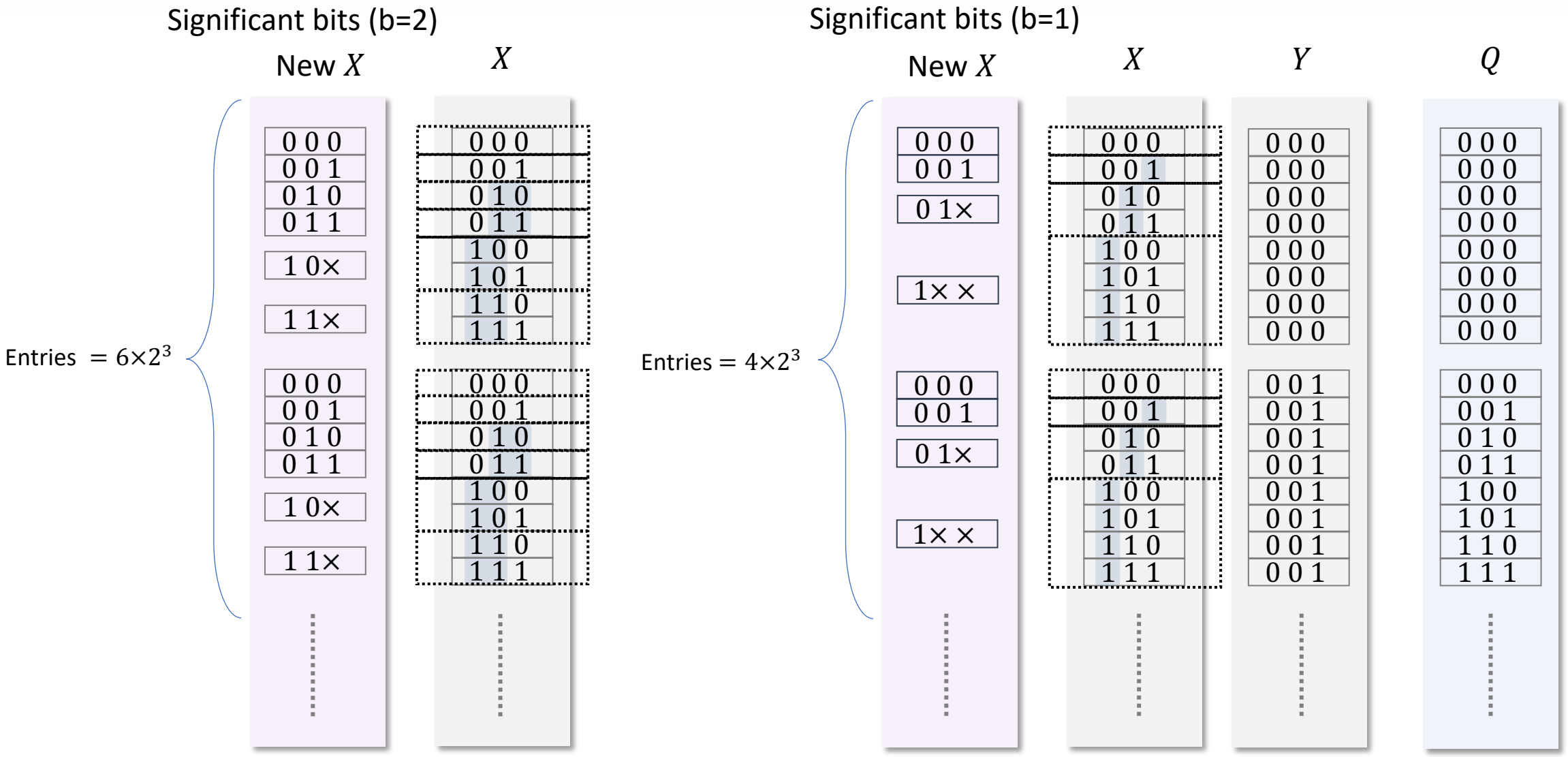


Slightly improved solution: longest prefix match (LPM)

- Using longest prefix match to group entries
  - Less entries is needed
  - No flexibility in adjusting error rate and number of entries



LPM with adjustable significant bit



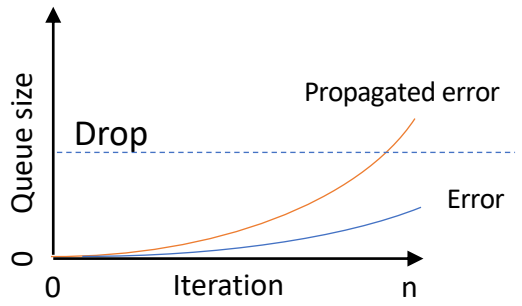
Problems with TCAM population

Error propagation

- It can make the system unstable

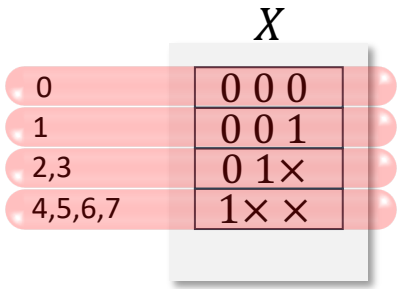
$$x_{new} = f(x_{old})$$

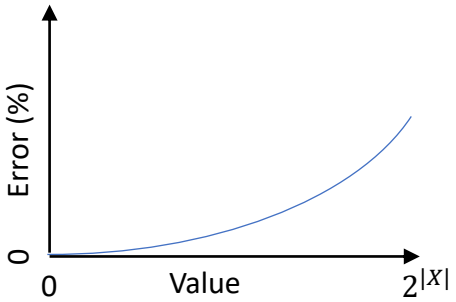
$$RTT_n = \alpha * rtt + (1 - \alpha) * RTT_{n-1}$$



Large error for larger values

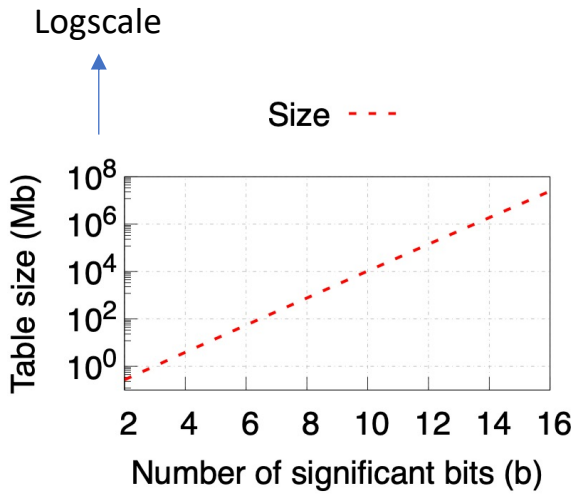
- Regardless of the number of significant bits, large numbers incur large error





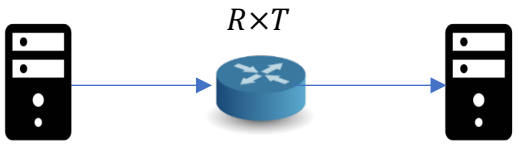
Large table sizes

- There are less than 300 entries
- Very small portion of the variables can fit in



How to use limited size TCAMs and avoid these problems?

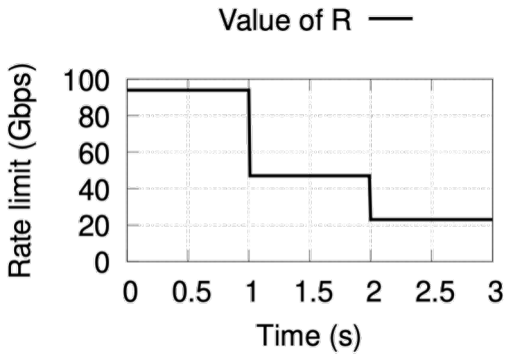
Key insights



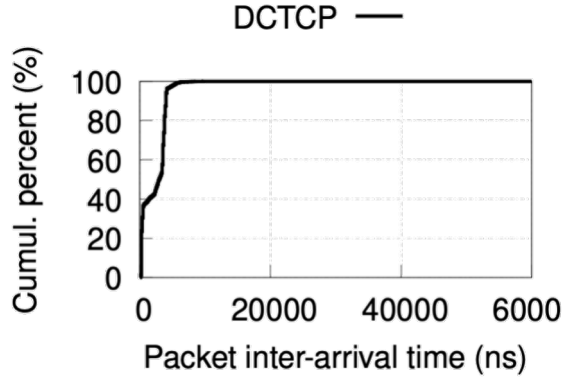
R=100,50,25Gbps

T=[0,6000]

Repopulate when distribution change



Populate based on distribution



Naïve

100Gbps	6000ns
50Gbps	0ns
⋮	⋮
50Gbps	6000ns
25Gbps	0ns
⋮	⋮
25Gbps	6000ns

Finding the distribution of variables is the key to reduce TCAM usage without sacrificing accuracy.

Distribution of R, T

	R	T
0-1s	100Gbps	0ns
		⋮
	100Gbps	6000ns

	R	T
0-1s	100Gbps	0ns
		2ns
		⋮
		300ns
		1000ns
	100Gbps	4000ns
		6000ns

Agenda

- Problem statement
- Background
- Motivation
- Design
  - Control plane
  - Data plane
- Evaluation

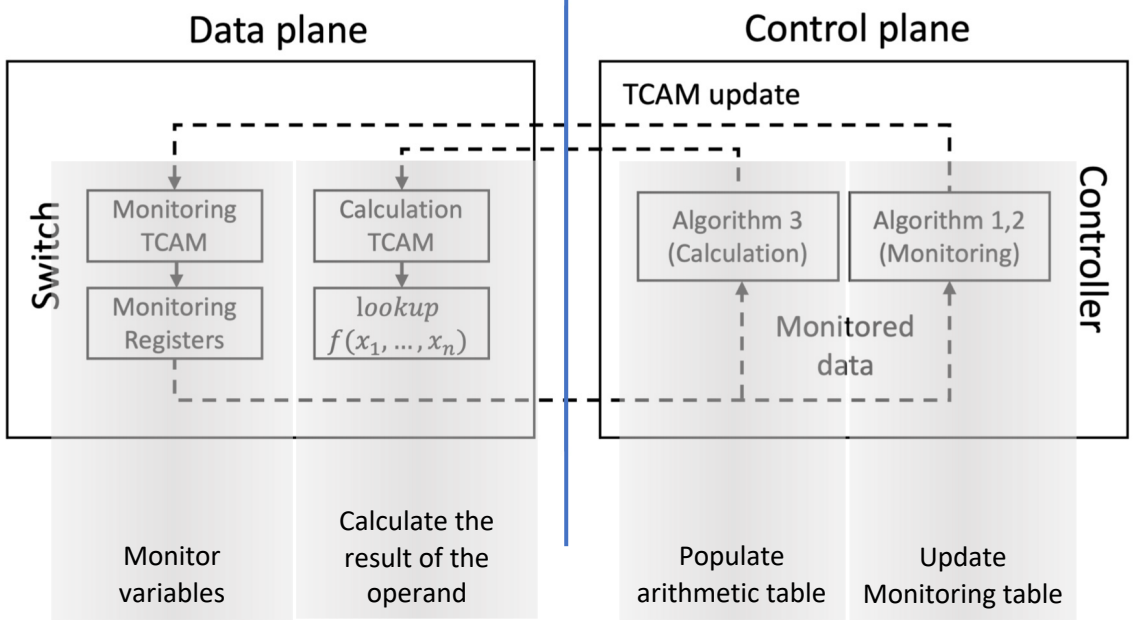
ADA design

Data plane goals

Find the distribution of the variables
Lookup the multiplication

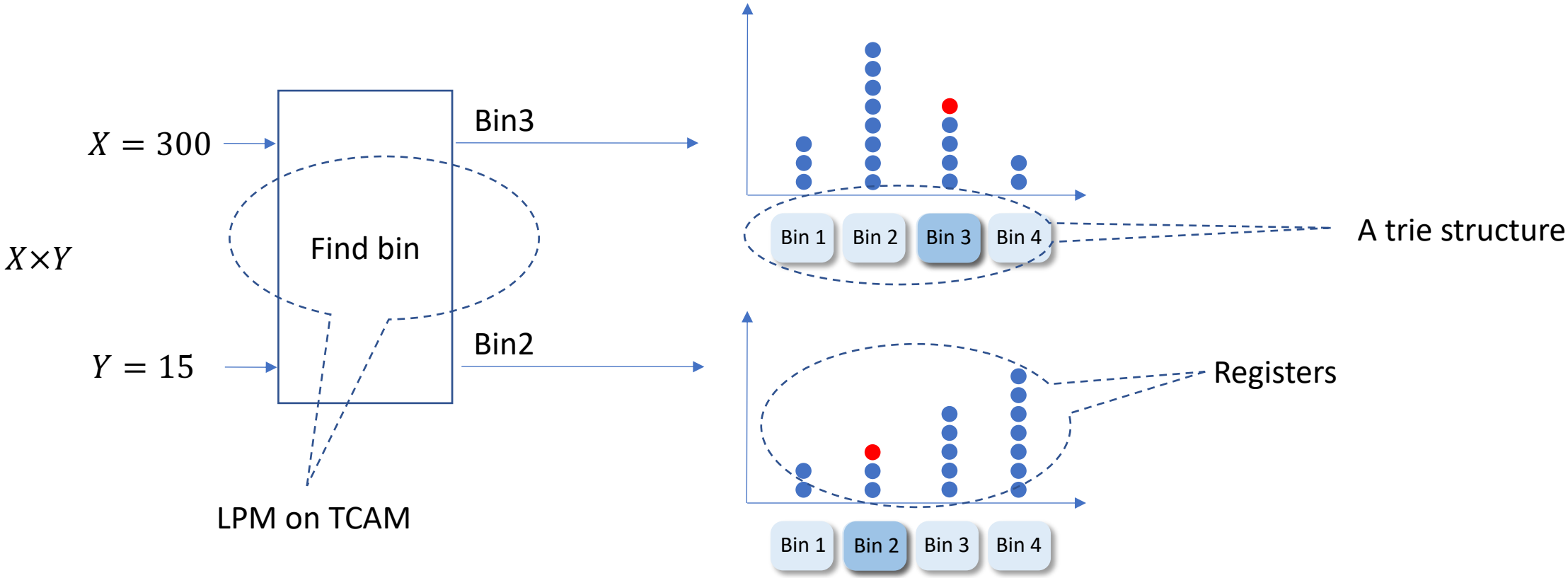
Control plane goals

Collect distribution data
Generate a new multiplication TCAM table
Change the monitoring TCAM table if needed



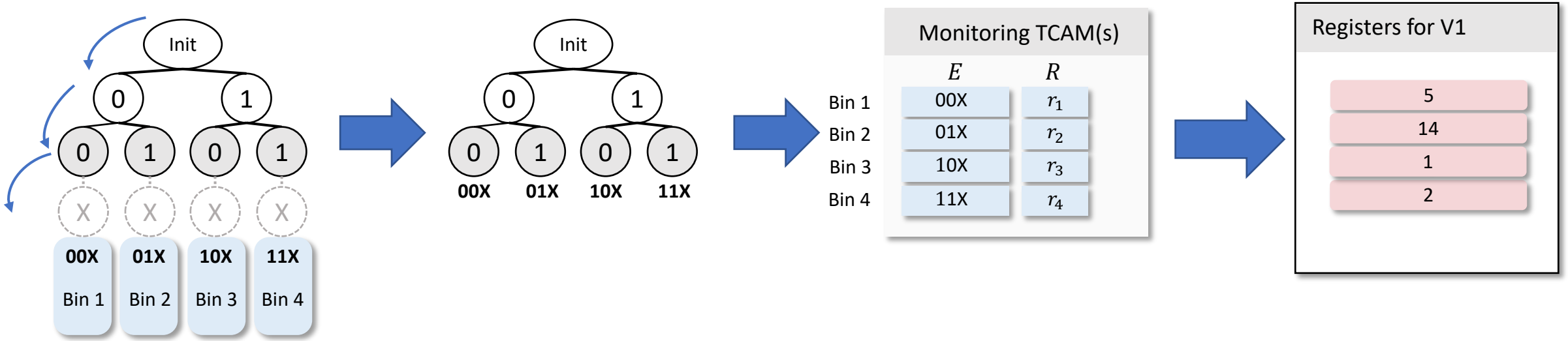
How to detect distribution?

- Using binning mechanism to find the distribution
- Using TCAM to implement bins and trie structure to abstract bins
- Using registers to keep statistics

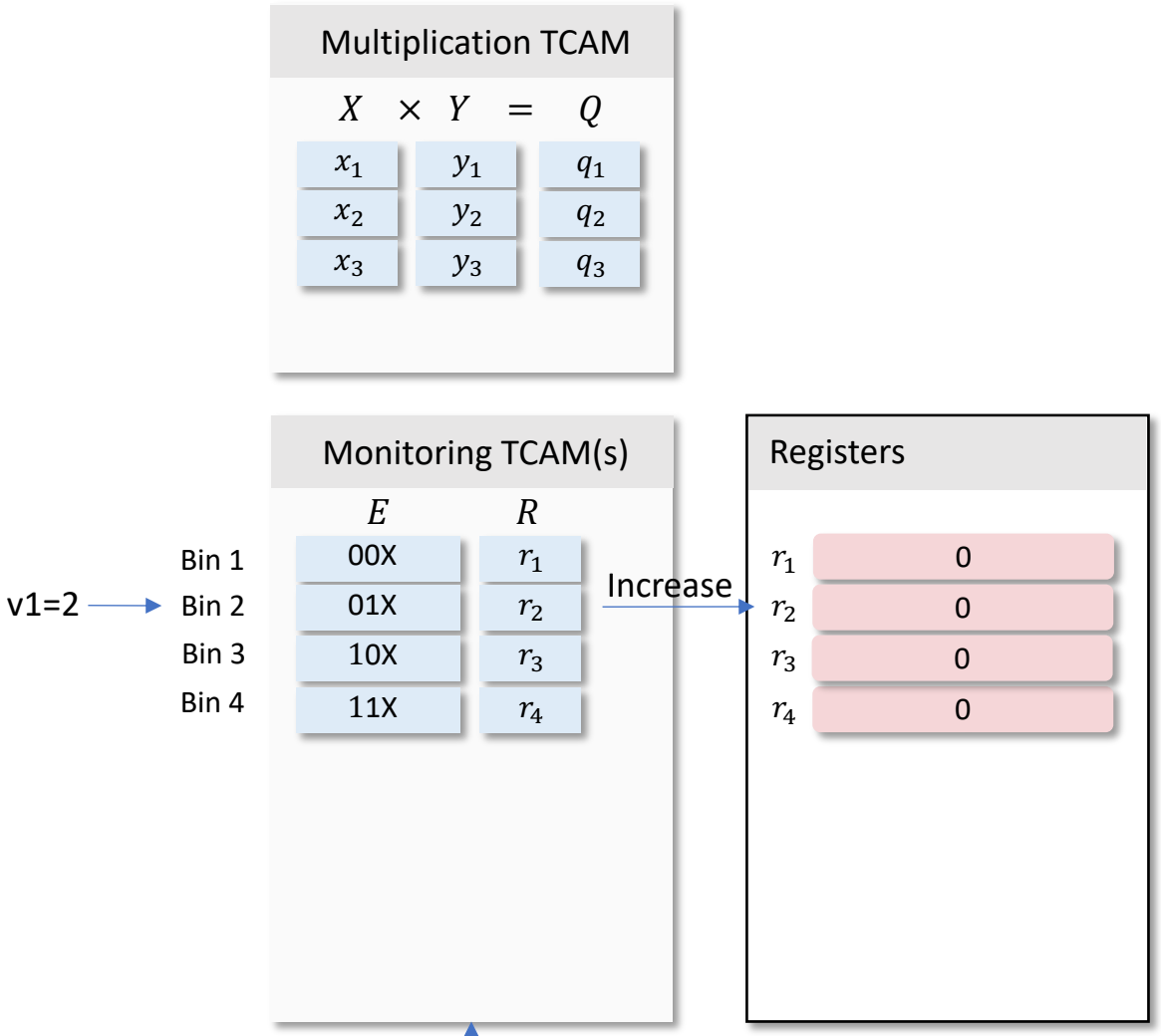


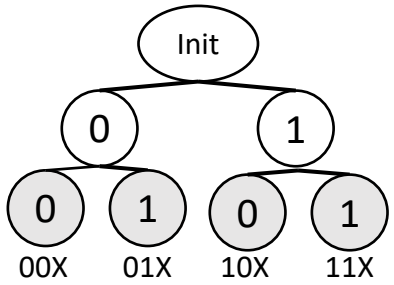
Binnig abstraction

- Using a trie to model the bins
  - Traverse from root to leaf → find an entry for TCAM
  - Example: 3 bits:



ADA design





Update v1

ADA design

Multiplication TCAM

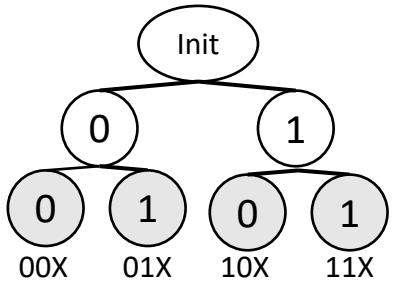
X	$\times$	Y	$=$	Q
x_1		y_1		q_1
x_2		y_2		q_2
x_3		y_3		q_3

Monitoring TCAM(s)

	E	R
Bin 1	00X	r_1
Bin 2	01X	r_2
Bin 3	10X	r_3
Bin 4	11X	r_4

Registers

r_1	3
r_2	14
r_3	1
r_4	2



Update v1

ADA design

Example: proportion to frequency

Bin 1	00X	15% of entries
Bin 2	01X	80% of entries
Bin 3	10X	5% of entries
Bin 4	11X	10% of entries

Multiplication TCAM

X	Y	Q
x_1	y_1	q_1
x_2	y_2	q_2
x_3	y_3	q_3

Monitoring TCAM(s)

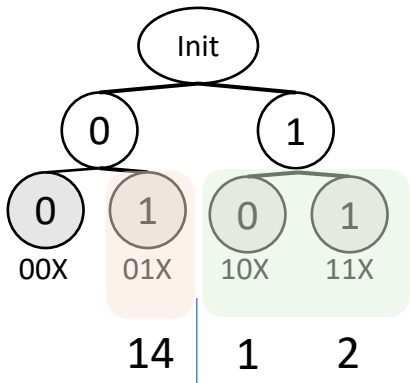
	E	R
Bin 1	00X	r_1
Bin 2	01X	r_2
Bin 3	10X	r_3
Bin 4	11X	r_4

Registers

r_1	3
r_2	14
r_3	1
r_4	2

Update TCAM.
Based on
Algorithm 3

Feedback



Update v1

ADA design

Example: proportion to frequency

Bin 1	00X	15% of entries
Bin 2	01X	80% of entries
Bin 3	10X	5% of entries
Bin 4	11X	10% of entries

Multiplication TCAM

X	Y	Q
x_1	y_1	q_1
x_2	y_2	q_2
x_3	y_3	q_3

Monitoring TCAM(s)

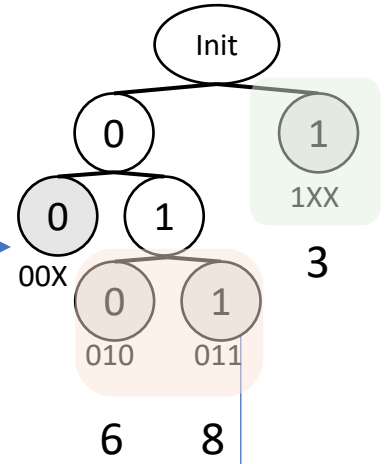
	E	R
Bin 1	00X	r_1
Bin 2	01X	r_2
Bin 3	10X	r_3
Bin 4	11X	r_4

Registers

r_1	0
r_2	0
r_3	0
r_4	0

Update TCAM.
Based on
Algorithm 3

Feedback



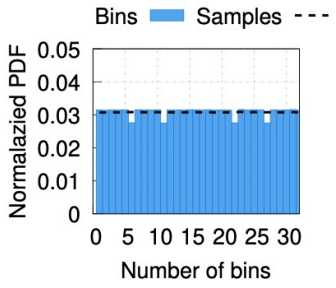
Update v1

Evaluation

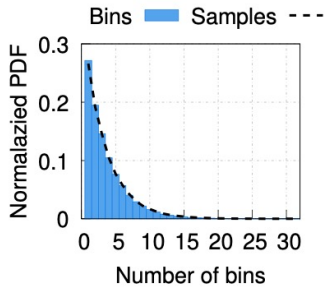
- Network Stand-alone simulator
  - Accuracy and integrity
  - Adaptive increment
- Testbed Experiments
  - Experiment with limited entries
- Large scale simulation

Stand-alone simulator - Accuracy and integrity

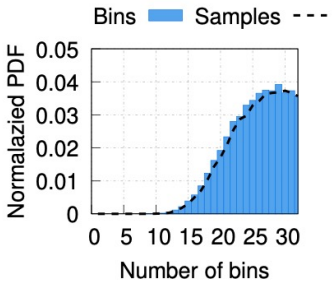
- Binning of different distributions



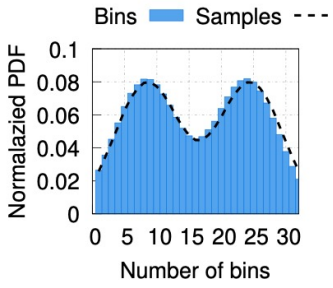
(a) Uniform dist.



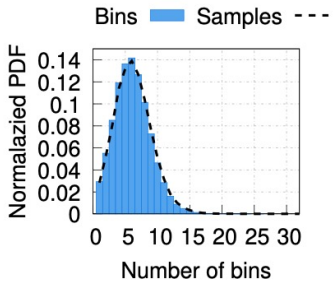
(b) Exponential dist.



(c) Fisher F dist.



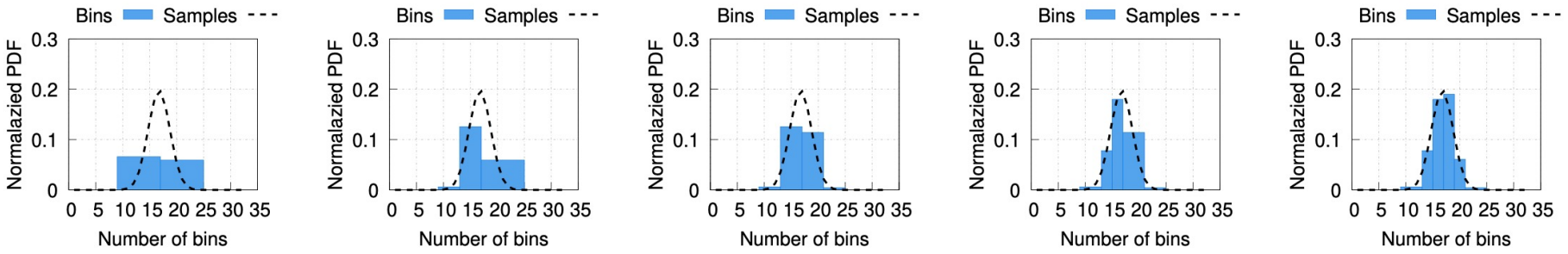
(d) Two Gaussian dist.



(e) Gaussian + Fisher F

Stand-alone simulator - Adaptive increment

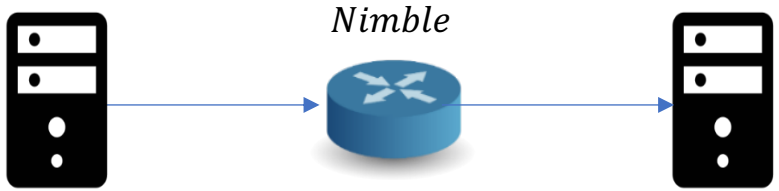
- Convergence if ADA with increasing entries in each iterations



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4 (e) Iteration 5

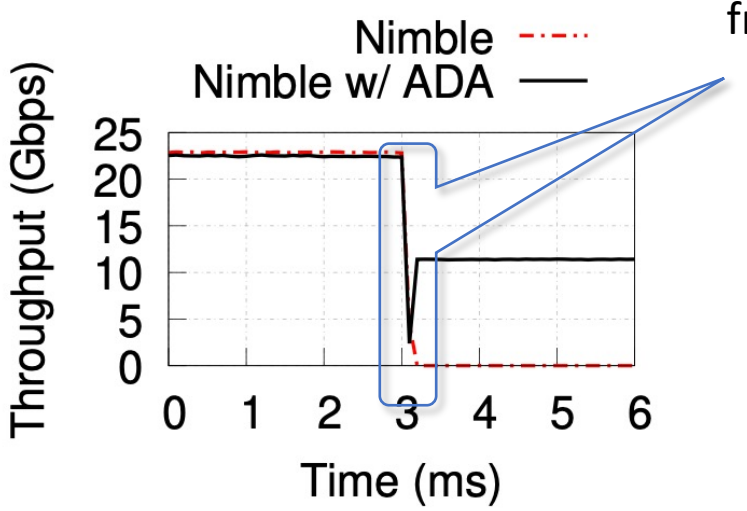
Real implementation in Tofino

- Language: ADA Implementation in P4, and Python
- Running agent: Barefoot Tofino Wedge 100BF-32X Ethernet switch
- Traffic:
  - Single flow between two machines
    - Using 16 parallel iperf3 connections
    - Traffic from a DCTCP enabled client to a server at a full line
- ADA parameters
  - Total of 128 entries for approximate multiplication
  - Total of 12 entries for the monitoring each variable



Real implementation - ADA with limited entries

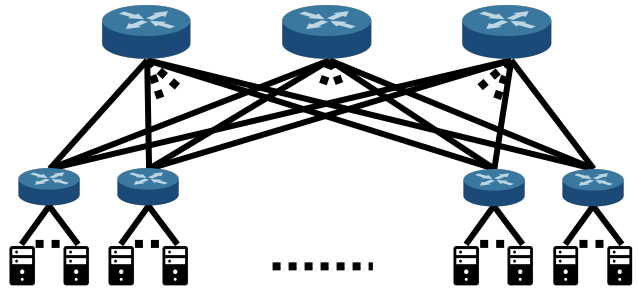
- Nimble fails without ADA due to the match with entries with large errors
- ADA change the table adaptively and doesn't interrupt the system

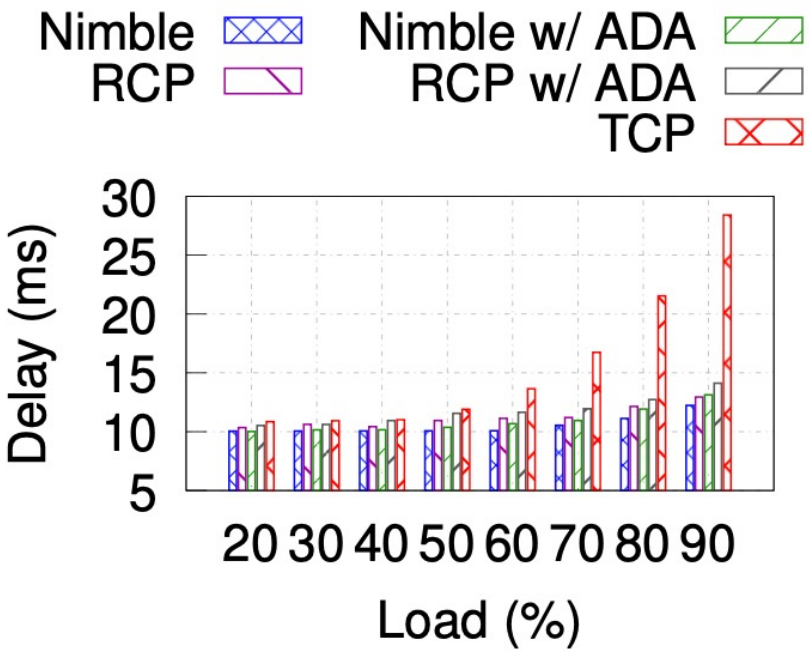


ADA prevents the system from being unstable

Large scale simulation

- Topology
  - Leaf-spine topology with 400 servers and 20 ToR switches
  - 100 Gbps links with a link delay of 1 µs
- Workloads
  - Mix of short flows (8 - 32 KB) and long flows (1 MB)
- Testing 99th percentile FCT for *ideal Nimble* and RCP with and without ADA
  - All approaches with ADA work as good as the ideal case





Thank you for your attention

Questions?

Mojtaba Malekpourshahraki

Email: mmalek3@uic.edu
Website: cs.uic.edu/~mmalekpo