Strategies for Mitigating TCAM Space Bottlenecks

Kirill Kogan[1,2], Sergey Nikolenko[3,4], Patrick Eugster[1,5], Eddie Ruan[6]

Purdue University[1]

IMDEA NETWORKS[2]

Steklov Institute of Mathematics at St. Petersburg[3]

National Research University Higher School of Economics[4]

TU Darmstadt[5]

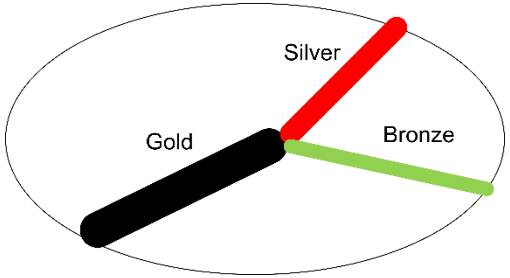
Cisco Systems[6]

Outline

- Current state of the art in packet classification
- Impact of invariants on representation efficiency
- Proposed solutions
- Evaluation and summary
- Ongoing and future work

Types of classifiers

1. Flow: forwarding (path+traffic aggregation)



2. Policies (represent economic models)

- Both can be represented as hierarchical packet match with set actions
- Can have different prioritization schemes, update requirements, etc.

Suggestion: decouple policies from flows

This paper is about efficient representation of policies.

Terminology

$$P \begin{cases} C_1 \begin{cases} F_1 & = (100*, 001*) \rightarrow A_1 \\ F_2 & = (1010, 0001) \rightarrow A_1 \end{cases} \\ C_2 \{ \; F_3 = (001*, 010*) \rightarrow A_2 \end{cases}$$

F – filter
A - action block
C - class
P - policy

Two policies are **semantically equivalent** if in both same actions are applied for any packet.

A policy P is **order-independent** if a policy P' with any order of filters in P is semantically equivalent to P

SW-based vs. TCAM-based solutions (single instance)

SW-based: $N = 3$ rules $K = 2$ fields ~~prefixes ranges~~

$(100*, 001*)$
$(1010, 0001)$
$(001*, 111*)$

Memory	Lookup time
$O(N)$	$O(log^{k-1}N)$
$O(N^k)$	$O(logN)$

TCAM-based: $N = 3$ rules $K = 3$ prefixes ~~ranges~~

$([1,3],[4,31],[1,28])$
$([4,4],[2,30],[4,27])$
$([7,9],[5,21],[3,18])$

Encoding	#TCAM entries
Binary	42+28+50=120
Gray	24+8+32=64

Exploiting structural properties: Kogan et al, SIGCOMM 2014

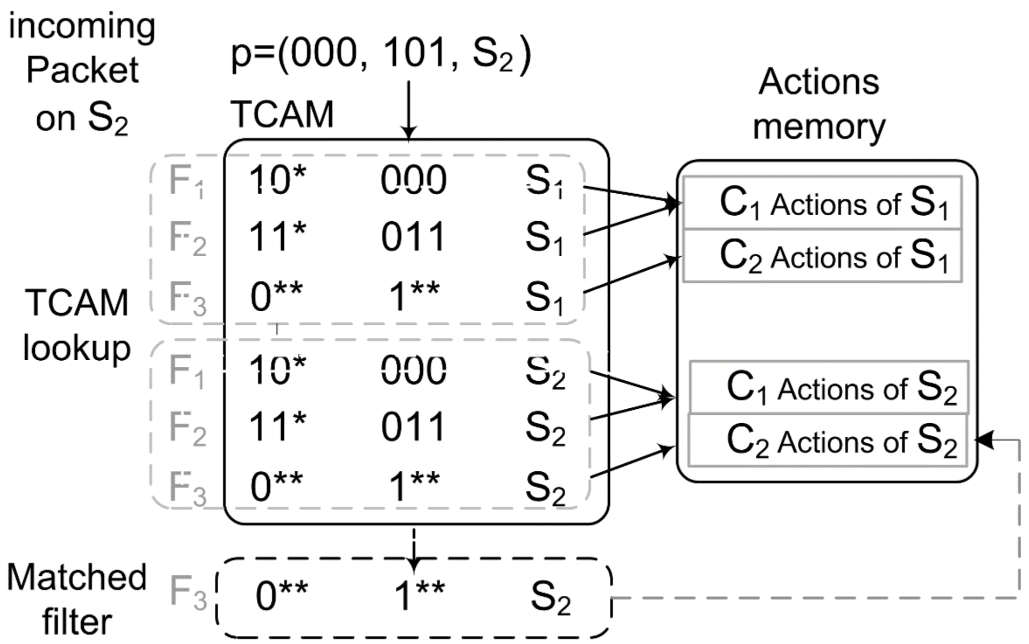
Motivation and problem statement

- Can we optimize if a representation of classifier is optimal (memory perspective)?
- **Invariant**: Number of policies significantly smaller than number of ``flows'‘ (10-100k vs. Gold, Silver, Bronze).

Class-per-Flow-Actions (CFA) requirement: (actions can be flow specific (unsharable): rate-limit, shape or non-flow-specific (sharable) as set ToS);

Problem: Find a semantically equivalent representation of a given policy *P* that share at least part of *P*'s classifier among all flows with *P* configured.

Traditional attachment model and CFA



Pros:

- simple management of action blocks
- independent from classifier structure
- Independent from type of actions

Cons:

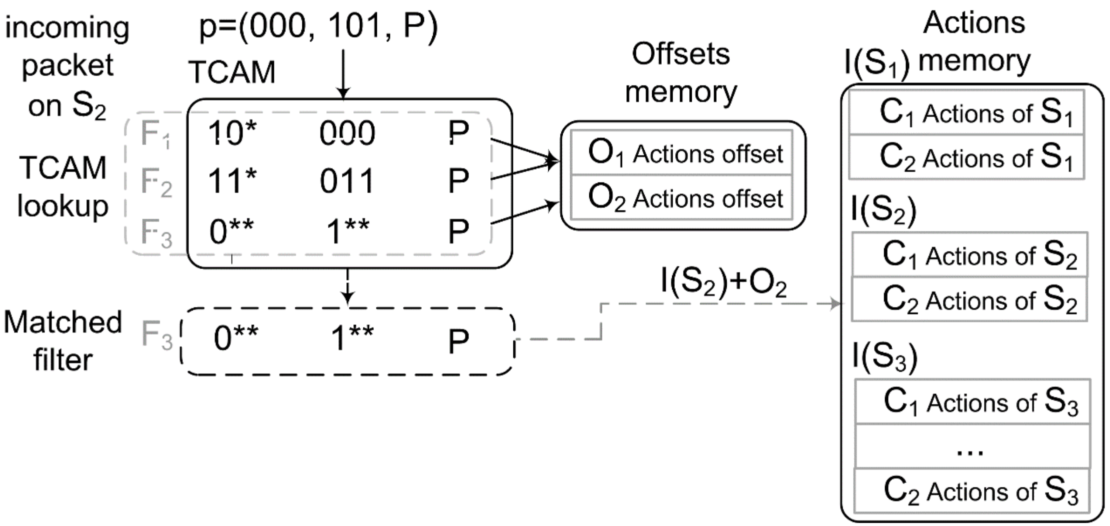
- TCAM memory proportional to #flows

- To implement CFA - a policy instance is configured per flow
- S_i is a flow identifier

Summary: N flows with P configured require $|P| * N$ memory

Can we do it better by still implementing CFA?

Equivalent Actions Layout



Pros:

- optimal implementation of CFA
- independent from classifier structure
- Independent from action types

Cons:

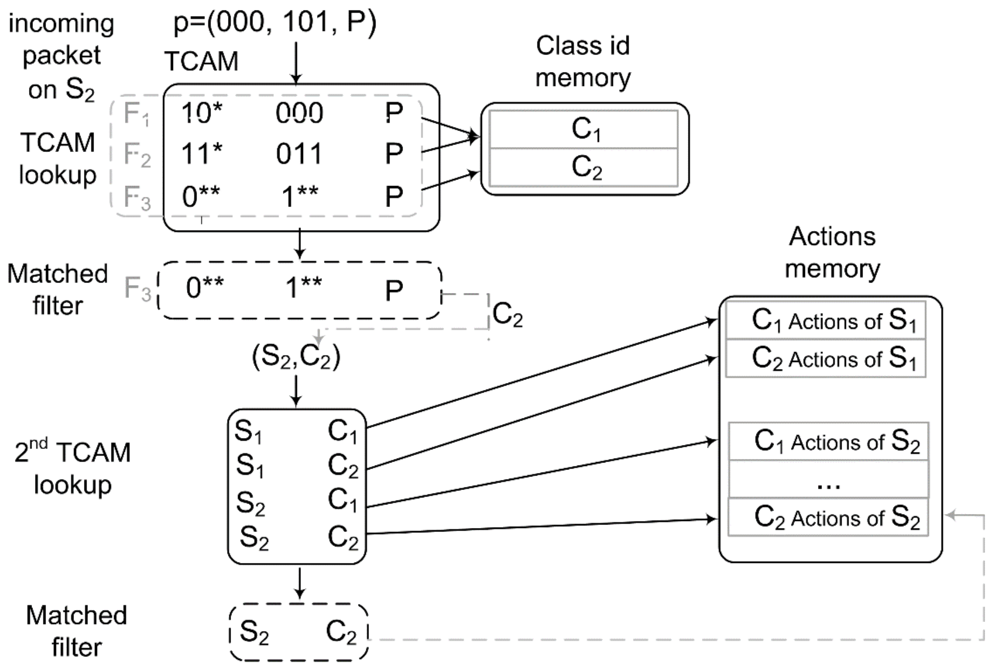
- complex actions block management
- implementation of start+offset logic

- A single classifier instance is configured in TCAM
- Every flow $S_i$maintains a start offset of the flow-specific actions block $I(S_i)$
- An offset O_j of any action block of a class C_j in P from $I(S_i)$ is the same for any flow S_i with P configured
- P is a policy identifier
- Address of action block is $O_j + I(S_i)$

Summary: N flows with P configured require only $|P|$ memory

Can we simplify actions management and avoid start+offset logic?

Two Serial Lookups



Pros:

- a single instance of a policy
- Independent from action types
- simple action block management
- no start+offset logic

Cons:

- 1 TCAM entry per class
- depend on a classifier structure:

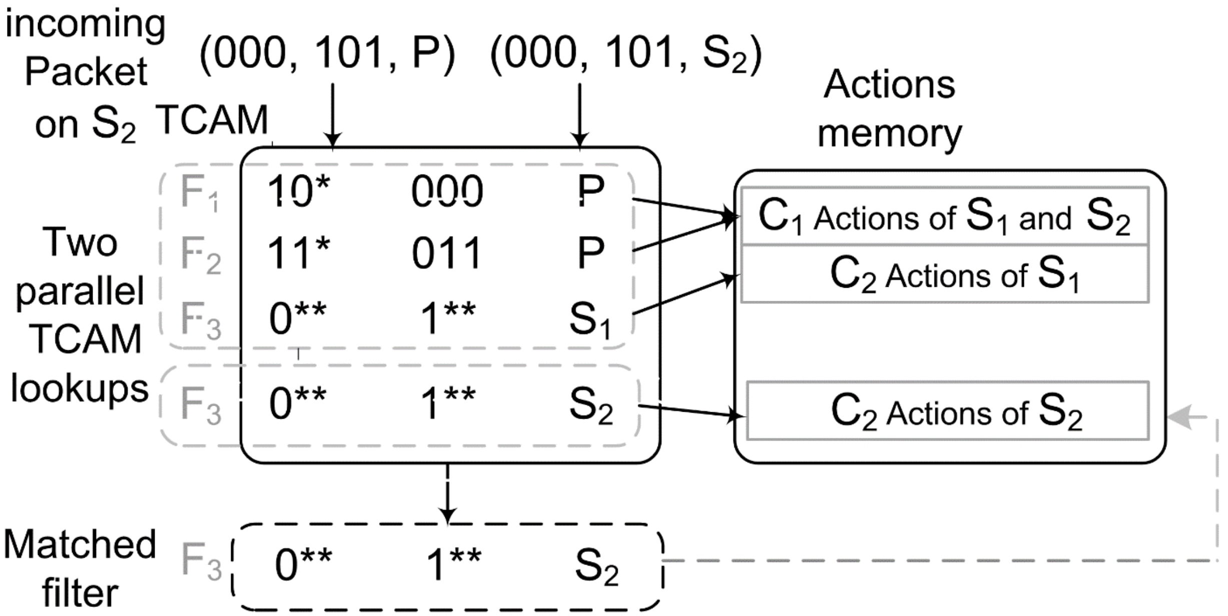
results depend on sizes of classes and classification width

- two serial lookups

- A single classifier instance is configured in TCAM
- The result of the first TCAM lookup is a class id C_i.
- The second lookup is based on a class id C_i and a flow id S_j
- P is a policy identifier

Summary: N flows with P configured require only $|P|$+$N_c * N$ memory, N_c is a number of classes in P

Relaxed CFA requirement cont.



- Two types of actions:
  sharable (e.g. set fields)
  none sharable (e.g. rate-limit)
- Idea: split all filters of $P = P_U + P_S$
  $P_u = \{all\ filters\ with\ unsharable\ actions\}$
  $P_S = \{all\ filters\ with\ sharable\ actions\}$

$P_S = \{F_1, F_2\}$ $\quad P_U = \{F_3\}$

How to compute P_u and P_S?
P is order-independent, simple…
P is order-dependent, lookup in parallel and choose a filter with higher priority (additional prioritization level)
But how to compute P_u and P_S if there is no additional prioritization level?

Order-dependent policy

Idea: exploit merge algorithm $OptimizeClassifier(\,)$ as a ``black box''

Algorithm ComputeShared (P)

$P_s = P$
for each filter F of P_s **do**
    if F has unsharable action **then**
        change action of F to 'TRANSMIT'
P_s=OptimizeClassifier(P_s) ▷ reduce unnecessary filters in P_s

Algorithm ComputeUnshared (P)

$P_u = P$
for each filter F of P_u **do**
    if F doesn't contain any unsharable action **then**
        change action of F to 'TRANSMIT'
P_u=OptimizeClassifier(P_u) ▷ reduce unnecessary filters in P_u.

#	Filter								Action
F_1	1	1	1	*	*	*	0	1	A_1
F_2	1	0	*	*	*	*	*	1	A_2
F_3	0	1	1	0	1	0	0	1	A_3
F_4	1	0	0	1	*	1	0	0	B_1
F_5	1	1	*	*	*	*	*	1	A_4
F_6	1	0	*	*	*	1	*	*	B_1
F_0	*	*	*	*	*	*	*	*	TRANSMIT

#	Filter								Action
F_1	1	1	1	*	*	*	0	1	A_1
F_2	1	0	*	*	*	*	*	1	A_2
F_3	0	1	1	0	1	0	0	1	A_3
F_4	1	0	0	1	*	1	0	0	TRANSMIT
F_5	1	1	*	*	*	*	*	1	A_4
F_6	1	0	*	*	*	1	*	*	TRANSMIT
F_0	*	*	*	*	*	*	*	*	TRANSMIT

#	Filter								Action
F_1	1	1	1	*	*	*	0	1	TRANSMIT
F_2	1	0	*	*	*	*	*	1	TRANSMIT
F_3	0	1	1	0	1	0	0	1	TRANSMIT
F_4	1	0	0	1	*	1	0	0	B_1
F_5	1	1	*	*	*	*	*	1	TRANSMIT
F_6	1	0	*	*	*	1	*	*	B_1
F_0	*	*	*	*	*	*	*	*	TRANSMIT

Two parallel lookups

- P_u is allocated per flow; P_s is sharable
- Only one lookup results action block, the other is just $`transmit'$

Summary: requires TCAM space depends on a structure of filters in P

Pros:

- simple action block management
- no start+offset logic

Cons:

- dependent on classifier structure (in general complex computation)
- dependent on action types
- Two parallel lookups

General mechanism for policy splitting

Policy updates

- Since policies represents economic models, changes are infrequent…

But if updates are still required?

- Action updates (equivalent actions layout is complex, two parallel is simpler, other as traditional)
- Filter updates (all non traditional schemes require less bandwidth to TCAM, two parallel lookups can be more complex)

Summary of methods

	Sharable part in TCAM	Unsharable part in TCAM	Dependency on classifier structure	Additional serial TCAM lookup	Additional parallel TCAM lookup	Start+offset logic	Management complexity of actions memory
Traditional	empty	$\|P\|$	no	no	no	no	no
Equivalent Layout	$\|P\|$	empty	no	no	no	yes	yes
Two Serial	$\|P\|$	1 entry per class per flow	depends on the no. of classes	yes	no	no	no
Two Parallel	$\|P_s\|$	$\|P_u\|$	yes	no	yes	no	no

Evaluation and summary

Policy	Traditional			Equivalent layouts			Two serial			Two parallel		
	shared	unshared	TCAM space, 1000 flows	shared	unshared	TCAM space, 1000 flows	shared	unshared	TCAM space, 1000 flows	shared	unshared	TCAM space, 1000 flows
Cisco1	0	10	10000	10	0	10	10	5	5010	6	4	4006
Cisco2	0	16	16000	16	0	16	16	4	4016	15	1	1015
Cisco3	0	20	20000	20	0	20	20	5	5020	11	9	9011
Cisco4	0	296	296000	296	0	296	296	5	5296	296	0	296
Cisco5	0	158	158000	158	0	158	158	3	3158	158	0	158

Summary: exploiting invariants can achieve better efficiency of representation by several orders of magnitude.

Ongoing and Future

How to increase chances of policies to be sharable?

- Taxonomy of classification fields (sharable, unsharable)

Outcome: Standardization of well-known classifiers and provisioning dependent on capabilities.

- Virtual pipeline architecture

Flows=data   Policies=programs

if policies become too complex to be implemented in existing network infrastructure?

Answer: policy splitting (two parallel lookups scheme can be used as a general splitting infra)

Hay et al, Infocom 2013

Kang et al, CONEXT 2013

Thank You