Aggregated Bit-Vector (ABV) Cross Producting

2015 年度 前期輪講

"Survey and Taxonomy of Packet Classification Techniques"

Abstract and Introduction

原田崇司

神奈川大学大学院 理学研究科 情報科学専攻 田中研究室

2015 年 6 月 16 日

目次

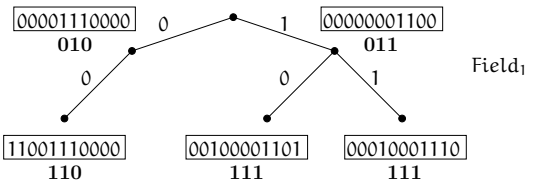
ABV前に提案されたフィルタリング法の問題点

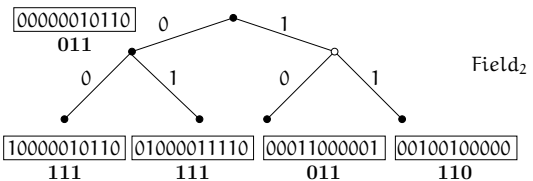
- 特殊なハードを使用　　(TCAM)
- フィールド数 2 以下で構成されるルールに特化
  (CrossProducting, Hicuts, RFC, etc)

  ⇨ フィールド数 3 以上では使用メモリ量が膨大

Aggregated Bit Vector

Rule	$Field_1$	$Field_2$
F_0	00∗	00∗
F_1	00∗	01∗
F_2	10∗	11∗
F_3	11∗	10∗
F_4	0∗	10∗
F_5	0∗	11∗
F_6	0∗	0∗
F_7	1∗	01∗
F_8	1∗	0∗
F_9	11∗	0∗
F_{10}	10∗	10∗





四角で囲われたのが Bit Vector，その下の太字が ABV

Aggregated Bit Vector

$H_1 = 00\ldots$, $H_2 = 11\ldots$ のパケットを BV で探索

Field_1	11001110000
Field_2	00100100000
AND	00000100000

$H_1 = 00\ldots$, $H_2 = 11\ldots$ を ABV も用いて探索

Field_1	**110**
Field_2	**110**
AND	**110**

Field_1	1100111
Field_2	0010010
AND	0000010

初めに ABV の AND をとって，BV の探索する場所を絞り込む

Aggregated Bit Vector

ABV を用いた場合に残念なことが起こる例

Filter	Field_1	Field_2
F_0	00000∗	11∗
F_1	1∗	1010∗
F_2	00000∗	0∗
F_3	01∗	1010∗
F_4	00000∗	100∗
F_5	100∗	1010∗
F_6	00000∗	000∗
F_7	001∗	1010∗
F_8	00000∗	01∗
F_9	11∗	1010∗
F_{10}	0000∗	0∗
F_{11}	010	1010∗
F_{12}	0000∗	1010∗

Aggrgeation Size $= 2$

$H_1 = 00000\ldots,\ H_2 = 1010\ldots$

F_1	1111111	(AVB of 00000)
F_2	1111111	(AVB of 1010)
	1111111	

F_1	1010101010101	(VB of 00000)
F_2	0101010101011	(VB of 1010)
	0000000000001	

Aggregated Bit Vector

ポリシーに違反しないようにフィルタを並び替える

Filter	$Field_1$	$Field_2$
F_0	00000∗	11∗
F_1	1∗	1010∗
F_2	00000∗	0∗
F_3	01∗	1010∗
F_4	00000∗	100∗
F_5	100∗	1010∗
F_6	00000∗	000∗
F_7	001∗	1010∗
F_8	00000∗	01∗
F_{19}	11∗	1010∗
F_{10}	0000∗	0∗
F_{11}	010	1010∗
F_{12}	0000∗	1010∗

Filter	$Field_1$	$Field_2$
F'_0	00000∗	11∗
F'_1	00000∗	0∗
F'_2	00000∗	100∗
F'_3	00000∗	000∗
F'_4	00000∗	01∗
F'_5	0000∗	0∗
F'_6	0000∗	1010∗
F'_7	1∗	1010∗
F'_8	01∗	1010∗
F'_9	100∗	1010∗
F'_{10}	001∗	1010∗
F'_{11}	11∗	1010∗
F'_{12}	010	1010∗

Aggregated Bit Vector

F_1	1111111000000	(VB of 00000)
F_2	0000001111111	(VB of 1010)
	0000001000000	

ルールを並び替えた後に $H_1 = 00000\ldots$, $H_2 = 1010\ldots$ を探索

F_1	1111000	(AVB of 00000)
F_2	0001111	(AVB of 1010)
	0001000	

$7 + 13 = 20 \rightarrow 7 + 2 = 9$

メモリアクセス数減少

F_1	10	(search only 6, 7 bits)
F_2	11	(search only 6, 7 bits)
	10	

Set Pruning Tree

Filter	Destination	Source
F_1	0∗	10∗
F_2	0∗	01∗
F_3	0∗	1∗
F_4	00∗	1∗
F_5	00∗	11∗
F_6	10∗	1∗
F_7	∗	00∗

Table : RuleList1

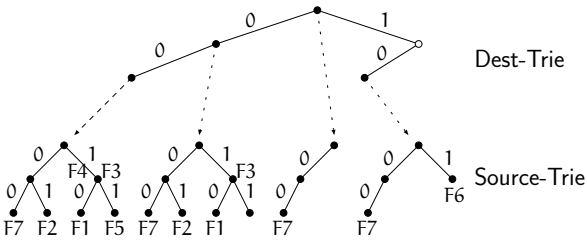
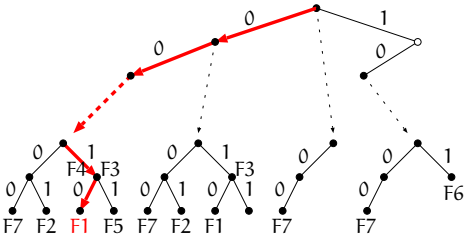


Figure : Set Pruning Tree of RuleList1

Dest-Trie の白丸ノードは，ルートからそのノードへのパスで構成されるビット列が，Destination フィールドにないことを表現

Set Pruning Tree



$D = 00\ldots$
$S = 10\ldots$

Dest-Trie を $0 \to 0$，Source-Trie を $1 \to 0$ と辿り，F_1 を返す．

(Longest Prefix Matching なので，途中の F_3, F_4 は無視)

Grid of Tries

- 二つのフィールドしか持たないルール限定の方法
- フィールドはプレフィックスで指定
  (レンジルールはプレフィックスルールへ変換)
- パケットとルールのマッチングは Longest Prefix Matching

Grid of Tries

Filter	Destination	Source
F_1	0∗	10∗
F_2	0∗	01∗
F_3	0∗	1∗
F_4	00∗	1∗
F_5	00∗	11∗
F_6	10∗	1∗
F_7	∗	00∗

Table : RuleList1

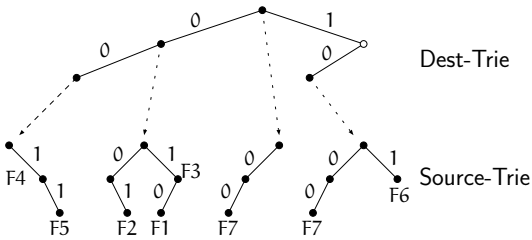
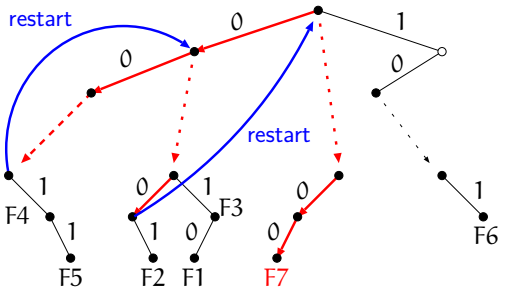


Figure : Grid of Tries of RuleList1

ルールの重複を避けるために,
Destination に完全に一致する箇所にのみ Source のトライを構成

Grid of Tries



Dest-Trie

$D = 001\ldots$

$S = 001\ldots$

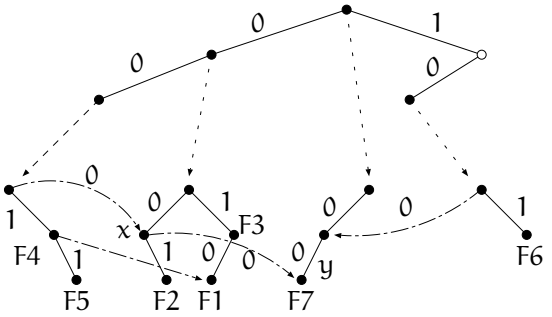
Source-Trie

D-Trie を $0 \rightarrow 0$ と辿るが，S-Trie を辿れず，D-Trie の $0*$ へ

S-Trie を 0 と辿るがフィルタに合致しないので，D-Trie の $*$ へ

S-Trie を $0 \rightarrow 0$ と辿って，F_7 を返す.

Grid of Tries (with Switch Pointers)



探索時間計算量を $O(dW^2)$ から $O(dW)$ とするために

スイッチポインタを与える

Extend Grid of Tries

Filter	DA	SA	DP	SP	Prot
F_1	0∗	10∗	∗	80	TCP
F_2	0∗	01∗	∗	80	TCP
F_3	0∗	1∗	17	17	UDP
F_4	00∗	1∗	∗	∗	∗
F_5	00∗	11∗	∗	∗	TCP
F_6	10∗	1∗	17	17	UDP
F_7	∗	00∗	∗	∗	∗

Table : ポート番号，プロトコル追加

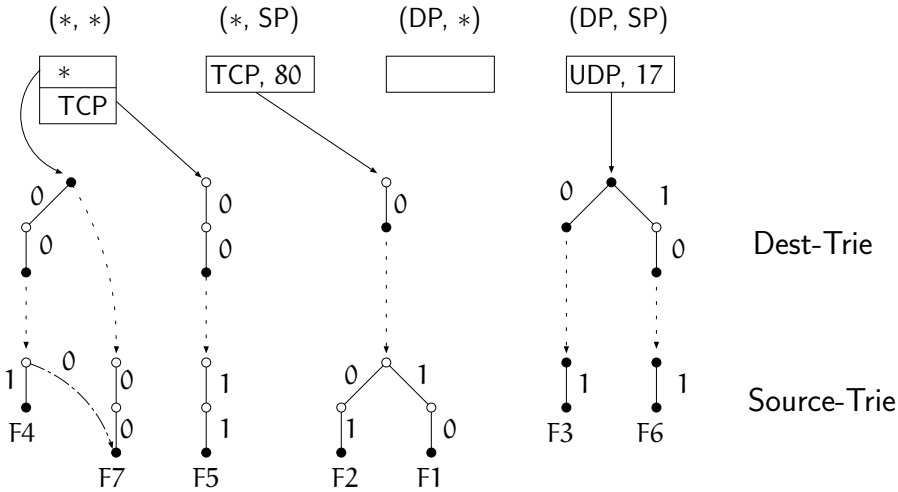
- Port 番号
- プロトコル

もフィルタリングの基準として使用
(ポート番号は範囲指定不可)

ポート番号の四つ組合わせに対してハッシュ表を作成

(DA, SA) の組 =
{ (∗, ∗), (∗, 指定), (指定, ∗), (指定, 指定) }

Extend Grid of Tries（未完）



Cross Producting

Filter	DA	SA	DP	SP	Prot
F_1	0∗	10∗	∗	80	TCP
F_2	0∗	01∗	∗	80	TCP
F_3	0∗	1∗	17	17	UDP
F_4	00∗	1∗	∗	∗	∗
F_5	00∗	11∗	∗	∗	TCP
F_6	10∗	1∗	17	17	UDP
F_7	∗	00∗	∗	∗	∗

DA	SA	DP	SP	Prot
0∗	10∗	∗	80	TCP
00∗	01∗	17	17	UDP
10∗	1∗		∗	∗
∗	11∗			
	00∗			

各フィールドにおいて異なるルールを集めてその直積をとる．

そして，各々の組み合わせに対して最優先ルールを与える．

On Demand Cross-Producting

Cross-Producting は，空間計算量が $O(N^K)$ となり実用的でない．
(N はルール数，K はフィールド数)

⇨ フィルタリングを行いながら Cross-Producting 表を作成

参考文献

[1] F. Baboescu, and G. Varghese, “Scalable Packet Classification,” SIGCOMM Comput. Commun. Rev., vol.31, no.4, pp.199—210, Aug. 2001.

[2] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, “Fast and Scalable Layer Four Switching,” SIGCOMM Comput. Commun. Rev., vol.28, no.4, pp.191—202, Oct. 1998.