L-021 パケットフィルタリング最適化のためのルール集合分割法

A dividing method of rule set for packet filtering optimization

池本泰斗* 田中賢* 三河賢治†
Taito Ikemoto Ken Tanaka Kenji Mikawa

1 まえがき

パケットフィルタリングは，外部から到達するパケットを監視して，その通過の可否を判断することによって不正アクセスを防止するための仕組みである．[1] では，パケットフィルタリングに起因する遅延を軽減するために，ルール集合を独立ルール集合に分割するアルゴリズムを提案した．しかし，このアルゴリズムは独立ルール集合をナイーブに分割するものであった．本稿では，より遅延を減少させられる独立ルール集合への分割法を示す．

2 独立ルール集合について

2.1 ルール間の関係

パケットフィルタリングルールには，独立と重複の二つの関係がある．二つのルール $R = b_1 b_2 \ldots b_{dw}$ と $R' = b'_1 b'_2 \ldots b'_{dw}$ について，三つの条件 $b_i \neq b'_i$ ，$b_i \neq$ ‘$*$’，$b'_i \neq$ ‘$*$’，を同時に満たすビットが存在するとき，R と R' は独立である．二つのルール $R = b_1 b_2 \ldots b_{dw}$ と $R' = b'_1 b'_2 \ldots b'_{dw}$ について $i = 1, 2, \ldots, dW$ の各ビットが三つの条件 $b_i = b'_i$, $b_i =$ ‘$*$’，$b'_i =$ ‘$*$’ を満たすとき，R と R' は重複である．

2.2 独立ルール集合

フィルタ R に属するルールをルール集合 D と D' に分割する．$D, D' \subset R$ とする．$\forall r \in D$ と $\forall r' \in D'$ について，r と r' が互いに独立であるとき，D と D' は独立なルール集合という．

3 木を用いたアルゴリズム

[1] のアルゴリズムは木を用いてルール集合の関係を判定し，ポリシーに違反しない位置にルールを追加していくことによって，ルール集合を独立ルール集合に分割する手法である．表 1 のルール集合を小出 [1] のアルゴリズムによって分割すると，図 1 のようになる．白ノードの子は独立関係，黒ノードの子は重複関係にある．

表 1：ルール集合の例

Filter	F_1	F_2	Filter	F_1	F_2
R_1	0000	0001	R_6	$*10*$	$1***$
R_2	$11*0$	1110	R_7	$0*1*$	$0*1*$
R_3	$000*$	0000	R_8	$0**1$	$0*1*$
R_4	$10**$	$11**$	R_9	0010	0001
R_5	$**00$	$10*1$	R_{10}	$**10$	$000*$

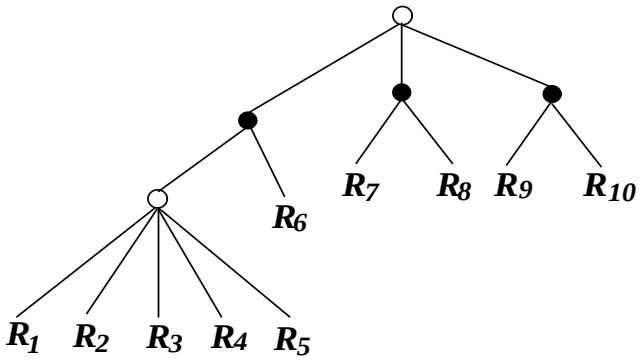

図 1: 木を用いたアルゴリズムによる分割

表 1 のルール集合の遅延 [1] は 531 である．図 1 に分割したものを遅延を最小にするために独立ルール集合の平均評価パケット数が降順になるように並び替えると，遅延は 436 に減少する．

4 ブロック化・パーティション化アルゴリズム

遅延をより減少させるアルゴリズムとして，ルールの重複関係から芋づる式にルールをまとめるブロック化（Algorithm 1）と，ブロックの中を区切ることによって，ブロック内で新しくブロックを作れるようにするパーティション化（Algorithm 2）を交互に行うものを提案する．
以下の疑似コードにおいて，B はブロック，R はフィルタリングルール，E は除外ルールを表している．

*神奈川大学大学院理学研究科情報科学専攻
†新潟大学学術情報基盤機構情報基盤センター

Algorithm 1 Blocking(R)

```
01:while rules other than a default rule remain do
02:     B_i ← GenerateBlock()
03:     top rule is added to B_i
04:     R_j ← top rule in B_i
05:     while R_j that are not searched in B_i exists do
06:         while R_k that are not searched exists do
07:             R_j is compared with rule set
08:             if R_j overlaps with R_k in rule set then
09:                 R_k is added to B_i
10:             end if
11:         end while
12:         R_j ← next rule in B_i
13:     end while
14:end while
```

Algorithm 2 Partitioning(R)

```
01:while R_i that are not searched in block
02:   if number of overlap is 2 or more then
03:     up ← search the number of roll of upper
04:     lw ← search the number of roll of lower
05:     if up > lw then
06:         R_o ← number of overlap - lw
07:     end if
08:     else if up < lw then
09:         R_o ← number of overlap - up
10:     end if
11:     else if up = lw then
12:         R_o ← number of overlap - lw
13:     end if
14:     if R_o > E_o then
15:         E_i ← R_i
16:         E_o ← R_o
17:     end if
18:   end if
19:end while
20:exclude E_i and its roll rules to small
```

このアルゴリズムによって表 1 のルール集合を分割したものが図 2 である．また，図 2 を図 1 の木と同じ形式に変換したものが図 3 である．
この分割を小出法と同様に平均評価パケット数が降順になるように並び替えると，遅延は 365 に減少し，従来の木を用いたアルゴリズムよりも減少していることが分かる．
n 個のルールが与えられたとき，ブロック化，パーティション化ともに空間計算量は $O(n^2)$ となる．

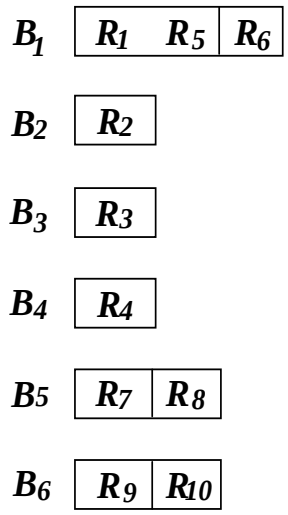

図 2: ブロック化・パーティション化による分割 1

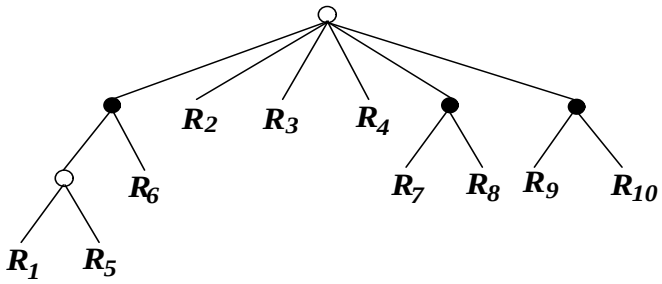

図 3: ブロック化・パーティション化による分割 2

5 今後の課題

木を用いたアルゴリズムでは，前述のように独立ルール集合を評価パケット数の平均によって降順に並び替えることで，遅延を減少させていた．しかし，独立ルール集合は他の独立ルール集合と順番を並び替えられるだけではなく，独立ルール集合の任意の位置に独立ルール集合かそのルールの一部を挿入することができる．この性質を利用すれば，さらなる遅延の減少が期待できる．今後は，この点に着目した並び替えアルゴリズムを考える必要がある．
また，実際のフィルタリングルールにおいて，重複関係の深さはある範囲に収まると考えられる．重複数を制限した状況での効率的なアルゴリズムの開発を検討することも今後の課題である．

参考文献

[1] 三河賢治，田中賢，小出淳一，“ブロック分割によるパケットフィルタ最適化問題の一解法 ”，電子情報通信学会論文誌，Vol.J94-B,No.10,pp.1408-1417,Oct.2011.