•可信计算与信息安全•

DOI:10.15961/j.jsuese.201700608

一种基于单元空间划分的快速防火墙包分类算法

程玉柱, 王伟平*, 王建新

(中南大学 信息科学与工程学院, 湖南 长沙 410083)

摘　要:针对现有包分类算法存在分类时间长和需要较大存储空间的问题, 提出一种基于单元空间划分的快速防火墙包分类方法(Uscuts)。方法主要包括规则预处理、规则空间划分以及决策树构建阶段。首先, 基于多维矩阵设计模型, 将原始规则按逆序依次映射到多维矩阵, 得到与原始规则语义相同但规则空间相互独立的目标规则。随后, 对目标规则对应空间进行划分并构建分类决策树, 决策树的每条树支与各独立多维规则子空间一一对应, 即决策树的每个叶子节点恰好只关联1条规则。因此, 在Uscuts包分类方法中, 当数据包匹配到叶子节点时, 可以直接判定数据包的分类决策为“accept”, 不同于传统包分类方法需要在叶子节点关联的规则分组内继续执行顺序匹配, 该性质显著地提高了包分类速度。此外, 在划分规则空间时, Uscuts方法采用基于单元空间边界的划分方式, 比传统分类算法的平均划分方式, 有效减少了规则子空间数目, 节省了存储空间。为验证方法的包分类效果, 设计了不同规模大小的规则和数据集测试方法的有效性。由测试结果可以看出, Uscuts分类方法的时间复杂度在一般情况下能达到$O(0.75k\cdot\mathrm{lb}(n))$, 即便在最坏情况下也不超过$O(k\cdot\mathrm{lb}(n))$, 其中, n和k分别为规则条数和维数。理论分析表明, 与现有基于决策树的分类方法相比, Uscuts方法具有更高的分类效率, 且所需存储空间更小。

关键词:包分类; 防火墙; 单元空间; 决策树算法

中图分类号:TP393　　**文献标志码**:A　　**文章编号**:2096-3246(2018)04-0144-09

A Fast Firewall Packet Classification Algorithm Using Unit Space Partitions

CHENG Yuzhu, *WANG Weiping**, *WANG Jianxin*

(School of Info. Sci. and Eng., Central South Univ.,Changsha 410083, China)

Abstract: In order to solve the problems of long classification time and large storage space incurrent packet classification algorithms, a fast packet classification approach (Uscuts) based on unit space partition was proposed. The method mainly includes rule preprocessing, rule space partition and decision tree construction stage. First, based on the multidimensional matrix design model, the original rules were mapped in reverse order to the multidimensional matrix, and the target rules which have the same semantics with the original rules and mutually independent rule spaces were obtained. Then, the corresponding space of the target rules was divided and a classification decision tree was constructed. Each branch of the decision tree corresponds to an independent multidimensional rule subspace, which means each leaf node of the decision tree just relates only one rule. Therefore, when the packet was matched to the leaf node, the classification decision of the packet can be directly determined to be accept. Unlike the traditional packet classification methods, the sequence matching should be continued within the rule group associated with the leaf nodes, which significantly raised the speed of packet classification. In addition, when dividing the rules space, Uscuts used the division method based on the unit space boundary. Compared with the current classification algorithms, the number of the rule subspace was effectively reduced and the storage space was saved. To further validate the efficiency and effectiveness of the proposed algorithm, different sizes of rules and packets were generated to test the time required for packet classification. The test results showed that the time complexity of Uscuts could reach $O(0.75k\cdot\mathrm{lb}(n))$, even in the worst case, it was not more than $O(k\cdot\mathrm{lb}(n))$, here n and k were the number and dimension of the rules, respectively. Theoretical analysis and experimental results demonstrated that, compared with existing classification methods based on decision tree, Uscuts method has higher classification efficiency and smaller storage space.

收稿日期:2017 – 08 – 01

基金项目:国家自然科学基金资助项目(61672543); 湖南省研究生科研创新项目资助(CX2014B081); 湖南省科技计划重点研发计划资助项目(2016JC2009); 湖南省教育厅科研项目资助(17B022)

作者简介:程玉柱(1980—), 男, 博士生. 研究方向: 网络信息安全. E-mail: peter_cheng@csu.edu.cn

* 通信联系人 E-mail: wpwang@csu.edu.cn

网络出版时间:2018 – 07 – 11 16 : 30 : 00　　**网络出版地址**: http://kns.cnki.net/kcms/detail/51.1773.TB.20180711.1630.010.html

Key words: packet classification; firewall; unit space; decision tree algorithms

防火墙是指设置在不同网络安全域之间的安全元件，其根据预设的安全策略控制（允许、拒绝）出入网络的数据包[1]。近年来，随着网络应用的不断发展，防火墙策略包含的规则越来越多，使得包分类速度受到了很大影响。但同时，链路速度变得越来越快，使得包括防火墙在内的许多网络设备的数据包分类速度需求更高[2]。防火墙包分类技术的基本原理是根据预先设置的防火墙规则检查数据包的相关字段，例如，源/目的IP地址、协议号等信息，以判断数据包所匹配的规则；再按照规则所定义的动作处理数据包[3]。显然，最直观的包分类方法为顺序匹配，其基本思路是将规则集按照优先级从高到低的顺序，组成线性表；接收到数据包时，顺序访问线性表中的每一条规则，直到找到匹配规则为止。顺序匹配算法的时间复杂度与规则条数呈线性关系。随着防火墙规则数目的增多，顺序匹配所需要的时间会越来越多，导致防火墙吞吐量下降，影响网络性能。因此，有必要研究如何提高防火墙包分类速度，提高防火墙性能。

在此背景下，研究人员提出了许多优秀的包分类算法，包括基于硬件TCAM的包分类算法[4]、基于软件的包分类算法如Hicuts、HyperCuts等[5–12]。通常，TCAM能够并行地访问存储在其中的每一个规则条目，基于TCAM的包分类算法时间复杂度为常量。但是，TCAM自身的一些不足限制了其使用范围，其不足包括：1）范围扩展不易，一条以范围形式表示的k维规则，转换成以前缀形式表示时，最坏情况下有$(2w-2)^k$条规则，其中，w为该范围所在维的域长；2）能量消耗过大，TCAM每比特能量消耗是SRAM每比特能量消耗的150倍；3）费用较高，1 MByte的TCAM通常需要200～250美元[4]。

基于软件的包分类方法一般是指在分类数据包之前对规则空间进行划分，而根据划分准则的不同可分为：1）基于哈希表的划分。为每个不同的前缀长度构建一个哈希表，前缀长度相同的规则分量被存储在同一个哈希表中。当分类数据包时，顺序访问所有哈希表直至找到最长匹配前缀为止。代表算法有2维数据包分类算法RS（rectangle search）[5]、多维数据包分类算法TSS（tuple space search）[6]和BSOL（binary search on leaves）算法[7]等。2）基于决策树的划分。按照一定方式，对规则所在的多维空间进行递归分解，并建立一棵决策树。决策树的根节点代表整个多维空间，非根节点则代表子空间。递归分解的结束条件是节点关联的规则条数小于等于预设的阈值τ。当接收到数据包时，先访问决策树的根节点，再根据空间的划分方式决定如何访问下一个节点。通常，基于决策树的包分类算法会最终访问叶子节点。当访问叶子节点时，算法顺序访问该节点所关联的至多τ条规则，以确定数据包所能匹配的规则。代表性算法有Hicuts算法[8]、HyperCuts[9]算法、Hierarchical Trie算法等[10–12]。

现有基于软件的包分类方法中，虽然遍历哈希表或决策树能达到对数级时间复杂度，但是在找到最长匹配前缀或叶子节点之后需要在τ条规则内继续执行顺序匹配[3]，而顺序匹配的执行时间效率与规则条数呈线性关系，该过程较大程度上降低了方法的包分类效率。此外，在对规则所在多维空间进行划分时，一条规则可能需要被复制到多个规则分组，导致方法需要大量的存储空间。对此，提出一种基于单元空间划分的快速防火墙包分类方法（Uscuts），该方法在对原始防火墙规则进行预处理的基础上构建分类决策树，使得每个叶子节点关联规则条数有且只有一条，避免了在遍历到叶子节点后需要继续执行规则顺序匹配，从而有效地提高了包分类速度。此外，基于单元空间的划分方法尽可能地减少了所划分子空间数目，从而节省了存储空间。

1　问题分析

一般而言，防火墙规则可用形如“$F_1 \in D(F_1) \wedge F_2 \in D(F_2) \wedge \cdots \wedge F_k \in D(F_k) \rightarrow decision$”的区间形式表示，其中，$F_i(1 \le i \le k)$表示源地址、目标地址、源端口和目标端口等，$D(F_i)$表示对应的域值区间，$decision$表示规则的决策（接受或拒绝），$k$为防火墙规则维数。

基于决策树的包分类方法，首先对规则所在的多维空间进行划分以构建分类决策树。当分类数据包时，从根节点出发遍历决策树找到叶子节点，然后，顺序访问该叶子节点所关联的一组规则。

以Hicuts分类算法为例，其按平均划分方式，对规则所在的多维空间进行递归分解以建立分类决策树。如图1所示，设原始规则集包含11条2维规则$\boldsymbol{r}_1:(0111*,011*)$；$\boldsymbol{r}_2:(010*,011*)$；$\boldsymbol{r}_3:(*,111*)$；$\boldsymbol{r}_4:(000*,*)$；$\boldsymbol{r}_5:(1*,1*)$；$\boldsymbol{r}_6:(0111*,*)$；$\boldsymbol{r}_7:(001*,0*)$；$\boldsymbol{r}_8:(*,011*)$；$\boldsymbol{r}_9:(01110*,110*)$；$\boldsymbol{r}_{10}:(0001*,1*)$；$\boldsymbol{r}_{11}:(11*,0*)$。先根据规则前缀将所有规则映射到2维空间，并以斜线表示规则所映射的区域，一般情况下，规则之间存在重叠，图1中，以交叉线标注的区域表示规则有重叠的部分，如区域$\boldsymbol{r}_3 \wedge \boldsymbol{r}_5$。

根据Hicuts算法的决策树构建思路，在F_1维度上

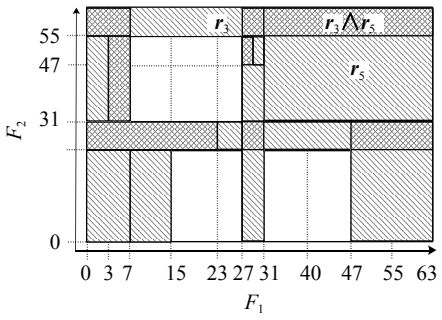


图 1 规则前缀空间表示

Fig. 1 Rules in a prefix plane

将整个2维空间平均划分为8个子空间。从根节点出发，可构建一棵包含8个叶子节点的一层决策树，如图2(a)所示，第1个子节点对应空间$F_1 \in [0, 7] \wedge F_2 \in [0, 63]$，其关联了4条规则$\boldsymbol{r}_3$、$\boldsymbol{r}_4$、$\boldsymbol{r}_8$和$\boldsymbol{r}_{10}$。如果要求各叶子节点所对应子空间内关联的规则条数不超过3条，则需要继续在F_2维度上对1维子空间$F_2 \in [0,63]$进行划分，此时，子空间$F_1 \in [0,7] \wedge F_2 \in [0,31]$关联两条规则$\boldsymbol{r}_4$和$\boldsymbol{r}_8$，而子空间$F_1 \in [0,7] \wedge F_2 \in [32,63]$关联3条规则$\boldsymbol{r}_3$、$\boldsymbol{r}_4$和$\boldsymbol{r}_{10}$。各子空间均满足所关联规则条数不大于3的要求。对应该划分过程，最后可构建决策树如图2(b)所示。要求每个子空间关联规则条数不超过3，即决策树叶子节点对应规则阈值τ等于3。

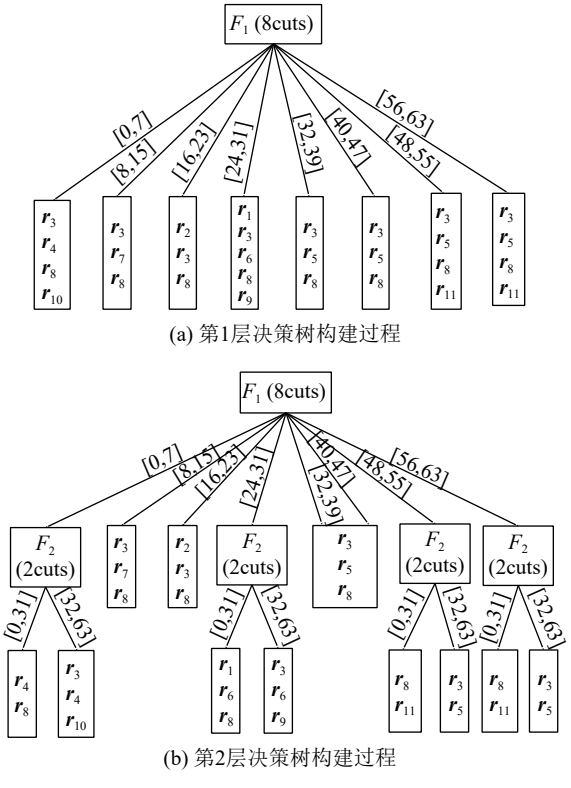


图 2 基于Hicuts算法构建分类决策树示例

Fig. 2 An example to construct the decision tree based on the Hicuts algorithm

注意到，阈值τ设置越大，则分类数据包时从根节点出发访问到叶子节点后需要执行顺序匹配的规则越多。但阈值τ也不能设置过小，因为τ设置得越小，需要对规则空间进行划分的次数越多，决策树规模越大，相应所需存储空间也会越大。

对现有基于决策树的包分类方法进行研究，发现当前方法存在3个主要的问题：1）匹配时间较长。一般规则之间会存在重叠，因此在对规则所在多维空间进行划分时，子空间内重叠的规则会关联到决策树的叶子节点，且相互重叠的规则越多，规则分组内的规则条数也越多，相应地执行顺序匹配所需的时间越长。2）所需存储空间大。以Hicuts为例，其采用平均划分方法对规则空间进行划分，使得一条规则可能被同时复制到多个叶子节点所分别对应的规则分组。例如图2(a)中，规则$\boldsymbol{r}_3$被复制到8个规则分组，即该规则需要存储8次，导致了存储空间的大量增加。3）一般基于决策树的划分方法要求叶子节点所关联规则条数阈值为τ，但如果有超过τ条规则同时映射到一个子空间，则按照现有划分方法，该子空间对应节点所关联的规则条数一定会大于τ，这意味着现有分类方法中阈值为τ的目标将会无法实现。如图1所示的规则集，规则$\boldsymbol{r}_3$、$\boldsymbol{r}_4$和$\boldsymbol{r}_{10}$同时映射到子空间$F_1 \in [4,7] \wedge F_2 \in [56,63]$，显然，无论如何划分，该子空间所关联的规则条数均不小于3。

2 基于单元空间划分的快速防火墙包分类算法

2.1 算法思想

针对现有基于决策树的包分类方法所存在的问题，提出了一种基于单元空间划分的快速防火墙包分类方法（Uscuts）。对问题1），首先，利用基于FDM(firewall design matrix)设计模型[14]的规则映射方法对原始规则进行预处理，使得目标规则与原始规则语义相同，但规则之间相互独立，不再有重叠（规则映射结果为一系列独立的单元空间，映射过程示例见第2.2.1节）。对问题2），不同于Hicuts等算法对规则空间进行平均划分，Uscuts方法采用基于单元空间边界的划分方式，可有效减少规则子空间数目，节省存储空间。对问题3），因采用Uscuts方法对规则进行预处理，使得目标规则之间不再有重叠，相应地基于目标规则所构建的分类决策树中，每条树支均唯一对应一个独立多维规则子空间，直观上看，即决策树的每个叶子节点所关联的规则条数等于1。

因此，在Uscuts包分类方法中，当数据包匹配到叶子节点时，可以直接判定数据包的分类决策为accept，不同于传统包分类方法需要在规则分组内继续执行

顺序匹配，该性质显著地提高了包分类速度。下面，结合一个2维访问控制规则实例对算法执行过程进行描述。

2.2 Uscuts算法步骤

算法过程包括以下3个步骤：1）对原始规则作预处理，通过规则映射方法将规则在多维矩阵空间进行映射，形成一系列独立的单元空间；2）根据所有单元空间在各个维度上的坐标投影区间，对规则所在的多维矩阵空间依次进行划分以构建分类决策树；3）基于分类决策树对数据包进行快速分类。算法的输入为原始k维防火墙规则$\boldsymbol{fr}$，经过规则预处理和决策树构建，可输出一棵k层分类决策树；第2.2.3节描述了根据分类决策树对数据包进行快速分类的过程。

2.2.1 规则预处理

按照基于FDM设计模型的规则映射思想，任一具有形如“$F_1 \in D(F_1) \wedge F_2 \in D(F_2) \wedge \cdots \wedge F_k \in D(F_k) \rightarrow decision$”的规则均可映射到$k$维矩阵空间$\boldsymbol{M}_k$。因此，对输入的防火墙规则$\boldsymbol{fr}$，可利用规则映射方法将各规则按逆序依次进行映射。在映射过程中，将最后决策为accept的区域用相互独立的单元空间$\boldsymbol{us}$（在k维矩阵空间中对应一个k维矩形）表示：$[(l_1,l_2,\cdots,l_k)(d_1,d_2,\cdots,d_k)]$，其中，$l_i$和$d_i$分别为区域在各维度上的最小边界值和范围。根据文献[14]所述，设规则条数为n，则规则预处理（对应FDM设计模型中的规则映射方法）的时间复杂度为$O(kn)$，其中，k为规则维数。图3（a）为一个2维的矩阵空间，两个维度的坐标值区间均为[0,9]。图3（b）描述了将规则“$\boldsymbol{r}$: $F_1 \in [1,7] \wedge F_2 \in [2,6] \rightarrow$accept”映射到2维矩阵空间$\boldsymbol{M}_2$后的结果，图3（b）中，方框区域“1, $\boldsymbol{r}$”表示规则$\boldsymbol{r}$的决策为accept。此时$\boldsymbol{M}_2$内包含一个单元空间，记为[(1,2)(7,5)]。

2.2.2 构建分类决策树

以2维情形为例，设原始访问控制策略(access control list, ACL)包含9条规则：

$$\boldsymbol{r}_2: F_1 \in [5,7] \wedge F_2 \in [3,4] \rightarrow \text{deny};$$

$$\boldsymbol{r}_3: F_1 \in [2,9] \wedge F_2 \in [7,8] \rightarrow \text{accept};$$

$$\boldsymbol{r}_1: F_1 \in [0,4] \wedge F_2 \in [4,6] \rightarrow \text{deny};$$

$$\boldsymbol{r}_4: F_1 \in [5,7] \wedge F_2 \in [3,9] \rightarrow \text{accept};$$

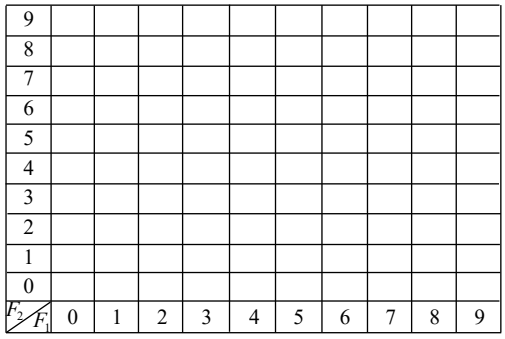
$$\boldsymbol{r}_5: F_1 \in [4,7] \wedge F_2 \in [0,2] \rightarrow \text{accept};$$

$$\boldsymbol{r}_6: F_1 \in [0,9] \wedge F_2 \in [0,1] \rightarrow \text{deny};$$

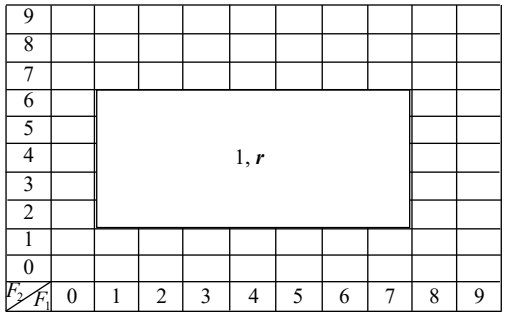
$$\boldsymbol{r}_7: F_1 \in [0,4] \wedge F_2 \in [9,9] \rightarrow \text{deny};$$

$$\boldsymbol{r}_8: F_1 \in [0,1] \wedge F_2 \in [0,9] \rightarrow \text{deny};$$

$$\boldsymbol{r}_9: F_1 \in [0,4] \wedge F_2 \in [2,9] \rightarrow \text{accept}。$$



(a) 2 维矩阵空间 $\boldsymbol{M}_2$

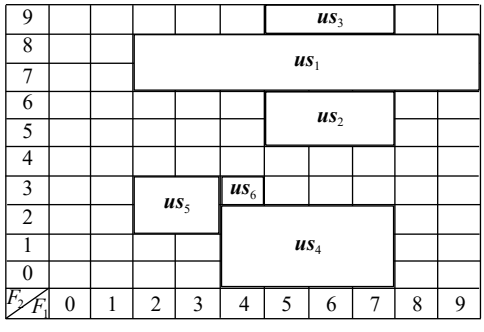


(b) 映射规则 $\boldsymbol{r}$ 后的 $\boldsymbol{M}_2$

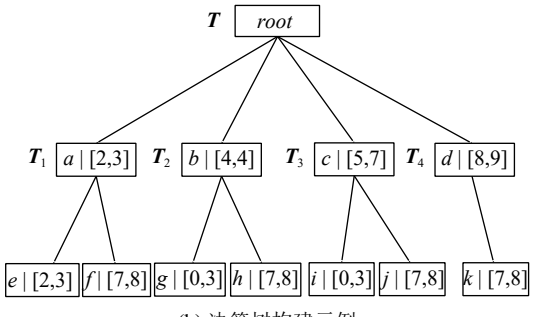
图 3　FDM规则映射示例$\boldsymbol{r}$: $F_1 \in [1,7] \wedge F_2 \in [2,6] \rightarrow$accept

Fig. 3　An mapping example of $\boldsymbol{r}$: $F_1 \in [1,7] \wedge F_2 \in [2,6] \rightarrow$ accept

以该ACL作为输入，通过映射操作可以得到最终映射结果如图4（a）所示。



(a) 规则映射结果



(b) 决策树构建示例

图 4　基于Uscuts算法构建分类决策树

Fig. 4　Constructing decision tree based on Uscuts

此时，6个单元空间分别为：$\boldsymbol{us}_1$: [(2,7)(8,2)]；$\boldsymbol{us}_2$: [(5,5)(3,2)]；$\boldsymbol{us}_3$: [(5,9)(3,1)]；$\boldsymbol{us}_4$: [(4,0)(4,3)]；

$\boldsymbol{us}_5$:[(2,2)(2,2)]; $\boldsymbol{us}_6$:[(4,3)(1,1)] 。根据单元空间$\boldsymbol{us}_1$~$\boldsymbol{us}_6$在各维度上的坐标投影区间,可对矩阵空间依次进行划分以构建分类决策树,如图4(b)所示。

一般而言,在某个给定维度F_i上,任意两个单元空间$\boldsymbol{u}$和$\boldsymbol{v}$的空间关系$R(\boldsymbol{u},\boldsymbol{v},F_i)$必定满足{相交,覆盖,包含,分离,相邻,重叠}6种关系之一,如图5所示。下面给出单元空间$\boldsymbol{u}$和$\boldsymbol{v}$在维度F_i上坐标投影区间$\boldsymbol{S}(\boldsymbol{u},\boldsymbol{v},F_i)$的定义。

定义 1 已知单元空间为$\boldsymbol{u}=(l_1^{(u)},l_2^{(u)}\cdots,l_k^{(u)})\cdot(d_1^{(u)},d_2^{(u)},\cdots,d_k^{(u)}),\boldsymbol{v}=(l_1^{(v)},l_2^{(v)},\cdots,l_k^{(v)})(d_1^{(v)},d_2^{(v)},\cdots,d_k^{(v)})$,如果:

1)$R(\boldsymbol{u},\boldsymbol{v},F_i)$=“相交”,则

$$\boldsymbol{S}(\boldsymbol{u},\boldsymbol{v},F_i)=\left[l_i^{(u)},l^{(v)}{}_i\right],\left[l_i^{(v)},l_i^{(u)}+d_i^{(u)}\right],\left[l_i^{(u)}+d_i^{(u)},l_i^{(v)}+d^{(v)}{}_i\right];$$

2)$R(\boldsymbol{u},\boldsymbol{v},F_i)$=“覆盖”,则

$$\boldsymbol{S}(\boldsymbol{u},\boldsymbol{v},F_i)=\left[l_i^{(u)},l_i^{(v)}\right],\left[l_i^{(v)},l_i^{(v)}+d_i^{(v)}\right],\left[l_i^{(v)}+d_i^{(v)},l_i^{(u)}+d_i^{(u)}\right];$$

3)$R(\boldsymbol{u},\boldsymbol{v},F_i)$=“包含”,则

$$\boldsymbol{S}(\boldsymbol{u},\boldsymbol{v},F_i)=\left[l_i^{(v)},l_i^{(u)}\right],\left[l_i^{(u)},l_i^{(u)}+d_i^{(u)}\right],\left[l_i^{(u)}+d_i^{(u)},l_i^{(v)}+d_i^{(v)}\right];$$

4)$R(\boldsymbol{u},\boldsymbol{v},F_i)$=“分离”,则

$$\boldsymbol{S}(\boldsymbol{u},\boldsymbol{v},F_i)=\left[l_i^{(u)},l_i^{(u)}+d_i^{(u)}\right],\left[l_i^{(v)},l_i^{(v)}+d_i^{(v)}\right];$$

5)$R(\boldsymbol{u},\boldsymbol{v},F_i)$=“相邻”,则

$$\boldsymbol{S}(\boldsymbol{u},\boldsymbol{v},F_i)=\left[l_i^{(u)},l_i^{(u)}+d_i^{(u)}\right],\left[l_i^{(v)},l_i^{(v)}+d_i^{(v)}\right];$$

6)$R(\boldsymbol{u},\boldsymbol{v},F_i)$=“重叠”,则$\boldsymbol{S}(\boldsymbol{u},\boldsymbol{v},F_i)=[l_i^{(u)},l_i^{(u)}+d_i^{(u)}]$。

以2维情形为例,两个单元空间$\boldsymbol{u}$和$\boldsymbol{v}$在维度F_1上的空间关系和坐标投影区间如图5所示。

基于定义1,下面简要描述构建分类决策树$\boldsymbol{T}$过程的步骤:

设k维矩阵空间包含N个单元空间$\boldsymbol{us}_i(i=1,2,\cdots,N)$,初始情况下$\boldsymbol{T}$仅包含根节点*root*。

1)求N个单元空间在F_t维度(初始时t=1)上的坐标投影区间$\boldsymbol{S}(\boldsymbol{us}_1,\boldsymbol{us}_2,\cdots,\boldsymbol{us}_N,F_t)$,记投影区间个数为$n$。

2)将n个坐标投影区间作为*root*的子节点$node_i(i=1,2,\cdots,n)$依次添加到决策树$\boldsymbol{T}$,每个子节点分别构成$\boldsymbol{T}$的一棵子树$\boldsymbol{T}_i(i=1,2,\cdots,n)$,而各节点$node_i$分别为对应子树$\boldsymbol{T}_i$的根节点。

3)维度t++。

4)分别求各子树$\boldsymbol{T}_i(i=1,2,\cdots,n)$的根节点对应区间所关联的全部单元空间(记为{$\boldsymbol{us}$}),及其在维度$F_t$上的坐标投影区间$\boldsymbol{S}(\{\boldsymbol{us}\},F_t)$,记投影区间个数为$n_i$。例如图4中节点$a$对应区间[2,3],其关联的全部单元空间{$\boldsymbol{us}$}为$\boldsymbol{us}_1$和$\boldsymbol{us}_5$。

5)将全部n_i个投影区间$\boldsymbol{S}(\{\boldsymbol{us}\},F_t)$作为$node_i$的子节点依次添加到$\boldsymbol{T}_i$。同理,每个子节点分别构成$\boldsymbol{T}_i$的一棵子树,而各子节点则成为对应子树的根节点。

6)迭代执行步骤3)到步骤5),当$t>k$时决策树构建过程结束,返回决策树$\boldsymbol{T}$。

以图4为例,初始情况下$\boldsymbol{T}$为一层决策树,仅包含根节点*root*。2维矩阵空间内6个单元空间($\boldsymbol{us}_1$~$\boldsymbol{us}_6$)在F_1维度上形成4个坐标投影区间为{[2,3], [4,4], [5,7], [8,9]}。依次将各投影区间作为*root*的子节点{a,b,c,d}添加到决策树$\boldsymbol{T}$,每个子节点分别构成$\boldsymbol{T}$的一棵子树$\boldsymbol{T}_i$(i=1,2,3,4)。

子树$\boldsymbol{T}_1$的根节点a对应区间[2,3],其所关联的全部单元空间$\boldsymbol{us}_1$和$\boldsymbol{us}_5$在F_2维度上形成两个坐标投影区间为[2,3]和[7,8]。分别将两个投影区间作为根节点a的子节点{e,f}添加到子树$\boldsymbol{T}_1$。同理,将投影区间[0,3]、[7,8]作为子节点{g,h}添加到子树$\boldsymbol{T}_2$,投影区间[0,2]、[5,9]作为子节点{i,j}添加到子树$\boldsymbol{T}_3$,投影区间[7, 8]作为子节点{k}添加到子树$\boldsymbol{T}_4$,最后形成决策树$\boldsymbol{T}$如图4(b)所示。

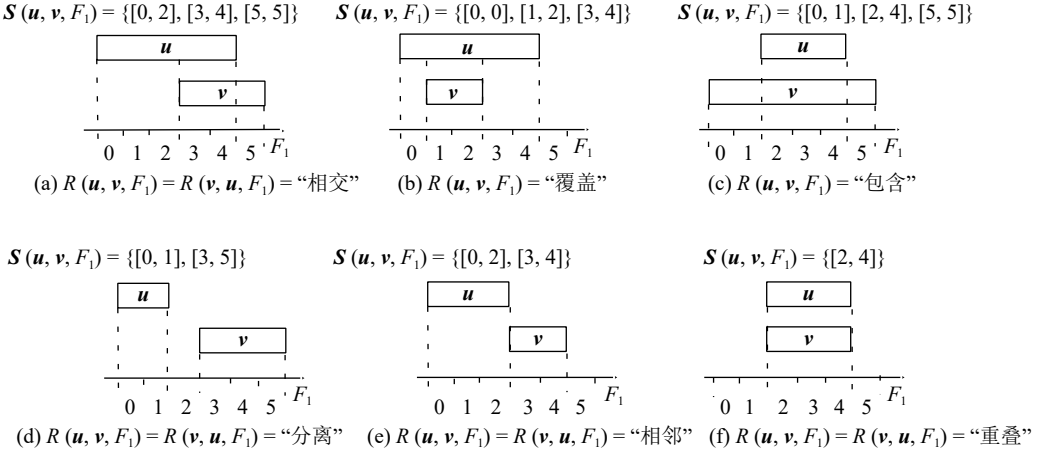


图 5 单元空间u和v在维度F_1上的空间关系和坐标投影区间

Fig. 5 Spatial relations and the coordinate projection intervals of two unit spaces u and v in the dimension F_1

2.2.3　根据决策树对数据包进行快速分类

在基于决策树的包分类方法中，包分类本质上是一个查询操作。对决策树或其任意一棵子树，其根节点的子节点所对应区间坐标值均为严格递增，故在查找时可直接应用折半查找法。如图6所示，决策树$\boldsymbol{T}$的根节点*root*有4个子节点，所对应区间坐标值分别为[2,3]、[4,4]、[5,7]和[8,9]，满足严格递增关系。

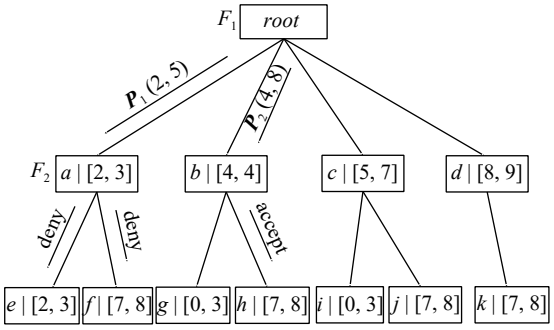


图 6　基于分类决策树对数据包进行快速分类

Fig. 6　Fast packet classification based on decision tree

考虑一个k元组数据包$\boldsymbol{P}(x_1,x_2,\cdots,x_k)$，对其分类时从决策树的根节点*root*开始，首先在根节点的所有子节点上进行折半查找，判断数据包的第一元数据x_1是否包含于某节点对应区间。如果不包含，则可判定数据包$\boldsymbol{P}$不能匹配任一节点，记该数据包决策为deny；否则，在以该节点为根节点的子树上继续查找。如果数据包$\boldsymbol{P}$的各元组$x_i(i=1,2,\cdots,k)$分别包含于决策树某条树支的各层节点所对应区间，则可判定该数据包$\boldsymbol{P}$能匹配决策树，记$\boldsymbol{P}$的决策为accept。

如图6所示，设有两个待分类数据包分别为$\boldsymbol{P}_1$=(2,5)，$\boldsymbol{P}_2$=(4,8)。首先，判定数据包$\boldsymbol{P}_1$(2,5)，已知x_1=2、x_2=5。从根节点*root*出发，x_1=2匹配第一个节点*a*的对应区间[2,3]，因此继续在以*a*为根节点的子树上进行查找，因x_2=5不能匹配子树根节点*a*的任一子节点所对应区间[2,3]或[7,8]，故可判定$\boldsymbol{P}_1$的分类决策为deny。

对数据包$\boldsymbol{P}_2$(4,8)，已知x_1=4、x_2=8，从根节点*root*出发，在其子节点上折半查找，可快速判定x_1匹配第2个子节点*b*的坐标区间[4,4]；继续在以*b*为根节点的子树上进行查找，x_2=8，包含于*b*的第2个子节点所对应区间[7,8]，因此，可判定数据包$\boldsymbol{P}_2$的分类决策为accept。数据包分类的具体过程如算法1所述。

算法1　数据包分类算法

Input：数据包$\boldsymbol{P}(x_1,x_2,\cdots,x_k)$，决策树$\boldsymbol{T}$（根节点*root*）

Output：数据包决策（‘accept’ or ‘deny’）

Begin

1. for (i=1 to k) do{

/* 在根节点*root*的子节点中折半查找$\boldsymbol{P}$的第i元数据x_i，如果x_i包含于*root*的第t个子节点*Child*(*root*, t)所对应区间，则继续在以*Child*(*root*, t)为根节点的子树上进行查找x_{i+1}*/

2. if (*Binarysearch* (*root*,x_i)==true)do{
3. *root* = *Child*(*root*, t);
4. continue;
5. } else $\boldsymbol{P}$→deny ;
6. break;
7. }
8. if (i==k+1)
9. $\boldsymbol{P}$→accept

End

从算法1执行过程可知，数据包分类速度取决于在决策树中查找数据包各元组的效率，下面具体分析Uscuts方法的查找效率。

3　理论分析与仿真结果

3.1　时间复杂度分析

Uscut方法运行过程包括线下和线上两个阶段，线下计算包括规则预处理和构建分类决策树两个步骤。线上计算为根据决策树对数据包进行快速分类。从实际需要出发，重点关注Uscuts方法的包分类时间。下面分别从最坏情况和平均情况两个方面对包分类算法的时间复杂度进行定量分析。

1）最坏情况下时间复杂度

根据Uscuts方法所构建的决策树，其各叶子节点所关联的规则条数等于1。因此，在数据包分类过程中，当访问到决策树的叶子节点时，不需要像传统基于决策树的包分类方法一样继续在规则分组中执行顺序查找，而是可以直接确定数据包的分类结果。同时，对决策树或其任意一棵子树，其根节点的子节点所对应区间坐标值均为严格递增，所以可采用折半查找法。

设原始规则条数为n，根据文献[14]分析，使用规则映射方法对原始规则进行映射后所得到的单元空间个数会小于n。而n个单元空间在任一维度上的投影区间个数不大于$2n-1$。如图5所示，任意两个单元空间在F_1维度上的投影区间不会超过3个。按决策树的构建过程可知，决策树或其任意一棵子树根节点的子节点数均不会超过$2n-1$个，对应最坏情况下的查找时间为lb($2n-1$)，其时间复杂度为O(lb(n))。因此，在k层决策树上进行查找，最坏情况下的时间复杂度为$O(T_{\text{worst}})=O(k\cdot\text{lb}(n))$。

表1列出了Uscuts方法和几种经典包分类方法在最坏情况下的时间复杂度。表1中：n和k分别指规则

条数和维数；w为规则域长；τ为叶子节点所关联的规则条数阈值；$RuleSize$为单条规则所占的字节数；C为CacheLine的字节数，其大小一般为16～256 Byte不等，目前主流CacheLine为64 Byte。

表 1 最坏情况下算法时间复杂度对比

Tab. 1 Comparison of the worst time complexity

算法	$O(T_{worst})$
TSS	$O(n^k)$
Linear search	$O(n)$
Grid-of-Tries	$O(2w)$
Hicuts	$O(w+\tau\times Rulesize/C)$
Uscuts	$O(k\cdot \mathrm{lb}\, n)$

由表1可以看到，相比Grid-of-Tries[11]、HiCuts等算法而言，Uscuts分类算法的执行时间有比较明显的优势。

2）平均时间复杂度分析

为进一步说明Uscuts分类算法性能，接下来，对该算法在一般情况下的平均时间复杂度进行分析。

因为实际应用中数据包和规则均是动态变化的，考虑一般情况，首先提出两点假设并分别进行实验验证。已知原始访问控制规则条数为n，假设：1）待分类的m个数据包约有50%分类为deny；2）分类为deny的数据包分别有$1/k$左右的概率在$F_i(i=1,2,\cdots,k)$维度上找不到匹配的节点。实验中，用Classbench[15]自动生成了10个不同规模（规则条数分别为9～1 000条）的ACL规则。

针对假设1），待分类的m个数据包约有一半分类为accept或deny，选取一个规则条数为200的ACL，并随机生成10个数据集（大小为1～100 kByte）进行分类测试，测试结果如表2所示。

表 2 数据包分类验证结果

Tab. 2 Results of packet classification

数据包大小/kByte	分类率/%	
	accept	deny
1	33.02	66.98
2	37.62	62.38
5	37.52	62.48
10	38.76	61.24
12	37.14	62.86
15	36.94	63.06
18	41.58	58.42
20	37.95	62.05
22	37.17	62.83
25	37.58	62.42
平均	37.53	62.47

由表2可以看到，各数据集平均有60%左右分类为deny，基本符合假设。

针对假设2），分类为deny的数据包分别有$1/k$左右的概率在$F_i(i=1,2,\cdots,k)$维度上找不到匹配的节点区间，对表1中分类为deny的数据包进行分析，分别列出各维度上未能找到相匹配节点区间的数据包占比，如表3所示。由表3可以看到，2维情况下（k=2）分类为deny的数据包分别有大约1/2的概率在各维度上不能匹配对应的节点，假设成立。

表 3 分类为deny的数据包在各维上未能匹配到节点数占比

Tab. 3 Proportion of the deny packets that failed to match in each dimension

数据包大小/kByte	F_1/%	F_2/%
1	44.12	55.88
2	44.80	55.20
5	46.71	53.29
10	49.49	50.51
12	49.38	50.62
15	51.78	48.22
18	44.37	55.63
20	44.78	55.22
22	52.19	47.81
25	48.41	51.59
平均	48.60	51.40

设待分类数据包数为m，则约有$m/2$个数据包分类决策为accept，该$m/2$个数据包在各维度上分别需要至多$\mathrm{lb}(2n)$的访问时间，因此总计需要时间为$(m/2)\cdot[k\cdot\mathrm{lb}(2n)]$；而另外$m/2$个决策为deny的数据包，各维度上分别有$1/k$的概率找不到匹配的坐标区间，故在第$i(i=1,2,\cdots,k)$维度上找不到匹配的坐标区间时，其所耗时间为$(m/2k)\cdot i\cdot\mathrm{lb}(2n)$。因此，平均情况下查找时间为：

$$T_{avg}=[(mk/2)\cdot\mathrm{lb}(2n)+(m/2k)\sum_{i=1-k}\mathrm{lb}(2n)]/m=[(3k+1)/4]\cdot\mathrm{lb}(2n) \quad (1)$$

即在k层决策树上进行查找，平均情况下的时间复杂度为$O(T_{avg})=O(0.75k\mathrm{lb}(n))$。令$c=0.75k$，有$O(T_{avg})=O(c\cdot\mathrm{lb}(n))$。对一般2维ACL，$k=2$，有$O(T_{avg})=O(1.5\mathrm{lb}(n))$；对一般5维防火墙规则，$k=5$，此时有$O(T_{avg})=O(3.75\mathrm{lb}(n))$。

3.2 空间复杂度分析

在算法实现过程中，采用链表结构对决策树内各个节点进行存储。具体而言，为树中每个节点设置一个孩子链表，并将这些节点及相应的孩子链表的头指针存放在一个向量中，其中，各链表元素分别对应决策树的节点。设决策树内节点数为N_d，则采用链

表结构存储对应决策树时，需要存储N_d个指针信息和$2\cdot N_d$个节点信息。在32位机器上，指针大小为4 Byte；而每个节点均由一个地址区间表示，最大对应2个32 bit地址存储。因此，对应Uscuts方法所需存储空间不大于$N_d\cdot 20/2^{20}$ MB。

一般情况下，对基于决策树的分类方法而言，除最后一层的叶子节点外，其他各层节点的存储结构均相同。但注意到，传统基于决策树的分类方法不仅要存储节点信息，而且要存储叶子节点所对应规则分组内的所有规则。设N_s为决策树的叶子节点所对应的规则数之和，而以区间形式表示的单条防火墙规则对应28 Byte的存储空间，因此对应总规则存储空间为$N_s\cdot 28/2^{20}$ MB。

显然，与现有其他基于决策树的分类方法相比，Uscuts方法中决策树内每个叶子节点所关联的规则条数有且只有1条。因此，该方法不需要额外空间存储分组规则，在一定程度上降低了存储空间需求。

3.3　仿真实验

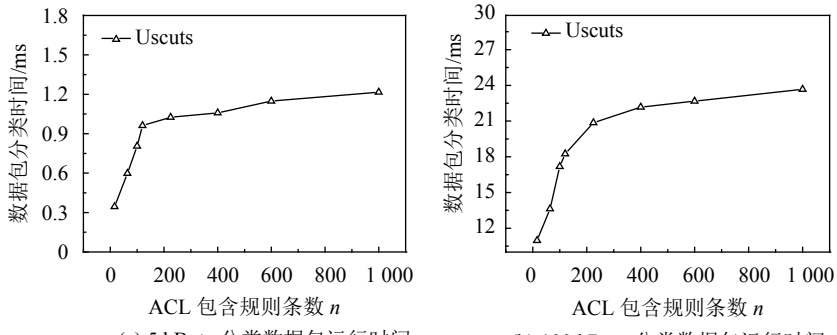
为进一步验证Uscuts分类算法效率，选取8个不同规模（规则条数分别为16～1 000条）的ACL规则，及9个不同大小（分别为1～1 000 kByte）的数据集，测试了用UsCuts算法分类数据包所需时间，算法执行环境为Java JDK1.6（操作系统Windows 7，内存4.0 GB，CPU为Intel(R)Core(TM)Processor of 2.60 GHz）。测试结果见表4。对规模为1 kByte的ACL，当数据集达1 000 kByte时，分类这些数据包所需时间也不超过220 ms。

表 4　不同规模ACL(n)下的数据包分类时间

Tab. 4　Packets classification time on different sizes of ACLs(n)

数据包大小/ kByte	数据包分类时间/ms							
	n=16	n=64	n=100	n=120	n=225	n=400	n=600	n=1 000
1	0.345	0.598	0.805	0.962	1.025	1.059	1.148	1.216
5	1.418	2.268	2.405	2.736	2.927	3.114	3.227	3.515
10	1.334	3.273	3.638	3.985	4.528	4.578	5.236	5.617
15	3.582	4.417	6.352	7.065	7.359	7.642	8.648	9.768
20	4.741	7.448	9.105	9.442	9.802	10.228	12.035	13.597
100	10.968	13.632	17.183	18.245	20.865	22.178	22.67	23.669
200	14.729	22.325	27.474	30.96	34.015	41.409	43.926	45.298
500	41.673	52.613	69.946	83.092	92.815	100.88	107.236	114.313
1 000	66.756	79.032	112.532	152.328	165.671	180.333	206.05	211.841

以ACL条数n为横坐标，数据包分类时间为纵坐标，选取两个不同大小数据包（这里分别选取5和100 kByte）在不同规模ACL下的分类速度测试结果，并在坐标图中加以映射，如图7所示。



(a) 5 kByte 分类数据包运行时间　　(b) 100 kByte 分类数据包运行时间

图 7　Uscuts分类算法速度测试结果

Fig. 7　Running time for Uscuts

由图7可以看到：分类数据包所需时间曲线分布均近似符合对数函数$c\cdot\mathrm{lb}(n)$，其中，c为常量，且随数据包大小近似呈线性变化。如数据包为5 kByte时，c=0.125；当数据包为100 kByte时，c=2.5。测试结果也进一步验证了用Uscuts方法对数据包进行分类，其时间复杂度符合$O(c\cdot\mathrm{lb}(n))$。

4　结　论

随着网络应用的发展，对网络数据包分类速度提出了更高的要求。针对现有包分类方法存在的不足，提出了一种基于单元空间划分的快速防火墙包分类方法（Uscuts）。首先，对规则进行预处理生成一

系列单元空间；然后，根据单元空间划分构建分类决策树；最后，基于决策树实现数据包快速分类。理论分析与测试结果表明，与现有基于决策树的分类方法相比，本文方法具有更高的分类效率，且针对决策树所需的存储空间更小。本文方法的应用不仅局限于防火墙包分类，也为路由器、SDN快速包分类等问题求解提供了一种新的思路。下一步工作将研究分布式防火墙包分类问题及如何将本文方法应用到SDN之中。

参考文献:

[1] Daly J,Liu A X,Torng E.A difference resolution approach to compressing access control lists[J].IEEE/ACM Transactions on Networking,2016,24(1):610–623.

[2] Ding linxuan,Huang Kun,Zhang Dafang.Low-power TCAM for regular experssion matching[J].Journal on Communications,2014,35(8):162–168.[丁麟轩,黄昆,张大方.基于 TCAM 的低能耗正则表达式匹配算法[J].通信学报,2014,35(8):162–168.]

[3] Qi Yaxuan,Li Jun.Theoretical analysis and algorithm design of high-performance packet classification algorithms[J].Chinese Journal of Computers,2013,36(2):408–421.[亓亚烜,李军.高性能网包分类理论与算法综述[J].计算机学报,2013,36(2):408–421.]

[4] Rottenstreich O,Keslassy I,Hassidim A,et al.On finding an optimal TCAM encoding scheme for packet classification[C]//Proceedings of the 2013 IEEE Infocom.Turin:IEEE,2013:2049–2057.

[5] Srinivasan V,Varghese G,Suri S,et al.Fast and scalable layer four switching[C]//Proceedings of the ACM SIGCOMM'98 Conference on Applications,Technologies,Architectures,And Protocols for Computer Communication.Vancouver:ACM,1998:191–202.

[6] Srinivasan V,Suri S,Varghese G.Packet classification using tuple space search[J].ACM SIGCOMM Computer Communication Review,1999,29(4):135–146.

[7] Lu H,Sahni S.*O*(log *W*) multidimensional packet classification[J].IEEE/ACM Transactions on Networking ,2007,15(2):462–472.

[8] Gupta P,McKeown N.Classifying packets with hierarchical intelligent cuttings[J].IEEE Micro,2000,20(1):34–41.

[9] Singh S,Baboescu F,Varghese G,et al.Packet classification using multidimensional cutting[C]//Proceedings of the 2003 Conference on Applications,Technologies,Architectures,And Protocols for Computer Communications.Karlsruhe:ACM,2003:213–224.

[10] Gupta P,McKeown N.Algorithms for packet classification[J].IEEE Network,2001,15(2):24–32.

[11] Lu Sheng,Gong Jian.A multi-dimension packet classification algorithm:NONintersection trie[J].Chinese Journal of Computers,2003,26(11):1502–1509.[陆晟,龚俭.一种新的高维报文分类算法——无相交树算法[J].计算机学报,2003,26(11):1502–1509.]

[12] Baboescu F,Singh S,Varghese G.Packet classification for core routers:Is there an alternative to CAMs?[C]//Proceedings of Twenty-second Annual Joint Conference of the IEEE Computer and Communications.San Francisco:IEEE,2003:53–63.

[13] Spitznagel E,Taylor D,Turner J.Packet classification using extended TCAMs[C]//Proceedings of 11th IEEE International Conference on Network Protocols.Atlanta:IEEE,2003:120–131.

[14] Cheng Y,Wang W,Min G,et al.A new approach to designing firewall based on multidimensional matrix[J].Concurrency and Computation:Practice and Experience,2015,27(12):3075–3088.

[15] Taylor D E,Turner J S.Classbench:A packet classification benchmark[J].IEEE/ACM Transactions on Networking,2007,15(3):499–511.

（编辑　赵　婧）

引用格式: Cheng Yuzhu,Wang Weiping,Wang Jianxin.A fast firewall packet classification algorithm using unit space partitions[J].Advanced Engineering Sciences,2018,50(4):144–152.[程玉柱,王伟平,王建新.一种基于单元空间划分的快速防火墙包分类算法[J].工程科学与技术,2018,50(4):144–152.]