第36卷 第19期 计 算 机 工 程 2010年10月
Vol.36 No.19 Computer Engineering October 2010
·软件技术与数据库· 文章编号：1000—3428(2010)19—0099—03 文献标识码：A 中图分类号：TP393.01

基于多域的计算几何流分类改进算法

刘 杉，侯整风
(合肥工业大学计算机与信息学院，合肥 230009)

摘 要：为使包分类具有快速点定位和良好的可扩展性，结合 cross-producting 表与线性查找提出一种新的基于计算几何的流分类算法。该算法通过控制规则的数目调整存储使用情况，使数据包中越来越多的规则被一维数据结构搜索到，进一步降低算法中 cross-producting 表需要的存储量。实验结果表明，该算法不仅改进了 cross-producting 的存储性能，而且能提高时间性能。

关键词：流分类；计算几何；前缀树

Improved Flow Classification Algorithm of Computational Geometry Based on Multiple Fields

LIU Shan, HOU Zheng-feng
(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

【Abstract】To make the packet classification both fast point location and scalable, this paper presents a new efficient packet classification algorithm of computational geometry. It combines cross-producting with linear search. The proposed algorithm can adjust storage usage by controlling the number of filters through one-dimensional searches, with more filters searched by one-dimensional data structure, the storage needed for the cross-producting table can be further decreased. Experimental result shows that the algorithm not only improves storage performance, but also increases time performance.

【Key words】 flow classification; computational geometry; prefix tree

1 概述

流分类的问题类似于多维空间的点选址定位问题[1]，该问题寻求一个点所在集合里的非重叠范围内的围绕区域。流分类问题可以简单地描述为：根据事先定义的一定的策略和规则，对报文报头的一个或多个域进行分析，以识别该报文所属的流。现有的流分类算法主要分为 4 类：基本数据结构算法，基于计算几何的算法，启发式算法和只能硬件实现的算法。其中，基于计算几何流分类算法(geometry-based)的思想是受分类器中的每个域表示为前缀/长度对，运算符/数字形式的启发。从几何的观点看，每一类都能用数据轴上的一个范围(或区间)表示。二维的规则表示了 2-D 欧几里得空间的一个矩形，d 维的规则表示了 d 维的超矩形。一个分类器就是与优先级相联系的超矩形集合。对于一个给定的报头 P，它表示 d 维空间的一个点，而流分类问题就是发现包含点 P 的优先级最高的那个超矩形。

文献[2]提出 cross-producting 算法，它属于计算几何流分类的范畴，此算法适合于任意个数的域和任意形式的域，主要思路是将多维的流分类问题建立在多个一维的流分类基础上，利用多个一维的流分类的结果查找 cross-producting 表获得最终的流分类结果。

2 相关工作

2.1 独立集算法

独立集思想是根据一个特定的域来对多维的规则进行分类。每个独立集里设置的规则是相互独立的[3]。因此，该算法通过查找求出每个独立集里的匹配规则。每个独立集里准确地存储各自的规则，从而减少内存消耗。由于独立集执行搜索的过程是一维搜索，这和 cross-producting 算法执行的一维搜索匹配，因此本文采用独立集算法来搜索最佳匹配前缀。

2.2 基于决策树的算法

以决策树为基础的算法是把规则用决策树分成多个组。每一个组对应于一个决策树的叶子节点，以线性查找遍历整个组[4]。每个组的规则数量由一个预定义的值限定，节点的裁剪规则可能是任意域的一个值或位。一套优化的裁剪规则能够最小化需要的存储量和搜索时间[6]。考虑到决策树关于前缀的数目和长度的可测性，本算法同时采取了多路查找树进行一维查找。

2.3 基于哈希表的解决方案

在以往的研究[5]中，基于哈希表的思想是在多维规则中提出的。一个哈希表存储同一前缀长度的规则组合，并且产生一个哈希密钥关联这些前缀。例如，二维规则 F=(10*,110*)和 G=(11*,001*)两者都属于元组 $T_{2,3}$。当搜索 $T_{2,3}$ 时，算法通过连接 2 个源地址域的比特位和 3 个目的地址域的比特位产生一个哈希密钥。可以采取哈希表数据结构来进行一维查找搜索最佳匹配前缀。

3 基于多域的 cross-producting 改进算法

3.1 改进算法的思想

本文提出了一个新的流分类算法，改进 cross-producting

基金项目：广东省教育部产学研结合基金资助项目(2008B090500240)；安徽省自然科学基金资助项目(090412051)

作者简介：刘 杉(1983－)，女，硕士研究生，主研方向：计算几何，网络信息安全；侯整风，教授

收稿日期：2010-02-20 **E-mail：**roamerls@126.com

算法通过利用线性查找来处理规则库规模扩大的问题。由于各个域内的不同前缀数的增加会引起一维搜索的搜索性能的下降[6],并且 cross-producting 表的消耗存储呈指数倍数增长，因此在 cross-producting 表找到匹配前缀前，通过一维搜索来查找每个域的最佳匹配前缀，利用域的规格区分出更多的规则，然后通过一维搜索查找到这些规则，再将它们从 cross-producting 表中移除。因此，总体的存储性能随数量级改变得到提高。

表 1 使用包含 10 条规则和 5 个域的一个规则库。表 2 列出了每个域的不同的规格。在共有 480 种不同的组合中，表明每条规则平均有 48 个组合。表 3 是一个生成的 cross-producting 表。在最坏的情况下，每条规则都有一个单独的域规范，因此，N 条规则会产生最多 N^d 个组合。

表 1　5 个域 10 条规则的规则库

规则	f_1	f_2	f_3	f_4	f_5	动作
F_1	000*	111*	[10:10]	*	UDP	Act_0
F_2	000*	111*	[01:01]	[10:10]	UDP	Act_0
F_3	000*	10*	*	[10:10]	TCP	Act_1
F_4	000*	10*	[00:10]	[01:01]	TCP	Act_2
F_5	000*	10*	[10:10]	[11:11]	TCP	Act_1
F_6	0*	111*	[10:10]	[01:01]	UDP	Act_0
F_7	0*	111*	[10:10]	[10:10]	UDP	Act_0
F_8	0*	1*	*	*	TCP	Act_2
F_9	*	01*	[00:10]	*	TCP	Act_2
F_{10}	*	0*	*	[01:01]	UDP	Act_0

表 2　每个域中不同的规格项

域	规格项
f_1	000*,　0*, *
f_2	111*, 10*, 1*, 01*, 0*
f_3	[10:10], [01:01], [00:10], *
f_4	*, [10:10], [01:01], [11:11]
f_5	UDP, TCP

表 3　表 1 中规则对应的 cross-product

索引	Crossproduct 项	匹配规则
1	000*,111*,[10:10],*,UDP	F_0
2	000*,111*,[10:10],*,TCP	F_7
3	000*,111*,[10:10],[10:10],UDP	F_0
4	000*,111*,[10:10],[10:10],TCP	F_7
5	000*,111*,[10:10],[01:01],UDP	F_0
6	000*,111*,[10:10],[01:01],TCP	F_7
…	…	…
479	*,0*,*,[11:11],UDP	F_9
480	*,0*,*,[11:11],TCP	F_8

本文将 cross-producting 算法和独立集思想结合起来，并利用独立集的思想优化 cross-producting 算法，其设计思路是合并 2 个算法把规则划分为 2 个组：(1)包括用一维搜索来查找的规则；(2)用 cross-producting 表搜索的规则。第(1)个组的挑选过程包括使用域规范。根据查找没有被其他规则中指定域规范的规则，把这些规则归类到一维搜索组中。剩余的规则用 cross-product 来进行搜索。重复迭代这一过程以实现进一步的缩减，直到规格条目减少到本文所预定义的某个值为止。因此，改进算法可以调整存储需求来适应存储空间。

3.2　改进算法数据结构

改进算法构建数据结构如下：

第 1 步 把规则中的范围转换为前缀。在最早的 cross-producting 算法中，范围必须划分为简单的相互分离的子范围。这样，插入新的范围可能会导致新的范围数的增长，为支持增量更新，通过拆分每个范围成多个子范围来把范围转换成为前缀，从而每个子范围独立地对应于一个前缀。在最坏的情况下，$2(W\text{-}1)$个数目的前缀需要覆盖全部范围，其中，W 是一个范围域的长度。因此，有 2 个范围域的规则可以转换成为最多 $4(W\text{-}1)^2$ 完全前缀域的规则。不同的域规格可能也会增加。

第 2 步 算法在每个基于新规则的域规格的域中构造一棵前缀查找树，树的节点称作“前缀节点”，对应于规则中域的规格。每个前缀节点设置一张随后即将执行的规则列表，该表插入在前缀节点的规则列表中，表中域规格与前缀节点相对应。一条规则从规则列表中移除，进入一维搜索组中。挑选规则的标准是在一维数据结构中用叶子节点极大数的前缀树查找到的那部分规则。当规则的前缀节点列表长度减少到 0 时，移除与前缀节点对应的域规格以减少 cross-producting 组合的数目。所有的前缀树在剩余规则域规格的基础上重新计算。该算法通过剩余 cross-producting 的项数来决定需要的存储量是否在一个预定义的阈值之内，如果不在预定的阈值之内，上述步骤包括用最大叶子节点数来选择前缀树和挑选另一组规则，在规则分类过程完成前重复进行。由于每次迭代中选中的规则至少在一个域内是独立的，因此这些规则属于某个独立集，而且一维搜索被看作一套独立集。由于每个独立集将导致至多一条额外规则，因此可以设想算法的目标是用空规则列表最大化前缀节点的数目，同时，最小化一维搜索组中的独立集数目。

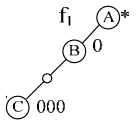
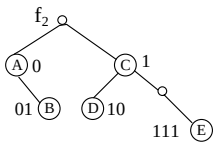
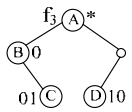
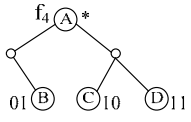
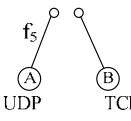
第 3 步 搜索最佳匹配前缀。这个搜索过程由 2 个部分组成：一维搜索和一个到 cross-producting 表的访问。注意到前缀树数据结构可以由哈希表或多路查找树替代来搜索最佳匹配前缀。在本文中，搜索最佳匹配前缀采用了多路查找树的方法，因为规则共用它们的域值。拟定算法的时间复杂度为 $O(d \text{ lb } N+I)$，其中，I 是独立集的数目。

3.3　改进算法的过程

用表 1 中的例子来说明：每条规则包括 2 个范围的域：f_3 和 f_4，它们必须被转换成前缀的格式。范围[10: 10]转换为前缀<10*>,而范围[00: 10]被转换成 2 个前缀：<0*>和<10*>。一旦一个范围转换成为更多的前缀，规则也同样重复进行，最初的规则库在作了转变后转换为 12 个规则。

阐述规则挑选过程，表 4 是 5 个域的前缀树。

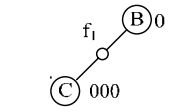
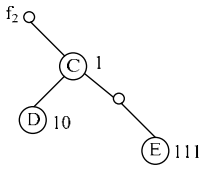
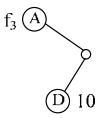
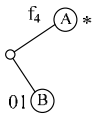
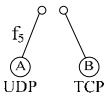
表 4　规则前缀尝试

图例	前缀树
	f_1 前缀：*, 0, 000; A(3):F_j,F_k,F_l B(3):F_g,F_h,F_i C(6):F_a~F_f
	f_2 前缀：0, 01, 1, 10, 111; A(1): F_l B(2):F_j,F_k C(1):F_i D(4):F_c~F_f E(4):F_a,F_b,F_g,F_h
	f_3 前缀：*，0，1，，01，10； A(3):F_c,F_i,F_l B(2):F_d,F_j C(1):F_b D(6):F_a,F_e~F_h,F_k
	f_4 前缀：*，01，10，11； A(4):F_a,F_i~F_k B(4):F_d,F_e,F_g,F_l C(3):F_b,F_c,F_h D(1):F_f
	f_5 前缀：UDP, TCP； A(5):F_a,F_b,F_g,F_h,F_l; B(7):F_c~F_f,F_i~F_k

对一棵构造好的二叉树，每个前缀节点都有一个唯一的标识符来简化随后的说明，列出了每个前缀节点的相关联规则，小括弧里的数目表示相关联的规则数目。首先用最大的叶子节点数来选择一棵前缀树。这一步可以选择前缀树 f_2 或 f_4，随机选择前者 f_2，再选择前缀树 f_2 每个叶子节点里的规则作为待选规则。这些待选规则依次放入到相应的每个叶子节点的独立集中。在前缀树 f_2 的节点 C 中有 2 条待选规则，规则 F_j 和 F_k。对规则 F_j，它的第 3 个域规格相对应于前缀树 f_3 的节点 B，节点 B 与 2 条规则关联。由于规则 F_k 的相应节点关联到至少 5 条规则，因此选择规则 F_j 并从所有相关联的前缀节点中移除，包括前缀树 f_1 的节点 A，前缀树 f_2 的节点 C 等。

同样在节点 E 中选中规则 F_f，还有在节点 F 中选中规则 F_b。当这些选中的规则和它们相关联的节点分离之后，前缀树 f_3 和 f_4 中都减少了一个特定的域规格，f_3 中节点 C 和 f_4 中节点 D 结束，并且 cross-producting 表中的组合数目从 480 (3×5×4×4×2)减少到 270(3×5×3×3×2)。直到 cross-producting 表中不同的组合数减少到比预定义的门限值(比如 100 h)之前，上述步骤反复进行。在第 2 次和第 3 次迭代过程中，仍然选择前缀树 f_2。第 2 次迭代中选中规则 F_d、F_h 和 F_k，前缀树 f_3 的节点 B 和前缀树 f_2 的节点 B 移除，cross-producting 表中的组合数进一步减少到 144(3×4×2×3×2)。第 3 次迭代中选择规则 F_c、F_g 和 F_l。3 次规则选择迭代过后生成了 3 个独立集组合，生成的前缀树如表 5 所示，3 次迭代生成的 3 组独立集依次是(F_j,F_f,F_b)、(F_d,F_k,F_h)、(F_c,F_l,F_g)。在第 3 次迭代以后，cross-producting 的组合数由最初的 480 降低到 48(2×3×2×2×2)，存储率减少了 90%，由于定义门限值是 100，因此步骤转为停止。

表 5　3 个独立集的前缀树

图例	前缀树
	f_1 前缀：0,000; B(1): F_i C(3):F_a,F_b,F_e\
	f_2 前缀：1,10，111; C(1):F_i　D(1):F_e E(1):F_a 独立集: (F_j,F_f,F_b) (F_d,F_k,F_h) (F_c,F_l,F_g)
	f_3　前缀：*,10； A(1): F_i　D(2):F_a,F_e
	f_4　前缀：*,01； A(2):F_a,F_i　B(1):F_e
	f_5　前缀：UDP,TCP; A(1):F_a B(2):F_e,F_i

4　算法比较

改进算法与原始的 cross-producting 算法不同之处在于它不需要额外比较独立集中的规则，并且通过控制规则的数目来调整存储使用情况。与原始算法相比，改进算法为一维搜索组织的数据包含 cross-product 表的索引和访问独立集必需的域规格，因此，cross-producting 表需要的存储量可以随交换查找的加速而进一步降低，如图 1 所示。

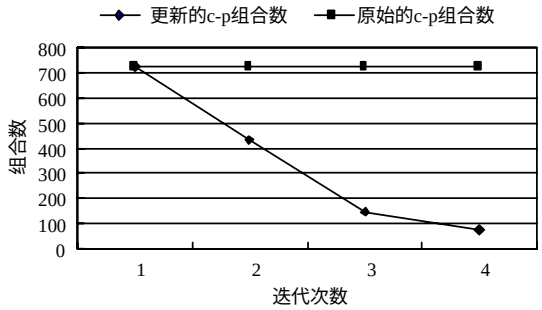


图 1　cross-product 组合数目比较

改进算法根据前缀的数目和长度的可估性利用多路查找树数据结构来进行搜索，改进算法的空间复杂度取决于 cross-producting 表使用的存储。假设在每个域中有 N 个不同的域规格，其中，至少 N/W 规格对应于左叶子节点，因此，在最小的独立集里有 N/W 个规则。在生成了 I 个独立集之后，cross-producting 组合数为$(N-I\times N/W)d = O(((W-I)N/W)d)$。由此可见，改进算法明显提高了流分类在存储上的性能，并且灵活地支持各种规模的规则库。

5　结束语

本文提出了一种新的基于计算几何流 cross-producting 的算法，利用线性查找来处理规则库规模扩大的问题。通过设计几何流分类模型，将 cross-producting 方案和独立集的优势结合起来构建新颖的查找过程和数据结构，调整存储状况，形成新型流分类方法，以适度的时间消耗作为权衡，改进存储效率，而关于如何改进速度和存储之间的平衡有待进一步地研究。

参考文献

[1] Gupta P, McKeown N. Algorithms for Packet Classification[J]. IEEE Network, 2001, 15(2): 26-30.

[2] Gupta P, McKeown N. Packet Classification on Multiple Fields[J]. ACM Computer Communication Review, 1999, 29(4): 150-158.

[3] Woo T Y C. A Modular Approach to Packet Classification: Algorithms and Results[C]//Proc. of IEEE INFOCOM'00. [S. l.]: IEEE Press, 2000: 1213-1222.

[4] Iyer S, Kompella R, Shelat R, et al. An Architecture for Fast and Flexible Packet Classification[J]. IEEE Network, 2001, 15(2): 33-41.

[5] Taylor D E, Turner J S. Scalable Packet Classification Using Distributed Crossproducting of Field Labels[C]//Proc. of IEEE INFOCOM'05. Miami, USA: [s. n.], 2005.

[6] 肖金阁，赵荣彩，单　征，等. 包分类算法中规则转换方法研究[J]. 计算机工程, 2009, 35(9): 46-47.

编辑　索书志