基于多维立方体的正则表达式匹配算法

宫阳阳[1],刘勤让[1],邵翔宇[1],朱圣平[1],邢池强[1],彭志彬[1],贺业里[2]
(1.国家数字交换系统工程技术研究中心,河南郑州 450002;2.65711部队司令部,辽宁大连 116599)

摘 要: 针对特定条件下含有".*"的正则表达式规则相互作用产生的状态爆炸问题,本文提出一种基于多维立方体的确定性有限自动机(Deterministic Finite Automaton,DFA)结构,将冗余状态按维度划分并压缩,并设计相应的多维立方体确定性有限自动机(Multi-Dimension-Cube-DFA,M-D-Cube-DFA)算法,通过构造动态交点的方法实现等价的状态转移.理论分析和仿真实验表明,与DFA算法相比,在维持时间复杂度不变的基础上对状态数目和存储空间进行了对数级别压缩.

关键词: 正则表达式;特征匹配;自动机;确定性有限自动机;非确定性有限自动机;多维立方体
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2014)09-1818-05
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.09.024

A Regular Expression Matching Algorithm Based on Multi-Dimensional Cube

GONG Yang-yang[1], LIU Qin-rang[1], SHAO Xiang-yu[1], ZHU Sheng-ping[1], XING Chi-qiang[1], PENG Zhi-bin[1], HE Ye-li[2]
(1. *National Digital Switching System R&D Center, Zhengzhou, Henan* 450002, *China*; 2. 65711 *Troops Headquarters, Dalian, Liaoning* 116599, *China*)

Abstract: A deterministic finite automaton(DFA) structure based on multi-dimensional cube is proposed aiming at the state explosion problem generated by the interaction among regular expression rules containing ".*" under certain conditions. It divides and compresses redundant states by dimension. Then the algorithm M-D-Cube-DFA is proposed. It achieves equivalent state transition by constructing dynamic intersections. Theory and simulation results show that, compared with the conventional DFA algorithm, M-D-Cube-DFA achieves a logarithm-level compression of the number of states and the storage space without changing the time complexity.

Key words: regular expression; signature matching; finite automaton; deterministic finite automaton; nondeterministic finite automaton; multi-dimensional cube

1 引言

目前,正则表达式匹配(Regular Expression Matching, REM)技术广泛应用于网络入侵检测和防御系统,作为深度数据包检测的核心算法来进行非法内容检测、恶意代码检测、入侵检测、协议分析等.近年来,随着计算机网络的持续发展及网络带宽的逐年增加,REM面临如何满足高速网络数据包处理的挑战.

在理论上,正则表达式等价于有限自动机语言[1],因此正则表达式通常采用非确定性有限自动机(Nondeterministic Finite Automaton, NFA)和确定性有限自动机(Deterministic Finite Automaton, DFA)实现.由于两者原理上的不同,NFA匹配效率非常低,而DFA匹配速度较高,但占用的存储空间较大,尤其是随着规则集的扩大会出现状态爆炸,存储急剧增加的现象.为了借助DFA速度方面的优势,当前有大量研究通过缩减其存储空间对DFA算法进行改进和优化.

状态爆炸的一种典型情况是,多条正则表达式编译成一个DFA的时候,正则表达式之间相互重叠和影响导致DFA的状态数目大规模的增长,从而引起状态爆炸.具有 n 个状态DFA的主要存储消耗为一个 $n \times |\Sigma|$ (Σ 为ASCII字母表)的二维表(Single Transition Table, STT),它占用DFA空间的95%以上[2],因此文献[3~7]针对STT冗余进行压缩,然而这类方法的缩减策略是线性的,与DFA状态数多项式级别乃至指数级别的增长相比,很难从根本上解决空间爆炸问题.因此,文献[8~13]从自动机的结构出发,力图从根本上解决空间爆炸问题.其中代表性的算法有基于规则分组的mDFA[8]、基于混合结构的Hybrid-FA[9]、基于历史记录的H-FA[10]以及扩展自动机XFA[11].

以上算法在很大程度上缩减了DFA存储空间,然而建立在共同的数学模型的基础上,使用二维跳转图或

收稿日期:2013-07-17;修回日期:2013-11-18;责任编辑:覃怀银
基金项目:国家863高技术研究发展计划(No.2011AA01A103、2011AA01A101);国家973重点基础研究发展计划(No.2012CB315901,No.2013CB329104);国家科技支撑计划(No.2011BAH19B01)

跳转表来表征 DFA 的状态转移,这很大程度上限制了研究者的思维.本文将多维立方体概念引入到 DFA 的结构设计中,以{.*SubStr_1.*SubStr_2}类型且".*"后接字符不同的规则集为例,提出多维立方体结构(Multi-Dimension-Cube,M-D-Cube),并设计相应的多维立方体确定性有限自动机算法(Multi-Dimension-Cube-DFA,M-D-Cube-DFA),解决了特定条件下含有".*"的规则相互作用造成的 DFA 状态空间爆炸问题,为基于自动机结构改进的空间缩减算法提供了新的数学模型和思路.

2　M-D-Cube 结构及 M-D-Cube-DFA 算法

本节介绍含".*"的正则表达式相互作用产生的状态爆炸问题,以一种特殊规则集为例引入多维立方体结构,继而设计相应的 DFA 算法,最后对算法时空复杂度进行理论分析.

2.1　状态空间爆炸问题

在实际的应用系统中,规则集包含了多条规则,为了快速完成对多条规则的匹配,一般将它们编译成一个联合 DFA,联合 DFA 可能存在状态空间爆炸问题,其中最常见的一种情形是规则间".*"相互作用造成 DFA 状态数目的急剧增长.为了叙述方面,本文后续介绍统一用 M 表示规则的数目,N 表示规则对应的状态数,X 表示规则含有的".*"数目.一般而言,当 M 条规则编译为一个 DFA 时,每个表达式中出现 X 次".*",则需要 $O(X^M)$个状态来记录所有可能出现的前缀的幂集,在这种情况下,即使每个规则只有 1 个".*"操作符,增加一个类似的规则也会使得 DFA 的状态数目增加 1 倍.在 L7-filter 中有很多类似的规则,例如,识别远程桌面协议的{.*rdpdr.*cliprdr.*rdpsnd}和 Internetradio 协议的{.*membername.*session.*player};Snort 中也存在大量类似规则,而且某些规则中,".*"符号的数目甚至达到了 6 个[2].

假设有 $M>1$ 个正则表达式联合编译,第 i 个正则表达式($i=1,2,\cdots,M$)有 N_i 个字符,那么将 M 个正则表达式编译成一个 NFA 对应的状态数:

$$S=\sum_{i=1}^{M}N_i \tag{1}$$

NFA 通过子集构造法生成 DFA,因此 NFA 的状态数目对应了编译为 DFA 的状态数目的理论下界.M 个{.*SubStr_1.*SubStr_2⋯.*SubStr_X_m}类型的规则合并编译,每个子串 SubStr_j 对应的长度为 L_j($j=1,2,\cdots,X_m$),则合并编译得到的 DFA 的状态空间理论下界为

$$S_{\text{lmt}}=\sum_{m=1}^{M}\left(\sum_{j=1}^{X_m}(L_j)+X_m-1\right) \tag{2}$$

取 $L_j=2(j=1,2,\cdots,X_m)$,$X_m=2(m=1,2,\cdots,M)$,则

$$S_{\text{lmt}}=\sum_{i=1}^{M}\left(\sum_{1}^{2}(2)+2-1\right)=5M \tag{3}$$

综上,M 个{.*cha1cha2.*cha3cha4}类型的规则联合编译生成的 DFA 状态数目的理论下界

$$S_{\text{lmt}}=O(5M)=O(M) \tag{4}$$

2.2　M-D-Cube 结构设计

本节将多维立方体概念引入到 DFA 的结构设计中,以规则集 A:"Sig1:{.*ab.*cd};Sig2:{.*ef.*gh};Sig3:{.*ij.*kl}"为例介绍我们的方法.图 1 所示的是规则 Sig1:{.*ab.*cd}对应的 DFA 转移图,该规则中的元字符"."代表匹配任意字符,"*"代表匹配 0 次或任意多次,该规则代表顺序包含 ab 及 cd 的字符串.

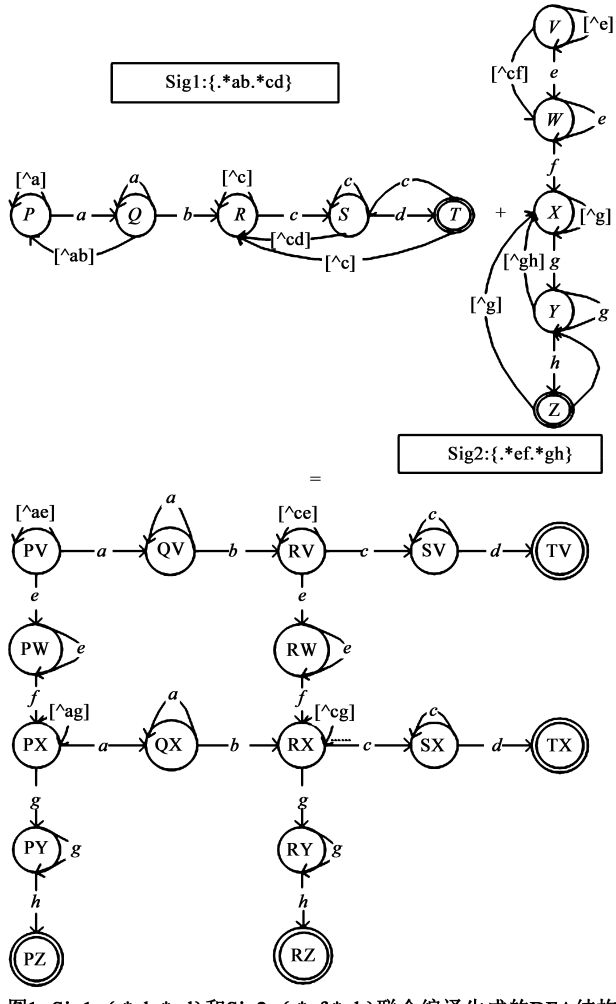
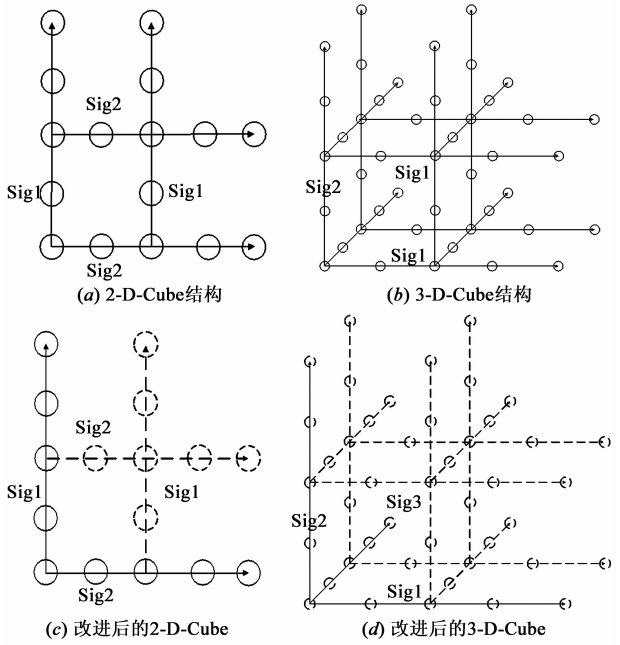


图1　Sig1: {.*ab.*cd}和Sig2: {.*ef.*gh}联合编译生成的DFA结构

每个规则构造的 DFA 有 5 个状态,当 Sig1、Sig2 合并编译时,生成的 DFA 结构状态数为 16.如果不考虑在交点状态处不同的跳转,状态标注 RV、RW、RX、RY、RZ 的一行复制了状态标注 PV、PW、PX、PY、PZ 的一行,状态标注 PX、QX、RX、SX、TX 的一列复制了状态标注 PV、

QV、RV、SV、TV 的一列.实际上由于".*"匹配任意字符串,*Sig1* 的".*"包含了 Sig2 的所有状态可能,同样 Sig2 的".*"包含了 Sig1 的所有状态可能,两者的交叉互联产生了与 Sig1 和 Sig2 相同的两个规则跳转链,状态数增加了 $5\times2-4=6$.如果把两者状态相连成一条直线,就构造出一个 2-D-Cube 结构,如图 2(a).当 Sig1、Sig2、Sig3 合并编译时,得到 3-D-Cube 结构,如图 2(b).当维数 $M>3$ 时,将会相应的得到 M-D-Cube,超立方体结构无法给出图示.



(a) 2-D-Cube结构 (b) 3-D-Cube结构

(c) 改进后的2-D-Cube (d) 改进后的3-D-Cube

图2 2-D-Cube和3-D-Cube结构

下面给出 M-D-Cube 的格点数目的一般公式:边数 $L=M\cdot2^{M-1}$,顶点数 $V=M\cdot2^M$,设每条边的格点数为 N,则 M-D-Cube 的格点数:

$$T=(N-2)\cdot L+V \tag{5}$$

因此 M 个规则编译成一个 DFA 对应的状态数目为:

$$S=T=(N-2)\cdot M\cdot2^{M-1}+2^M \tag{6}$$

令每个规则含有的".*"数目 $X=2$,状态数 $N=5$,取 $M=1,2,\cdots,7,8$,通过式(6)计算得到的状态数目和实验得到的状态数目如表 1 所示:

表 1 $X=2, N=5, M=1,2,\cdots,7,8$ 时状态数目理论与实验对比

M	1	2	3	4	5	6	7	8
S(理论)	5	16	44	112	272	640	1472	3328
S(实验)	5	16	44	112	272	640	1472	3328

可以看出,在不同的维度下,式(6)得到的状态数目都与实验相一致,因此采用 M-D-Cube 结构的方法是合理的.考虑到每一维上状态跳转的相似性,我们采用如图 2(c)、(d)所示的实线部分的结构作为其基本框架,保存基本结构对应的子 DFA,同一维度的 DFA 映射到对应的坐标轴上,使用少量的附加存储寄存交点信息.M-D-Cube 将状态冗余按维度进行合并和压缩,缩减了每个规则与另外 $M-1$ 个规则相互交互产生的巨大的状态空间,从而状态空间数目与维数 M 呈正比.

2.3 M-D-Cube-DFA 算法设计

本节介绍 3-D-Cube-DFA 算法,M-D-Cube-DFA($M>3$)算法的设计可以据此进行线性扩展.规则集 A 的三个规则对应的 DFA 二维跳转表如表 2 ~ 4 所示.表中第一列有方框的状态,表示该 DFA 中的交点状态索引,第一行有方框的字符表示对应的转移字符.

表 2 Sig**1**:{.*ab.*cd}对应的 DFA 的 STT

	…	[a]	b	[c]	d	e	f	g	h	i	j	k	l	…
[0]	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	0	0	0	0	0	0	0	0	0	0	0
[2]	2	2	2	3	2	2	2	2	2	2	2	2	2	2
3	2	2	2	2	4	2	2	2	2	2	2	2	2	2
4	2	2	2	3	2	2	2	2	2	2	2	2	2	2

表 3 Sig**2**:{.*ef.*gh}对应的 DFA 的 STT

	…	a	b	c	d	[e]	f	[g]	h	i	j	k	l	…
[0]	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	2	0	0	0	0	0	0	0
[2]	2	2	2	2	2	2	2	3	2	2	2	2	2	2
3	2	2	2	2	2	2	2	2	4	2	2	2	2	2
4	2	2	2	2	2	2	2	3	2	2	2	2	2	2

表 4 Sig**3**:{.*ij.*kl}对应的 DFA 的 STT

	…	a	b	c	d	e	f	g	h	[i]	j	[k]	l	…
[0]	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	0	2	0	0	0	0	0	0	0	2	0	0	0
[2]	2	2	2	2	2	2	2	2	2	2	2	3	2	2
3	2	2	2	2	2	2	2	2	2	2	2	2	4	2
4	2	2	2	2	2	2	2	2	2	2	2	3	2	2

每个规则内".*"后的字符我们定义在数组 Node_State[]内,例如对于 Sig1,Node_State[0] = a,Node_State[1] = c,三个规则的 Node_State[]中的元素构成 3-D-Cube 结构的 8 个交点,采用 3bit 来记录交点坐标信息.由于 DFA 当前激活状态仅有 1 个,因此通过动态存储 DFA 当前激活状态相邻的两个交点(Current_Node,Neighbour_Node)信息来判断所在边,从而得到当前激活状态位置,具体结构如图 3 所示.

值得注意的是在交点处增加了从当前边跳转到其他边的判断,由当前交点信息决定.数据存储结构如图 4 所示,存储 3 个 DFA 的跳转表、当前状态所在边(由当

前交点和相邻交点决定)、当前边交点跳转信息、当前的 DFA 指针.当前边交点跳转信息通过对 3 个子 DFA 的读取获得.

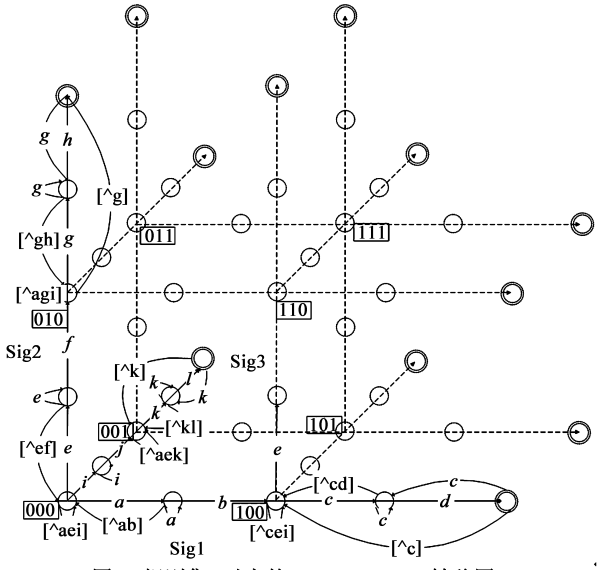


图3　规则集*A*对应的3-D-Cube-DFA转移图

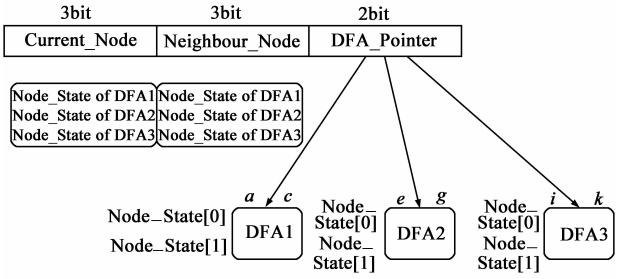


图4　规则集*A*对应的3-D-Cube-DFA数据存储结构

M-D-Cube-DFA 算法进行字符串匹配时循环执行如下步骤:

Step1:判断当前状态是否交点,如果是,转 Step3,如果不是,转 Step2;

Step2:读取当前子 DFA 跳转表进行跳转,如果跳转状态为交点状态,转 Step3,如果不是,转 Step6;

Step3:读取交点信息,判断需要读取的子 DFA,转 Step4;

Step4:判断 Step3 读取的子 DFA 是不是与当前子 DFA 一致,如果不是,转 Step5,如果是,转 Step6;

Step5:更新交点信息以及 DFA 指针,转 Step6;

Step6:读取当前 DFA 跳转表进行跳转.

2.4　复杂度分析

设规则集含有 M 个正则表达式{.*SubStr_1.*SubStr_2},每个规则的子串长度相同,状态数均为 N.考虑最差情况,算法依次执行 Step1 ~ Step6,由于每一步算法的时间复杂度都是 $O(M)$,因此整个算法的时间复杂度为 $O(6)=O(1)$.状态空间方面,算法要存储 M 个子 DFA 的对应的状态,因此状态数目复杂度为 $O(MN)=O(M)$,达到了 DFA 状态数目的理论下界.存储空间方面,除了 M 个子 DFA 的 STT 外,还需要存储两个动态节点信息、DFA 指针信息,由于每个 DFA 的 STT 存储规模为常数,动态节点的位宽与规则数目成正比,因此存储空间复杂度为 $O(MN+M+1)=O(M)$.由表 5 和式(4)可知,与 DFA、NFA 算法相比,M-D-Cube-DFA 算法的状态空间和存储空间在保证时间复杂度的基础上达到了理论下界,但是时间复杂度为非严格的 $O(1)$,付出了增加访存次数和判断指令的代价.

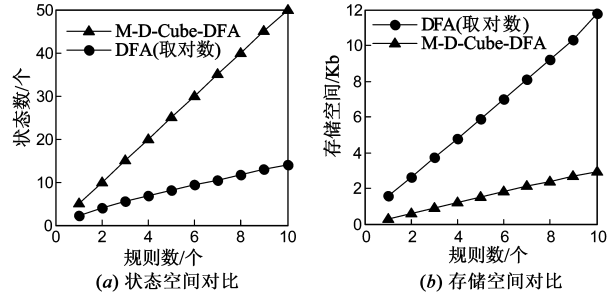
表 5　算法复杂度对比

算法	时间复杂度	状态/存储复杂度
NFA	$O(M)$	$O(M)$
DFA	$O(1)$strictly	$O(M)\sim O(2^M)$
M-D-Cube-DFA	$O(1)$non-strictly	$O(M)$

3　仿真验证

为了进一步验证本文的结论,选取规则集性质如下:包含 1 ~ 10 条形式为{.*SubStr1.*SubStr2}的正则表达式,其中 $X=2$,每个子串的长度 $L=2$ 且具有不同的前缀.利用 Regular Expression Processor 仿真得到 DFA 和 M-D-Cube-DFA 生成的状态数以及存储空间.

为了将两者放在同一个图中比较,将 DFA 的状态数以及存储均取对数处理.图 5 中 DFA 的曲线显示,随着规则数的增加,取过对数之后的 DFA 状态个数和存储空间随规则数线性增长,因此 DFA 状态个数和存储空间随规则数增长指数增长.M-D-Cube-DFA 的曲线显示,M-D-Cube-DFA 的状态个数和存储空间随规则数增加线性增长.因此,M-D-Cube-DFA 对 DFA 状态个数和存储空间进行了对数压缩,验证了本文的结论.



(*a*) 状态空间对比　(*b*) 存储空间对比

图5　M-D-Cube与DFA仿真对比

4　总结

针对特定条件下含“.*”的正则表达式间相互作用产生状态空间爆炸的问题,本文提出一种基于多维

立方体的 DFA 结构 M-D-Cube,将冗余状态按维度划分并压缩到坐标轴上,并设计相应的算法 M-D-Cube-DFA,通过构造动态交点的方法记录当前状态所在的位置,从而实现与 DFA 等价的状态转移.与传统的 DFA 算法相比,M-D-Cube-DFA 算法时间复杂度为 $O(1)$,状态空间复杂度为 $O(M)$,存储空间复杂度为 $O(M)$.该算法在保证时间复杂度不变的前提下对状态数目和存储空间进行了对数压缩,使状态空间复杂度达到了理论下界,解决了规则间由".*"产生的状态空间爆炸问题,但是付出了增加访存次数和判断指令的代价.与当前基于 STT 冗余线性压缩的算法比,M-D-Cube-DFA 算法的压缩级别是对数级别压缩;与基于自动机结构改进的 DFA 算法进行比较,M-D-Cube 模型具有启发意义,将二维理论模型扩展到多维,为解决 DFA 空间爆炸问题提出了新的数学模型和努力方向.

本文针对的规则要求符合{.*SubStr_1.*SubStr_2}类型且".*"后接字符不同,而实际的规则需在此基础上扩展,例如:每个规则".*"的个数 $X \neq 2$;".*"的后接字符相同;".*"被替换为"[^\n]* "、".{m}"或"[a-z]{m}"等闭包操作.针对扩展类型规则的 M-D-Cube 结构构造以及相应的算法设计是下一步研究的重点.

致谢 感谢姜鲲鹏、叶狄秋、贺炜师兄给本文提出的参考意见;感谢 Michela Becchi 提供的仿真软件.

参考文献

[1] Hopcroft J E, Ullman J D. Introduction to Automata Theory, Languages and Computation [M]. 2nd Edition. US: Addison Wesley, 2001. 201 – 203.

[2] 张树壮,罗浩,方滨兴.面向网络安全的正则表达式匹配技术[J].软件学报,2011,22(8):1838 – 1853.
Zhang Shu-zhuang, Luo Hao, Fang Bin-xing. Regular expression matching for network security[J]. Journal of Software, 2011, 22(8): 1838 – 1853. (in Chinese)

[3] Kumar S, et al. Algorithms to accelerate multiple regular expressions matching for deep packet inspection[A]. Proceedings of ACM SIGCOMM 2006[C]. Pisa: ACM Press, 2006. 339 – 350.

[4] Ficara D, Giordano S, Procissi G, et al. An improved DFA for fast regular expression matching[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(5): 29 – 40.

[5] Becchi M, Cadambi S. Memory-efficient regular expression search using state merging[A]. Proceedings of IEEE INFOCOM 2007[C]. Anchorage: IEEE Press, 2007. 1064 – 1072.

[6] Qi Y, Wang K, Fong J, et al. Feacan: Front-end acceleration for content-aware network processing[A]. INFOCOM, 2011 Proceedings IEEE[C]. Shanghai: IEEE Press, 2011. 2114 – 2122.

[7] Liu Tingwen. An efficient regular expressions compression algorithm from a new perspective[A]. INFOCOM, 2011 Proceedings IEEE[C]. Shanghai: IEEE Press, 2011. 2129 – 2137.

[8] Yu F, et al. Fast and memory-efficient regular expression matching for deep packet inspection[A]. Proceedings of ACM/IEEE ANCS 2006[C]. San Jose: ACM Press, 2006. 93 – 102.

[9] Becchi M, Crowley P. A hybrid finite automaton for practical deep packet inspection[A]. Proceedings of ACM CoNEXT Conference[C]. New York: ACM Press, 2007. 1 – 12.

[10] Kumar S, et al. Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia[A]. Proceedings of ACM/IEE ANCS 2007[C]. Orlando: IEEE Press, 2007. 155 – 164.

[11] Smith R, Estan C, Jha S, et al. Fast signature matching using extended finite automaton(XFA)[A]. ICISS'08[C]. Heidelberg: Springer-Verlag, 2008. 158 – 172.

[12] 徐乾,等.深度包检测中一种高效的正则表达式压缩算法[J].软件学报,2009,20(8):2214 – 2226.
Xu Qian, et al. Efficient regular expression compression algorithm for deep packet inspection[J]. Journal of Software, 2009, 20(8): 2214 – 2226. (in Chinese)

[13] 张大方,张洁坤,黄昆.一种基于智能有限自动机的正则表达式匹配算法[J].电子学报,2012,40(8):1617 – 1623.
Zhang Da-fang, Zhang Jie-kun, Huang Kun. A regular expression matching algorithm with smart finite automaton[J]. Acta Electronica Sinica, 2012, 40(8): 1617 – 1623. (in Chinese)

作者简介

宫阳阳 男,1988 年 12 月出生于河南濮阳.2007 年毕业于北京大学物理系,现为国家数字交换系统工程技术研究中心在读硕士,主要研究方向为信息安全、网络安全.
E-mail:842903190@qq.com

刘勤让 男,1975 年 11 月出生于河南商丘,博士.国家数字交换系统工程技术研究中心副教授,硕士生导师,主要研究方向为网络业务识别与控制.
E-mail:qinrangliu@sina.com

邵翔宇 男,1992 年 01 月出生于河南濮阳.2008 年毕业于电子科技大学通信工程,现为国家数字交换系统工程技术研究中心在读硕士,主要研究方向为信息安全、网络安全.
E-mail:690627819@qq.com