文章编号: 1001- 2486(2008) 04- 0082- 06

基于内容分析的协议识别研究*

陈曙晖, 苏金树
(国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: 为解决多模式同时匹配的协议识别性能问题, 提出了一种多模式组合有限状态机; 以 Thompson 算法为基础, 提出了一种压缩 ε 的 NFA 构造算法, 该算法通过减少 ε 边及其对应状态, 有效提高在协议模式编译时, NFA 转换成 DFA 及 DFA 最小化过程的性能; 基于上述理论与算法实现了一种 One-Pass 的组合多模式协议识别系统。实验表明: 结合上述技术实现的系统, 编译性能比标准 DFA 构造过程提高了 7 倍以上, 匹配性能比 L7-Filter 提高了近 20 倍。

关键词: 网络安全; 协议识别; 模式匹配; 正则表达式
中图分类号: TP301 文献标识码: A

Protocol Identification Research Based on Content Analysis

CHEN She-hui, SU Jin-shu
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: To solve the performance problem in Regular Expression matching of Protocol Identification, this paper introduces a Multi-pattern FSM (MPFSM), which can use one FSM to match several Regular Expressions. Based on Thompson algorithm, an Epsilon Compressed NFA Construction Algorithm is also put forward and implemented. This algorithm enhances the performance of conversion from NFA to DFA by decreasing the epsilon edges and the corresponding states. A One-pass Multiple-pattern protocol identification system is also implemented using the Multi-pattern FSM and corresponding algorithms. Experiments based on actual traffic are employed to show that the compile speed would be 7 times faster than the usual transfer process, and the Matching speed would be 20 times faster than the L7-Filter.

Key words: network security; protocol identification; pattern matching; regular expression

协议识别用于确定网络流量所属的协议类型。在基于内容分析的协议识别中, 报文体与一组模式进行匹配, 如 Linux 上的 L7-Filter[1] 和 IPP2P[2]。由于正则表达式的强大功能, 入侵检测系统开始使用正则表达式取代串模式, Snort[3] 从 2003 年开始使用正则表达式, Bro[4] 使用的模式也为正则表达式。有关正则表达式的基础理论来源于文献[5]。

文献[6]根据报文深度检测的特点, 改写规则并降低由正则表达式所构造的 DFA(确定性有限自动机)的复杂性, 通过分组降低存储空间。文献[7]将 DFA 中一个状态的多个边用单个缺省边代替, 引入一种称为输入延迟的 DFA(D^2FA)来降低边的存储空间。文献[8]采用一种转移表, 通过将多个输入压缩成一个输入, 构建输入为 Σ^n 的状态机, 提高了匹配性能。文献[9]采用 FPGA 构造 FSM(有限自动机), 适合于模式固定的应用。本文主要从正则表达式的编译与匹配性能两方面优化协议识别的性能。

1 协议识别

协议识别的工作包括归纳各协议特征, 将每个协议总结为包含正则表达式的模式, 对每一个流上的前 n 个报文进行模式匹配, 直到识别协议, 或者报文个数超过一定数目(匹配失败)。L7-Filter 性能较低的主要原因有:

* 收稿日期: 2008- 02- 10
基金项目: 国家自然科学基金资助项目(90604006); 国家部委资助项目
作者简介: 陈曙晖(1974—), 男, 副研究员, 博士生。

(1) L7-Filter 的匹配引擎基于 GNU 的 regex 实现, GNU 的 regex 采用 NFA(非确定性有限自动机) 匹配引擎, 而 NFA 对单个正则表达式、单个字节的最差处理复杂性为 $O(n^2)$ [6], 对长度为 l 的报文, 其最差处理复杂性为 $O(l \times n^2)$。

(2) L7-Filter 针对每个协议采用分离状态机进行匹配, 若协议个数为 m, 则其最差处理复杂性为 $O(m \times l \times n^2)$。

(3) 因为采用 NFA, 当前活跃状态较多, L7-Filter 难以保存每匹配完一个报文后的状态, 某些报文需匹配多次, 如图 1 所示。同一条流上的前几个报文需反复进入多个匹配引擎进行匹配, 降低了整个系统的性能。

2 MPFSM 与 ECNFA

传统状态机基于单模式, 匹配多个模式时, 被处理内容需分别多次进入匹配引擎进行匹配, 多模式的状态机(Multi-pattern FSM, MPFSM) 将多个模式组合在同一个状态机中, 起到加速匹配的作用。

定义 1 多模式组合状态机 $MPFSM = \{Q, \Sigma, \delta, q_0, MF\}$, 其中, Q 为所有状态的集合, Σ 为输入符号, δ 为转换函数, $q_0(q_0 \in Q)$ 是初始状态, $MF = \{< q, p > \mid q \in Q, p \in PatternSet\}$ 是二元有序对< 接受状态, 所匹配模式> 的集合。

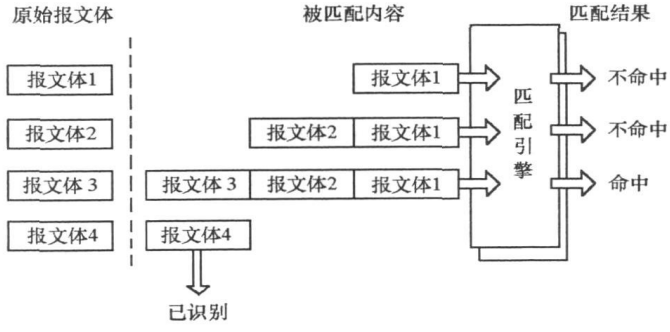


图 1 L7-Filter 的扫描过程示例

Fig. 1 Example of L7-Filter scanning

当 $|\{p \mid < q, p > \in MF\}| = 1$ 时, 定义针对五元组的一一映射 g: $MPFSM \rightarrow FSM$, 其中, g 对五元组 $MPFSM$ 中的 Q, Σ, δ, q_0 为同射; 对 $MPFSM$ 中的 MF, 若 $mf \in MF$, 因 $|\{p \mid < q, p > \in MF\}| = 1$, 假设 p 从 0 开始编号, 则 $mf = < f, 0 >$, 定义 $g(mf) = f, f \in F$。g 是双射, 所以 $|\{p \mid < q, p > \in MF\}| = 1$ 时, $MPFSM$ 与 FSM 同构。

定义 2 包含关系 $\in_p$, $FSM_i \in_p MPFSM$ 当且仅当存在一个与 FSM_i 重命名状态后等价[5] 的单模式状态机 $FSM_j = \{Q_j, \Sigma_j, \delta_j, q_{j,0}, MF_j\}$, $Q_j \subseteq Q$, $\Sigma_j \subseteq \Sigma$, $q_{j,0} = q_0$, $\delta_j \subseteq \delta$, 且对任意 $f_j \in F_j$, $< f_j, i > \in MF$。

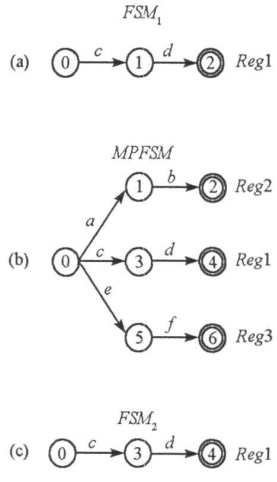
图 2(b) 中的 $MPFSM$ 包含图 2(a) 中的 FSM_1, 因为存在一个与 FSM_1 重命名后等价的 FSM_2(图 2(c)), $Q_2 \subseteq Q$, $\Sigma_2 \subseteq \Sigma$, $q_{2,0} = q_0$, $\delta_2 \subseteq \delta$, 且对唯一的结束状态 4, 有 $< 4, 1 > \in MF$。

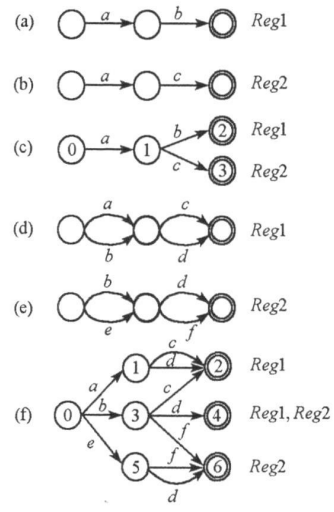
公共前缀对组合状态机的状态数产生重要影响。图 3(a) 的 $N(ab)$ 与图 3(b) 的 $N(ac)$ 合成时, 减少了 2 个状态(图 3(c))。在图 3(c) 中, 当到达状态 2 时, 识别 $Reg1$; 到达状态 3 时, 识别 $Reg2$。而图 3(d) 的 $N((a \mid b)(c \mid d))$ 与图 3(e) 的 $N((b \mid e)(d \mid f))$ 合成时, 增加了 1 个状态(图 3(f))。在图 3(f) 中, 当到达状态 2 时, 识别 $Reg1$; 到达状态 4 时, 同时识别 $Reg1$ 与 $Reg2$; 到达状态 6 时, 识别 $Reg2$。

$MPFSM$ 由 m 个 FSM 组合而成, 当且仅当对每个单模式状态机 FSM_i ($1 \leqslant i \leqslant m$), 有 $FSM_i \in_p MPFSM$, 且对 $MPFSM$ 中从初态到终态 $< f, i >$ 的任意路径 P, 总能从 $MPFSM$ 中找到一个同构的单模式状态机 FSM_j, P 是 FSM_j 中从初态到终态的路径。

Thompson 构造算法[10] 的主要方法为: 对 Σ 中的任意字节 a, 其 NFA 如图 4(a) 所示。假设 $N(s)$ 和 $N(t)$ 分别为正则表达式 s 和 t 所接受的语言, 则 $N((s)*)$、$N((s)+)$、$N(s?)$、$N(st)$、$N(s \mid t)$ 分别如图 4(b) ~ (f) 所示。Thompson 算法只需扫描一次正则表达式, 其构造复杂性为 $O(l)$, 其中 l 为正则表达式

的长度。Thompson 算法所构造 NFA 的数据结构简单,但是引入了过多的状态。多个等价状态增加了子集构造算法中 ε*closure* 与 *move* 的复杂性。





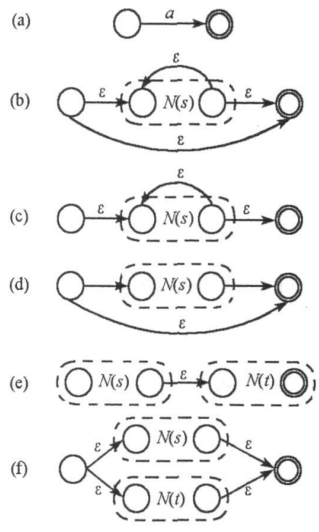


图 2 MPFSM 与 FSM 的关系 图 3 分离状态机与合成状态机的状态数 图 4 Thompson 构造算法

Fig. 2 Relationship of MPFSM & FSM Fig. 3 States of separate FSM vs. MPFSM Fig. 4 Thompson algorithm

对 q 中的任意状态 t, ε*closure*$(t)=\{s \mid <t, \varepsilon^{i}> \rightarrow s \in \delta, i \geqslant 0\}$ 是以 t 为起点,由一个或多个 ε 边可达状态以及状态 t 本身的集合。ε*closure*$(T)=\bigcup_{t \in T}$ε*closure*(t) 是以 T 中所有状态为起点,由一个或多个 ε 边可达的状态集与状态集 T 的并集。*move*$(T, a)=\{s \mid <t, a> \rightarrow s \in \delta, t \in T\}$ 是以 T 中所有状态为起点,由 a 边可达的状态的集合。

Thompson 算法构造的 $N([a_{[0]}-a_{[n-1]}][b_{[0]}-b_{[n-1]}])$ 如图 5 所示。$|$ ε*closure*$(A)|=2n-1$, ε*closure*(A) 求取一次之后,针对 ε*closure*(A) 对 Σ 上的所有可能输入(假设有 k 种输入)求取 *move* 的复杂性为 $O(nk)$。求取 ε*closure*(*move*(ε*closure*$(A), a_{[i]})$)的复杂性为 $O(n)$,因此针对 A 的所有输入求取 *move* 和 ε*closure* 的复杂性将高达 $O(kn^2)$,正则表达式由 m 个连续选择性子表达式组成时,计算复杂性将高达 $O(kmn^2)$。

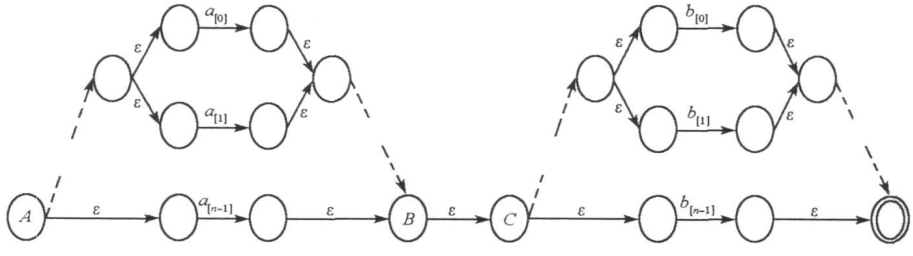


图 5 Thompson 算法构造 $N(a_{[0]}-a_{[n-1]})|(b_{[0]}-b_{[n-1]})$ 的 NFA

Fig. 5 $N(a_{[0]}-a_{[n-1]})|(b_{[0]}-b_{[n-1]})$ constructed by Thompson algorithm

在子集构造算法中,需要对所有构造子集的所有输入 256 种可能进行探测,针对每一个子集 T_i 求取 *move* 的次数为 $|T_i||\Sigma|$。针对 T_i 用 *move* 获得的每个集合 $M_{i,a}$,求取 ε*-dosure* 的运算次数为 $2|M_{i,a}|$。如能降低由于 ε 引入的状态,则可降低每个子集的状态个数,从而有效提高子集构造算法的性能。

压缩 ε 的 NFA(Epsilon-compressed NFA, ECNFA)构造算法去除不必要的 ε,如图 6 所示。ECNFA 构造算法通过压缩 ε,减少了边的个数,$N(s*)$、$N(s+)$ 和 $N(s?)$ 分别减少了 2 个状态,减少了 2 条边;$N(st)$ 减少了 1 个状态,减少了 1 条边;$N(s|t)$ 减少了 2 个状态,减少了 4 条边。对 m 个子表达式选择性连接的模式,其状态减少 $2m-2$ 个,边减少了 $4m-4$ 个。

ECNFA 在减少状态的同时, 引入了所构造 NFA 的扩大化问题。$N(a^*\ b^*)$ 采用 ECNFA 所构造的 NFA 如图 7(a) 所示, 经过子集构造算法之后, 其 DFA 将如图 7(b) 所示, 图 7(b) 的 NFA 所表示的语言为 $N((a|b)^*)$, 显然 $N(a^*\ b^*) \subset N((a|b)^*)$, 前者识别 0 个或多个 a 后紧跟 0 个或多个 b 的串, 后者识别 0 个以上 a 或 b 的串。

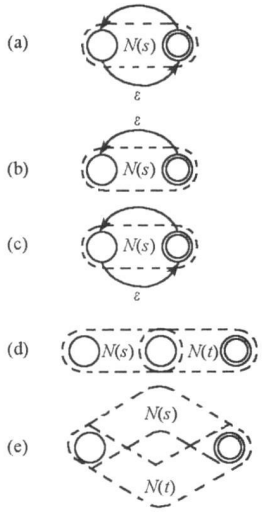


图 6 ECNFA 构造算法

Fig. 6 ECNFA construction algorithm

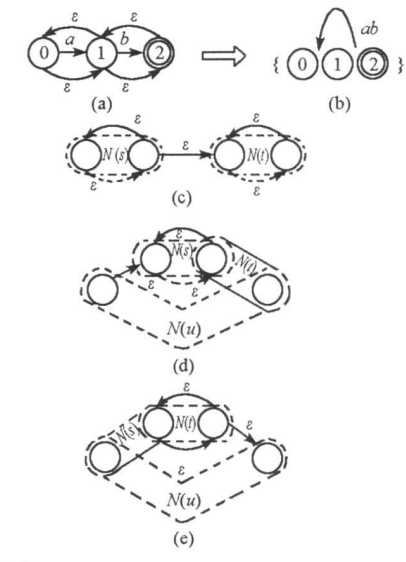


图 7 解决 ECNFA 扩大化问题的方法

Fig. 7 Method to solve ECNFA enlarging problem

ECNFA 的 FSM 扩大化问题是由 ε 回边引入的(图 6(a) 与图 6(b) 的加粗 ε 边), 扩大化问题出现在以下几种情形中: (1) 两个 ε 回边子表达式连接, 即 $N(s^*\ t^*)$、$N(s+\ t+)$、$N(s^*\ t+)$ 和 $N(s+\ t^*)$; (2) ε回边子表达式位于"或"分支的开始处, 即 $N(s^*\ t|u)$ 和 $N(s+\ t|u)$; (3) ε回边子表达式位于"或"分支的结束处, 即 $N(st^*|u)$ 和 $N(st+|u)$。

为解决FSM 扩大化问题, 在图 6 的基础上, 于多个子元素的连接处增加 ε 进边, 通过分割不同子元素区分不应合并的状态。图 7(c) 在两个子元素之间增加一个 ε 进边; 图 7(d) 在 ε 回边子元素前增加一个ε进边, 图 7(e) 在 ε 回边子元素后增加一个ε 进边。

构建过程采用分治策略, 需对输入序列扫描两次, 一次确定或关系的边界以及子元素之间的关系, 一次生成NFA。MP-NFA2DFA 在子集构造算法的基础上, 判断接受状态时考虑多模式问题, 其最差计算复杂性与原子集构造算法相同。子集构造算法的最差复杂性为 $O(2^n)$, 其中 n 为 NFA 的状态数。虽然对一般情况下的复杂性分析较为困难[5], 但减少 NFA 的状态数有利于提高其运行性能。

3 One-Pass 扫描

DFA 中包含状态与边, 报文匹配过程通过"当前状态地址+ 输入"得到下一个状态的地址, 匹配过程的复杂性为 $O(1)$, 对每一个状态有 2^8 个表项, 分别指向对应输入的下一状态地址。

使用标准 DFA 构造过程构造的 DFA, 在匹配过程中, 需要对输入进行重复搜索。这类识别通常用于语言解析过程中, 针对报文内容处理则性能低下。为了加快匹配速度, 针对没有"^"(首锚) 的模式, 在正则表达式前面加上".*", 表示模式可从输入的任意状态开始匹配。随着输入从开始到结束, DFA 可以在输入的任何位置识别模式, 而无需重复扫描。

为解决多个报文多次送入匹配引擎的性能问题, 针对未识别且报文个数未超过指定范围的流, 记录每个流所处的状态, 在下一个报文到达时, 从当前状态而非初始状态继续匹配, 这样已经过了匹配引擎的报文无需再次送入匹配引擎, 减少了匹配引擎的负载。One-Pass 扫描算法专用于协议识别过程, 在每一条流上判断前 $pNum$ 个报文, 通过一次扫描确定报文所属的协议。One-Pass 扫描过程针对长度为 n 的报文, 匹配复杂性为 $O(n)$。图 1 的匹配过程将改变为图 8 所示的过程, 减少了匹配引擎的负载。

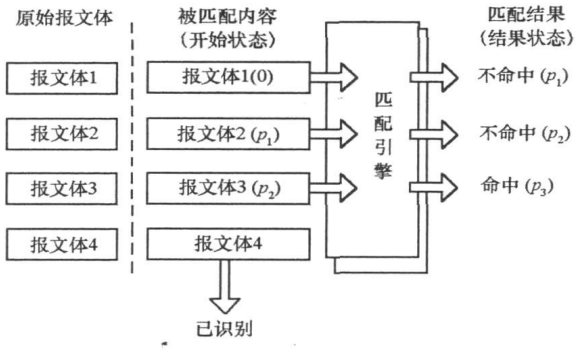


图 8 One-Pass 扫描过程示例

Fig. 8 Example of One-Pass scanning process

4 评估与分析

首先分析压缩 ε 的 NFA 构造算法的性能, 对 L7-Filter 所提供的模式中假阳性比例与假阴性比率较低的 6 种协议, 单个编译与组合编译, 在 Intel Pentium ® 2. 8GHz(1GB 内存, 2MB L2 Cache) 计算机上, 基于 Linux 采用 ANSI C 完成编译与匹配过程。

表 1 NFA 转 DFA 的状态数与时间

Tab. 1 State number and time of transfer from NFA to DFA

协议模式	Thompson NFA 构造算法		ECNFA 构造算法	
	NFA 状态数	NFA 转 DFA 时间	NFA 状态数	NFA 转 DFA 时间
HTTP	643	19. 23s	75	0 22s
DNS	679	124. 72s	24	0 11s
QQ	15	0. 01s	7	< 0. 01s
Telnet	31	0. 02s	9	< 0. 01s
Edonkey	517	95. 1s	29	0 83s
SMTP	251	0. 29s	21	< 0. 01s
合成 6 种协议	2136	1738. 5s	155	201. 67

ECNFA 使编译过程显著加速, 尤其对于选择性子表达式较多, 且选择性子表达式连续的模式, 如 DNS 和 Edonkey, 其编译性能提高了 1000 倍以上, 对于合成 6 个协议的状态机, 其性能也提高了 7 倍以上。采用 ECNFA 构造的多数单模式, 其所生成的 NFA 已接近 DFA, 因此 NFA 转 DFA 时间明显减少。

在匹配性能测试中, 报文不直接通过网络捕获, 而是采用两组静态数据: 第 1 组数据来自 DARPA (2000 年 DMZ 数据, 96MB, 共 304 089 个报文)。测试结果如表 2 所示。从实验结果中发现, L7-Filter 识别 DNS、QQ 等包含通配元字符较多的协议模式性能较差, 这是由其 NFA 匹配引擎决定的。L7-Filter 扫描的性能不稳定, 不同模式的识别性能相差很大。MPDFA One-Pass-Scan 性能较 L7-Filter 的匹配方式有显著提高, 单个协议的匹配性能提高至少 1 倍以上, 而针对多个协议的识别, MPDFA One-Pass-Scan 的性能是 L7-Filter 的 20 倍以上。另外, One-Pass-Scan 的性能相对稳定, 各种不同协议之间相差不大, 合成协议的 MPDFA 性能比单个协议性能更高。而 L7-Filter 针对多个协议识别时, 性能显著降低, 6 个协议时, 处理能力降低到接近 10MB/s。

表 2 匹配性能比较

Tab. 2 Matching performance comparison

协议	L7-Filter		MPDFA One-Pass-Scan	
	匹配时间(s)	吞吐量(MB/s)	匹配时间(s)	吞吐量(MB/s)
HTTP	7.95	96.60	3.98	192.96
DNS	26.61	28.89	3.56	215.73
QQ	26.78	28.67	4.91	156.41
Telnet	7.83	98.08	4.92	156.09
Edonkey	6.48	118.51	3.74	205.34
SMTP	7.92	96.96	3.61	212.74
合成 6 种协议	75.65	10.15	3.14	244.58

5 结束语

本文首先提出了一种多模式组合的 DFA(MPDFA),对 MPDFA 与 DFA 的关系进行了探讨。针对协议识别的模式,结合构造 MPDFA 过程中的性能问题,提出了一种压缩 ε 的 NFA 构造算法,能够降低 NFA 转 DFA 的复杂性。最后,为加快匹配过程,提出了一种 One-Pass 的报文扫描算法。对系列方法所实现的协议识别软件进行测试表明,组合多模式的编译性能提高了 7 倍,匹配性能提高了 20 倍。相关方法与算法可用于 IDS、病毒过滤软件等。

下一步,将研究软硬件结合协议识别系统,软件完成 MPDFA 的构建,定制硬件完成流管理与匹配过程。MPDFA 存在状态爆炸问题,因此本文中未对包含所有协议的编译与匹配进行测试,下一步还将研究可行的状态表压缩技术。

参 考 文 献:

[1] Application Layer Packet Classifier for Linux [Z]. http://www.ipp2p.org.

[2] IPP2P[Z]. http://www.ipp2p.org.

[3] Snort Network Intrusion Detection System [Z]. http://www.snort.org.

[4] Bro Intrusion Detection System [Z]. http://bro-ids.org/Overview.

[5] Hopcroft J E. Ullman J D. Introduction to Automata Theory, Languages, and Computation (Second Edition) [M]. Boston: Addison-wesley, 2002.

[6] Yu F, Chen Z F, Diao Y L. Fast and Memory-efficient Regular Expression Matching for Deep Packet Inspection[R]. Berkeley: University of California, 2006.

[7] Kumar S, Dharmapurikar S, Yu F. Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection [C]//SIGCOMM'06, 2006: 11-15.

[8] Brodie B, Cytron R, Taylor D. A Scalable Architecture for High-throughput Regular-expression Pattern Matching[C]//ISCA'06 34(2), 2006: 137-145.

[9] Sidhu R, Prasanna V K. Fast Regular Expression Matching Using FPGAs [C]//Proc. of the IEEE Symp. on Field-programmable Custom Computing Machines, 2001: 227-238.

[10] Thompson K. Regular Expression Search Algorithm [C]//Communications of the ACM, 1968, 11(6): 410-422.