Efficient Regular Expression Grouping Algorithm Based on Label Propagation

Xi Chen[a], Shuqiao Chen[b] and Ming Mao[c]

National Digital Switching System Engineering & Technology Research Center, Zhengzhou, China

[a]1157123878@qq.com, [b]bfw0901@163.com, [c]517282681@qq.com

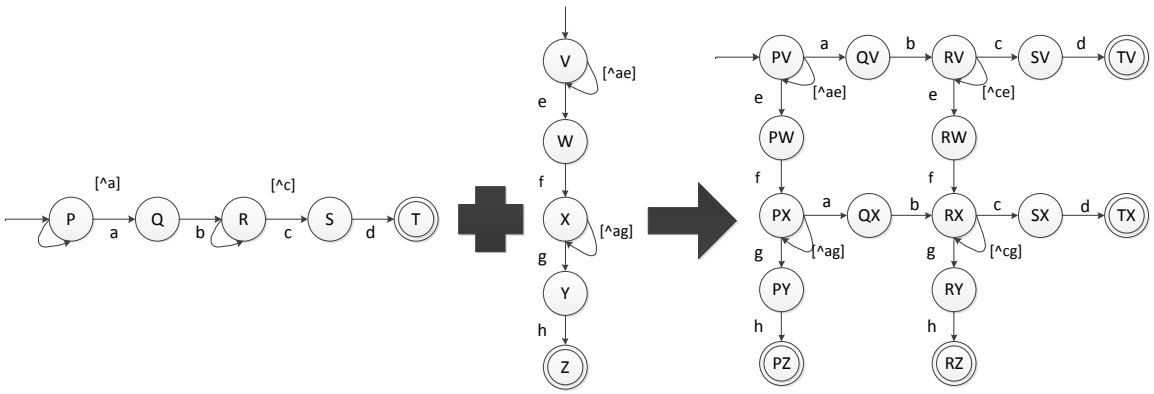
Keywords: regular expression, deep packet inspection, grouping algorithm, label propagation.

Abstract. Regular expression grouping is the practical way to address the state explosion problem of Deterministic Finite Automaton (DFA). Previous grouping algorithms have poor grouping time, and difficult to meet application needs. In this work, we present an efficient regular expression grouping algorithm based on label propagation. Through the deterministic initial grouping and based on similarity of the propagation process to achieve faster convergence. Experimental results show that. Compared with other algorithms, GBLP algorithm has the minimum total number of states and grouping time for the same number of groups.

Introduction.

With the increasing network security threats, Deep Packet Inspection (DPI) is becoming more and more important in security application and research. It describes the threats (e.g. spam, virus or anonymous intrusion) by default description of the signatures, to identify the network flows with malicious information in the payload of their packets. Nowadays, regular expressions have gradually become the main description language of DPI signatures owing to its expressiveness and flexibility.

Regular expression is usually achieved by Finite Automata (FA). The Deterministic Finite Automata (DFA) with its linear and predictable matching speed becomes the research hotspot. However, the DFA may require a huge amount of memory become of the state explosion problem. The number of DFA states can be exponential in size of regular expression as shown in Fig.1. So many algorithms for the DFA storage optimization have been proposed.



ab.*cd ef.*gh /ab.*cd/ and /ef.*gh/

Fig.1 Example of state explosion

But in the current era of big dada, the scale and complexity of the DPI rule set is also increasing. The problem of the DFA state explosion may cause it difficult to generate. The rule set of L7-Filter is compiled to DFA which needs more than 16GB of memory. The general machine is difficult to achieve. At this case, most of the storage optimization algorithms (D^2FA, &FA, etc.) cannot be applied. But grouping algorithm can effectively avoid inflation between regular expressions.

However, as the demand for the real-time detection under the high-speed network environment is gradually prominent. The grouping time of the existing regular expression grouping algorithm is

longer and it is difficult to be applied in practice. Therefore, in this work, an efficient regular expression grouping algorithm based on label propagation (GBLP) is designed. It can achieve the effective balance between grouping time and grouping effect.

Finally, we implement the GBLP algorithm and the other 3 kinds of grouping algorithms in L7-Filter, Snort and Cisco security application. The experimental results based on practical rule sets show that under the same number of clusters, the GBLP algorithm has improved the total states number and grouping time, especially on grouping time reduced from 2 to 5 times.

Related Work

DFA main memory consumption is an $n\times\sum$ (n is the number of state, $\sum$ is alphabet of ASCII) of the State Transition Table (STT). It occupies more than 95% of the DFA storage space. Therefore, the method of solving the DFA memory can be divided into two directions: reduced state transition edge [1-4] and reduced state number [5-10]. Each of these States is corresponding to the 256 transition edges, so the compression of the edge is limited. The reduction in the number of states can avoid the space explosion by the nature. In this work, we choose to design a fast and efficient algorithm to reduce the number of states.

F. Yu et al. [5] first proposed regular expression grouping algorithm, called mDFA. However, the algorithm only uses the mutual relationship between the regular expressions by 0 and 1. Grouping strategy is too simple. Becchi et al. [6] used the grouping method of dichotomy to achieve faster grouping time, but the partial grouping effect is sacrificed. Q. Xu et al. [7] defined the Dilate Ratio (DR) to describe the expansion effect of regular expressions, and put forward a kind of selective clustering algorithm REGADG to improve the grouping effect. But the algorithm may lose information. Rohrer [8] and Fu Z [9] abstracted the grouping problem, and combined with the existing optimization algorithm (integer linear programming, simulated annealing, genetic algorithm and ant colony optimization algorithm) to achieve better effect of grouping. But even though Fu Z designed a DFA estimation algorithm to accelerate the execution speed, grouping time is still higher than the traditional mDFA. Liu Tingwen [10] proved that the optimum k-grouping problem of regular expression set was NP-hard, and the local optimization algorithm GRELS was designed. But the grouping time has not been improved significantly.

Obviously, the previous regular expression grouping algorithm has gradually improved the grouping effect, less memory footprint. However, with the increasing demand for network security and service quality, the importance of data real-time detection is also more prominent. And the grouping time of the grouping algorithm is closely related to detection efficiency. At the same time, the analysis found that the previous grouping methods usually adopt the method of random traverse and multiple iterations. And no deterministic initial group was given.

GBLP Algorithm.

The regular expression grouping is to divide the regular expression set into multiple sub sets. The goal is to minimize the DFA expansion states number. The label propagation algorithm is simple and easy to implement. The implementation time is short, especially suitable for large scale dense graph grouping. Therefore, this work will reference the idea of label propagation. We optimize the distribution of the initial label and the transmission process. Finally, the reduction of grouping time is realized in the premise of effective grouping.

In practical application, the parallel processing capability of hardware platform will limit the maximum group number. In general, the number of groups is determined by the maximum parallel processing capability of the device. So we set the known number of clusters K, minimizing the total number of states.

Introduction to Label Propagation Algorithm.

Label propagation algorithm [11] (LPA) is proposed by Zhu et al. in 2002. Its basic idea is to construct a sample connection diagram, which gives a different initial label to each vertex. Then according to the principle of voting, the label that has the largest number of tags in the adjacent

vertices is transmitted to the vertex. Finally, the same label vertices are divided into a group.

Label propagation algorithm including initialization, label update, and add the same tag vertex to the same group three parts. Each vertex is assigned a unique label. And the optimal group is achieved after multiple iterations. But in the process of transmission, a large number of small groups can be generated. Many meaningful labels are fuzzy. At the same time, the communication effect based on the principle of simple voting is poor. And simple voting principle is not suitable for complete graph of connected to each other.

Therefore, label propagation algorithm cannot be directly applied to the regular expression grouping. We design a fast and efficient regular expression grouping algorithm by improving it. Mainly includes two steps: deterministic initial group selection and label propagation based on similarity.

Design and Implementation of GBLP.

In view of a regular expression set R= {r_1, r_2, ..., r_n}, we construct a undirected complete graph G= (V, E). Which V= {v_1, v_2, ..., v_n} for vertex set, where each vertex v_i represents a regular expression r_i. Every pair of vertices v_i and v_j has an edge weight e_{ij}. In order to describe the relationship between the regular expressions, the concept of the Expansion Rate (ER) is introduced as the edge weight.

Definition 1: For any regular expression r_i and r_j, ER formula is as follows:

$$ER(r_i, r_j) = \frac{T(r_i, r_j)}{T(r_i) + T(r_j)} \quad (1)$$

$T(r_i)$ and $T(r_j)$ respectively corresponding to the regular expression r_i and r_j compiled into state of DFA. $T(r_i, r_j)$ said the regular expression r_i and r_j combined generated DFA state number. When ER > 1, the corresponding regular expression is merged and the expansion is generated. The greater the ER is, the more serious the state expands. The corresponding regular expressions should be divided into different groups. When ER is less than or equal to 1, said the corresponding two regular expressions do not interact. ER is close to 0, the corresponding regular expressions are more likely to be classified into the same group.

A. Initial Group Selection.

The original label propagation algorithm assigns each vertex a unique label, which leads to many small groups in the iterative process. The meaningful label cannot be emerged a large number at first. This reduces the convergent speed and accuracy limit problems. In the case of the known packet number K, the "critical" K vertices may be selected as the initial group and the initial tag is given. This can effectively reduce the number of labels, speed up the convergence rate, and improve the accuracy of the group.

The selection of "critical" K vertices, namely choose K and all the vertices do not produce or produce the expansion of the state of the vertices as little as possible. The concept of Peak Intensity (PI) is introduced in this paper to describe the expansion effect of the regular expression in the vertices.

Definition 2: For regular expression r_i corresponding vertex v_i, PI formula is as follows:

$$PI(v_i) = \sum_{j=1, i \neq j}^{j=n} ER(r_i, r_j) \quad (2)$$

PI is the weight of all edges on the corresponding vertices. The greater the PI indicates that the vertex and the rest of the merger have a larger expansion rate. The vertex corresponds to a regular expression is more complex. On the contrary, the smaller PI vertex corresponds to a regular expression is more simple, the smaller the state expansion caused by. According to the above analysis, we first calculate the PI of n vertices and prioritize. Then, according to the PI value, the minimum K vertices are selected as the initial group G_i and assigned the initial labels.

B. Label Propagation Grouping Process.

In this process, the traditional labeling algorithm is based on the principle of voting to select the

common label. In order to avoid the label shocks, usually adopt the strategy of asynchronous to update the label. The updating formula is as follows:

$$C_{vi}(t) = f(C_{vi1}(t),\ldots,C_{vi(i-1)}(t),C_{vi(i+1)}(t-1),\ldots,C_{vin}(t-1)) \quad (3)$$

Among them, v_{i1}, …, v_{in} is the vertex v_i of all the adjacent vertices, v_{i1}, …, $v_{i\,(i-1)}$ indicates that the vertex of the label has been updated before this iteration, $v_{i\,(i+1)}$, …, v_{in} said the iteration of the label has not updated the vertex, *f* function as described above. But it cannot be directly used in here. Each vertex is connected to each other in a complete graph G. And we can achieve the minimum of the expansion state for the purpose of grouping. So we need to modify it. The concept of vertex Similarity (S) is introduced.

Definition 3: For vertex v_i and v_j, S formula is as follows:

$$S(v_i,v_j) = \frac{1}{ER(r_i,r_j)} \quad (4)$$

The S (v_i, v_j) between v_i and v_j is the reciprocal of ER (r_i, r_j). Obviously, the greater the similarity between vertices, the corresponding two vertices should be for the same group. So in order to achieve the least number of expansion state. We design a method of label propagation based on similarity, label propagation formula is:

$$C_{vi}(t) = g(C_{vi1}(t),\ldots,C_{vi(i-1)}(t),C_{vi(i+1)}(t-1),\ldots,C_{vin}(t-1)) \quad (5)$$

In the above formula, *g* returns the sum of the similarity and the maximum of the neighbor vertices in the vertex v_i. If the sum of similarity is equal, then randomly choose one. The propagation method can effectively limit the spread of the label randomness, and improve the convergence precision.

C. Algorithm Implementation.

Based on the above two steps, the final group according to the label information on each vertex. Label propagation grouping algorithm steps are as follows:

Step1: Calculate the PI of each vertex in the complete graph G. Since the childhood prioritize put in X;

Step2: Calculate the similarity between arbitrary vertices;

Step3: According to the number of clusters K, PI's smallest K vertices are selected, and the initial label is assigned;

Step4: Set up t =1, update label, for($v_i \in V$),

$C_{vi}(t) = g(C_{vi1}(t),\ldots,C_{vi(i-1)}(t),C_{vi(i+1)}(t-1),\ldots,C_{vin}(t-1))$;

Step5: If all vertices are assigned labels and not change, the algorithm ends, or set t =t+1, to **Step4.**

Algorithm Analysis.

All kinds of regular expression grouping algorithms need to calculate expansion relationship between the rules. This step can be considered as the preprocessing stage of the regular expression grouping, and the time complexity is $O(n^2)$ (n number of rules in regular expression set R). So we neglect this step in the process of analysis. The time complexity of the grouping process is considered only. We put forward an effective regular expression grouping algorithm based on label propagation. The initial grouping and the label propagation based on similarity are linear. The time complexity of the grouping process is O(n). For the previous grouping algorithm, the expansion relationship needs to be calculated in each iteration process. The time complexity is generally $O(n^2)$. Therefore the time complexity of GBLP is especially improved.

Experiment and Analysis.

In this section, we conducted a series of experiments to evaluate the performance of GBLP algorithm. All experiments were performed on Virtual Machine 8.0. The virtual machine

configuration is as follows: operation system: Ubuntu 11.10, internal memory: 8 GB, Intel dual core 2.5 GHz CPU. The Regular Expression Processor [6] is used to calculate the accurate number of states of a regular expressions distribution, as the metric of memory consumption. Three rule sets picked from L7-Filter、Snort and commercial corporation Cisco are used, as shown in Table 1.

Table 1. RULE SETS INFORMATION

Rule sets	# of RegEx	% RegEx w/ wildcards (*，+)
Snort-158	158	84.25
L7-Filter-80	80	65.33
Cisco-233	233	13.65

We generated mDFA [5], Becchi [6], GRELS [10] and GBLP respectively with different rule sets in Table 1. We compared them on the total states number of the DFAs constructed from groups and grouping time.

Table 2. GROUPING RESULTS FOR Snort-158

Grouping algorithm	K=5		K=6		K=7	
	states number	Grouping time(s)	states number	Grouping time (s)	states number	Grouping time (s)
mDFA	13637	1030	8420	418	6157	279
Becchi	43695	775	32316	403	17040	143
GRELS	7677	938	5736	441	4672	244
GBLP	7374	473	5236	286	3681	119

Table 3. GROUPING RESULTS FOR L7-Filter-80

Grouping algorithm	K=5		K=6		K=7	
	states number	Grouping time (s)	states number	Grouping time (s)	states number	Grouping time (s)
mDFA	29256	2308	20742	1177	16119	986
Becchi	216093	1953	180702	1035	130277	541
GRELS	21072	2139	15519	1009	10052	717
GBLP	20153	718	15051	491	9334	318

Table 4. GROUPING RESULTS FOR Cisco-233

Grouping algorithm	K=5		K=6		K=7	
	states number	Grouping time (s)	states number	Grouping time (s)	states number	Grouping time (s)
mDFA	20839	3059	18005	2611	15971	1540
Becchi	22205	1762	17001	1127	16893	774
GRELS	17877	1538	14136	714	10383	663
GBLP	15157	495	12355	367	9003	189

As shown in Table 2. On Snort-158 rule set, under the same group number, the states number of mDFA algorithm is less than that of the Becchi algorithm, but the grouping time is the longest. The GRELS algorithm of local optimization on the total number of state has obviously improved than the above two groups, but the grouping time is still longer. While GBLP algorithm proposed in this work is the smallest in two indicators. The total number of states is reduced by about 10% compared with the GRELS algorithm. The grouping time is reduced by more than 47% of the Becchi algorithm.

Table 3 shows both effectiveness and efficiency for L7-Filter-80 rule set. Although the regular expression of the rule set is more complex, the state expansion is more serious. GBLP algorithm is compared with the current GRELS algorithm, on the basis of the total number of State to be reduced, the grouping time is reduced by more than half.

As shown in Table 4, under the same group number, the state number of the existing grouping algorithm is far greater than the GBLP algorithm. GBLP algorithm runs 5 times and 3 times faster

than Becchi algorithm and GRELS algorithm, respectively.

By the above analysis, in the same number of clusters, the states number of GBLP algorithm and the grouping time are improved compared with the previous grouping algorithm. In similar about the complex L7-Filter、Snort rule sets, GBLP algorithm can reduce the number of states slightly, and grouping time is reduced by about half. While in the Cisco security applications and other simple rule sets, the algorithm performance is more prominent. Compared with the latest GRELS algorithm, the grouping time decreases by 3 times.

Conclusions

In view of the current high-speed network environment, the grouping time of the existing regular expression grouping algorithms is longer. And they are difficult to meet the needs of real-time detection in practical application. We propose a new fast and efficient grouping algorithm, which is efficient regular expression grouping algorithm based on label propagation (GBLP). Learn from simple and quick label propagation ideas. We give a method to determine the initial groups, and to design the grouping process based on the similarity. The experimental results show that the proposed GBLP algorithm has less number of states and shorter grouping time than the current grouping algorithms, which can achieve the effective balance between memory occupancy and grouping time. The future work is to consider the expansion influence of different characteristics, and further improve the grouping accuracy of the algorithm.

Literature References

[1] Kumar S, Dharmapurikar S, Yu F, et al. Algorithms to accelerate multiple regular expressions matching for deep packet inspection[J]. ACM SIGCOMM Computer Communication Review, 2006, 36(4): 339-350.

[2] Becchi M, Cadambi S. Memory-efficient regular expression search using state merging[A]. 26th IEEE International Conference on Computer Communications (INFOCOM 2007)[C]. IEEE, 2007: 1064-1072.

[3] QI Y, Wang K, Fong J, et al. FEACAN: Front-end acceleration for content-aware network processing[A]. 2011 Proceedings IEEE INFOCOM[C]. IEEE, 2011:2114-2122.

[4] Becchi M, Crowley P. A hybrid finite automaton for practical deep packet inspection[A]. Proceedings of the 2007 ACM CoNEXT conference[C]. ACM, 2007:1-12.

[5] Yu F, Chen Z, Diao Y, et al. Fast and Memory-efficient Regular Expression Matching for Deep Packet Inspection[C]. ANCS ’06. New York, NY, USA: ACM, 2006:93–102.

[6] Becchi M, Franklin M, Crowley P. A workload for evaluating deep packet inspection architectures[C]. In: Proc. of the IISWC 2008. Piscataway: IEEE Computer Society. IEEE ,2008:79− 89.

[7] Xu Qian, E Yue-peng, Ge Jing-guo, et al. Efficient regular expression compression algorithm for deep packet inspection[J]. Journal of Software, 2009,20(8):2214−2226.

[8] Rohrer J, Atasu K, Lunteren JV, et al. Memory-Efficient distribution of regular expressions for fast deep packet inspection[C]. CODES+ISSS 2009 Proceedings, Grenoble, France, Oct 11-16, 2009:147 − 154.

[9] Fu Z, Wang K, Cai L, et al. Intelligent grouping algorithms for regular expressions in deep inspection[C]. Computer Communication and Networks (ICCCN), 2014 23rd International Conference on. IEEE, 2014: 1-8.

[10] Liu Ting-wen, Sun Yong, Bu Dong-bo, et al. 1/(1-1/k)-Optimal algorithm for regular expression grouping[J]. Journal of Software, 2012,23(9):2261-2272.

[11] Zhu X, GHAHRAMANI Z. Learning from labeled and unlabeled data with label propagation[R]. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.