An Improved String Matching Algorithm for HTTP Data Reduction

Lei Zhang, Yong Peng, Jie Liang, Xifeng Liu, Jin Yi, Zhe Wen
China Information Technology Security Evaluation Center,
Beijing 100085, China
{zhangl, pengy, liangj, liuxf, yij}@itsec.gov.cn

***Abstract*—In this paper, we analyze both BMH algorithm and QS algorithm, focus on the improvement of QS, and present FQS algorithm which can apply to HTTP data reduction. For building this algorithm, we considered character features of pattern strings in HTTP application data, improved the collating sequence of characters in matching process, and had the bad characters jumping strategy improved for increasing the jumping distance. Experimental results show that FQS algorithm effectively reduces matching time and a better time performance.**

Keywords- HTTP protocol; pattern matching; FQS algorithm;

I. INTRODUCTION

With the rapid development of Internet technology, network security issues become increasingly prominent. Effective detection and monitoring of information has become an important issue. The HTTP protocol is one of the most widely used and important communication protocols in the Internet. So real-time monitoring and reduction over its communication data is very necessary. When restoring the HTTP packet data, we mainly use string matching techniques to extract useful information. At this point, the pattern matching algorithm is an important method for string matching. In order to identify and restore the content more quickly and accurately, the matching algorithm and its matching efficiency need to be improved.

In this paper, we first analyze the advantages and the disadvantages for both the typical single pattern matching algorithms, and then improve the algorithms mainly on the basis of QS [1] with taking into account the characteristics of the pattern string characters in HTTP data. We propose an improved algorithm named Fast Quick Search algorithm which is referred to as FQS for short.

II. TWO CLASSICAL ALGORITHMS

In order to explain the algorithm, we make the following definition: For a text string $T = T_1T_2...T_n$ with length n, and a pattern string $P = P_1P_2...P_m$ with length m (generally $n \geq m$), we say that the match is good when the equation $T_iT_{i+1}...T_{i+m}T_{i+m-1} = P_1P_2...P_{m-1}P_m$ ($1 \leq i \leq n-m+1$) exists with returning the position of the pattern string from the text string, otherwise we say that the matching fails.

A. BMH Algorithm

BMH algorithm[2] is an improved algorithm which is presented by Horspool on the basis of BM algorithm[3]. BMH algorithm only uses bad character rule in the preprocessing stage, and it simplifies the matching process. When a character mismatches, the algorithm only considers the last character of the text string in the current window for calculating the right offset distance.

The idea for matching from BMH algorithm is: left-align the pattern string and the text string, first match the last character in the current window, if the matching is successful, match the rest of the $m-1$ characters in order; when a character in the text string mismatches, the last character of the text string in the current window inspires the pattern string to move rightward. Misfit character table during the matching process is as the following formula:

$$BMH\_J(c) = \begin{cases} m; c \neq P_j (1 \leq j < m) \\ m - \max(k); \{k \mid P_k = c, 1 \leq k < m\} \end{cases} \quad (1)$$

BMH algorithm simplifies the initialized process, and the efficiency is better than the BM algorithm in actual using.

B. QS algorithm

QS algorithm is first proposed by Daniel M. Sunday in 1990, the algorithm only uses bad character rule. The pattern string left-aligned with the text string when matching happens, the characters of the text string and the pattern string can be compared either from left to right or from right to left.

The idea of QS is as follows. When a mismatch occurs, the next character of the text string in this current matching window will be considered to determine the jumping distance, and the pattern string moves rightward according to distance, then create a new window to continue the next round for matching. The misfit character table in the matching process shows as the following formula:

$$QS\_J(c) = \begin{cases} m; c \neq P_j (1 \leq j \leq m) \\ m - \max(k) + 1; \{k \mid P_k = c, 1 \leq k \leq m\} \end{cases} \quad (2)$$

Comparing with BMH algorithm, QS reduces the comparison times and increases the jumping distance. We can say that QS algorithm is more efficient.

C. Proposal of an Improved Algorithm

The analysis for both the algorithms shows that the maximum moving distance of QS algorithm can reach $m+1$, which is larger than BMH algorithm. Therefore, QS

978-1-5090-0188-0/15 $31.00 © 2015 IEEE
DOI 10.1109/IIH-MSP.2015.18

has the priority in consideration of using a single pattern matching algorithm to restore the data from HTTP protocol.

However, QS algorithm uses only one character to calculate the right offset distance, which will increase some unnecessary cost for character comparison and moving. In addition, QS algorithm is not as efficient as BMH algorithm in some cases [4]. In the end, appropriate character comparing sequence can improve matching efficiency to some extent. As suggested above, we propose FQS algorithm which is an improved single pattern matching algorithm for QS algorithm.

III. IMPLEMENT OF FQS ALGORITHM

A. The Improvement of QS Algorithm

Through the above analysis, we find that QS algorithm has a lot of space for increasing jumping distance and reducing character comparison times .There are mainly three aspects for improving the algorithm.

1) QS algorithm uses only one character to calculate the right offset distance, which will increase some unnecessary characters comparison and moving. Many improved algorithms have been raised aiming at this problem .Among which, Sunday2 algorithm proposed by Zeng uses window for text strings slicing, so that the maximum right offset distance of pattern string increases to $2m+1$ from $m+1$, which effectively reduces characters matching times and improves the performance of the algorithm[5]. However, there is also a limitation of Sunday2 algorithm, for it uses two bad character functions in the preprocessing stage, that is, to establish two pre-treatment tables, which increases the computational overhead.

FQS algorithm makes such improvement. It only uses bad character rule of QS algorithm in the preprocessing stage, and improves the matching process (detailed in 2.2). The maximum right offset distance of FQS algorithm can still reach to $2m+1$, as shown in Figure 1:

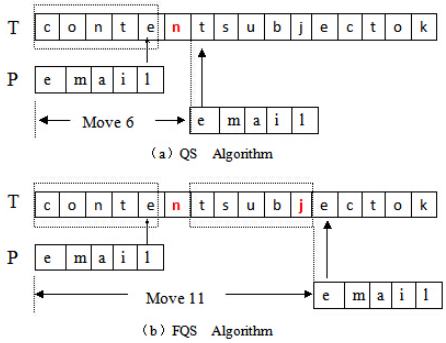

Figure 1. The comparison of maximal right shift

As we can see from Figure 1, when the character mismatch occurs, QS algorithm makes the pattern string have a right offset distance 6, while FQS algorithm moves the string pattern rightward by 11. So it is obvious that the latter algorithm makes the pattern string move further, which reduces the characters matching times.

2) Consider the case in which the last character T_{i+m-1} of the text string in the current window does not appear in the pattern string but appears in T_{i+m}. As shown in Figure 2, where the text string is 'contenttitle' and the pattern string is 'match', and both the algorithms start to match from the same position.

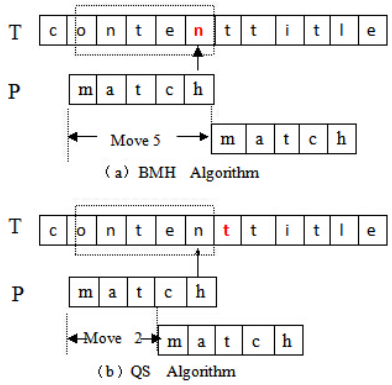

Figure 2. the contrast between BMH and QS

As shown in Figure 2, when mismatch occurs, the right offset distance is 2 by taking QS algorithm, and the offset distance is 5 by taking BMH. So we can see that the efficiency of QS is inferior to BMH in this case.

The improvements of FQS algorithm makes: When a character mismatch occurs, if the next character T_{i+m} of the last character in the current window is in the pattern string, continue to check whether the last character T_{i+m-1} in the current window is in the pattern string. If not, increase the right offset distance to m, otherwise, disregard it.

3) Design an appropriate character comparing sequence aiming at the features of the pattern string in HTTP application data restore, which can effectively reduce the number of character comparisons and improve the matching efficiency of the algorithm.

This paper sorts out some pattern strings which are useful when restoring HTTP protocol, such as title, email, content, message, username, nickname, subject and so on. The characteristics of the pattern strings are as follows: the length of these strings are generally short, and most of them are common words or very similar to common words, the suffixes are quite common and there are a lot of words which have the same suffixes as these pattern strings[6].

Because QS algorithm compares the character in right-to-left order, if a lot of strings with the same suffixes with the pattern strings exist in a text string, then the match will be the case as shown in Figure 3: that it matches several characters successfully continuously near the last character T_{i+m-1} of the pattern string until matching characters near the first character T_i of the pattern string, which obviously adds a lot of unnecessary character comparisons, so there is room for improvement for QS algorithm in this application background.

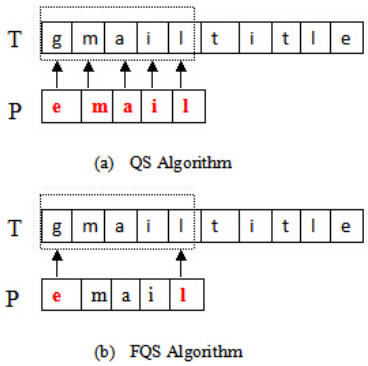

Figure 3. the comparison of matching sequence

FQS algorithm makes improvements in this way: It makes improvements basing on that the QS algorithm compares characters from right to left. In the matching process, match the last character of the text string in current window first, if matches, no further match the character before the last character, but to match the first character of the text string in the current window. And then match the rest of the characters one by one on condition that the first character and the last character is a successful match.

B. Implementation of the FQS algorithm

The implementation of FQS algorithm is divided into two parts, including preprocessing and matching process. In the preprocessing, FQS is the same as QS algorithm, it uses the bad character table function to calculate the offset of each character in the text string. While in the matching process, FQS algorithm's matching steps can be expressed as follows:

a) Let the pattern string P and text strings T to be left-aligned, forming the current window.

b) Start the matching from the last character of the pattern string in the current window, if succeed, match the first character of the pattern string, when both of them match, then match the rest characters. If all the characters in this current window match successfully or mismatching occurs, terminate this round of matching and calculate the right offset distance of the pattern string.

c) If the right offset distance is not $m+1$, then continue to determine whether the T_{i+m-1} is in the pattern string, if not, increase the right offset distance to m, otherwise keep its value.

d) If the right offset distance is $m+1$, then judge whether the T_{i+2m} in the pattern string, if not, increase the right offset distance to $2m+1$,otherwise keep its value.

e) According to the right offset distance, the pattern string moves its corresponding distance, forming a new matching window, then start the next round of the matching. If the pattern string and text string are matched, return the position of matching.

f) When the text string's pointer beyond the length of the text string, this matching process is completed.

The whole process is shown in figure 4.

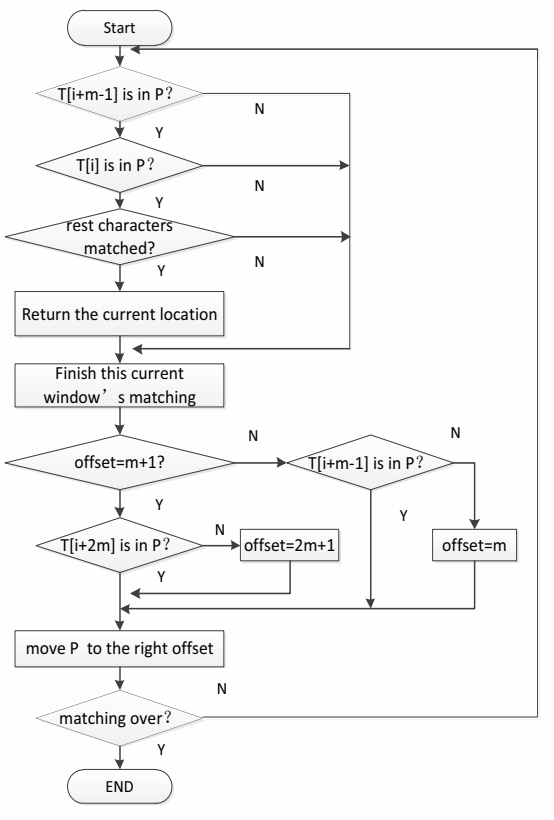

Figure 4. the flow chart of CIQS matching process

C. Performance Analysis of FQS

In order to analyze the advantages of the improved algorithm, an example will be given to illustrate the matching process of FQS algorithm. Suppose the text string is 'catchpostteachmatch', and the pattern string is 'match', so the matching process can be shown in TABLE 1:

TABLE I. MATCHING PROCESS OF FQS ALGORITHM

	c	a	t	c	h	p	o	s	t	t	e	a	c	h	m	a	t	c	h
1	**m**	.	.	.	**h**														
2															.	.	.	.	**h**
3															**m**	**a**	**t**	**c**	**h**

From the Table 1, we can see that FQS algorithm significantly reduced the number of comparisons when the text strings have many same character suffix compared with pattern string (Here is 'atch'). And FQS algorithm can increase the right offset distance to $2m+1$ in some cases, it increases the moving distance.

IV. EXPERIMENTS AND RESULTS

From the above analysis, FQS algorithm has better performance in theory. In order to test the real efficiency of FQS algorithm, it will be validated by experiments.

The experiment is performed on a virtual machine, the experiment environment is as follows: Physical system with Windows 7, Intel(R) Core(TM) 2 T5750, 2.10GHZ，2GB Memory. The virtual machine software is VMware Workstation 8, its operating system is CentOS5.2 with 1GB Memory, C++ programming.

First we collect some communication packets from browsing some forums, web searching, sending and receiving e-mails with data-capturing software, Randomly get some HTTP application layer data which has been restructured as the text string sample, then store them into a txt file with the size around 7.6 MB. These pattern strings randomly selected from the txt file, the length of them are 3, 4, 5, 6, 7, 8.

For each pattern string, test with BMH algorithm, QS algorithm and FQS algorithm to match the text strings, count the number of successful matching times and the time cost for each of the matching processes.

Through the experiments, we found that BMH algorithm, QS algorithm and FQS algorithm can reach the same successful matching times. It shows that the FQS algorithm has the same reliability on the accuracy of matching.

Compare the matching time of this three algorithms by repeating the matching process ten times for each pattern string, and get the statistical results after averaging as shown in Figure 5.

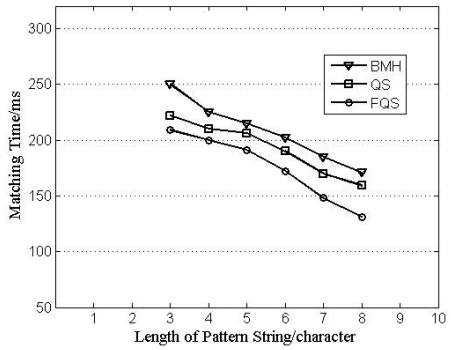

Figure 5. matching time of different algorithms

As shown in Figure 5, we can see that these algorithms' matching time are all gradually decreasing as the pattern string's length increases. The reason is that the maximum safety jumping distance is concerned with the pattern string's length, the larger the pattern string's length is, the greater the jumping distance will be, which comes out the shorter matching time.

In addition, we compared the matching time of these algorithms with the same pattern string, we can conclude that FQS algorithm generally takes the least matching time. When the length of the pattern string increases, FQS shows better time performance than QS algorithm, because the probability of the occurrence that the same string suffix appears both in text string and pattern string increases, so the advantage of the improved algorithm will be reflected.

The experiments show that FQS algorithm's jumping distance is more safety and it also takes fewer character matching times than QS algorithm.

V. CONCLUSION

This paper presents an improved algorithm after the research on BMH algorithm and QS algorithm, we proved the feasibility of FQS algorithm in theory, and then proved the good performance in the HTTP protocol reduction. However, with the development of network technology, the pattern matching algorithm is gradually designed to match the development of multi-pattern in the network information reduction, thus how to improve the FQS algorithm from single pattern matching to multi-pattern matching is the future work.

References：

[1] SUNDAY Daniel M.A Very Fast Substring Search Algorit-hm [J].Communications of the ACM, 1990, 33(3): pp.132-142.

[2] HORSPOOL R Nigel.Practical Fast Searching in Strings [J] .Software Practice and Experience, 1980, 10(6): pp. 501-506.

[3] BOYER Robert Stephen,MOORE J Strother . A Fast String Searching Algorithm [J]. Communications of the ACM, 1977, 20: pp.762-772.

[4] ZHANG Yu-xin,LI Cheng-hai,BAI Rui-yang. Improved single pattern matching algorithm [J]. Manufacturing Automation, 2012, 34(1): pp.208-212.

[5] ZENG Chuan-huang,DUAN Zhi-hong. Design and Realization of the Improved Sunday Pattern Matching Algorithm [J].Journal of Harbin University of Science and Technology, 2011, 32(3): pp.22-25.

[6] CHEN Jie.A New Algorithm for Pattern Match Based on BMH Algorithm[EB/OL].Beijing: Sciencepaper Online, 2011-08-03[2014-11-08]. http://www.paper.edu.cn/releasepaper/content/201108-50.