Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2013, 5(9):515-521

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

The application of a kind of duplex layer KMP algorithm within flow filtering technology

Lei Li[1], Yuwan Gu[1,2], Yan Chen[1] and Yuqiang Sun[1*]

[1]*ChangZhou University International Institute of Ubiquitous Computing, Jiangsu, Changzhou, China*
[2]*JiangSu University College of Electronic and Information Engineering, Jiang Su, Zhenjiang, China*

ABSTRACT

This paper analyses the theory of KMP algorithm, using the irrelevant mismatching feature of main string and substring, proposed a kind of duplex layer structure hierarchical nesting string matching algorithm, and apply the improved algorithm to flow filtering technology. Under the condition of "large text - big Mode" string matching, the improved algorithm solves the problem of strong dynamic memory share and time-consuming about calculating the next value in the KMP algorithm. The experimental results suggest that the algorithm has higher matching efficiency under the condition of large data sets matching.

Key words: flow filtering, TCP message, KMP algorithm, string matching

---

INTRODUCTION

Flow filtering technology is a new type of firewall architecture, which is a representative technology of deep Packet Inspection proposed by Neusoft. The technology overcomes the defects that the packet filtering cannot cache and restore the application layer data and the application proxy firewall can only provide very limited protection scope of the protocol, and integrate the advantage of packet filtering and application proxy, which not only conduct single check of TCP/IP message, but also cache and reassemble several TCP messages, accordingly achieve the filtering check of the whole data flows. Filtering check is content matching of reassembled message complying with search rules. At present most commonly used matching algorithm is KMP, BM and KR. This paper proposed an improved KMP algorithm, which conduct the filtering string matching to the reconstructed TCP packet, and make use of the master string mismatching irrelevant feature of KMP to construct secondary KR substring , and get the next[j]rapidly, therefore the algorithm can accomplish the matching in shorter time[1,2].

KMP ALGORITHM AND THE IMPROVEMENT STRATEGY

KMP Algorithm

KMP Algorithm is a classical algorithm of matching prefix, which evade the backtrack phenomenon of traditional algorithm. When the comparison of characters are not equal during the matching process, there is no need to backtrack pointers, but exploit part of the matching results to keep the pattern string sliding distance to the right as far as possible, then continue to compare. The time complexity of the algorithm is O(n+m)[3].

The core of the KMP string matching algorithm is the next construction function, whose complexity is O（m）. Assuming that at this moment it need to compare to the (k（k< j）)-th character in the pattern string, the substring of the first k-1characters in this pattern satisfy the relationship as follows:

$$y_1y_2y_{k-1} = x_{i-k+1}x_{i-k+2}x_{i-1} \quad (1)$$

the matched result it has got:

$y_{j-k+1}y_{j-k+2}y_{j-1} = x_{i-k+1}x_{i-k+2}x_{i-1}$ (2)

the following formula can be deduced:

$y_1y_2y_{k-1} = y_{j-k+1}y_{j-k+2}y_{j-1}$ (3)

For next[j]=k, if there are two substrings that satisfied the relationship as above exist in the pattern string. When a mismatch generated during matching process (i.e. xi ≠ yj), the matching is just need to compare the k(k < j)-th character of the patternto the current character of the main string. Therefore, next[j] definition table is as follows:

Table 1.The definition of next[j]function

j	Next[j]
1	0
$1 < k < j$ && $y_1y_2y_{k-1} = y_{j-k+1}y_{j-k+2}y_{j-1}$	max
others	1

For instance, match the main string X = abacabadaeacabae with pattern string Y = abae, according to the matching step of KMP algorithm as in Table 2

Table 2.The matching process of KMP algorithm

Text string:	a b a c a b a d a e a c a b a e
pattern string :1	a b a e
2	a b a e
3	a b a e
4	a b a e
5	a b a e
6	a b a e
7	a b a e

Hierarchical nesting exact string matching algorithm

Basic idea: Hierarchical nesting exact string matching algorithm (Figure 1) is based on KMP algorithm on the premise of large amount of characters of text string and pattern string. When constructing the next function,the primary pattern string can be seen as secondary text string and the substring of calculating the next function can be seen as secondary pattern string as since the construction of next function only related to pattern string. Using the thinking of KR algorithm to construct a kind of uniform Hash function[4]and exploiting this Hash function to transform secondary pattern string whose length is p and every secondary text substring to a couple of unique integer value, then replace the comparison of secondary pattern string and secondary text substring with the comparison of corresponding integer value, and calculate next[j] quickly. The improved algorithm solved the problem of high dynamic memory occupancy and long time-consuming calculation under the condition of "large text - big Mode" string matching, Accordingly, the matching could be accomplished in shorter time.

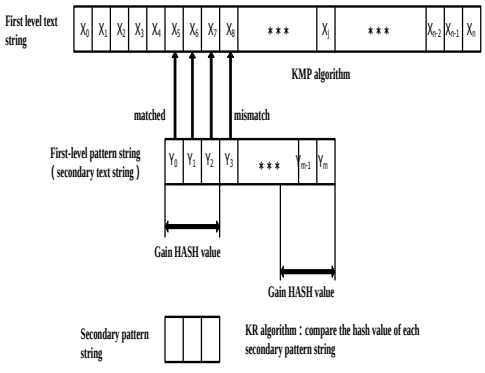

Figure 1. The structure of hierarchical nested exact string matching algorithm

The implementation of algorithm: KMP is a classic algorithm of prefix matching whose core is constructed next function. The traditional next function can be solved by induction method while the improved algorithm define the secondary string and solved by KR algorithm. The formal description of the algorithm is as below:

Algorithm1. Exact string matching with 1 processor.

Input: Text string X[0:n], Pattern string Y[0:m];

Output: The starting position of matching i, 0≤i≤n-m;
Begin
(1) Match the initial index i and j in the definition of text string X[0:n] and pattern string Y[0:m]
(2) Initialize i , j =0;
(3) If $X_i = Y_j$, continue to check whether Xi +1 matches Yj +1;
(4) If $X_i \neq Y_j$, and j=1, executing the matching check of X_{i+1} and Y_j: Conducting the the matching check from beginning after moving the text string and pattern string one character to the right separately,
(5) If $X_i \neq Y_j$, and $1 < j \leq m$, slide j to the position of next[j], and get the value of next[j] by KR algorithm;
(6) Detected the matched pattern string before mismatching at this time, and define it as secondary pattern string Z_p;
(7) Mapping the secondary pattern string Z to an integer pattern_hash by using Hash function;
(8) Define the count variable i as 1;
(9) Mapping the first substring Y[i,i+1,…,i+p-1] of secondary text string whose length is p into an integer text_hash;
(10) while（i<m-p+1）, perform (11)(12);
(11) if(pattern_hash=text_hash), output the matched position at this moment;
(12) else if（i=i+1）, return to (9);
(13) Repeat (5) until j＞m or i＞n-m+1. When the comparison of all the characters in the pattern P[0:m] was finished,return to the starting subscript of the matching, or else return 0;
End.

This improved algorithm combines the advantages of KMP algorithm and KR algorithm and some improvements of calculating the next function in the KMP algorithm was made under the condition of " large text - big Mode " string matching. When a mismatching occurs, the first j-1 pattern substrings will be saved and its hash value will be calculated, and secondary KR text-pattern string will be constructed. According to the improved algorithm that needs 1+2[m-(j-1)+1]=2m-2j+5 times matching operation in the worst cases of match strings. When 2m-2j+5<m, that is j>(m+5)/2, the matching efficiency of improved algorithm is far higher than KMP algorithm.

Case Analysis: In the secondary text string, comparing the matching times between needed in calculating next value and the next function in the KMP. Define the pattern string(secondary text string) as: abbcddacbdaacda； the length of the character string is 15, then the matching times of calculating the next function is 15 times in KMP algorithm. The changes of matching time of comparing the secondary text string and secondary pattern string by the way calculating hash value is as follows:

(1) When the secondary pattern string is: daacda, the amount of matching is: 1 + 2*(15-6+1) = 21;
(2) When the secondary pattern string is: bdaacda, the amount of matching is: 1 + 2*(15-7+1) = 19;
(3)When the secondary pattern string is: cbdaacda, the amount of matching is: 1 + 2*(15-8+1) = 17;
(4) When the secondary pattern string is: acbdaacda, the amount of matching is: 1 + 2*(15-9+1) = 15
(5) When the secondary pattern string is: dacbdaacda, the amount of matching is: 1 + 2*(15-10+1) = 13
…… ……
(i-1) When the secondary pattern string is: $z_1z_2\ldots z_{i-1}$($0< i < 15$), the amount of matching is:1 + 2*(15- i+1+1) = 35-2i

(i) When the secondary pattern string is: $z_1z_2\ldots z_i$($0< i < 15$), the amount of matching is: 1 + 2*(15- i+1) = 33-2i

The above results show that when the number of the characters of secondary pattern strings gradually becomes longer, the corresponding number of comparisons gradually decreased, and the efficiency gradually increased. The amount of comparison is the same as the traditional in (4), but lower in (5)…(i-1)(i). This proves that when the amount of the character in the secondary text string (the pattern string) and the secondary pattern string is huge, the efficiency will continue to increase.

Efficiency analysis: The analysis of the time complexity of KMP algorithm can be divided into two parts, one is the matching times of scanning x_n and y_m , another one is the workload of calculating the next[j] according to pattern. It has relationship to the length m of pattern y of calculating next[j], instead of the main string, and its time complexity is O(m). As can be seen from the improved algorithm ,the matching times of scanning x_n and y_m is the same as the traditional but the workload of calculating next[j] was reduced.

In the construction of Hash function, it needs 1 time operation in (1). Since it requires to find out the hash value of m-p+1 secondary text substring in （2）and（3）, so needs m-p+1 times operation in all. It requires to compare the pattern_hash obtained from (1) and text_hash of (m-p+1) substrings obtained from (2) (3) in (4) , which needs m-p+1 times at the worst. Therefore it needs 1+m-p+1+m-p+1=2(m-p)+3 times calculation altogether. When the amount of the pattern string is huge, the mantissa 3 has very little effect on the time complexity. When change 2(m-p)+3 to 2(m-p), it just need to compare the time complexity of the traditional way to calculate next function O(m) with O(2(m-p)). The longer the length of the secondary pattern string, the lower the time complexity is.

THE APPLICATION OF THE IMPROVED ALGORITHM IN THE FLOW FILTERING TECHNOLOGY

The principle of the flow filtering technology

Flow filtering technology is a new type of firewall architecture proposed by Neusoft. This technology integrated the security and advantages of packet filtering and application proxy, and overcomes not only the difficulty that the application proxy firewall can only check single TCP message and unable to cache and restore the data flow in the application layer and limitation of the scope of application protection protocol the application proxy firewall provided[5,6]. The separated treatment strategy was adopted, and simple state detection of single TCP/IP message was taken(as shown in Figure 2), to identify, intercept, cache and reassemble multi messages which contained blocking data flow, thereby achieve the check of data stream in the application layer (as shown in Figure 3)

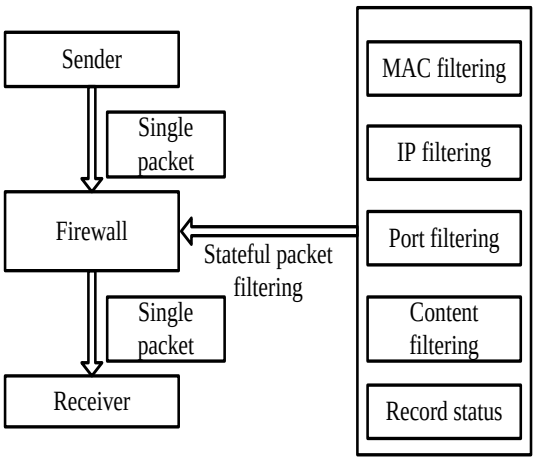

Figure 2. Flow filtering of single data block

The fundamental principle of the flow filtering technology is achieving the protection of application layer in the form of Stateful Packet Filtering. Perform transparent reassembling, checking and transmission to blocking data flow through dedicated TCP protocol stack. Outside the firewall, flow filtering is still the form of packet filtering, but for the allowed data access there are two separate TCP sessions inside the firewall and data communicated in the way of flow between sessions. Flow data firewalls can take place of any end of communication to take part in the session of application layer at any time, which play the same control capabilities as application proxy firewall.

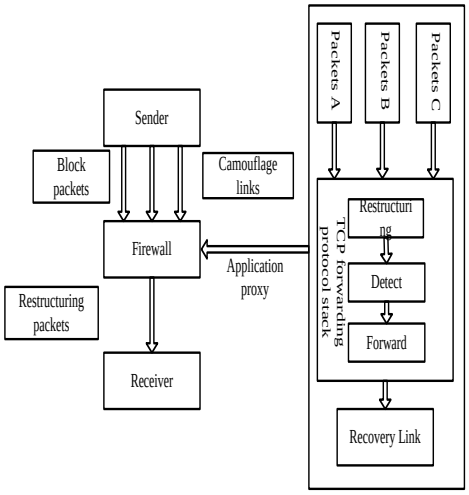

Figure 3. Flow filtering of multi-data block

The improved algorithm of filtering content matching

The message reassembling of TCP blocking data: The filtering content matching is to match the reassembled data according to character the item of the filtering content specified. This paper will discuss the filtering content matching after using the improved KMP algorithm to reconstruct the data. Using specified data structure to save the head of MAC address, the information of port and the TCP message. Realizing the reassembling of message through secondary pointer, and using the improved KMP algorithm to perform filtering match to illegal data after the reorganization of big data message skillfully(as shown in Figure 4).

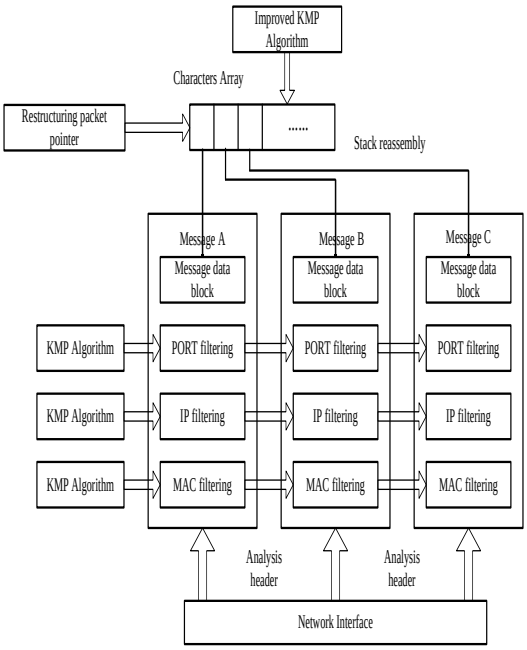

Figure 4.The reorganization of TCP message

Here is going to use the reassembled big data message to perform filtering matching of illegal content data. Setting the length of illegal long enough, when mismatch occurs in the text string X [n] - pattern string Y [m], and the mismatch position meets j> (m +5) / 2, matching efficiency will far higher than the KMP algorithm.

The implementation of improved algorithm:(1)Matching by calling KMP algorithm, when mismatching occurs, saved the first k-1 character substring matched each other and defined it as secondary pattern string: Z[p](p = k-1), then obtain its hash value; int Hash(char* Z)

```
{
int sum = 0;
while(*Z!= '\0')sum +=(int)*Z++;
retum sum%q;  //q is a prime selected randomly
}
```

(2)Define the pattern string as secondary text string, then calculate the value of Hash function of any secondary text string whose length is p, define Hash_ap(T,i,Hash_p,p)(i stand for the ith comparison);intHash_ap(char*Y, int i, intprevHash, int p)

```
//preHash represent the hash value getting from the (i-1)th loop
{
intaphash=(prevHash-((int), Y[i])+((int)Y[i+p]))%q;
// q is a prime selected randomly
return ( val< 0 ) ? ( val + q ): val;
}
```

(3)Match secondary KR substring once, and strike the value of next [j] rapidly;

```
for(int a = 0; a <= i-1; a++)
{
if (hash_ap == hash)
return a;
else
```

```
return 0
}
(4)Put the calculated value of next into KMP algorithm;
intkmp(SString Y, SString X, int next[] )
{
int i = 0, j = 0;
m = length( Y ), n = length( X );
while( j < m && i < n );
if (j = = -1 ‖ Y[j] = = X[i] ){
i++;
j++;
}
else
j = next[j];\
if( j >= m)
return( i-m );
else
return(0);
}
```

The experiment: Using two computer whose model is DELL Pentium(R)Dual 2.20GHz、memory is 2.00GB in the test environment, one of them was used for testing. The operation system of the computer is Linux ubuntu6.06 (kernel version: 2.6.1), he filter system is Linux Redh at 9 (Kernel version: 2.6.18). Insert the improved KMP algorithm into the kernel of the filter, then start data flow generator of both computer, and test the matching efficiency of data in different levels of magnitude, the number of test times is 20. After the data is filtered through every layers of data protocol, the reorganized data in different levels of magnitude will match through improved KMP algorithm. The matched results was got as follow, the result is average of 20 times matches:

Table 3.The result of string matching (unit: ms)

length(byte)	KMP(ms)	KR(ms)	Improve algorithm(ms)
50	102.46	543.28	147.31
100	125.47	531.91	152.59
300	147.81	548.74	157.71
500	151.44	581.04	163.18
800	157.02	491.35	166.06
1000	162.47	575.23	167.11
1500	188.73	541.31	171.20
3000	201.19	489.66	174.31
5000	229.41	472.05	183.78

The results of the experiment show that when the length of the text string is under1500 bytes, the average computing time of improved algorithm will be higher than the average KMP algorithm, however, when the length of text string higher than 1500 bytes, the average computing time of improved algorithm will be less than the matching time of KMP algorithm. Meanwhile the average matching time of the improved algorithm is much higher than the KR algorithm. Experiment shows that the computing efficiency of the improved algorithm is far higher than both KMP algorithm and KR algorithm under the condition of “large text - big Mode” string matching.

CONCLUSION

In this paper, we improved the KMP algorithm on the basis of the classical string matching algorithm, and exploit the improved KMP algorithm to conduct the illegal data filtering matching to the reorganized big data message. The result of the experiment shows that the improved algorithm has higher matching efficiency on the condition of “large text - big Mode”.

Filtering technology is an improved technology which combined "packet filtering" and "Application agent". This paper proposed an improved duplex layer matching algorithm for filtering match the illegal data of TCP reorganized message by the analysis of the filtering technology. This algorithm solved the problem of high dynamic memory occupancy and long time-consuming calculation in KMP algorithm. At research shows, since flow filtering is not completely transparent to the application, it will be the further research direction to match small text of single message.

Acknowledgement
Supported by Natural Science Fund in JiangSu (BK2009535) and Jiangsu Province ordinary university innovative research project (CXZZ13_0691).

REFERENCES

[1] Guo-liang Cheng.Design and Analysis of Parallel Algorithms [M]. Beijing: Higher Education Press, **2009**.
[2] Frantisek Franek, Christopher G. Jennings, W.F. Smyth. *Journal of Discrete Algorithms*, **2007**, 5(4): 682-695
[3] Wei-Min Yan, Wei-Ming Wu. Data Structure [M]. Beijing: Tsinghua University Press, **2007**.
[4] Dana Shapira, Ajay Daptardar. *Information Processing & Management*, **2006**, 42(2): 429-439
[5] Yue-Hua Zhao, Wan-Sheng Zhou. *Computer Engineering*, **2012**, 38(2): 135-140
[6]YuwanGuetc, *International Review on Computers and Software*.**2012**,6(7):3042-3046