Statistics of Frequent Sequence of Link Layer Bitstream Data based on AC-BMHS-CO Algorithm

Jibo Han[1], Lin Qi[2] and Zheng Dou[3]

[1]College of Information and Communication Eng., Harbin Engineering University
(145 Nantong St., Harbin, Heilongjiang, China)
E-mail:hanjibo@ hrbeu.edu.cn
[2]Professor, College of Information and Communication Eng., Harbin Engineering University
(145 Nantong St., Harbin, Heilongjiang, China)
E-mail: qilin@hrbeu.edu.cn
[3]Professor, College of Information and Communication Eng., Harbin Engineering University
(145 Nantong St., Harbin, Heilongjiang, China)
E-mail: douzheng@hrbeu.edu.cn

At present, the AC algorithm and its improved algorithms are widely used in big data statistics and character matching, but the analysis objects of these algorithms are mostly ordinary characters and text. When these algorithms are used for frequent sequence statistics of link layer bitstream data, the problem of matching technique failure and matching efficiency is reduced will occur. In response to the above problems, a method for frequent sequence statistics of link layer bitstream data based on the AC-BMHS-CO (Coded AC-BMHS) algorithm is proposed. The AC-BMHS-CO algorithm first compresses and encodes the link layer bitstream data to expand the character set, and then uses an improved AC-BMHS algorithm with fuzzy matching rules to perform jump style matching. Through theoretical analysis and actual data comparison test, it is found that the AC-BMHS-CO algorithm can accurately count frequent sequences in the bitstream data environment. Compared with the improved algorithm of AC algorithm, AC-BMHS-CO algorithm has higher efficiency and accuracy.

Key Words : *AC-BMHS-CO algorithm, frequent sequence statistics, link layer bitstream, compression coding, fuzzy matching*

1. INTRODUCTION

In today's wireless communication environment, whether it is signal detection in electronic countermeasures or security maintenance in communication networks, especially in the military field, unknown protocols are widely used. Therefore, identification of unknown protocols has very important research significance[1]. Only after the link layer recognizes the complete data frame and analyzes and recognizes the frame format, the high-level protocol can perform further identification. Therefore, link layer protocol identification occupies an important position in the field of protocol identification[2]. However, because the link layer protocol data exist in the form of a bit stream and does not have any semantic features, the research progress on the recognition of unknown bit stream protocols at home and abroad is slow. In existing unknown existing protocol recognition technologies for bitstreams, pattern matching and data mining techniques are usually used. The basic processes are: frequent sequence statistics, association rule mining, data frame segmentation, and protocol format speculation. The effects of the latter three modules depend to a large extent on the correctness of frequent sequence statistics[3]. And frequent sequence statistics are at the heart of this article's algorithm research. When performing frequent sequence statistics of unknown protocol bitstream data, scholars at home and abroad mostly use the AC algorithm[4]. However, due to the defects of the algorithm's specific design, the algorithm has problems that urgently need to be solved, such as high time complexity, high space complexity, and short pattern sequence. Scholars at home and abroad have used the AC algorithm to perform frequent sequence statistics of bitstreams. The algorithm realizes that sequence statistics can be completed by scanning the sequence only once, but the algorithm time complexity is high[5]. The AC-BM algorithm is an improved algorithm of the AC algorithm, which can achieve skip matching of multiple pattern strings. Compared with

the traditional AC algorithm, the theoretical time efficiency has been greatly improved, but due to the particularity of the binary sequence, the algorithm cannot achieve the efficiency of the theory[6]. The AC-BMH algorithm is another improved algorithm of the AC algorithm. The algorithm uses the idea of the BMH algorithm to improve the AC algorithm, but the time efficiency of the algorithm is not much improved compared to the AC-BM algorithm[7]. In 2017, Cao proposed an AC-IM (Improved AC) algorithm, which measures the dissimilarity of pattern matching by constructing a string maximum jump distance table and two hash tables. The algorithm improves the efficiency and accuracy of statistics, but it is more complicates[8].

There are three main innovations in the algorithm in this paper:

An improved algorithm of AC-BMHS algorithm is proposed, and fuzzy matching rules are added on the basis of the original algorithm. The improved algorithm improves the efficiency and accuracy of pattern matching.

The AC-BMHS-CO (Coded AC-BMHS) algorithm is proposed. Firstly, the bitstream data are compressed and encoded, and then the AC-BMHS algorithm is employed for pattern matching. The improved algorithm improves the efficiency of frequent bitstream sequence statistics and reduces the missed detection rate.

Using actual data, the accuracy and efficiency of the AC-BMHS-CO algorithm and the literature algorithm were compared and tested, and the ideal experimental results were generated.

The rest of the paper is structured as follows: Section 2 introduces the basic principles of the algorithm; Section 3 is the theoretical analysis of the algorithm; Section 4 is the algorithm performance comparison test; Section 5 is the summary of the full text.

2. ALGORITHM PRINCIPLE

The AC algorithm and its improved algorithms are widely used in big data statistics and character matching, but the analysis objects of these algorithms are mostly ordinary characters and texts. When used in the frequent statistics of link layer bitstream data, the character matching skills will fail. The efficiency of algorithm identification is drastically reduced, which is close to the worst case of the algorithm, and the actual application effect is not obvious. After analysis, it is found that the failure of the string matching algorithm in the face of bitstream application scenarios is due to the limited character set elements, and the target string and pattern string have the same character set.

Aiming at the above problems, the AC-BMHS-CO algorithm is proposed. The method to improve the efficiency of the algorithm is to first compress and encode the original data to expand the character set size, and then use an efficient multi-pattern matching algorithm for pattern matching. Expansion of character set elements means that the frequency of mismatches increases and the number of window moves increases, so it is more suitable to use multi-pattern matching algorithms. Using the improved and efficient AC-BMHS algorithm to perform pattern matching on the compressed and encoded data will further improve the efficiency of the algorithm for frequent sequence statistics of the link layer bit stream.

(1) AC-BMHS Algorithm

The AC-BMHS algorithm is an improved algorithm that combines the idea of "bad characters" in the BMHS algorithm and the idea of a finite state automata in the AC algorithm[9]. The algorithm is divided into preprocessing stage and pattern matching stage:

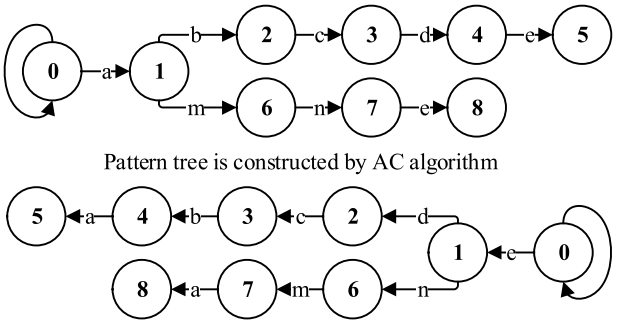

Fig.1 Two different pattern trees

Pretreatment stage: Build a reverse finite state automata. Unlike the AC algorithm, the pattern tree constructed by the AC-BMHS algorithm is an "inverted tree". For example, for the pattern string set $P=\{abcde,amne\}$, the pattern tree constructed by the AC algorithm and the AC-BMHS algorithm is shown in Figure 1. At the same time, the improved algorithm does not need to construct a failure function.

Calculate the jumping distance of the pattern tree under the "bad character" rule: AC-BMHS algorithm draws on the "bad character" rule of the BMHS algorithm. Set the shortest pattern string length as *minlength*. When a mismatch occurs, consider the first character x from the right of the last character in the current matching window. If x do not appear within the *minlength* depth of the pattern tree, move the pattern tree to the right (*minlength+1*) characters. If x appears in a pattern string with a depth of p

($p<minlength$) in the pattern tree (if x appears in multiple positions, take the minimum value of p), move the pattern tree to the right by p characters.

Pattern matching stage: Initial state: The root node of the pattern tree is aligned with the *minlength* character of the text string, and the *minlength* layer node is aligned with the first character of the target string. The moving direction of the pattern tree is from left to right, and the character comparison direction is right to left.

Pattern matching: In the current matching window, read the target substring characters from right to left, and uses the transfer function of the pattern tree for state transfer. When there is a mismatch, the pattern tree is moved to the right according to the jumping distance calculated by the "bad character" rule; when the output function is not empty, the matching success message is output.

Repeat the above steps until the matching ends when the pattern tree is aligned with the end character of the target string.

(2) AC-BMHS-CO Algorithm

The AC-BMHS-CO algorithm is divided into the compression and encoding stage and the AC-BMHS algorithm matching stage. The algorithm first compresses and encodes the original bitstream data to expand the character set before pattern matching, and then uses the improved AC-BMHS algorithm for pattern matching.

The original link layer data is a bit stream, and its basic unit is a bit. A link layer data frame, the frame start bit is fixed. Once the start bit is wrong, it will cause a lot of error analysis. The common pro-cessing method is: before the bit stream is sent to the analysis program, the bit stream is converted into a character stream, that is, bit “0” in the data is mapped to the character "0", and bit “1” is mapped to the character "1". After such processing, the problem of positioning the first bit is weakened, and the data is easier to be analyzed by the program. Before the AC-BMHS-CO algorithm formally uses the string matching algorithm, the data need to be compressed and encoded. These data include the target bit string and the pattern bit string.

The specific process of compression encoding is as follows: the algorithm uses the "1" in the bit string as the statistical node, counts the number of "0" before each "1", and stores the number of "0" "l" as the character L; Then the character "L" is a new character formed by encoding the "l" bits "0" and the "1" immediately after it. A special case needs to be considered: if the bit string ends with a bit "0", then the algorithm adopts the strategy of complementing "1" by default. Then a bit string shaped like "000100" is equivalent to the bit string "0001001", which can be compressed and encoded into the string "3,2". This article refers to the character formed by encoding the bit string ending with the bit "0" as the "fake" character, and the "fake" character should be treated differently from the "true" character when the pattern is matched.

Figure 2 contains two processing schemes for the original bit string: In the traditional processing scheme, the bit string is mapped to an 8-character target string, and the corresponding character set is {0,1}; In the compression encoding processing scheme, the bit string is mapped to a 3-character target string, and the corresponding character set is {1,2,3}. Through comparison, it can be found that the compressed and coded data processing scheme has expanded the character set. At the same time, due to the limited length of the pattern string, the character set of the pattern string must be smaller than the character set of the target string.

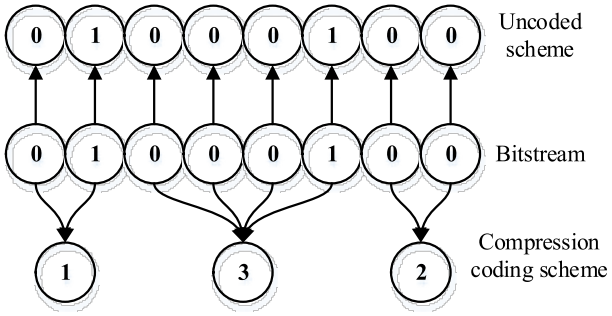

Fig.2 Two processing schemes for bit string

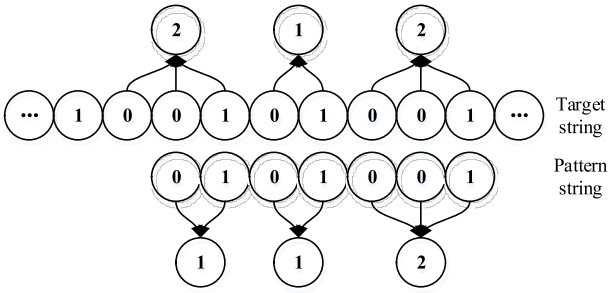

Fig.3 Fuzzy matching example 1

Because the algorithm adopts the compressed and encoded data preprocessing method, the data characteristics of the original bit string are overwritten, and some special situations will occur when the string is matched. As shown in Figure 3, the target string is "...,2,1,2,..." and the pattern string is "1,1,2". According to the bit string data comparison, there is a sequence matching the pattern string in the target string. But if according to the AC-BMHS algorithm from right to left exact matching (exactly equal) rules, "...,2,1,2,..." in the string does not have a substring that is exactly equal to "1,1,2", so they do not match. In view of this change, fuzzy matching (not exactly equal) rules have been added to the

AC-BMHS algorithm. Various situations of fuzzy matching are described in detail below.

Fuzzy matching rule for the first character of the pattern string: When the first character of the pattern string matches the target string, the fuzzy matching rule of "not less than" is adopted. As long as the size of the target string character is not less than the pattern string character, then the two are considered to match. As shown in Figure 3, the target substring "2,1,2" matches the pattern string "1,1,2".

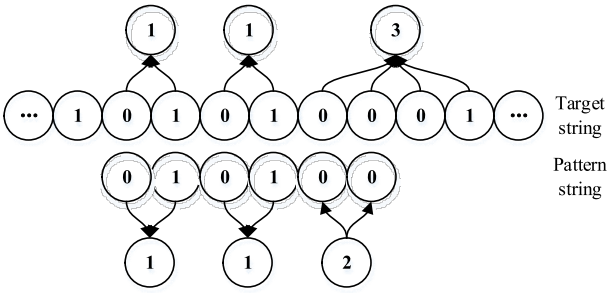

Fig.4 Fuzzy matching example 2

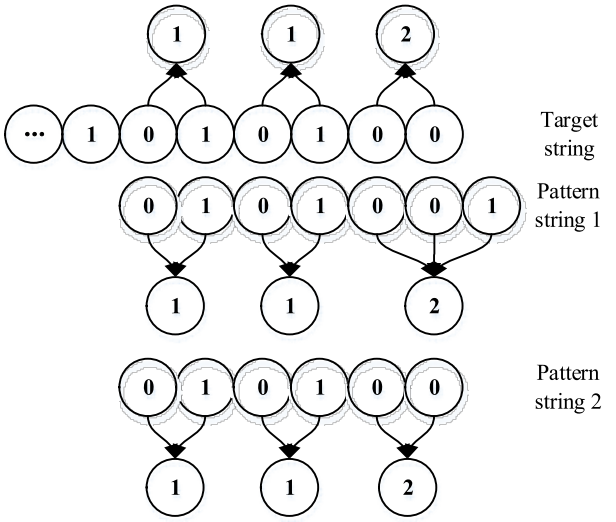

Fig.5 Fuzzy matching example 3

Fuzzy matching rule when the last character of the pattern string is "fake": When the last character of the pattern string is "fake" and matches the non-end character of the target string, the fuzzy matching rule of "not less than" is adopted. As long as the size of the target string character is not less than the pattern string character, then the two are considered to match. As shown in Figure 4, the target substring "1,1,3" matches the pattern string "1,1,2".

Fuzzy matching rules when the end character of the target string is "fake": When the end character of the target string is "fake" but the end character of the pattern string is "true", the two will not match even if they are equal. When the end character of the target string is "fake" and the end character of the pattern string is also "fake", the fuzzy matching rule of "not less than" is adopted. As shown in Fig.5, the target string does not match the pattern string 1; it matches the pattern string 2.

Pattern string single-character fuzzy matching rule: When a single-character of the pattern string is a "true" character and matches a non-tail character in the target string, the rule in Figure 3 is obeyed. When a single-character in the pattern string is a "fake" character and matches a non-tail character in the target string, the rule in Figure 4 is obeyed. When a single-character in the pattern string matches a character at the end of the target string, the rule in Figure 5 is obeyed.

The bitstream data are first compressed and encoded. Then the AC-BMHS algorithm containing fuzzy matching rules is used for pattern matching, and finally the frequent sequence determination is performed. The algorithm can efficiently complete the statistics of the frequent sequence of bitstream data.

3. ANALYSIS OF ALGORITHM

(1) Matching Speed

The optimal matching efficiency of the algorithm is determined by the maximum jump distance of the pattern tree. The larger the maximum jump distance, the higher the optimal matching efficiency of the algorithm. In the case of no missed detection, the maximum jump distance of each algorithm is shown in Table 1, where *minlength* represents the shortest pattern string length.

Table 1 Maximum jump distance

Type	Maximum jump distance
AC-BM	*minlength*
AC-BMH	*minlength*
AC-IM	*minlength+3*
AC-BMHS-CO	*minlength+1*

It can be seen from Table 2: The maximum jump distance of the AC-BMHS-CO algorithm is the shortest pattern string plus 1. Although it is not optimal, the AC-BMHS-CO algorithm uses compressed and encoded strings, which actually have higher efficiency. Link layer bitstream data can be regarded as randomly occurring, so the probability of each bit of "0" or "1" is 50%. After compression coding, the data compression rate can reach 50%. Then even in the case of *minlength=1*, the maximum hop distance corresponding to the AC-IM algorithm is not greater than that of the AC-BMHS-CO algorithm. This shows that the AC-BMHS-CO algorithm improves the matching efficiency through the preprocessing method of compression coding.

The processing object of the algorithm in the literature is the bit stream, the character set is only {0,1}, and the pattern string is the same as the target string character set, so the possibility of the maximum jump during data matching is extremely small. The AC-BMHS-CO algorithm performs compression and coding preprocessing on the bit stream data, so that the pattern string character set is no larger than the target string, which improves the mismatch efficiency and increases the possibility of reaching the maximum jump distance during matching.

(2) Matching Time Efficiency

The time efficiency of the algorithm is determined by the time complexity. The higher the time complexity, the lower the time efficiency of the algorithm. Table 2 is the time complexity data table. In the table, *n* represents the length of the target string, *minlength* represents the length of the shortest pattern string, and *maxlength* represents the length of the longest pattern string.

Table 2 Time complexity

Type	Worst	Average	Best
AC-BM	$O(n\times maxlength)$	$O(n)$	$O(n/minlength)$
AC-BMH	$O(n\times maxlength)$	$O(n)$	$O(n/minlength)$
AC-IM	$O(n\times maxlength)$	$O(n)$	$O(n/(minlength+3))$
AC-BMHS-CO	$O(n\times maxlength)$	$O(n)$	$O(n/(minlength+1))$

It can be seen from Table 2 that the average time complexity and worst time complexity of the AC-BMHS-CO algorithm and the literature algorithm are the same, and the best time complexity is not the worst. The AC-BMHS-CO algorithm uses compression and encoding data preprocessing methods, which will surely have higher time efficiency in the actual data matching process.

4. ALGORITHM VERIFICATION

(1) Algorithm Accuracy Verification

In order to verify the accuracy of the algorithm in the application of link layer bitstream frequent sequence statistics, the experiment uses Wireshark software to capture 10,000 TCP data frames in the link layer of the wireless WiFi network over a period of time. The data frame is converted into continuous binary bit stream data, as the algorithm test data set. Compare AC-BMHS-CO algorithm with AC-BM, AC-BMH, AC-IM algorithm. First, perform frequent sequence statistics on the 2-16-bit pattern string (the threshold is set to 0.9), calculate the algorithm's screening rate, and then compare with the characteristic sequence of the standard TCP data frame to obtain the correct rate of screening. Fig.6 and Fig.7 respectively show the screening rate and the correct rate of the corresponding algorithm when the mode string is 2 to 16 bits.

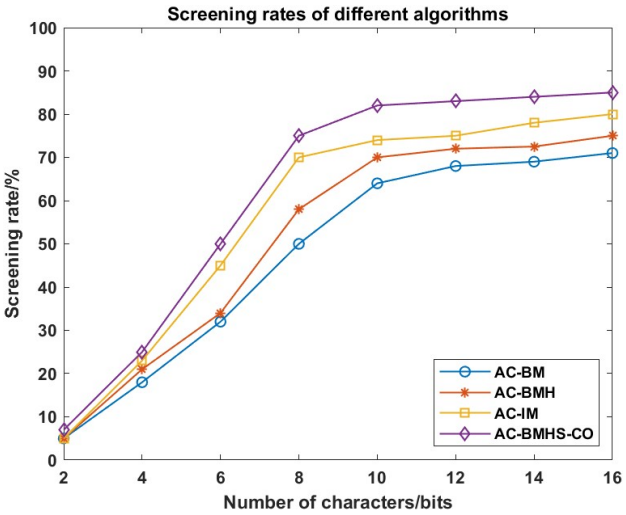

Fig.6 Screening rate of different algorithms

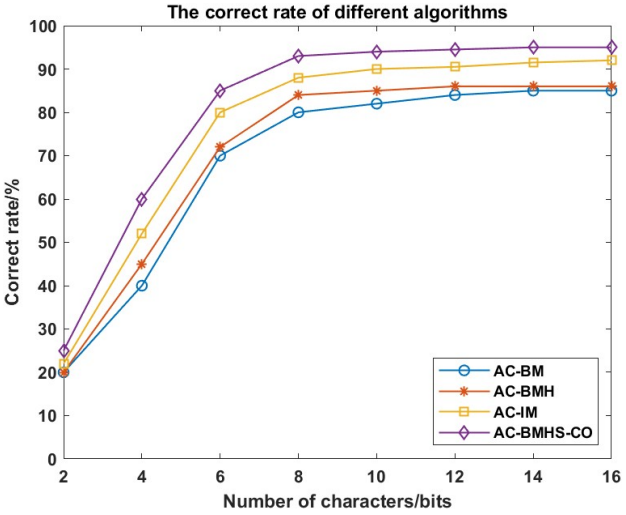

Fig.7 The correct rate of different algorithms

It can be seen from Fig.6 and Fig.7 that within a certain range, as the length of the pattern string increases, the algorithm's screening rate and the correct rate gradually increase. Compared with the literature algorithm, the AC-BMHS-CO algorithm has certain advantages in the screening rate and accuracy of frequent sequences.

(2) Algorithm Efficiency Verification

Use Wireshark software to capture 10000 ARP data frames, 10000 ICMP data frames, and 10000 TCP data frames of the wireless WiFi network link layer, and then mix the three different data frames and splice them into continuous binary bit stream data. And use this as a data set to verify the efficiency of the algorithm. First, fix the length of the pattern string to 16 bits, change the number of pattern strings

for comparison testing; then fix the number of pattern strings to 500, change the length of the pattern string for comparison experiments. The experimental results are shown in Figure 8 and Figure 9.

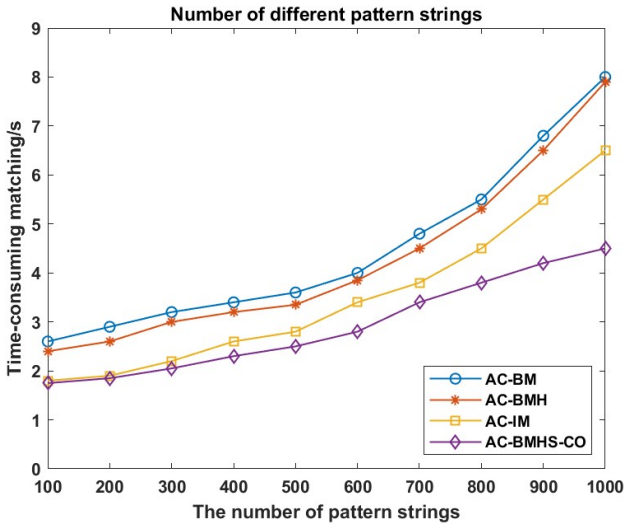

Fig.8 Different number of pattern strings

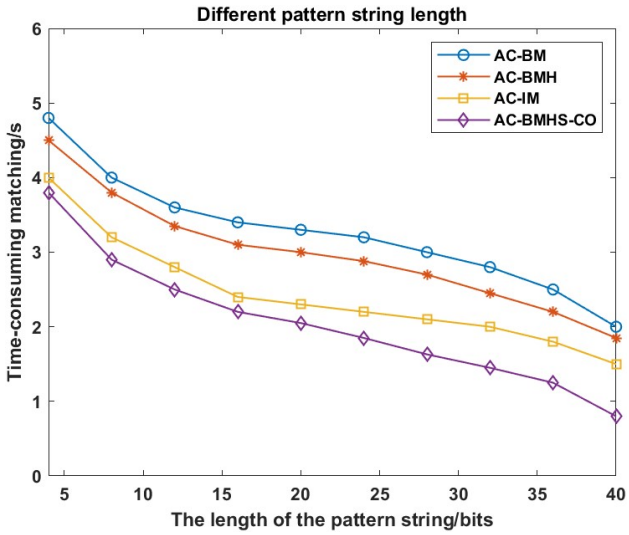

Fig.9 Pattern strings of different lengths

It can be seen from Figure 8 that as the number of pattern strings increases, the algorithm matching speed shows a downward trend; as can be seen from Figure 9, as the length of the pattern strings increases, the algorithm matching speed shows an upward trend. Because the AC-BMHS-CO algorithm compresses and encodes the bitstream data before pattern matching, the overall matching speed is faster than the literature algorithm. Figures 8 and 9 fully demonstrate that the AC-BMHS-CO algorithm has higher efficiency.

5. CONCLUSION

The AC-BMHS-CO algorithm proposed in this paper aims to solve the problem of character matching skills fails in the AC algorithm and its improved algorithm when the link layer bit stream data frequent sequence statistics are performed, and then show the decline in matching efficiency and accuracy. Theoretical analysis and experimental verification show that the AC-BMHS-CO algorithm can accurately count the frequent sequences of link layer bit stream data, and the algorithm consumes less time and is highly efficient. The AC-BMHS-CO algorithm lays the foundation for the analysis and identification of the unknown protocol frame format at the link layer in the next step, and has certain practical application value.

ACKNOWLEDGMENT: This work is supported by the National Natural Science Foundation of China (Grant No. 62071139).

REFERENCES

1) Yin, S. and Wang, T. : An unknown Protocol improved k-means clustering algorithm based on Pearson distance, *Journal of Intelligent & Fuzzy Systems*, Vol. 38, No. 4, pp. 4901-4913, 2020.
2) Wang, Y., Xue, H. and Liu, Y. : Statistical network protocol identification with unknown pattern extraction, *Annals of Telecommunications*, Vol. 74, No. 7-8, pp. 473-482, 2019.
3) Zheng, J. : An association analysis and identification for unknown protocol of bitstream oriented, *Concurrency and Computation-Practice & Experience*, Vol. 28, No. 15, pp. 4067-4081, 2016.
4) Zheng, J. and Li, J. : The Research of Address Message of an Unknown Single Protocol Data Frame, *Intelligent Automation and Soft Computing*, Vol. 24, No. 1, pp. 139-144, 2018.
5) Li, T. and Liu, Y. : A Novel Method for Delimiting Frames of Unknown Protocol, *IEEE Workshop on Electronics, Computer and Applications*, pp. 552-555, 2014.
6) Jia, M., Chen, M. and Li, H. : An improved AC pattern matching algorithm, *Information Technology Applications in Industry II, Pts 1-4,* Vol. 411-414, pp. 1594-1597, 2013.
7) Wang, M. and Zhu, L. : Research about Pattern Matching Algorithm, *Instrumentation, Measurement, Circuits and Systems*, Vol. 127, pp. 27-32, 2012.
8) Cao, C. and Lei, Y. : Frequent Statistics of Link-layer Bit Stream Data based on AC-IM Algorithm, *Green Energy and Sustainable Development I*, Vol. 1864, 2017.
9) Xue, K., Liu, B. and Wang, J. : Data Link Bit Stream Oriented Association Analysis on Unknown Frame, *Journal of Electronics & Information Technology*, Vol. 39, No. 2, pp. 374-380, 2017.