ICIC Express Letters
Volume **11**, Number **3**, March **2017**
 ISSN 1881-803X
pp. **649–655**

REDUNDANT NETWORK TRAFFIC IDENTIFICATION ALGORITHM BASED ON SLIDING WINDOW

Ling Xing[1,2], Qiang Ma[2] and Hong Zheng[2]

[1]School of Information Engineering
Henan University of Science and Technology
No. 263, Kaiyuan Avenue, Luoyang 471023, P. R. China
xingling_my@163.com

[2]School of Information Engineering
Southwest University of Science and Technology
No. 59, Qinglong Road, Mianyang 621010, P. R. China
{ mqaing_my; zhenghong_my }@163.com

Received August 2016; accepted November 2016

Abstract. *Redundant traffic identification (RTI) is vital to protocol-independent redundant traffic elimination (RTE) since RTI is the premise of RTE based application. The core issue of RTI is the chunk selection strategy, which makes decisions on how to divide the packet payloads into chunks used for caching. Current chunk selection algorithms take blocks of fixed size, which fails to deal with the problem of payloads bits shift. To tackle this problem, an RTI algorithm based on sliding window (SW-RTI) is proposed. In this algorithm we divide payloads into chunks of variable sizes, employ the sliding window to search for the best dividing points and calculate the fingerprints between two adjacent points. The dividing points are obtained by Rabin hashing method. Experimental results show that SW-RTI, which optimizes the selections of sliding window size and average chunk size, can enhance the stability of chunking. Also it is effective for identifying network traffic redundancy.*
Keywords: Network traffic analysis, Redundant traffic elimination, Redundant traffic identification, Chunk selection strategy, Sliding window

1. **Introduction.** As Internet technology continues to develop, large volumes of Internet applications (e.g., file sharing, video distributing and web interactions) are growing fast. Due to large number of users and the long-tailed distribution of popularity of shared resources on the web, many similar or same contents are repeatedly transferred over the Internet, which are unnecessary and cause redundant network traffic [1-3]. The redundant traffic is harmful to network bandwidth [4,5]. Thanks to the high performances and fast processing abilities, network storage devices are able to identify the redundant traffic and can even remove some redundancy. Redundant traffic elimination (RTE) is a technique to detect and eliminate redundant blocks of data from packets at network layer [6-8]. The RTE is aimed to improve network bandwidth usage and save energy for communication.

The pioneer work of protocol-independent RTE is the chunk selection algorithm MODP (MODe Partition), which was proposed by Spring and Wetherall [9]. It is based on mode operation and it selects chunks with robust fingerprints. However, its major disadvantages are that its selection is too random and many large blocks may not be selected. Thus the blocks selected are too sparse or too dense. Anand et al. [10] improved the MODP and advanced a selection method called MAXP (MAXima Partition), which chooses blocks with the largest byte value. MAXP was actually based on WINN (WINdow Numerically partition) [11], which sets bounds on the distances of adjacent chunks and ensures that within certain distance at least one chunk is selected. Agerwal et al. [12] proposed a relatively faster algorithm named by SAMPLEBYTE (SAMPLE BYTE partition), which

selects chunks with specific byte value based on lookup tables. It obtains the most redundant byte through training and executes more efficient than MAXP. Nevertheless, it cannot handle situations of dynamic changes of network packets and its sampling manner is not predictable.

The methods mentioned before are all based on the fixed size of chunks and question arises from the perturbations of packet payloads, where how to correctly deal with the bits shift of payloads is a question affecting the performances of RTE. In this paper we propose a novel chunk selection algorithm to enhance the stability of chunking. The algorithm is based on sliding window for RTE (SW-RTI) and uses sliding window technique to divide chunks. It increases the detection of redundancy and improves the efficiency of RTE for highly redundant traffic.

The rest of this paper is organized as follows. Section 2 states through examples our idea of sliding window-based RIT method. The SW-RTI algorithm is explained in detail in Section 3 and we conduct parameters evaluations of the proposed method in Section 4. Section 5 concludes this paper.

2. Related Work.

The RTI is the most important step of RTE algorithm. Usually RTI is implemented via fingerprints computing technique to select and identify redundant chunks [13]. Fingerprinting technique is suitable to find the redundancy within and/or between packet contents [14]. Every chunk within a packet is calculated by hashing method and its fingerprint is obtained. Those fingerprints are stored in a set. When indices are constructed and searched, each candidate compares with those fingerprints. If match exists, then the redundant chunk is claimed and its relative position in the packet is marked and used for elimination. This means that there is no need to transfer this chunk anymore and instead its position together with some meta-data is transferred, which reduces packet payload and increases bandwidth efficiency. At the receiver side when redundancy is detected the meta-data and position would be replaced by original chunks.

Most methods implicitly define that chunks are all fixed size but when packet payloads undergo deletions and/or additions of minor bits, they are not able to detect those repeated blocks in the same packets. Redundancy detection is decreased and therefore we advance stability definition to measure the efficiency of chunking.

Definition 2.1. *Suppose by a chunk selection algorithm the packet payload A is chunked into set $A = \{A_1, A_2, \ldots, A_n\}$. If small changes are conducted on the some content, the newly obtained chunk set is A^*. The ratio of the number of the same chunks between set A and A^* divided by its total number of chunks is defined as stability of the chunking.*

Example 2.1. *Figure 1 shows the bits shift problem of payload by deletion. First for content A there are five chunks, i.e., $\{A_1, A_2, A_3, A_4, A_5\}$. When a similar content A^* arrives but with some deletion of bits in chunk A_4, then by the same chunk selection method with fixed chunk size, only $\{A_1, A_2, A_3\}$ are to be found the same and are regarded as redundancy. However, obviously chunk A_5 is still in content A^* and should be considered as redundancy. Thus this stability is 3/5.*

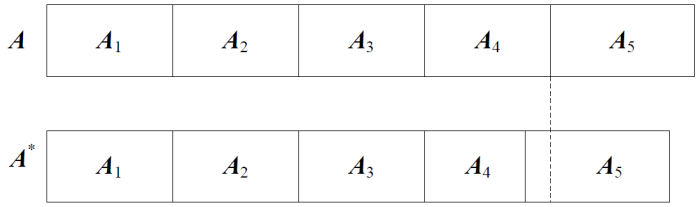

FIGURE 1. Bit shift problem by deletion of bits

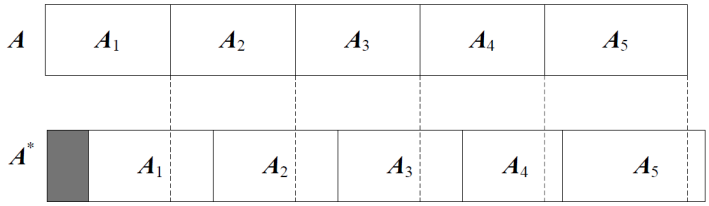

FIGURE 2. Bit shift problem by addition of bits

Example 2.2. *The bit shift problem caused by addition of bits is shown in Figure 2. The newly arrived packet A^* is inserted by some bits in front of previous chunk A_1 and by using fixed chunk size we obtained no redundancy. However, the original chunks actually still remain in A^* and the stability is 0/5.*

From these two examples we conclude that even small changes by addition and/or deletion of bits can result in inefficiency of chunking. The bit shift problem challenges chunk selection algorithm of RTI. We address this issue by using sliding window to chunk with various chunk sizes and divide the payload into blocks with better stability.

3. **SW-RTI Algorithm.** Distinguished from the conventional chunking methods which use fixed chunk size, the SW-RTI allows various chunk sizes, which means the divided blocks differ from each other in terms of block length. Figure 3 illustrates the processing flow of the proposed RTI, in which Rabin hash is first performed on the payload and the window size is increased by one byte if no Rabin hash codes match. On the contrary if there is a match for Rabin, fingerprinting is conducted on the data within this window to judge whether the data is redundant or not. Note that the size of the window is not fixed and it is determined by the Rabin hash to grow incrementally.

Suppose the fingerprinting algorithm is denoted by f and the window size by l, r is a binary data with length of k. A payload data string is denoted by $S = s_1 s_2 \ldots s_n$, and a substring of S is W with l, $W = s_k s_{k+1} \ldots s_{k+l-1}$. If the lower k bits of $f(S)$ equal r, then the substring W is the last chunk of S, i.e., the last byte is regarded as the dividing point.

When searching for dividing points we use the robust hashing method [15,16], which obeys rules as follows:

1) different hash codes imply different contents but the same hash codes do not mean the same contents and;

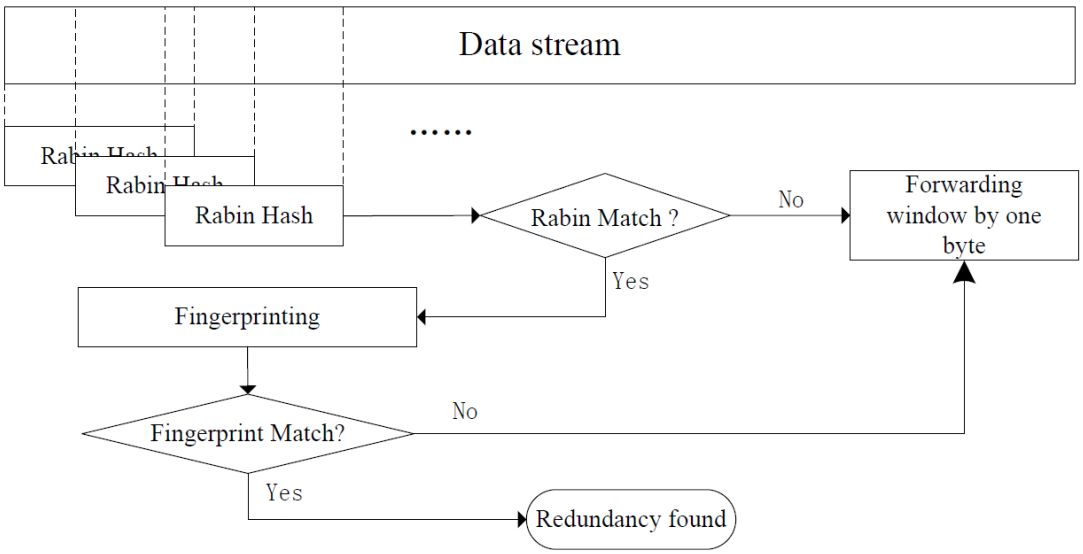

FIGURE 3. Illustration of SW-RTI

2) in order to keep chunk size from becoming too large the robust hashing should collide as much as possible.

Therefore we adopt Rabin hashing to seek dividing points in SW-RTI. Suppose A stands for a binary code with length m, $A = (a_1, a_2, \ldots, a_m)$. A polynomial $A(t)$ is constructed from A with degree of $(m-1)$, which is as follows:

$$A(t) = a_1 t^{m-1} + a_2 t^{m-2} + \ldots + a_m \tag{1}$$

in which t is treated as variable. Suppose there is another polynomial $P(t)$ with degree k for a binary code B, $B = (b_1, b_2, \ldots, b_k)$ and is defined as:

$$P(t) = b_1 t^k + b_2 t^{k-1} + \ldots + b_k \tag{2}$$

If $A(t)$ is divided by $P(t)$ then the result is a polynomial with degree of $(m-k-1)$. Let the fingerprint of A be $f(A)$ and it is defined as follows:

$$f(A) = A(t) \bmod P(t) \tag{3}$$

where mod operator stands for mode operation.

Rabin hashing exhibits two characteristics, i.e., 1) if $f(A)$ does not equal $f(B)$, then A is distinct from B and 2) if $f(A)$ equals $f(B)$ then A may be the same as B or may not. Rabin hashing satisfies the two requirements of robust hashing and therefore we choose it as for robust hash code generating. Besides this the Rabin hashing is good at finding dividing points. It can obtain the dividing point of next window from the content of current window. Suppose W_i is the string of length l falling into the i-th window, $W_i = A_i, A_{i+1}, \ldots, A_{i+l-1}$, and $A_k = (a_{k,1}, a_{k,2}, \ldots, a_{k,8})$ is the k-th byte with $k = 1, 2, \ldots, i, \ldots, i+l-1$. The polynomial for A_k is $A_k(t) = a_{k,1}t^7 + a_{k,2}t^6 + \ldots + a_{k,8}$ and we have

$$\begin{aligned} f(W_i) &= A_i(t) \bmod P(t) \\ &= \left(A_i(t)t^{8(l-1)} + A_{i+1}(t)t^{8(l-2)} + \ldots + A_{i+l-1}(t)\right) \bmod P(t) \end{aligned} \tag{4}$$

Thus if we obtain $f(W_i)$ then the hash for the next window W_{i+1} is calculated as follows:

$$\begin{aligned} f(W_{i+1}) &= A_{i+1}(t) \bmod P(t) \\ &= \left(A_{i+1}(t)t^{8(l-1)} + A_{i+2}(t)t^{8(l-2)} + \ldots + A_{i+l}(t)\right) \bmod P(t) \\ &= \left(W_i(t)t^8 - A_i(t)t^{8l} + A_{l+1}(t)\right) \bmod P(t) \\ &= \left(\left(W_i(t) - A_i(t)\right)t^{8l} + A_{l+1}(t)\right) \bmod P(t) \end{aligned} \tag{5}$$

Therefore we argue that the dividing points can be decided and it is shown in Figure 4, where the current dividing points are tactically determined by previous dividing points. The decision is based on the Rabin code of current candidate window. If it is equal to coefficient r, the current window is chosen as chunk. Otherwise move forward the window along sliding direction by one byte. Note that the parameter r is a factor which can be set according to practical analysis.

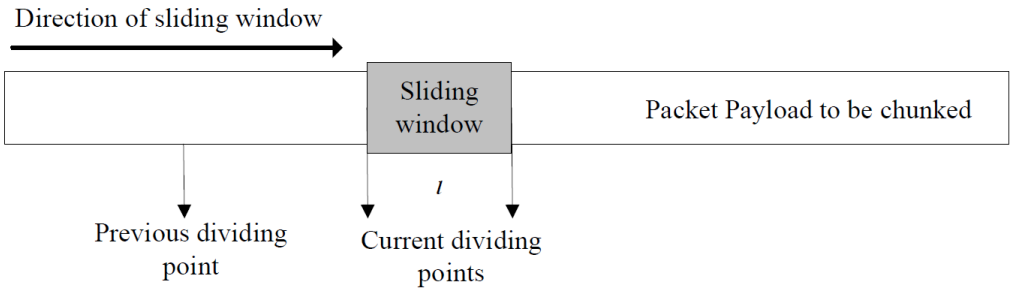

FIGURE 4. Dividing points in SW-RTI

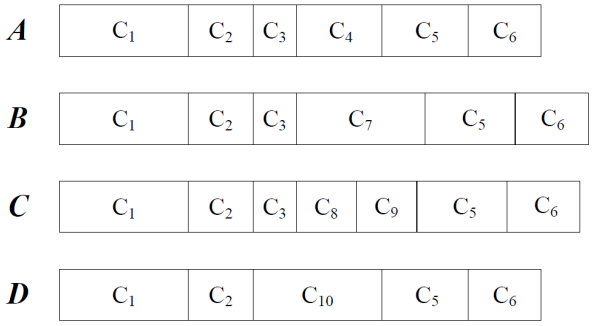

FIGURE 5. Chunking in SW-RTI

Here we apply the concept of stability to analyze the SW-RTI chunking performances. Suppose first we have payload content A chunked into $\{C_1, C_2, C_3, C_4, C_5, C_6\}$ in Figure 5. By bits addition to chunk C_4, payload A is changed into B and with our SW-RTI it is able to detect chunks $\{C_1, C_2, C_3, C_5, C_6\}$. The stability is 5/6. Similarly when chunk C_4 is broken into two chunks for payload C when added bits satisfy the Rabin hashing and chunks that are detected redundant are $\{C_1, C_2, C_3, C_5, C_6\}$. Also the stability is 5/6. For payload D bits of addition causes original chunks C_3 and C_4 disappear and four redundant chunks are detected. The stability is 4/6. Thus it can be seen with SW-RTI the stability of chunking performs well because it doest not affect the remaining redundant chunks detection even after certain modified chunks caused by bit shift problem.

4. **Performance Evaluations.** We evaluate our SW-RTI and concentrate on the two crucial parameters, i.e., the size of sliding window l and the average size of chunks r. To avoid too large or too small chunks we follow three rules, which are 1) if the length of packet payload is less than l then searching for dividing points stops and instead treats the whole payload to calculate its fingerprint, 2) if too many dividing points exist then we limit the least chunk size as m bytes and 3) if too few dividing points exist then we set the M bytes starting from the previous point as a new chunk and calculate its fingerprint. Note that the degree of too many or too few is defined empirically. We capture packets on our campus network for one hour by Wireshark tool and the total size of data is about 27.5G bytes.

4.1. **Effects of sliding window size.** We choose three different values of l, i.e., 16 bytes, 32 bytes and 48 bytes, to compare their effects on the results of redundancy detection. The results are shown in Table 1. As it can be seen that the value of l has little effects on the redundancy detection. On the contrary when we choose l, we should take into consideration the payload chunk. Smaller l results in lower correlations between chunks and payload contents while larger l results in much more computing time for Rabin calculations.

4.2. **Effects of average chunk size.** We choose the least k bits of the chunk for Rabin hashing. Suppose the k bits are randomly distributed, then every 2^k Rabin hash generates a dividing point. Note that r equals $2^k/8$ bytes. In the testing we choose r by 64 bytes, 128 bytes and 256 bytes to compare their contributions on the redundancy detection. The

TABLE 1. Effects of sliding window size on redundancy detection

Size of Window	Redundant traffic/GB	Ratio of redundancy
16 bytes	9.504	34.56%
32 bytes	9.517	34.61%
48 bytes	9.509	34.58%

Table 2. Effects of average chunk size on redundancy detection

Size of average chunk	Redundant traffic/GB	Ratio of redundancy
64 bytes	10.026	36.46%
128 bytes	9.792	35.61%
256 bytes	9.702	35.28%

corresponding value for k is 6, 7 and 8, respectively. The results are presented in Table 2. It is observed that the smaller the r is, the better the redundancy detection is, which means more redundant traffic is identified. This is due to the fact that smaller average chunk size causes more chunks, which increases the probability of blocks matching.

5. **Conclusions.** This paper addresses the issue of stability problem existing in redundant traffic identification (RTI) and proposes a sliding window based RTI. The main feature of SW-RTI is that it allows various lengths of chunks. SW-RTI utilizes Rabin hash to search for the dividing points and it improves the stability of chunk selection. Experimental results show the contributions of the size of sliding window and the average size of chunks to the redundancy detection. By choosing optimal parameters the RTI is more effective. In the future we will design efficient and effective redundant traffic elimination system based on our RTI works.

Acknowledgment. This research is supported by National Natural Science Foundation of China (Grant No. 61171109) and Basic Research Plan in SWUST (Grant No. 13zx9101). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] E. Halepovic, C. Williamson and M. Ghaderi, Enhancing redundant network traffic elimination, *Computer Networks*, vol.56, no.2, pp.795-809, 2012.

[2] S. H. Lim, Y. B. Ko and G. H. Jung, Inter-chunk popularity-based edge-first caching in content-centric networking, *IEEE Communications Letters*, vol.18, no.8, pp.1331-1334, 2014.

[3] L. Xing, Q. Ma and M. Zhu, Tensor semantic model for an audio classification system, *Science China Information Sciences*, vol.56, pp.1-9, 2013.

[4] A. X. Liu and M. G. Gouda, Complete redundancy removal for packet classifiers in TCAMs, *IEEE Trans. Parallel and Distributed Systems*, vol.21, no.4, pp.424-437, 2010.

[5] L. Xing, Q. Ma and L. Xu, A Cauchy-Laplace multifractal wavelet model for network redundant traffic, *Journal of Beijing University of Posts and Telecommunications*, vol.38, no.5, pp.54-57, 2015.

[6] S. A. Nsaif and J. M. Rhee, RURT: A novel approach for removing the unnecessary redundant traffic in any HSR closed-loop network type, *International Conference on ICT Convergence*, Jeju, Korea, pp.1003-1008, 2013.

[7] S. A. Nsaif and J. M. Rhee, RMT: A novel algorithm for reducing multicast traffic in HSR protocol networks, *Journal of Communications and Networks*, vol.18, no.1, pp.123-131, 2016.

[8] Q. Ma, L. Xing and B. Wu, Technique for authenticating scalable video coding streams with bandwidth awareness, *ICIC Express Letters, Part B: Applications*, vol.6, no.11, pp.3121-3126, 2015.

[9] N. T. Spring and D. Wetherall, A protocol-independent technique for eliminating redundant network traffic, *Proc. of ACM SIGCOMM*, Stockholm, Sweden, pp.87-95, 2000.

[10] A. Anand, C. Muthukrishnan and A. Akella, Redundancy in network traffic: Findings and implications, *Proc. of ACM SIGMETRICS*, Seattle, USA, pp.37-48, 2009.

[11] S. Schleimer, D. S. Wilkerson and A. Aiken, Winnowing: Local algorithms for document fingerprinting, *Proc. of ACM SIGMOD*, San Diego, CA, pp.76-85, 2003.

[12] B. Aggarwal, A. Akella and A. Anand, EndRE: An end-system redundancy elimination service for enterprises, *Proc. of USENIX NSDI*, San Jose, CA, pp.419-432, 2010.

[13] J. Sun, H. Chen, L. He and H. Tan, Redundant network traffic elimination with GPU accelerated rabin fingerprinting, *IEEE Trans. Parallel and Distributed Systems*, vol.27, no.7, pp.2130-2142, 2016.

[14] A. Muthitacharoen, B. Chen and D. Mazieres, A low-bandwidth network file system, *Proc. of ACM SOSP*, AB, Canada, pp.174-187, 2001.

[15] X. Bai, H. Yang and J. Zhou, Data-dependent hashing based on p-stable distribution, *IEEE Trans. Image Processing*, vol.23, no.12, pp.5033-5046, 2014.

[16] J. Wang, W. Liu and S. Kumar, Learning to hash for indexing big data – A survey, *Proc. of the IEEE*, vol.104, no.1, pp.34-57, 2016.