Vol.2013-MBL-67 No.12
Vol.2013-CDS-8 No.12
2013/9/13

Study of Compression Techniques for Congestion Control in Wireless Networks

Lee Chin KHO[1,a)] Yasuo TAN[1,b)] Azman Osman LIM[1,c)]

Abstract: Congestion definitely occurs when the offered traffic load exceeds available capacity at any point in the wireless networks. To reduce congestion in the wireless networks, a compression technique can be adopted as one of the viable solution. In this paper, we study the well-known lossless compression techniques and analyse their performance of the compression ratio. This paper also discusses the advantages and disadvantages of the compression techniques in the domain of congested wireless networks.

Keywords: congestion control, compression technique, compression ratio

1. Introduction

Congestion occurs when the resource demand exceeds the network capacity and the packets are lost due to too much queuing in the network. During congestion, the network throughput may drop to zero and the path delay may become very high [1]. The typical effects of congestion in networks are queuing delays, packet losses, blocked new connections, etc. causing the quality of service of the network to deteriorate; especially in wireless networks. There are numerous of congestion control for wired and wireless networks being introduced. Most of the congestion control include the algorithms of slow start, congestion avoidance, fast retransmit, and fast recovery to form a basic framework of TCP flow and congestion control. These four congestion control algorithms are executed at the source end and accomplish the congestion control through adjusting the parameters such as the congestion window (CWND), round trip time (RTT), retransmission timeout (RTO), the slow start threshold (ssthresh), etc. In spite of that, compression techniques can be adopted as one of the viable solution to reduce congestion, particularly in wireless networks.

With the growing coverage of wireless networks and the development of wireless technologies, the users are increasing exponentially and result to the limited wireless network bandwidth and easily congested networks. Compared to wired networks, wireless networks have a long propagation delay and higher bit error rate; therefore it is more difficult to maintain the quality of service. Through the help of compression techniques, the capacity of a communication channel can be increased by transmitting smaller data packets size and shorted transmission time. Compression techniques reduce the data packet size by removing redundant information in the data that should be transmitted.

There are numerous of compression techniques for the kind of application being introduced. For instance, the image compression such as GIF uses the Lempel-Ziv-Welch coding, BMP uses the run-length coding, JPEG uses the lossy discrete cosine transform, then Huffman or arithmetic coding, etc. In this paper, we aim to determine which compression techniques are more suits to the congestion control in wireless networks. Therefore, we study some well-known lossless compression techniques such as Run length coding (RLE), Huffman coding, arithmetic coding, and Lempel-Ziv-Welch (LZW) coding. Their performances of the compression ratio are also analyzed.

The main contribution of this research is to provide a comparative study of the four well-known lossless compression techniques: RLE, Huffman coding, arithmetic coding and LZW coding. This study enables us to understand the basic characteristics, target domain, complexity level, and features of the four lossless compression techniques. The advantages and disadvantages of those compression techniques in the domain of congested wireless networks are also discussed.

The rest of this paper is organized as follows. Section 2 is devoted to the related works of compression techniques and Section 3 discusses the taxonomy of compression techniques. The lossless compression technique of RLE, Huffman coding, arithmetic coding, and LZW coding are deeply discussed in the Section 4. The results and discussion are shown in Section 5 and conclude the paper in Section 6.

2. Related Works

There are a number of studies that have addressed the compression techniques in different kind of application. The most common applications are data compression, image compression, voice compression, etc. M. A. Laham al. et. had presented a comparative study between various algorithms of data compression

[1] School of Information Science, Japan Advanced Institute Science and Technology (JAIST),1-1 Asahidai, Nomi City, Ishikawa, 923-1211, Japan
[a)] s1120203@jaist.ac.jp
[b)] ytan@jaist.ac.jp
[c)] aolim@jaist.ac.jp

techniques in 2007 [2]. In the paper, the compression techniques such as RLE, Huffman, LZ 77, and LZW are briefly discussed. Then, the author compared the compression ratio of LZW and Huffman for the file of .DOC, .BMP, .JPG, and .GIF. Based on the results, LZW performed badly in image data compression, especially .GIF and .JPG file. LZW enlarges the file size to maximum 40% of the input file size, whereas Huffman enlarge 5% of the input file size. Moreover, LZW performed as well as Huffman in text format of .DOC file. Both techniques achieve 80% of compress ratio in text file. Unlikely, the results of compression ratio for both techniques are inconsistent. Huffman sometime performs better than LZW or the opposite way. We improve the above paper by determining the percentage of average maximum redundancy in a file before compress, so that the relationship of redundancy and compression ratio for compression techniques of RLE, Huffman, arithmetic and LZW can be identified.

In addition, S. Shanmugasundaram al. et. in [3] had provided a survey of text compression algorithms based on statistical and dictionary. The author used a parameter metric of Bit Per Character (BPC) to compare the performance of compression algorithms in twelve different text files. The results showed that the average BPC for RLE was around 8 bits, whereas Shannon Fano coding, Huffman coding, Adaptive Huffman coding and arithmetic coding of BPC is between 5 to 6 bits. Although this parameter metric showed the bits reduction per character among the different compression techniques, it cannot present the overall performance of the compression in term of complexity, time, efficiency, etc.

The others related compression survey such in [4-6] focus on the techniques to compress the data rather than the compression performance. They review different compression techniques by giving a few examples of the way to compress and decompress the source data.

3. Taxonomy of Compression Techniques

In a network flow, compression is performed at the source node before the data packet is fed into the transmitter and decompresses the data packet at the receiver node when the signal has detected. Thus, compression can be assumed as an end-to-end functionality. Therefore, the main consideration of the congestion control by applying compression in wireless networks will be type of compression techniques, type of data to be compressed, compression ratio, etc. Taxonomy is developed in this paper to clarify the types of compression techniques that is existed. Fig. 1 shows a taxonomy of compression techniques.

In terms of data packet point of view, compression techniques can be categorized into two parts: header compression and payload compression. Typically, packet header contents critical information about routing the data to its destination. Thus, no information loss can be accepted in the packet header. While in lossy compression, some information loss is acceptable since the loss information cannot be sensed by the human eye. For instance, the human eye is more sensitive to subtle variations in luminance than variation in color. JPEG image compression works in part of rounding off less important information. In other words, lossy compression technique cannot be used in header compression.

In the lossless compression techniques, the information after

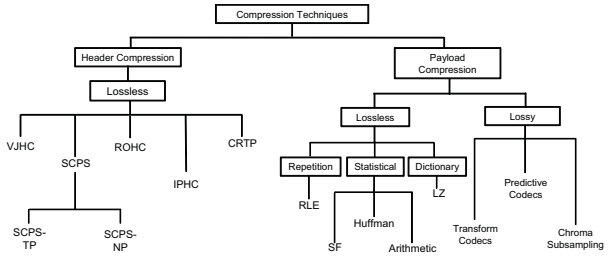

Fig. 1 Taxonomy of compression techniques

decompress should not be changed whatsoever. There are few examples of header compression techniques are shown in Fig.1. Van Jacobsons header compression (VJHC) is one of the compression techniques that improve the ratio of data to total bytes transferred over a link. It reduces the normal 40 byte TCP/IP packet headers to 3-4 bytes for the average case by sending the differences in the header field that change.

Space Communication Protocol Specification (SCPS) was designed to operate over any kind of space mission or infrastructure, regardless of complexity. SCPS redefines the network stack from the network layer down with variations to known protocol (i.e. SCPS-NP is used instead of IP). Modification to TCP is done through the use of TCP extensions or option as specified in SCPS-TP.

Robust Header Compression (ROHC) is a standardized method to compress the IP, UDP, UDP-Lite, RTP and TCP header of Internet packets. The main tasks of the ROHC compressor are 1) compress the receiving RTP/ UDP/ IP / TCP into the appropriate compressed packets and send them to the decompressed; 2) deal with the feedback information from the decompression, and finish the conversion of states and modes; 3) manage the CIDs reasonably and effectively

IP header compression (IPHC) is improved and extended from VJHC. It is mainly used for packet over TCP/IP and UCP/IP in low speed links. IPHC uses different compression techniques between TCP data streams and non-TCP streams. For the TCP based data streams, the compression algorithms are same with VJHC. For the non-TCP based data streams, the packets arrive randomly, thus the compressor cannot reduce the number of sending bytes by differential encoding, so IPHC sends the full changed fields without modification.

Real Time Protocol Compression (CRTP) is used to compress the typical real time multimedia data packets. However, CRTP also can compress the combination of RTP, UDP and IP header. The 40 bytes of RTP/ UDP/ IP packets with UDP checksum can be compressed to 4 bytes. CRTP can work well on the small RTT link. In the long RTT, the compressor and decompressor cannot achieve a good synchronization, which lead to a series of packet loss. Moreover, CRTP takes up a lot of bandwidth when it sends the complete header information to update the content. Therefore, CRTP is not suitable for wireless networks.

In the payload compression, lossy and lossless compression techniques are applicable depend on the type of data packets and the user requirements. The lossy compression techniques can be categorized into three types: transform codecs, predictive codecs, and chroma subsampling. The transform codec compression is

generally used for JPEG images only. In the transform codecs, the samples of picture are taken and chopped into smaller segments before transforming into a new image. Whereas in predictive codecs, previous and/or subsequent decoded data is used to predict the compressed image frame. For the chroma subsampling, it takes into account of the human eye perceives changes to average or drop some chroma information while maintaining Luma information.

In this paper, the study of lossless compression techniques will be focused. Therefore, the well-known techniques such as Run Length (RLE) coding, Huffman coding, arithmetic coding, and LZW will be further discussed in the section 4.

4. Lossless Compression Techniques

The lossless compression can be categorized into three models: repetition, statistical, and dictionary. The repetition model reduces the redundant information by encode the repeat symbols into the length of string and symbols. The compression technique that implements the repetition model is Run-Length (RLE) coding.

The statistical model is based on the symbols and the probability distribution of the source. The most common compression techniques that use statistical model are Huffman coding, Shannon Fano (SF) coding, and arithmetic coding. Meanwhile, the dictionary model relies upon the observation of correlations between the parts of data. It replaces those redundant by references to a dictionary that contain the original. The following is the detail of RLE, Huffman, arithmetic, and LZW coding.

4.1 Run-Length Coding (RLE)

RLE is created to encode the data with a string of repeated symbols. For example, the text with kkkkhoho is the source to compress; the first four letters are a run with length 4 since there is repetition of symbol k, while the next 4 letters are non-run with length 4. The RLE encoding algorithm compresses the runs of the original file and keeps the non-runs from the compression process. If the string of repeated symbols is large, the size of the output data will be significantly reduced. In contrast, the size of output data can be double the size of the input data during the worst case. Therefore, RLE is usually used as a pre-compress for other compression techniques such as Huffman and arithmetic.

4.2 Huffman Coding

Huffman coding is based on statistical method where the probability distribution of the character from the source is used to develop the code words for symbols. The frequency of each symbol is calculated to determine the probability distribution. The code words are assigned based on the probabilities. Code words with shorter length are for higher probabilities and code words with longer length are for smaller probabilities. This property can be achieved by constructing a binary tree where the symbols are acting as leaves based on their probability and the paths are acting as the code words. For example, the source of kkkkhoho;

If a block code of character is eight bits, then the source data consist of 64 bits. By using Huffman coding, the code words for k becomes one bits, h becomes two bits, and o becomes two bits.

Table 1 Huffman Coding Example

Element	Frequency	Huffman Code
k	4	1
h	2	01
o	2	00

Then, the source data can be compressed to 12 bits. However, the table includes character and codes for each character need to be transmitted together with the encoded output to the receiver for decompression process. The size of this table depends on the file being compressed, usually in range of 500 to 1200 bytes. By this, the transmitted packet size can not be reduce much.

There are two types of Huffman coding algorithms: static Huffman algorithms and adaptive Huffman algorithms. Static Huffman algorithm compresses the source data by determining the frequencies of character and generate a common tree for both compression and decompression processes. Unlikely, the detail of the tree need to be transmitted with the compressed file, which will enlarge the data packet. For the adaptive Huffman algorithms, there are two trees in the compression and decompression processes. A tree is generated with the flag symbol in the beginning and is updated as the next symbol is read.

4.3 Arithmetic Coding

In arithmetic coding, code words are used to represent a fraction that denote as the entire source message. The occurrence probability and cumulative probability are used to calculate the set of code words representation. The cumulative probabilities of a symbol range are calculated at the beginning of encoding process. The source of character is read one by one and selects the corresponding cumulative probability range of the character. When the character is read, the corresponding sub range is selected and divided into subsections according to the probabilities of the symbol. This process is repeated until the end of the message is encountered. A number from the final sub range of cumulative probabilities is taken as the output of the encoding process. This output will be a fraction that represents the entire message. The information of the probability distribution and number of characters of the source message is transmitted to a receiver for decoding process.

4.4 Lempel-Ziv Coding

Lempel-Ziv coding was proposed by Jacob Ziv and Abraham Lempel in 1977 and 1978. Lempel-Ziv coding is a lossless data compression that based on dictionary scheme. It can be divided into two types: LZ77 and LZ78 as shown in Fig. 2 for the evaluation of Lempel -Ziv. LZ77 family is based on the sliding window algorithm to generate the dictionary and LZ78 forms to the dictionary by parsing the source data. The detail of LZ77 and LZ78 is described in the following section.

4.4.1 LZ77

In the LZ77 approach, it exploits the words and phrases in the data source with repetition to encode. LZ77 encodes the repetition by a pointer to the number of characters that match with prior occurrence. The dictionary in LZ77 is the portion of the previously encoded sequence.

In the encoded processes, the input sequence is examined

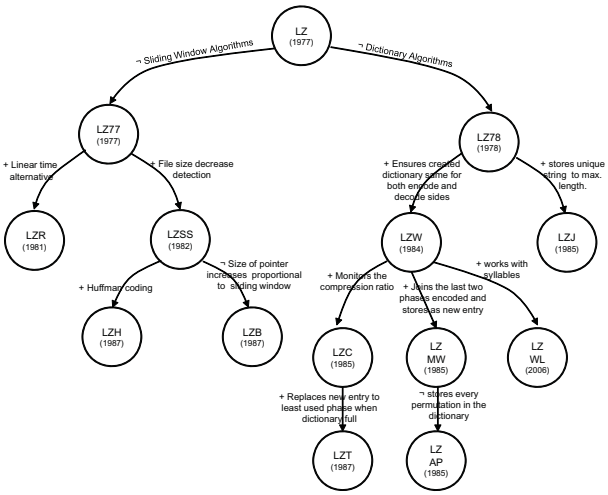

Fig. 2 Taxonomy of LZ compression techniques

through a sliding window that consists of two parts: search buffer that contains a part of encoded sequence and look forward buffer that contain the next part of the sequence to be encoded. The longest sliding window that matches with the beginning of look forward buffer is searched and then outputs a pointer to that match. If there is no match, the output cannot contain just pointers. Generally, the pointer in LZ77 is always output as (offset, length, symbol). The offset is referred as offset of the match, the length is referred as the length of the match, and symbol is referred as the next symbol after the match. If there is no match, the offset and the match length will be equal to 0 and the symbol is equal to the first symbol in the look forward buffer.

There are several variations on LZ77 technique, but in this paper, we only describe a few well know LZ techniques that under LZ77 such as LZR, LZSS, LZH, and LZB as shown in Fig.2. LZR can be stand for Lempel-Ziv-Rodel. It is a modification of LZ77 that aim to be a linear time alternative. Although the pointer in LZR can be pointed to any offset in the file, it consumes amount of memory. Therefore, it is an unfeasible variant.

Lempel-Ziv-Storer-Szymanski (LZSS) technique is an improvement of LZ77 with the function of determining whether a substitution will increase the file size or not before proceeding the encoder process. If there is no size reduction, the input will not be encoded. Otherwise, the input is replaced with (offset, length) pair of the character read from the position. Moreover, LZSS eliminates the next character or symbol from the output. It only uses offset-length pair. Lempel-Ziv-Huffman (LZH) and Lempel-Ziv-Bell (LZB) are the variant of LZSS. LZH uses the Huffman coding to compress the pointer to increase the compression ratio, while LZB gradually increases the size of the pointer as the sliding window increases to improve the encoded LZSS pointer compression. LZB achieves a higher compression ratio than LZSS and LZH, but it has taken more time to process due to the extra encoding step for pointers.

4.4.2 LZ78

Instead of using the sliding window method to build the dictionary, LZ78 form a dictionary as the file is parsed. LZ78 adds each newly encountered character to the dictionary while parsing the file. A dictionary entry with the form of (D_index, N_symbol) is generated for each symbol in the input is read. If a symbol exists in the dictionary, the dictionary will search for substrings of the current symbol and following symbol. The index of the longest substring match is selected for the D_index. The last character that data pointed for the D-index is added to N_symbol). If the current symbol is unknown, the D_index is set to 0 to indicate that it is a single character entry. Thus, the entries form a linked-list type data structure.

LZW is the most commonly used of LZ78 family. It removes redundant characters in the output and makes the output entirely out of pointers. It includes every character in the dictionary before start to compress and improves the compression by encode the last character of every new phase as the first character of the next phase.

Lempel-Ziv-Compress (LZC) is the modification of LZW that include the monitoring function on the compression ratio of the output. Once the ratio over a certain threshold, the dictionary is discarded and rebuilt. While Lempel-Ziv-Ticher (LZT) improves the LZC by replacing the least recently used phase with a new entry when the dictionary is full. Lempel-Ziv-Miller-Wegman (LZMW) uses the method of joining the last two phases encoded and stores the result as a new entry to solve the dictionary overload problem. Whereas Lempel-Ziv-All-Prefixes (LZAP) which modify from LZMW store every permutation in the dictionary rather than store a single phase in the dictionary for each iteration. LZWL is one of the variants of LZW. It is created to work on certain data sets such as XML.

Lempel-Ziv-Jakobson (LZJ) is one of the LZ78 variants that deviates from LZW. It stores every unique string until the maximum length in the dictionary and assigns codes to each of them. When the dictionary is full, entries that occurred once will be removed.

5. Results and Discussion

5.1 Performance Evaluation

In this paper, the lossless compression techniques such as RLE, Huffman coding, arithmetic coding, and LZW performance are examined with the following measurement.

Compression Ratio (CR) which is the ratio between the size of compressed file and the size of the source file.

$$CR = \frac{size\ after\ compression}{size\ before\ compression} \quad (1)$$

Improvement percentage (IP) is the reduction of the source file in percentage.

$$IP = \frac{size\ before\ compression\ -\ size\ after\ compression}{size\ before\ compression} \quad (2)$$

Moreover, the time efficiency and space efficiency of the compression techniques also measure through calculating the time and memory need to compress the file.

Instead of determining the relationship between the type of file and the compression ratio, we identify the relationship of redundant information with compression ratio. Therefore, we calculate the average maximum redundancy in percentage of the character in the file for RLE, Huffman coding and arithmetic coding compression and average maximum redundancy in percentage of

the words or phases pattern for LZW compression. The redundancy in this context is referred as the total number of reflections scanned divided by the total number of unique reflection. In this paper,a text format of three online articles from "Time" are selected as sample test.

5.2 Simulation Results

In order to obtain the average maximum redundancy from an article, the frequency of each character in an article is calculated and find the average maximum of the unique reflection. The Table 2 shows the data redundancy calculation for the selected three articles and AMR. in Table is denoted as Average Maximum Redundancy.

Table 2 Data Redundancy Calculation

File Name	Total Character	AMR. of Character	AMR. of word Pattern
Article 1	6247	12.0%	9.1%
Article 2	6242	12.1%	6.3%
Article 3	2982	10.9%	5.6 %

Then, the Compression Ratio (CR), Improvement Percentage (IP), compression processing time (Time) in seconds, memory space need during compression (Space) in byte are calculated for the RLE coding, Huffman coding, arithmetic coding, and LZW coding. The results are shown in Table 3 to Table 6.

Table 3 Compression results for RLE Coding

Article	Input Size	Compression Size	CR	IP (%)	Time (ms)
1	1200	1085	0.904	14.0	1700
2	1300	1159	0.891	10	1765
3	700	637	0.911	9	958

Table 4 Compression results for Huffman Coding

Article	Input Size	Compression Size	CR	IP (%)	Time (ms)
1	1200	741	0.618	38	2743
2	1300	754	0.580	42	2897
3	700	553	0.790	21	1876

Table 5 Compression results for Arithmetic Coding

Article	Input Size	Compression Size	CR	IP (%)	Time (ms)
1	1200	697	0.580	41	4741
2	1300	636	0.489	51	4912
3	700	418	0.598	40	2709

Table 6 Compression results for LZW Coding

Article	Input Size	Compression Size	CR	IP (%)	Time (ms)
1	1200	1025	0.854	1.45	2193
2	1300	882	0.678	31	1
3	700	379	0.542	45	1

5.3 Result Discussion

The simulation results showed that RLE coding perform poorly for text format file. This is due to a repetition sequence characters in text format seldom exists. However, RLE can perform well for the image having solid black pixels. In addition, RLE takes the least time to compress the data compare to others compression techniques.

5.4 General Discussion

In this section, the lossless compression techniques of RLE coding, Huffman coding, arithmetic coding, and LZW are classified into the target domain and complexity degree. Moreover, the advantages and disadvantages of the compression techniques in the domain of congested wireless networks will also discuss here.

The Table 7 shows the target domain and the complexity degree of these lossless compression techniques. The target domain in here is classified into three: text format, figure format and both. The aim is to categorize the type of data format that suit for these four compression techniques.

There are three degrees of complexity define in this context: low, medium, and high. In the low complexity, the time needed to compress a data is low. When the complexity degree is getting higher, the time needed to compress a data will get higher.

Table 7 Lossless Compression Techniques with Target Domain and Complexity Degree

Compression Techniques	Target Domain	Complexity Degree
RLE	Image	Low
Huffman	Both	Medium
Arithmetic	Both	High
LZW	Text	High

In the congested wireless network, the bandwidth capacity is overloaded. The congestion control in TCP will reduce the transmission rate by decrease congestion window to mitigate the congestion problems. However, congestion control in TCP has a difficulty to distinguish the congestion is due to buffer overflow or the medium contention and poor radio link for wireless networks. Therefore, instead of reducing the transmission rate to mitigate the congestion problem, the compression can be one of the solution in congestion control for the wireless networks. The advantages of compression techniques in congestion control is they have the ability to reduce the number of bits required to represent data and decrease the transmission time.

Unlikely, the compression techniques have some drawbacks in network point of view. There is no lossless data compression that can guarantee the compression of all input data. Sometimes, the lossless compression techniques may larger the file size instead of reduce. By this way, the compression may cause the congestion ever worst.

However, with the hybrid technology of prediction, compression, network coding, and TCP congestion control, the congestion in the wireless networks is believed can be significantly reduced.

6. Concluding Remarks

We have presented a comprehensive review of the four types of lossless compression techniques; RLE is coding, Huffman coding, arithmetic coding, and LZW coding. Based on this study, RLE coding is not suitable for text compression. While other compression techniques can be used to compress the data text format. In future work, the joint technique of compression and congestion control in TCP will be studied.

References

[1] R. Jain, and K. K. Ramakrishnan: Congestion avoidance in computer networks with a connectionless network layer: concepts, goals, and methodology, *Proc. Comput. New. Symp.* , Washington, USA, pp.134 143 (1988)

[2] M. A. Laham, and I. M. M. E. Emary: Comparative study between various algorithms of data compression techniques, *World Congr. on Eng. and Comput. Sci.* , San Francisco, USA, pp. 326 336, 2007.

[3] S. Shanmugsundaram, and R. Luordusamy: A comparative study of text compression algorithms, *Int. J. of Wisdom Based Comp.* , vol. 1, no. 3, pp. 68 67, December 2011.

[4] Sashikala, Y. Melwin, S. S. Arunodhayan, and M. N. Nachappa: A survey of compression techniques, *Int. J. of Recent Technol. and Eng. (IJRTE)* , vol. 2, no. 1, pp. 152 156, March 2013.

[5] D. Kaur, and K. Kaur: Analysis of lossless data compression techniques *Int. J. of Comp. Eng. Research* , vol. 3, no.4, pp. 123 127, April 2013

[6] V. S. Gulhane, and M. S. Ali: Survey over adaptive compression techniques *Int. J. of Eng. Sci. and Innovative Technol.(IJESIT)* , vol. 2, no. 1, pp. 152 156, January 2013

[7]