Research Journal of Applied Sciences, Engineering and Technology 5(22): 5320-5324, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: January 10, 2013 Accepted: January 31, 2013 Published: May 25, 2013

Research of SIP Compression Based on SigComp

[1]Derong Du, [2]Jianming Liu, [1]Hongzhou Li and [1]Zhiyong Peng
[1]School of Electric Engineering and Automation,
[2]School of Computer Science and Engineering, Guilin University of Electronic Technology,
Guilin 541004, China

Abstract: SIP (Session Initiation Protocol) has been chosen as the core signaling protocol of the NGN (Next Generation Network), but the large SIP message which is text-based is an obstacle with the planned usage of SIP in wireless mobile networks. Based on the SigComp (Signaling Compression) framework, some further improvements are made to the Deflate algorithm according to the characteristics of SIP in this study. Experiments show that the improved Deflate algorithm can compress the SIP message greatly and reduce the bandwidth requirements signally, so it is highly valued in IMS (IP Multimedia Subsystem), PTT (Push To Talk) and other wireless real-time SIP applications.

Keywords: Deflate, SigComp, sip, wireless mobile network

INTRODUCTION

SIP is developed by IETF to set up, modify and terminate sessions for multimedia communication (Rosenberg *et al*., 2002). Because of its convenience to implement, diagnose and extend, SIP has been chosen as the core signaling protocol of the NGN. Now, SIP is used widely and it still keeps on being updated and improved. However, SIP is text-based application protocol and it was originally engineered for bandwidth rich links. So, its messages have not been optimized in terms of size and the large SIP messages become an obstacle with the planned usage of SIP in wireless mobile networks.

The networks such as 2.5G, 3G or Wi-Fi are all shared wireless mobile networks. One single user only has very limited bandwidth in the peak of the networks or when many network applications are running. Right now, the large SIP messages will consume too much bandwidth and memory and also cause higher session setup delay. The higher session setup delay means the lower QoS(Quality of Service) of the real-time SIP applications(e.g., IMS and PTT).

The session setup delay mainly depends on message length (Wen *et al*., 2011) and compressing messages can improve the transmission performance and reduce the delay. So, 3GPP release 5 requires that SIP messages should be compressed before transmission in the wireless mobile networks.

For a better utilization of text-based application protocols (e.g., SIP), IETF defines SigComp (Price *et al*., 2003) as a new layer between the application and the underlying transport. SigComp can offer robust, lossless compression and decompression of application messages. The compression efficiency of SigComp largely depends on the selected compression algorithm, compression mechanisms and compression/ decompression memory size. In this study, we mainly study the compression algorithm.

SIGCOMP ARCHITECTURE AND THE CHARACTERISTICS OF SIP

SigComp architecture: For now, the main SigComp research organizations are 3GPP and IETF ROHC (Robust Header Compression) (Li, 2009) which have released a number of relevant standards. With the wide usage of the text-based application protocols in wireless mobile networks, SigComp has been increasingly concerned and emphasized. SigComp supports a wide range of compression algorithms and transports (e.g., TCP, UDP and SCTP) and its architecture is illustrated in Fig. 1.

SigComp consists of Compressor Dispatcher, Compressor, Decompressor Dispatcher, UDVM (Universal Decompressor Virtual Machine) and State Handler. During the compression process, the Compressor Dispatcher receives the messages with compartment identifier from local application and invokes a particular Compressor according to the compartment identifier; the Compressor compresses the messages by utilizing a certain compression algorithm and accessing existing state and returns SigComp messages to the Compressor Dispatcher; the

Corresponding Author: Derong Du, School of Electric Engineering and Automation, Guilin University of Electronic Technology, Guilin 541004, China

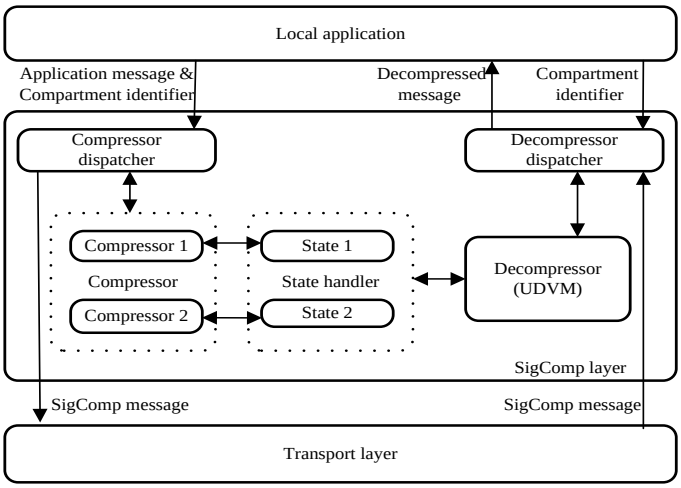

Fig. 1: SigComp architecture

Compressor Dispatcher then forwards the SigComp messages to the remote endpoint. During the decompression process, the Decompressor Dispatcher receives the SigComp messages and invokes an instance of the UDVM; the UDVM decompresses the SigComp messages by utilizing the corresponding decompression algorithm and accessing existing state and returns the decompressed messages to the Decompressor Dispatcher; the Decompressor Dispatcher then forwards the resulting decompressed messages to local application, which may return a compartment identifier in order to update state.

The characteristics of SIP: SIP is a text-based protocol that uses the UTF-8 charset. A SIP message is either a request from a client to a server, or a response from a server to a client. Both types of messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields and an optional SDP (Session Description Protocol) message-body.

Message = start-line (Request-Line/Status-Line)
*message-header
CRLF
[message-body]

The start-line, each message-header line and the empty line must be terminated by a Carriage-Return Line-Feed sequence (CRLF). Note that the empty line must be present even if the message-body does not exist. (Rosenberg *et al*., 2002)

By analyzing the literature (Johnston *et al*., 2003), we can see that SIP messages have three characteristics:

- The format of the message is fixed, but its size is not. A typical SIP message ranges from a few hundred bytes up to two thousand bytes or more. It consists of SIP instructions and user data and SIP instructions (e.g.,"SIP", "From" and "To") frequently appear in each SIP message.
- During the same session process, a large quantity of redundancy exists among SIP messages (requests/responses). For example, each SIP message has the same value of URI, IP and Call-ID.
- SIP header accounts for the majority of the size of the SIP message.

DEFLATE ALGORITHM

SIP is a text-based protocol, so it needs lossless compression. The lossless compression algorithm (e.g., Huffman, RLE, LZ and Arithmetic) (Li and Yang, 2010) compresses messages by reducing redundancy among messages. Deflate performs best among the common lossless compression algorithms (Jin and Mahendran, 2005) and it was specified in RFC1951. So, we choose Deflate as the basic compression algorithm in this study.

Deflate is a dual compression algorithm and it combines LZ77 algorithm and Huffman coding. Today, there are many software implementations such as PKZIP, zlib and 7-Zip/AdvanceCOMP (Alireza and Mahmoud, 2011).

LZ77 Algorithm: LZ77 algorithm has low compression ratio and it is ideally suited to compress the real-time messages (Tian, 2010). LZ77 is sometimes called sliding window algorithm and it maintains two buffers (Fig. 2): Historical Buffer and Forward Buffer. The former keeps the data existing earlier in the input (uncompressed) data stream and the latter keeps the current data that is about to be compressed.

LZ77 algorithm achieves compression by trying to search the match of current data of Forward Buffer and replacing the current data with a pointer to the match from Historical Buffer. If the match exists, LZ77

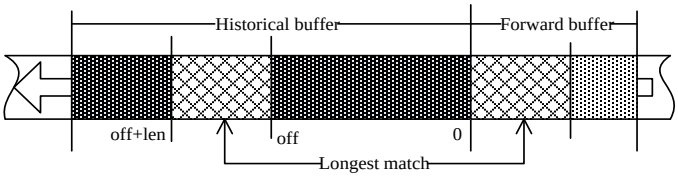

Fig. 2: LZ77 compression process

algorithm continues to search the longest match and it outputs the pointer of the longest match in the form like (off, len). At the same time, the current data moves into Historical Buffer and the len long earliest data of Historical Buffer moves out. If the match does not exist, LZ77 algorithm outputs the current (uncompressed) data. At the same time, the one current data moves into Historical Buffer and the one earliest data of Historical Buffer moves out. In order to optimize the search time, Deflate algorithm uses a hash table to maintain and organize Historical Buffer.

LZ77 algorithm sets the first bit as a flag to distinguish the raw data from the (off, len) pair. Flag = 0 indicates the raw data and Flag = 1 indicates the (off, len) pair.

LZ77 decompression algorithm is very simple. It also maintains a sliding window and gets the raw data from compressed data according to the (off, len) pair.

Huffman Coding: Huffman coding is an entropy encoding algorithm and it expresses the most common source symbols using shorter strings of bits than the less common source symbols. The core work of the Huffman coding is to build Huffman Tree (also called Optimal Binary Tree).

Huffman coding is the first truly practical coding method with high efficiency and it is also quite easy to meet the requirement. But, Huffman coding has not reliability protection mechanisms. For example, even just one bit error can cause a train of errors (called Error Propagation) during the Huffman decoding process (Li, 2009).

IMPROVEMENTS TO DEFLATE

In order to enhance the compression efficiency of SIP, some further improvements are made to the Deflate algorithm:

- **Pre-loading data:** The SIP/SDP Static Dictionary specified by the standard (Garcia-Martin *et al*., 2003) is pre-loaded as a Sig Comp state and the REGISTER message is pre-loaded into the Historical Buffer of LZ77 after SIP application registers successfully. The Static Dictionary is a collection of well-known strings that appear in the most of the SIP/SDP messages. The REGISTER message also contains very important information such as the value of user URI and IP. So, pre-loading data is very useful to compress SIP messages especially the first several messages.
- **Building the User-specific Dictionary and SIP Phrases Frequency Table:** Both sent and received SIP (uncompressed) messages move into the Historical Buffer as the sliding dictionary during the session setup process. Meanwhile, SIP Phrases Frequency Table is built to record the frequency of the SIP phrases (except the phrases that appear in the Static Dictionary). If the frequency of one phrase increases to a certain value, the phrase will be saved as a Sig Comp state. In addition, the phrases with the lowest frequency must move out if the Frequency Table is full. A large quantity of redundancy exists between SIP requests and responses in the same session, so saving both sent and received SIP (uncompressed) messages is necessary. The SIP Phrases Frequency Table is very important to build a high-efficiency dictionary for LZ77 algorithm and it is also useful to build Huffman Tree for Huffman coding.
- **Regularly updating Huffman Tree:** We set a variable *"F"* as the flag and *"F"* reduces by one for every Huffman coding. When *"F"* reduces to zero, the Huffman Tree will be updated and then *"F"* is set to *n*. Regularly updating Huffman Tree can avoid frequently updating Huffman Tree and it also signally reduces the system overhead and coding time.
- **A simple error detection and handling mechanism:** SIP application detects whether the value of Call-ID of every received message is right during the same session. If the value is wrong, Sig Comp will initialize the Deflate immediately. That is, Sig Comp empties the Historical Buffer, SIP Phrases Frequency Table and Huffman Tree and frees the state mentioned in (ii). At the same time, SIP application sends a SIP error message to the remote endpoint and the remote endpoint also initializes the Deflate immediately after receiving the SIP error message. The error mentioned here only comes from Sig Comp rather than the raw SIP message. The error must be handled, or else it will cause Error Propagation and retransmission. The value of Call-ID is unique and remains unchanged during the same session, so it can act as the detection string.

The improved Deflate compression process is illustrated in Fig. 3. The improved Deflate

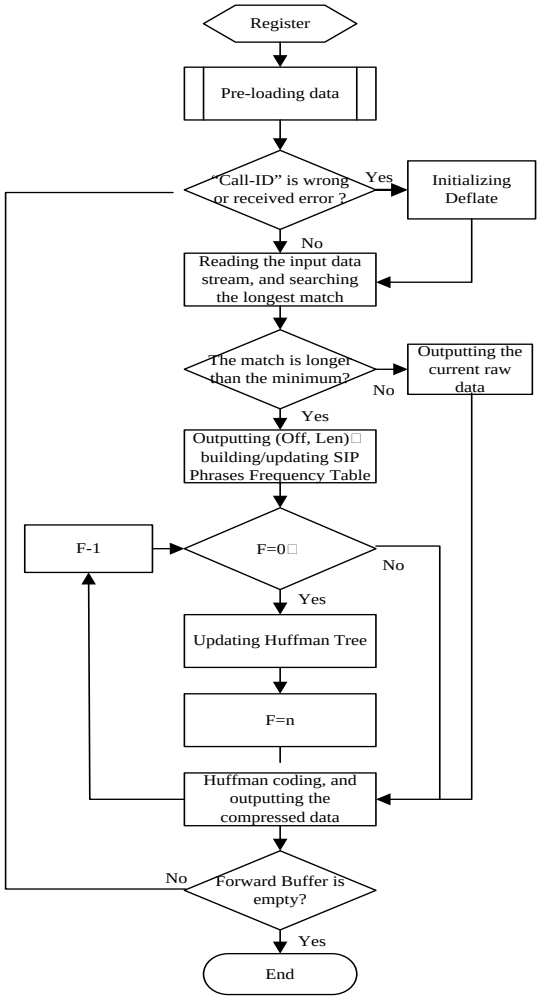

Fig. 3: The improved deflate compression process

decompression is simple, so we no longer give unnecessary details here.

PERFORMANCE ANALYSIS

To validate the performance of the improved Deflate, a simple session scene (Fig. 4) is constructed. During the session, X-Lite1 and X-Lite2 softwares act as the SIP clients and Mini Sip Server software acts as the SIP server. In addition, Wireshark software captures the sequence of SIP messages. The SIP messages are compressed using Deflate algorithm, Deflate+Static Dictionary algorithm (DefDic for short) and the improved Deflate algorithm (ImpDef for short). The compression results of every SIP message are shown in Table 1 and Fig. 5 and the several compression ratios of the total messages separately sent by X-Lite1, X-Lite2 and MiniSipServer are shown in Table 2.

The analysis of the results is as follows:

- As shown in the Table 2, ImpDef has the lowest total compression ratio among the three algorithms

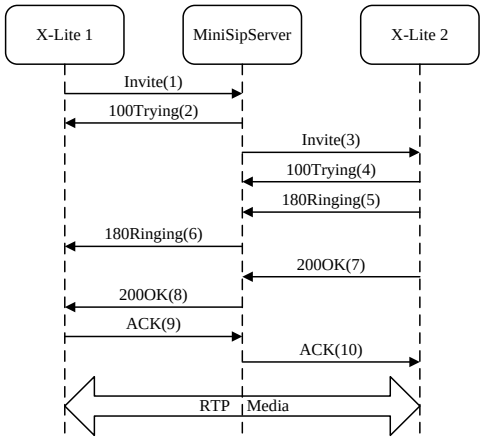

Fig. 4: A simple session setup process

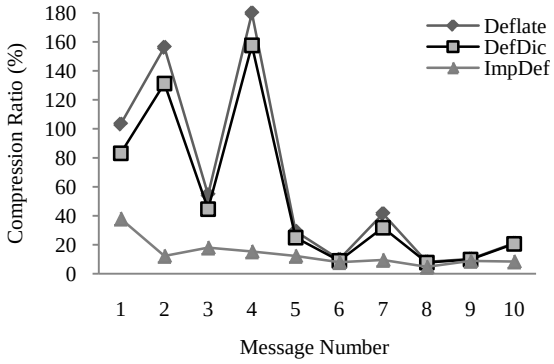

Fig. 5: Compression ratio of every SIP message

Table 1: The compression results

Number	Message (byte)	Deflate (byte)	DefDic (byte)	ImpDef (byte)
1	1024	1057	851	389
2	393	614	516	48
3	800	436	357	144
4	287	516	452	44
5	394	115	98	48
6	507	50	45	40
7	781	323	248	74
8	793	64	61	37
9	645	65	63	57
10	383	81	79	32
Sum	6007	3321	2770	913

Table 2: The compression ratios for senders

Sender	Deflate (%)	DefDic (%)	ImpDef (%)
X-Lite1	67.23	54.76	26.72
MiniSip server	43.29	36.79	10.47
X-Lite2	65.25	54.58	11.35
The total	55.29	46.11	15.20

Due to the Static Dictionary, the DefDic's compression ratio is lower 9.18% than Deflate. But, ImpDef has the more and the higher efficient dictionary resource, so its compression ratio can be as low as 15.2%.

- As shown in the Fig. 5, ImpDef performs best among the three algorithms in compressing the first

SIP messages (number 1, 2 and 4) sent separately by X-Lite1, MiniSipServer and X-Lite2. When compressing the first SIP message, Desflate has not any dictionary resource; and besides, it must add some additional information such as bytecodes and flag bits to the compressed message. As a result, Deflate performs worst and even its compressed message is longer than the raw message. Due to the Static Dictionary, the DefDic performs a little better than Deflate, but it is far from enough. ImpDef has enough dictionary resource by pre-loading data, so it performs best.

- According to the Fig. 4 and 5 and Table 2, we can see that the more messages the sender sends, the better Deflate and DefDic perform. For example, MiniSipServer sends the most messages in the session and the total compression ratio of its messages is lowest. Similarly, the more messages the sender sends and receives, the better ImpDef performs. But, the Compression ratio of the three algorithms changes more smoothly (especially when Historical Buffer is full) at the same time. However, the Compression ratio of ImpDef changes still faster than Deflate and DefDic due to its increasingly efficient dictionary.
- The results mentioned above are got on the condition that no error occurs during the compression/decompression process. If not, all the three algorithms would perform badly. However, ImpDef would still perform best among the three algorithms due to its simple error detection and handling mechanism. On the contrary, the error might cause Error Propagation or session setup failure during the compression/decompression process of Deflate or DefDic.

CONCLUSION

In general, wireless real-time SIP applications (e.g., IMS and PTT) require the session setup to be finished instantaneously. So, the large SIP messages should be compressed before transmission to reduce session setup delay.

In this study, we deep analyze the SigComp architecture, Characteristics of SIP and Deflate algorithm and give some further improvements to the Deflate algorithm. The test results prove that the improved Deflate algorithm can compress the SIP messages greatly and reduce the bandwidth requirements signally. So it is highly valued in the wireless real-time SIP applications and also has some reference for compressing other text-based protocols such as RTSP (Real Time Streaming Protocol).

ACKNOWLEDGMENT

The authors wish to thank the help of the funds: The National Natural Science Foundation of China under Grant No.61262074 and No.61162008; Guangxi Department of Education Fund, China (201204LX126, 201101ZD006); Open Project of Guangxi Key Lab of Trusted Software, China (kx201101); Program for Excellent Talents in Guangxi Higher Education Institutions, China (201065).

REFERENCES

Alireza, Y. and R.H. Mahmoud, 2011. A simple lossless preprocessing algorithm for hardware implementation of deflate data compression. Proceeding of IEEE 19th Iranian Conference on Electrical Engineering (ICEE). Tehran, pp: 1-5.

Garcia-Martin, M., C. Bormann, J. Ott, R. Price and A.B. Roach, 2003. The Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Static Dictionary for Signaling Compression (SigComp). IETF RFC3485, Retrieved from: http://www.getrfc.ru/list/3481/3500/.

Jin, H. and A.C. Mahendran, 2005. Using SigComp to compress SIP/SDP messages. Proceeding of IEEE International Conference on Communications (ICC), pp: 3107-3111.

Johnston, A., S. Donovan, R. Sparks, C. Cunningham and K. Summers, 2003. Session Initiation Protocol (SIP) Basic Call Flow Examples. IETF RFC3665, Retrieved from: http://www.apps.ietf.org/rfc/rfc-i3600.html.

Li, B. and F. Yang, 2010. SIP compression algorithm based on SigComp. J. Comp. Appl., 30(4): 881-883.

Li, S., 2009. SigComp technology and its performance analysis. M.A. Thesis, Nanjing University of Posts and Telecommunications, China.

Price, R., C. Bormann, J. Christoffersson, H. Hannu, Z. Liu and J. Rosenberg, 2003. Signaling Compression (SigComp). IETF RFC3320, Retrieved from: http://tools.ietf.org/html/draft-ietf-rohc-sigcomp-sip-01.

Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks *et al*., 2002. SIP: Session Initiation Protocol. IETF RFC3261, Retrieved from: http://tools.ietf.org/pdf/draft-ietf-sip-rfc2543bis-08.pdf.

Tian, Y., 2010. The research and implementation of the FPGA-based compression algorithm. M.A. Thesis, Xidian University, China.

Wen, T., D. Zhang and Q. Guo, 2011. A new SIP compression mechanism with pretreatment based on SIGCOMP in IMS. Proceeding of International Conference on Internet Technology and Applications (ITAP). Wuhan, pp: 1-6.