EMERGING ITU-T STANDARD G.711.0 — LOSSLESS COMPRESSION OF G.711 PULSE CODE MODULATION

Noboru Harada[1]*, Yutaka Kamamoto*[1]*, Takehiro Moriya*[1]*, Yusuke Hiwasaki*[2]*, Michael A. Ramalho*[3]*, Lorin Netsch*[4]*, Jacek Stachurski*[4]*, Lei Miao*[5]*, Hervé Taddei*[6] *and Fengyan Qi*[5]

[1]NTT Communication Science Labs./[2]NTT Cyber Space Labs., Japan
[3]Cisco Systems, Inc., USA [4]Texas Instruments Incorporated, USA
[5]Huawei Technologies, China/[6]Huawei Technologies Deutschland GmbH, Germany

ABSTRACT

The ITU-T Recommendation G.711 is the benchmark standard for narrowband telephony. It has been successful for many decades because of its proven voice quality, ubiquity and utility. A new ITU-T recommendation, denoted G.711.0, has been recently established defining a lossless compression for G.711 packet payloads typically found in IP networks. This paper presents a brief overview of technologies employed within the G.711.0 standard and summarizes the compression and complexity results. It is shown that G.711.0 provides greater than 50% average compression in typical service provider environments while keeping low computational complexity for the encoder/decoder pair (1.0 WMOPS average, <1.7 WMOPS worst case) and low memory footprint (about 5k octets RAM, 5.7k octets ROM, and 3.6k program memory measured in number of basic operators).

Index Terms— Speech coding, Standardization, ITU-T Recommendation G.711.0, G.711 Lossless compression

1. INTRODUCTION

The ITU-T Rec. G.711 [1] is the benchmark coding standard for narrowband telephony for many decades. Owing to its proven voice quality, ubiquity and utility, G.711 continues to enjoy widespread use in today's newest packet-based networks (e.g., Voice over IP) – even when neither endpoint interfaces to a telephony network. The ITU-T has recently established a *lossless* coding standard for G.711 payloads typically used in VoIP applications. This new standard is ITU-T Rec. G.711.0 [2].

The G.711.0 codec may be used as a traditional codec and its use negotiated (end-to-end) by the end terminals (IP phones, conference bridge endpoints, etc.). Additionally, owing to its lossless and stateless design, G.711.0 may also be used as a lossless compression mechanism on any intermediate link (e.g., service provider VoIP backbone links at voice gateways) where G.711 is used by the end systems. G.711.0 employed in these transcoding applications provides bandwidth savings with *no degradation in audio quality relative to G.711* since it is a lossless algorithm. For these gateway applications, low computational complexity is desired. A Figure of Merit (FoM), defined in the G.711.0 Terms of Reference (ToR) [3], was used to assess the tradeoff between complexity and signal compression during the design phase and the G.711.0 selection process [4].

In this paper, first overviews of the G.711 and G.711.0 standards are given followed by some compression and complexity results.

2. G.711 PULSE CODE MODULATION

G.711 coding is a form of a non-linear quantization whereby individual uniform PCM samples of 13 or 14 bit precision are compressed to 8 bits using one of two logarithmic conversion laws (A-law and μ-law).

3. GENERAL DESCRIPTION OF G.711.0

The G.711.0 codec accommodates both G.711 encoding laws and losslessly compresses frames consisting of 40, 80, 160, 240 or 320 G.711 samples. G.711.0 is lossless for all possible G.711 payloads and is most effective when compressing zero mean acoustic signals such as speech. Owing to its stateless and self-describing design (all information needed to reconstruct an original G.711 frame is contained in the G.711.0 compressed frame), the input frame lengths may be changed on-the-fly at the encoder. The algorithmic delay of G.711.0 is defined by the input frame length and is therefore, 5, 10, 20, 30, and 40 ms (assuming the usual 8 kHz sampling).

G.711.0 is a variable bit rate compression algorithm; the size of the (compressed) output frame depends on the input signal characteristics. The minimum size of an encoded frame is one byte. The maximum size of an encoded frame is the input frame size plus one byte which occurs when the input frame cannot be compressed by any of the available encoding tools.

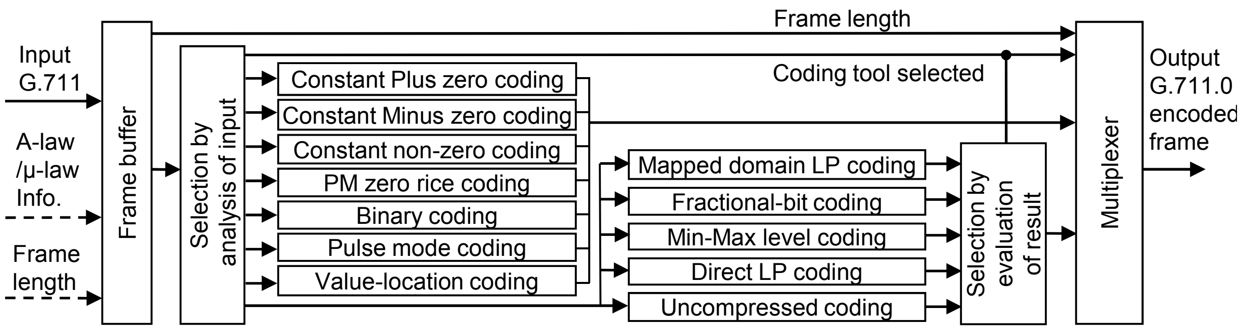

Fig. 1. High-level block diagram of the G.711.0 encoder.

4. OVERVIEW OF G.711.0 ENCODING

Fig. 1 shows the high-level block diagram of the G.711.0 encoder. The encoder selects among one of the coding tools shown in Fig. 1 to create the G.711.0 encoded output frame. A "prefix code" and the encoded information are sent as a part of the G.711.0 bitstream output. The prefix code defines the frame length and contains information related to the selected encoding tool. The G.711.0 decoder reads the prefix code and then presents the remainder of the encoded data (plus any side data contained in the prefix code) to the appropriate decoding tool. The G.711.0 coding tools are briefly described next.

4.1. Encoding tools

4.1.1. Uncompressed coding tool

If all tools fail to compress an input frame, the encoder produces a one-byte prefix for an uncompressed frame and simply reproduces the original G.711 bitstream (i.e., the input frame). In this case, the encoded bitstream size is equal to the input data size plus one byte.

4.1.2. Constant value coding tools

When all sample values in an input frame are the same, one of the constant value coding tools is applied. Constant plus zero values 0^+, such as 0xd5 for A-law and 0xff for μ-law, and constant minus zero values 0^-, such as 0x55 for A-law and 0x7f for μ-law, are signaled with a one-byte prefix and there are no trailing bytes in these cases. Those minimum magnitude values (0^+ and 0^-) are treated specially because they are often observed when the input signal is silence. Constant values other than those above are signaled by a one-byte prefix code followed by the actual constant value in one byte (two bytes total).

4.1.3. Plus-Minus zero Rice coding tool

When all sample values of an input frame are either plus zero 0^+ or minus zero 0^-, the encoder tries the Plus-Minus (PM) zero Rice coding tool. The case occurs when the input signal is silence but both the plus and minus minimum magnitude values (0^+ and 0^-) are observed in the frame. First of all, this coding tool counts the number of existing 0^+ and 0^- samples and detects which value has more occurrences. The value is called more zero 0_m. If numbers of occurrences of 0^+ and 0^- are the same, 0_m is set to the value 0^+. Then, the tool converts the input samples into sequential values of number of running more zero 0_m values followed by a less zero value 0_l. Finally, the tool encodes the sequence of the numbers using Rice coding with the best Rice parameter value for the frame. The Rice parameter is Huffman encoded.

4.1.4. Binary coding tool

When all sample values in an input frame are either 0^+ or 0^-, and if the PM zero Rice coding tool does not reduce the encoded data size, the Binary coding tool is applied. This tool generates a one-byte prefix code followed by the input sample values which are converted into one bit per sample. In the generated code, 0 indicates 0^+ and 1 indicates 0^-.

4.1.5. Pulse mode coding tool

When all sample values in a given input frame are either 0^+ or 0^- except one sample (the input frame is almost silence), the encoder applies the Pulse mode coding tool. After the position and the value of the non zero sample (called pulse) are stored, the sample values are encoded applying the same scheme as Plus-Minus zero Rice coding with considering the pulse sample as more zero 0_m. Huffman coding is applied to the Rice parameter. The pulse position index is binary encoded and the pulse value is differentially coded by Rice coding with Rice parameter 0, based on smaller difference from 0_m or 0_l value which is signaled by 1 bit.

4.1.6. Value-location coding tool

When the input frame is a low-level signal (the sample value that occurs most in an input frame is zero and the frame size and range of the remaining values satisfy a specified criteria [2]), the encoder uses the value-location coding tool. The tool sequentially encodes positions of all values within an input frame that differ from the reference

Table 1. Information of Corpora I and II.

	Corpus I	Corpus II
Speech duration	523 seconds	751 seconds
Languages	Mandarin Chinese, English, American English, Finnish, French, German, Italian, Japanese, Polish, and American Spanish	Cantonese Chinese, Mandarin Chinese, American English, British English, French, German, Japanese, and Spanish
Speakers	4 sentence pairs spoken by 4 different speakers (2 male and 2 female)	100 Cantonese sentences , 180 Mandarin sentences, 300 American English sentences, 150 Spanish sentences, and 200 sentences of other languages spoken by 2 different speakers (1 male and 1 female)
Test categories	(a1): Clean speech case of input levels -16, -26 and -26 dBov; voice activity factor (VAF) of 45 % +/- 1 %; both A-law and µ-law.	
	(a2): Noisy speech input level -26 dBov; VAF of 45 % +/- 1 %; both A-law and µ-law; SNRs of 15, 20 and 25 dB; noise conditions: cafeteria, street, office noise, interfering talker, background music	
	(a3): Tandem cases with G.711.1 R1, EFR+DTX, G.729, and G.726 for clean speech/noisy speech conditions same as above (a1) and (a2) (for EFR, also car noise added)	
Total duration of all test signals	185 hours	242 hours

0 value. The method effectively decomposes an input frame s as

$$\mathbf{s} = \sum_{k=1}^{L-1} v_k \ \mathbf{c}_k$$

where L represents the number of non-zero values, v_k, in the frame, and the vectors $\mathbf{c}_k$ represent their locations (the $\mathbf{c}_k$ code vector contains 1 at the locations at which v_k occurs, and 0 elsewhere). To enhance coding efficiency, the vectors $\mathbf{c}_k$ are encoded sequentially. First, the value locations that have already been encoded in all previous code vectors $\mathbf{c}$ are removed from the current $\mathbf{c}_k$ vector. The reduced-dimensionality vector $\mathbf{c}'_k$ is then coded using Rice coding, binary encoding, or explicit location encoding. The information about the $\mathbf{c}_k$ encoding sequence, the encoding method for each $\mathbf{c}_k$, and the corresponding values v_k are transmitted in the bit-stream.

4.1.7. Mapped domain Linear Predictive coding tool

The Mapped domain LP coding tool takes a sequence of N G.711 A-law or µ-law symbols. First, these N G.711 symbols are converted into uniform (linear) PCM domain and a short-term prediction is carried out using LP analysis with progressive linear prediction for the first few samples in the frame. The prediction residual signal lies in the range of $[-255, 255]$ since the predicted value is subtracted from the target value in the 8-bit logarithmic domain (not in the uniform PCM domain). The LPC parameters are quantized as PARCOR coefficients. Additional coding tools such as Bandwidth extension, Long term prediction and Plus-Minus zero mapping tools are further applied depending on the input samples and the frame lengths. The amplitude of the residual signal is calculated and coded in sub-frames using either Rice coding or Escaped-Huffman coding with Adaptive recursive Rice coding.

4.1.8. Fractional-bit coding tool

The fractional-bit coding tool identifies the total number of signal levels that exist within an input frame and then combines several samples for joint encoding. Five samples are used at a time to calculate the polynomial:

$$V = l_1 + l_2 L + l_3 L^2 + l_4 L^3 + l_5 L^4$$

where l_i represents the value of sample i, and L represents the number of levels within an input frame. Bit-rate saving is achieved by binary encoding the polynomial V (resulting in fractional-bit per sample) instead of encoding each sample individually. Several level-distribution cases that benefit most from this coding approach are identified. In the bit-stream, one of 30 states of the one-byte header prefix code indicates that the fractional-bit coding is applied and specifies the encoded input frame characteristics (frame length and levels present).

4.1.9. Min-Max level coding tool

The Min-Max level coding tool is used only for 40-sample frames. This tool calculates the minimum number of bits needed to encode the binary span of the G.711 amplitude levels represented in the input frame and then encodes each G.711 sample with precisely that number of bits per sample. Pre-appended to this encoded sample data is one, or on rare occasion two, additional bytes of overhead information specific to this tool. Lastly, pre-appended to this information is the one byte prefix code described earlier.

4.1.10. Direct Linear Predictive coding tool

For 40-sample frames, the Direct Linear Predictive coding tool is used when all the above coding tools fail to compress the input frame. It performs a 4th order LP coding directly in the 8-bit logarithmic domain on absolute values of the input samples. All prediction coefficients are fixed at 0.25. The prediction residual is encoded by Rice coding with the Rice parameter of 5 along with the sign bit of the input sample.

Table 2. Results for Corpora I and II.

	Corpus I	Corpus II	Total
Input data size [Mbytes]	5,327	6,969	12,296
Encoded size [Mbytes]	2,489	3,005	5,491
Comp. ratio [%]	53.27	56.87	55.33
Complexity [WMOPS, weighted millions of operations]			
Average enc.+dec.	1.03	0.94	1.01
Worst-case enc.+dec.	1.63	1.63	1.63
Worst-case enc.	1.08	1.08	1.08
Worst-case dec.	0.56	0.56	0.56

Table 3. Results for Each Test Category.

Test category		Compression ratio [%]	
		A-law	μ-law
(a1): Clean speech	-16 dBoV	59.56 %	50.67 %
	-26 dBoV	69.39 %	60.62 %
	-36 dBoV	77.01 %	72.55 %
(a2): Noisy speech	SNR 15 dB	50.90 %	44.52 %
	SNR 20 dB	54.43 %	47.15 %
	SNR 25 dB	60.64 %	52.43 %
(a1) and (a2) conditions in total		57.55 %	50.24 %
(a3): Tandem conditions in total		60.08 %	54.52 %
(b): Recorded (NTT) μ-law corpus		-	50.83 %

5. PERFORMANCE TEST RESULTS

The G.711.0 coding algorithm has been implemented in ANSI-C using the ITU-T Software Tool Library STL2005 v2.2 [5]. Complexity and compression performance per each set of conditions specified in the G.711.0 ToR [3] and processing plan [4] are presented. Table 1 shows information corresponding to test Corpora I and II. Corpus I consists of the P.501 speech corpus [6]. Corpus II was chosen from the "Multilingual Speech Database 2002" [7] and the "Ambient Noise Database" [8]. The durations of input speech signals for Corpora I and II are 523 seconds and 751 seconds, respectively, which resulted in over 425 hours of processed data for all categories of test conditions listed in Table 1. In addition, a μ-law corpus recorded from an in-service network operated in Japan was provided by NTT (1.4 GB) [9] and was also used (test category (b) in Table 3). Tables 2 and 3 provide results for Corpora I and II, and results for each test category, respectively.

The compression ratio is calculated as:

$$\text{Compression ratio [\%]} = \left(1 - \frac{\text{compressed size}}{\text{original size}}\right) \times 100 \cdot$$

Compression and complexity results are averaged over all input frame lengths and conditions. The computational complexity is reported in Weighted Millions of Operations Per Second (WMOPS). The G.711.0 ROM and RAM data requirements and program size (in number of basic operators) are shown in Table 4. Note that μ-law compression is less than A-law (A-law encodes low amplitude signals more coarsely) and greater than 50% compression is achieved for the recorded (service provider) corpus and all but the two high noise μ-law test conditions.

Table 4. Required ROM and RAM sizes and Number of Basic Operators for the G.711.0 C code.

ROM size [bytes]	Word16 and Word8 tables (including 2-byte pointers)	5,481 (5,721)
RAM size [bytes]	Encoder	3,586
	Decoder	1,372
	Total	4,958
Program size [number of basic operators]		3,554

6. CONCLUSION

This paper presented an overview of the G.711.0 standard together with compression and complexity results. G.711.0 provides more than 50% average compression in service provider environments while keeping low computational complexity for the encoder/decoder pair (1.0 WMOPS average, <1.7 WMOPS worst case) and low memory footprint (about 5k octets RAM, 5.7k octets ROM, and 3.6k basic operators).

7. ACKNOWLEDGEMENTS

The authors would like to thank Mr. Simão Campos, Mr. Zhang Dejun, Dr. Rongqiang Hu, Dr. Xu Jianfeng, Mr. Paul E. Jones, Mr. Csaba Kós, Dr. Claude Lamblin and Mr. Yushi Naito (alphabetical order) for their kind help during the ITU-T standardization process of the G.711.0 Recommendation.

8. REFERENCES

[1] ITU-T, Geneva, Switzerland, ITU-T Rec. G.711, "Pulse code modulation (PCM) of voice frequencies," Nov. 1988

[2] ITU-T, Geneva, Switzerland, ITU-T Rec. G.711.0, "Lossless compression of G.711 pulse code modulation," Oct. 2009

[3] ITU-T SG16 TD-33R1/WP3 Annex Q10.E, "Terms of Reference (ToR) and time schedule for G.711 lossless compression (G.711-LLC)," Study Period 2009-1012, Geneva, February 2009 (Source: Rapporteurs Q10/16)

[4] ITU-T WP3/16 Document AC-0908-Q10-18, "Selection phase processing test plan for lossless compression for G.711," July 2009

[5] ITU-T, Geneva, Switzerland, ITU-T Rec. G.191, "Software tools for speech and audio coding standardization," July 2005

[6] ITU-T, Geneva, Switzerland, ITU-T Rec. P.501, "Test signals for use in telephonometry," June 2007

[7] http://www.ntt-at.com/products_e/speech2002/index.html

[8] http://www.ntt-at.com/products_e/noise-DB/index.html

[9] ITU-T WP3/16 Document AC-0809-Q10-14, "Proposed processing plan for the corpus 11-B of G.711 LLC," Study Period 2005-2008, Geneva, September-October 2008.