Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 26, 2015

M. Ramalho, Ed.
P. Jones
Cisco Systems
N. Harada
NTT
M. Perumal
Ericsson
L. Miao
Huawei Technologies

March 25, 2015

RTP Payload Format for G.711.0
draft-ietf-payload-g7110-05

Abstract

This document specifies the Real-Time Transport Protocol (RTP) payload format for ITU-T Recommendation G.711.0. ITU-T Rec. G.711.0 defines a lossless and stateless compression for G.711 packet payloads typically used in IP networks. This document also defines a storage mode format for G.711.0 and a media type registration for the G.711.0 RTP payload format.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 26, 2015.

Table of Contents

1. Introduction

The International Telecommunication Union (ITU-T) Recommendation G.711.0 [G.711.0] specifies a stateless and lossless compression for G.711 packet payloads typically used in Voice over IP (VoIP) networks. This document specifies the Real-Time Transport Protocol (RTP) RFC 3550 [RFC3550] payload format and storage modes for this compression.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. G.711.0 Codec Background

ITU-T Recommendation G.711.0 [G.711.0] is a lossless and stateless compression mechanism for ITU-T Recommendation G.711 [G.711] and thus is not a "codec" in the sense of "lossy" codecs typically carried by RTP. When negotiated end-to-end ITU-T Rec. G.711.0 is negotiated as if it were a codec, with the understanding that ITU-T Rec. G.711.0 losslessly encoded the underlying (lossy) G.711 pulse code modulation (PCM) sample representation of an audio signal. For this reason ITU-T Rec. G.711.0 will be interchangeably referred to in this document as a "lossless data compression algorithm" or a "codec", depending on context. Within this document, individual G.711 PCM samples will be referred to as "G.711 symbols" or just "symbols" for brevity.

This section describes the ITU-T Recommendation G.711 [G.711] codec, its properties, typical uses cases and its key design properties.

3.1. General Information and Use of the ITU-T G.711.0 Codec

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, G.711.0, has been established for defining a stateless and lossless compression for G.711 packet payloads typically used in VoIP networks. ITU-T Rec. G.711.0 is also known as ITU-T Rec. G.711 Annex A [G.711-A1], as ITU-T Rec. G.711 Annex A is effectively a pointer ITU-T Rec. G.711.0. Henceforth in this document, ITU-T Rec. G.711.0 will simply be referred to as "G.711.0" and ITU-T Rec. G.711 simply as "G.711".

G.711.0 may be employed end-to-end; in which case the RTP payload format specification and use is nearly identical to the G.711 RTP

specification found in RFC 3551 [RFC3551]. The only significant difference for G.711.0 is the required use of a dynamic payload type (the static PT of 0 or 8 is presently almost always used with G.711 even though dynamic assignment of other payload types is allowed) and the recommendation not to use Voice Activity Detection (see Section 4.1).

G.711.0, being both lossless and stateless, may also be employed as a lossless compression mechanism for G.711 payloads anywhere between end systems which have negotiated use of G.711. Because the only significance between the G.711 RTP payload format header and the G.711.0 payload format header defined in this document is the payload type, a G.711 RTP packet can be losslessly converted to a G.711.0 RTP packet simply by compressing the G.711 payload (thus creating a G.711.0 payload), changing the payload type to the dynamic value desired and copying all the remaining G.711 RTP header fields into the corresponding G.711.0 RTP header. In a similar manner, the corresponding decompression of the G.711.0 RTP packet thus created back to the original source G.711 RTP packet can be accomplished by losslessly decompressing the G.711.0 payload back to the original source G.711 payload, changing the payload type back to the payload type of the original G.711 RTP packet and copying all the remaining G.711.0 RTP header fields into the corresponding G.711 RTP header. Negotiation specifics for this lossless G.711 payload compression for RTP use case is not in scope for this document.

It is special to note that G.711.0, being both lossless and stateless, can be employed multiple times (e.g., on multiple, individual hops or series of hops) of a given flow with no degradation of quality relative to end-to-end G.711. Stated another way, multiple "lossless transcodes" from/to G.711.0/G.711 do not affect voice quality as typically occurs with lossy transcodes to/ from dissimilar codecs.

Lastly, it is expected that G.711.0 will be used as an archival format for recorded G.711 streams. Therefore, a G.711.0 Storage Mode Format is also included in this document.

3.2. Key Properties of G.711.0 Design

The fundamental design of G.711.0 resulted from the desire to losslessly encode and compress frames of G.711 symbols independent of what types of signals those G.711 frames contained. The primary G.711.0 use case is for G.711 encoded, zero-mean, acoustic signals (such as speech and music).

G.711.0 attributes are below:

A1 Compression for zero-mean acoustic signals: G.711.0 was designed as its primary use case for the compression of G.711 payloads that contained "speech" or other zero-mean acoustic signals. G.711.0 obtains greater than 50% average compression in service provider environments [ICASSP].

A2 Lossless for any G.711 payload: G.711.0 was designed to be lossless for any valid G.711 payload - even if the payload consisted of apparently random G.711 symbols (e.g., a modem or FAX payload). G.711.0 could be used for "aggregate 64 kbps G.711 channels" carried over IP without explicit concern if a subset of these channels happened to be carrying something other than voice or general audio. To the extent that a particular channel carried something other than voice or general audio, G.711.0 ensured that it was carried losslessly, if not significantly compressed.

A3 Stateless: Compression of a frame of G.711 symbols was only to be dependent on that frame and not on any prior frame. Although greater compression is usually available by observing a longer history of past G.711 symbols, it was decided that the compression design would be stateless to completely eliminate error propagation common in many lossy codec designs (e.g., ITU-T Rec. G.729 [G.729], ITU-T Rec. G.722 [G.722]). That is, the decoding process need not be concerned about lost prior packets because the decompression of a given G.711.0 frame is not dependent on potentially lost prior G.711.0 frames. Owing to this stateless property, the frames input to the G.711.0 encoder may be changed "on-the-fly" (a 5 ms encoding could be followed by a 20 ms encoding).

A4 Self-describing: This property is defined as the ability to determine how many source G.711 samples are contained within the G.711.0 frame solely by information contained within the G.711.0 frame. Generally, the number of source G.711 symbols can be determined by decoding the initial octets of the compressed G.711.0 frame (these octets are called "prefix codes" in the standard). A G.711.0 decoder need not know how many symbols are contained in the original G.711 frame (e.g., parameter ptime in Session Description Protocol, SDP, [RFC4566]), as it is able to decompress the G.711.0 frame presented to it without signaling knowledge.

A5 Accommodate G.711 payload sizes typically used in IP: G.711 input frames of length typically found in VoIP applications represent SDP ptime values of 5 ms, 10 ms, 20 ms, 30 ms or 40 ms. Since the dominant sampling frequency for G.711 is 8000 samples per

second, G.711.0 was designed to compress G.711 input frames of 40, 80, 160, 240 or 320 samples.

A6 Bounded expansion: Since attribute A2 above requires G.711.0 to be lossless for any payload (which could consist of any combination of octets with each octet spanning the entire space of 2^8 values), by definition there exists at least one potential G.711 payload which must be "uncompressible". Since the quantum of compression is an octet, the minimum expansion of such an uncompressible payload was designed to be the minimum possible of one octet. Thus G.711.0 "compressed" frames can be of length one octet to X+1 octets, where X is the size of the input G.711 frame in octets. G.711.0 can therefore be viewed as a Variable Bit Rate (VBR) encoding in which the size of the G.711.0 output frame is a function of the G.711 symbols input to it.

A7 Algorithmic delay: G.711.0 was designed to have the algorithmic delay equal to the time represented by the number of samples in the G.711 input frame (i.e., no "look-ahead").

A8 Low Complexity: Less than 1.0 Weighted Million Operations Per Second (WMOPS) average and low memory footprint (~5k octets RAM, ~5.7k octets ROM and ~3.6 basic operations) [ICASSP] [G.711.0].

A9 Both A-law and mu-law supported: G.711 has two operating laws, A-law and mu-law. These two laws are also known as PCMA and PCMU in RTP applications RFC 3551 [RFC3551].

These attributes generally make it trivial to compress a G.711 input frame consisting of 40, 80, 160, 240 or 320 samples. After the input frame is presented to a G.711.0 encoder, a G.711.0 "self-describing" output frame is produced. The number of samples contained within this frame is easily determined at the G.711.0 decoder by virtue of attribute A4. The G.711.0 decoder can decode the G.711.0 frame back to a G.711 frame by using only data within the G.711.0 frame.

Lastly we note that losing a G.711.0 encoded packet is identical in effect of losing a G.711 packet (when using RTP); this is because a G.711.0 payload, like the corresponding G.711 payload, is stateless. Thus, it is anticipated that existing G.711 PLC mechanisms will be employed when a G.711.0 packet is lost and an identical MOS degradation relative to G.711 loss will be achieved.

3.3. G.711 Input Frames to G.711.0 Output Frames

G.711.0 is a lossless and stateless compression of G.711 frames. The following figure depicts this where "A" is the process of G.711.0 encoding and "B" is the process of G.711.0 decoding.

1:1 Mapping from G.711 Input Frame to G.711.0 Output Frame

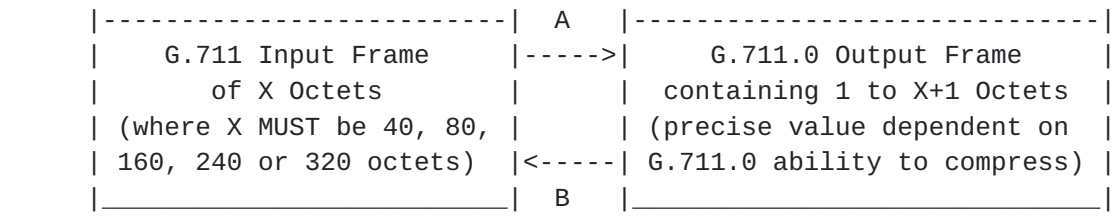

Figure 1

Note that the mapping is 1:1 (lossless) in both directions, subject to two constraints. The first constraint is that the input frame provided to the G.711.0 encoder (process "A") has a specific number of input G.711 symbols consistent with attribute A5 (40, 80, 160, 240 or 320 octets). The second constraint is that the companding law used to create the G.711 input frame (A-law or mu-law) must be known, consistent with attribute A9.

Subject to these two constraints, the input G.711 frame is processed by the G.711.0 encoder ("process A") and produces a "self-describing" G.711.0 output frame, consistent with attribute A4. Depending on the source G.711 symbols, the G.711.0 output frame can contain anywhere from 1 to X+1 octets, where X is the number of input G.711 symbols. Compression results for virtually every zero-mean acoustic signal encoded by G.711.0.

Since the G.711.0 output frame is "self-describing", a G.711.0 decoder (process "B") can losslessly reproduce the original G.711 input frame with only the knowledge of which companding law was used (A-law or mu-law). The first octet of a G.711.0 frame is called the "Prefix Code" octet; the information within this octet conveys how many G.711 symbols the decoder is to create from a given G.711.0 input frame (i.e., 0, 40, 80, 160, 240 or 320). The Prefix Code value of 0x00 is used to denote zero G.711 source symbols, which allows the use of 0x00 as a payload padding octet (to be described later in Section 3.3.1).

Since G.711.0 was designed with typical G.711 payload lengths as a design constraint (attribute A5), this lossless encoding can be

performed only with knowledge of the companding law being used. This information is anticipated to be signaled in SDP and will be described later in this document.

If the original inputs were known to be from a zero-mean acoustic signal coded by G.711, an intelligent G.711.0 encoder could infer the G.711 companding law in use (via G.711 input signal amplitude histogram statistics). Likewise, an intelligent G.711.0 decoder producing G.711 from the G.711.0 frames could also infer which encoding law in use. Thus G.711.0 could be designed for use in applications that have limited stream signaling between the G.711 endpoints (i.e., they only know "G.711 at 8k sampling is being used", but nothing more). Such usage is not further described in this document. Additionally, if the original inputs were known to come from zero-mean acoustic signals, an intelligent G.711.0 encoder could tell if the G.711.0 payload had been encrypted - as the symbols would not have the distribution expected in either companding law and would appear random. Such determination is also not further discussed in this document.

It is easily seen that this process is 1:1 and that G.711.0 based lossless compression can be employed multiple times, as the original G.711 input symbols are always reproduced with 100% fidelity.

3.3.1. Multiple G.711.0 Output Frames per RTP Payload Considerations

As a general rule, G.711.0 frames containing more source G.711 symbols (from a given channel) will typically result in higher compression, but there are exceptions to this rule. A G.711.0 encoder may choose to encode 20 ms of input G.711 symbols as: 1) a single 20 ms G.711.0 frame, or 2) as two 10 ms G.711.0 frames, or 3) any other combination of 5 ms or 10 ms G.711.0 frames - depending on which encoding resulted in fewer bits. As an example, an intelligent encoder might encode 20 ms of G.711 symbols as two 10 ms G.711.0 frames if the first 10 ms was "silence" and two G.711.0 frames took fewer bits than any other possible encoding combination of G.711.0 frame sizes.

During the process of G.711.0 standardization it was recognized that although it is sometimes advantageous to encode integer multiples of 40 G.711 symbols in whatever input symbol format resulted in the most compression (as per above), the simplest choice is to encode the entire ptime's worth of input G.711 symbols into one G.711.0 frame (if the ptime supported it). This is especially so since the larger number of source G.711 symbols typically resulted in the highest compression anyway and there is added complexity in searching for other possibilities (involving more G.711.0 frames) which were unlikely to produce a more bit efficient result.

The design of ITU-T Rec. G.711.0 [G.711.0] foresaw the possibility of multiple G.711.0 input frames in that the decoder was defined to decode what it refers to as an incoming "bit stream". For this specification, the bit stream is the G.711.0 RTP payload itself. Thus, the decoder will take the G.711.0 RTP payload and will produce an output frame containing the original G.711 symbols independent of how many G.711.0 frames were present in it. Additionally, any number of 0x00 padding octets placed between the G.711.0 frames will be silently (and safely) ignored by the G.711.0 decoding process Section 4.2.3).

To recap, a G.711.0 encoder may choose to encode incoming G.711 symbols into one or more than one G.711.0 frames and put the resultant frame(s) into the G.711.0 RTP payload. Zero or more 0x00 padding octets may also be included in the G.711.0 RTP payload. The G.711.0 decoder, being insensitive to the number of G.711.0 encoded frames that are contained within it, will decode the G.711.0 RTP payload into the source G.711 symbols. Although examples of single or multiple G.711 frame cases will be illustrated in Section 4.2, the multiple G.711.0 frame cases MUST be supported and there is no need for negotiation (SDP or otherwise) required for it.

4. RTP Header and Payload

In this section we describe the precise format for G.711.0 frames carried via RTP. We begin with RTP header description relative to G.711, then provide two G.711.0 payload examples.

4.1. G.711.0 RTP Header

Relative to G.711 RTP headers, the utilization of G.711.0 does not create any special requirements with respect to the contents of the RTP packet header. The only significant difference is that the payload type (PT) RTP header field MUST have a value corresponding to the dynamic payload type assigned to the flow. This is in contrast to most current uses of G.711 which typically use the static payload assignment of PT = 0 (PCMU) or PT = 8 (PCMA) [RFC3551] even though the negotiation and use of dynamic payload types is allowed for G.711. With the exception of rare PT exhaustion cases, the existing G.711 PT values of 0 and 8 MUST NOT be used for G.711.0 (helping to avoid possible payload confusion with G.711 payloads).

Voice Activity Detection (VAD) SHOULD NOT be used when G.711.0 is negotiated because G.711.0 obtains high compression during "VAD silence intervals" and one of the advantages of G.711.0 over G.711 with VAD is the lack of any VAD-inducing artifacts in the received signal. However, if VAD is employed, the Marker bit (M) MUST be set in the first packet of a talkspurt (the first packet after a silence

period in which packets have not been transmitted contiguously as per rules specified in [RFC3551] for G.711 payloads). This definition, being consistent with the G.711 RTP VAD use, further allows lossless transcoding between G.711 RTP packets and G.711.0 RTP packets as described in Section 3.1.

With this introduction, the RTP packet header fields are defined as follows:

V - As per [RFC3550]

P - As per [RFC3550]

X - As per [RFC3550]

CC - As per [RFC3550]

M - As per [RFC3550] and [RFC3551]

PT - The assignment of an RTP payload type for the format defined in this memo is outside the scope of this document. The RTP profiles in use currently mandate binding the payload type dynamically for this payload format.

SN - As per [RFC3550]

timestamp - As per [RFC3550]

SSRC - As per [RFC3550]

CSRC - As per [RFC3550]

Where V (version bits), P (padding bit), X (extension bit), CC (CSRC count), M (marker bit), PT (payload type), SN (sequence number), timestamp, SSRC (synchronizing source) and CSRC (contributing sources) are as defined in [RFC3550] and as typically used with G.711. PT (payload type) is as defined in [RFC3551].

4.2. G.711.0 RTP Payload

This section defines the G.711.0 RTP payload and illustrates it by means of two examples.

The first example, in Section 4.2.1, depicts the case when it is desired to carry only one G.711.0 frame in the RTP payload. This case is expected to be the dominant use case and is shown separately for the purposes of clarity.

The second example, in Section 4.2.2, depicts the general case when it is desired to carry one or more G.711.0 frames in the RTP payload. This is the actual definition of the G.711.0 RTP payload.

4.2.1. Single G.711.0 Frame per RTP Payload Example

This example depicts a single G.711.0 frame in the RTP payload. This is expected to be the dominant RTP payload case for G.711.0, as the G.711.0 encoding process supports the SDP packet times (ptime and maxptime, see [RFC4566]) commonly used when G.711 is transported in RTP. Additionally, as mentioned previously, larger G.711.0 frames generally compress more effectively than a multiplicity of smaller G.711.0 frames.

The following Figure illustrates the single G.711.0 frame per RTP payload case.

Single G.711.0 Frame in RTP Payload Case

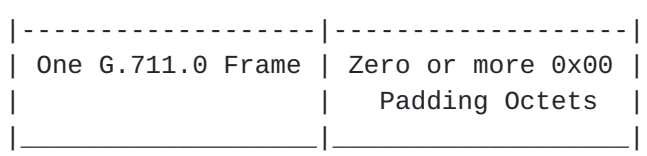

Figure 2

Encoding Process: A single G.711.0 frame is inserted into the RTP payload. The amount of time represented by the G.711 symbols compressed in the G.711.0 frame MUST correspond to the ptime signaled for applications using SDP. Although generally not desired, padding desired in the RTP payload after the G.711.0 frame MAY be created by placing one or more 0x00 octets after the G.711.0 frame. Such padding may be desired based on security considerations (see Section 10).

Decoding Process: Passing the entire RTP payload to the G.711.0 decoder is sufficient for the G.711.0 decoder to create the source G.711 symbols. Any padding inserted after the G.711.0 frame (i.e., the 0x00 octets) present in the RTP payload is silently ignored by the G.711.0 decoding process. The decoding process is fully described in Section 4.2.3 below.

4.2.2. G.711.0 RTP Payload Definition

This section defines the G.711.0 RTP payload and illustrates the case of when one or more G.711.0 frames are to be placed in the payload. All G.711.0 RTP decoders MUST support the general case described in this section (rationale presented previously in Section 3.3.1).

Note that since each G.711.0 frame is self-describing (see Attribute A4 in Section 3.2), the individual G.711.0 frames in the RTP payload need not represent the same duration of time (i.e., a 5 ms G.711.0 frame could be followed by a 20 ms G.711.0 frame). Owing to this, the amount of time represented in the RTP payload MAY be any integer multiple of 5 ms (as 5 ms is the smallest interval of time that can be represented in a G.711.0 frame).

The following Figure illustrates the one or more G.711.0 frames per RTP payload case where the number of G.711.0 frames placed in the RTP payload is N. We note that when N is equal to 1 that this case is identical to the previous example.

One or More G.711.0 Frames in RTP Payload Case

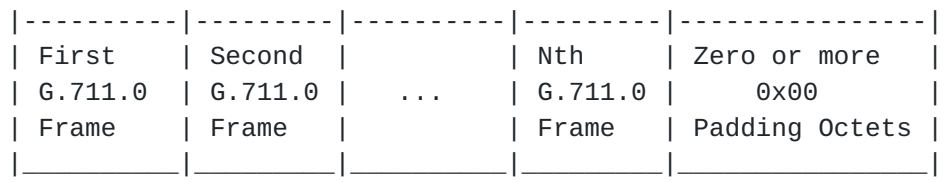

Figure 3

We note here that when we have multiple G.711.0 frames that the individual frames can be, and generally are, of different lengths. The decoding process described in Section 4.2.3 is used to determine the frame boundaries.

Encoding Process: One or more G.711.0 frames are placed in the RTP payload simply by concatenating the G.711.0 frames together. The amount of time represented by the G.711 symbols compressed in all the G.711.0 frames in the RTP payload MUST correspond to the ptime signaled for applications using SDP. Although not generally desired, padding in the RTP payload SHOULD be placed after the last G.711.0 frame in the payload and MAY be created by placing one or more 0x00 octets after the last G.711.0 frame. Such padding may be desired based on security considerations (see Section 10). Additional encoding process details and considerations are specified later in Section 4.2.2.1.

Decoding Process: As G.711.0 frames can be of varying length, the payload decoding process described in Section 4.2.3 is used to determine where the individual G.711.0 frame boundaries are. Any padding octets inserted before or after any G.711.0 frame in the RTP payload is silently (and safely) ignored by the G.711.0 decoding process specified in Section 4.2.3.

4.2.2.1. G.711.0 RTP Payload Encoding Process

ITU-T G.711.0 supports five possible input frame lengths: 40, 80, 160, 240, and 320 samples per frame and the rationale for choosing those lengths was given in the description of property A5 in Section 3.2. Assuming 8000 sample per second, these lengths correspond to input frames representing 5 ms, 10 ms, 20 ms, 30 ms or 40 ms. So while the standard assumed the input "bit stream" consisted of G.711 symbols of some integer multiple of 5 ms in length, it did not specify exactly what frame lengths to use as input to the G.711.0 encoder itself. The intent of this section is to provide some guidance for the selection.

Consider a typical IETF use case of 20 ms (160 octets) of G.711 input samples represented in a G.711.0 payload and signaled by using the SDP parameter ptime. As described in Section 3.3.1, the simplest way to encode these 160 octets is to pass the entire 160 octet to the G.711.0 encoder, resulting in precisely one G.711.0 compressed frame, and put that singular frame into the G.711.0 RTP payload. However, neither the ITU-T G.711.0 standard nor this IETF payload format mandates this. In fact 20 ms of input G.711 symbols can be encoded as 1, 2, 3 or 4 G.711.0 frames in any one of six combinations (i.e., {20ms}, {10ms:10ms}, {10ms:5ms:5ms}, {5ms:10ms:5ms}, {5ms:5ms:10ms}, {5ms:5ms:5ms:5ms}) and any of these combinations would decompress into the same source 160 G.711 octets. As an aside, we note that the first octet of any G.711.0 frame will be the prefix code octet and information in this octet determines how many G.711 symbols are represented in the G.711.0 frame.

Notwithstanding the above, we expect one of two encodings to be used by implementers: the simplest possible (one 160 byte input to the G.711.0 encoder which usually results in the highest compression) or the combination of possible input frames to a G.711.0 encoder that resulted in the highest compression for the payload. The explicit mention of this issue in this IETF document was deemed important because the ITU-T G.711.0 standard is silent on this issue and there is a desire for this issue to be documented in a formal Standards Developing Organization (SDO) document (i.e., here).

4.2.3. G.711.0 RTP Payload Decoding Process

The G.711.0 decoding process is a standard part of G.711.0 bit stream decoding and is implemented in the ITU-T Rec. G.711.0 reference code. The decoding process algorithm described in this section is a slight enhancement of the ITU-T reference code to explicitly accommodate RTP padding (as described above).

Before describing the decoding, we note here that the largest possible G.711.0 frame is created whenever the largest number of G.711 symbols is encoded (320 from Section 3.2, property A5) and these 320 symbols are "uncompressible" by the G.711.0 encoder. In this case (via property A6 in Section 3.2) the G.711.0 output frame will be 321 octets long. We also note that the value 0x00 chosen for the optional padding cannot be the first octet of a valid ITU-T Rec. G.711.0 frame (see [G.711.0]). We also note that whenever more than one G.711.0 frame is contained in the RTP payload, the decoding of the individual G.711.0 frames will occur multiple times.

For the decoding algorithm below, let N be the number of octets in the RTP payload (i.e., excluding any RTP padding, but including any RTP payload padding), let P equal the number of RTP payload octets processed by the G.711.0 decoding process, let K be the number of G.711 symbols presently in the output buffer, let Q be the number of octets contained in the G.711.0 frame being processed and let "!=" represent not equal to. The keyword "STOP" is used below to indicate the end of the processing of G.711.0 frames in the RTP payload. The algorithm below assumes an output buffer for the decoded G.711 source symbols of length sufficient to accommodate the expected number of G.711 symbols and an input buffer of length 321 octets.

G.711.0 RTP Payload Decoding Heuristic:

H1 Initialization of counters: Initialize P, the number of processed octets counter, to zero. Initialize K, the counter for how many G.711 symbols are in the output buffer, to zero. Initialize N to the number of octets in the RTP payload (including any RTP payload padding). Go to H2.

H2 Read internal buffer: Read min{320+1, (N-P)-1} octets into the internal buffer from the (P+1) octet of the RTP payload. We note at this point, N-P octets have yet to be processed and that 320+1 octets is the largest possible G.711.0 frame. Also note that in the common case of zero-based array indexing of a uint8 array of octets, that this operation will read octets from index P through index [min{320+1, (N-P)}] from the RTP payload. Go to H3.

H3 Analyze the first octet in the internal buffer: If this octet 0x00 (a padding octet) go to H4, otherwise go to H5 (process a G.711.0 frame).

H4 Process padding octet (no G.711 symbols generated): Increment the processed packets counter by one (set P = P + 1). If the result of this increment results in P >= N then STOP (as all RTP Payload octets have been processed), otherwise go to H2.

H5 Process an individual G.711.0 frame (produce G.711 samples in the output frame): Pass the internal buffer to the G.711.0 decoder. The G.711.0 decoder will read the first octet (called the "prefix code" octet in ITU-T Rec. G.711.0 [G.711.0]) to determine the number of source G.711 samples M are contained in this G.711.0 frame. The G.711.0 decoder will produce exactly M G.711 source symbols (M can only have values of 0, 40, 80, 160, 240 or 320). If K = 0, these M symbols will be the first in the output buffer and are placed at the beginning of the output buffer. If K != 0, concatenate these M symbols with the prior symbols in the output buffer (there are K prior symbols in the buffer). Set K = K + M (as there are now this many G.711 source symbols in the output buffer). The G.711.0 decoder will have consumed some number of octets, Q, in the internal buffer to produce the M G.711 symbols. Increment the number of payload octet processed counter by this quantity (set P = P + Q). If the result of this increment results in P >= N then STOP (as all RTP Payload octets have been processed), otherwise go to H2.

At this point, the output buffer will contain precisely K G.711 source symbols which should correspond to the ptime signaled if SDP was used and the encoding process was without error. If ptime was signaled via SDP and the number of G.711 symbols in the output buffer is other than what corresponds to ptime, the packet MUST be discarded unless other system design knowledge allows for otherwise (e.g., occasional 5 ms clock slips causing one more or one less G.711.0 frame than nominal to be in the payload). Lastly, due to the buffer reads in H2 being bounded (to 321 octets or less), N being bounded to the size of the G.711.0 RTP payload, and M being bounded to the number of source G.711 symbols, there is no buffer overrun risk.

We also note, as an aside, that the algorithm above (and the ITU-T G.711.0 reference code) accommodates padding octets (0x00) placed anywhere between G.711.0 frames in the RTP payload as well as prior to or after any or all G.711.0 frames. The ITU-T G.711.0 reference code does not have Step H3 and H4 as separate steps (i.e., Step H5 immediately follows H2) at the added computational cost of some additional buffer passing to/from the G.711.0 frame decoder

functions. That is the G.711.0 decoder in the reference code "silently ignores" 0x00 padding octets at the beginning of what it believes to be a G.711.0 encoded frame boundary. Thus Step H3 and Step H4 above are an optimization over the reference code shown for clarity.

If the decoder is at a playout endpoint location, this G.711 buffer SHOULD be used in the same manner as a received G.711 RTP payload would have been used (passed to a playout buffer, to a PLC implementation, etc.).

4.2.4. G.711.0 RTP Payload for Multiple Channels

In this section we describe the use of multiple "channels" of G.711 data encoded by G.711.0 compression.

The dominant use of G.711 in RTP transport has been for single channel use cases. For this case, the above G.711.0 encoding and decoding process is used. However, the multiple channel case for G.711.0 (a frame-based compression) is different from G.711 (a sample-based encoding) and is described separately here.

RFC 3551 [RFC3551] provides guidelines for encoding audio channels (Section 4) and for the ordering of the channels within the RTP payload (Section 4.1). The ordering guidelines in RFC 3551, Section 4.1 SHOULD be used unless an application-specific channel ordering is more appropriate.

An implicit assumption in RFC 3551 is that all the channel data multiplexed into a RTP payload MUST represent the same physical time span. The case for G.711.0 is no different; the underlying G.711 data for all channels in a G.711.0 RTP payload MUST span the same interval in time (e.g., the same "ptime" for a SDP-specified codec negotiation).

RFC 3551 provides guidelines for sample-based encodings such as G.711 in Section 4.2. This guidance is tantamount to interleaving the individual samples in that they SHOULD be packed in consecutive octets.

RFC 3551 provides guidelines for frame-based encodings in which the frames are interleaved. However, this guidance stems from the assumption that "the frame size for frame-oriented codecs is a given". However, this assumption is not valid for G.711.0 in that individual consecutive G.711.0 frames (as per Section 4.2.2) can:

1) represent different time spans (e.g., two 5 ms G.711.0 frames in lieu of one 10 ms G.711.0 frame), and

2) be of different lengths in octets (and typically are).

Therefore a different, but also simple, concatenation-based approach is specified in this RFC.

For the multiple channel G.711.0 case, each G.711 channel is independently encoded into one or more G.711.0 frames defined here as a "G.711.0 channel superframe". Each one of these superframes is identical to the multiple G.711.0 frame case illustrated in Figure 3 of Section 4.2.2 in which each superframe can have one or more individual G.711.0 frames within it. Then each G.711.0 channel superframe is concatenated - in channel order - into a G.711.0 RTP payload. Then, if optional G.711.0 padding octets (0x00) are desired, it is RECOMMENDED that these octets are placed after the last G.711.0 channel superframe. As per above, such padding may be desired based on security considerations (see Section 10). This is depicted in the following Figure 4 below.

Multiple G.711.0 Channel Superframes in RTP Payload

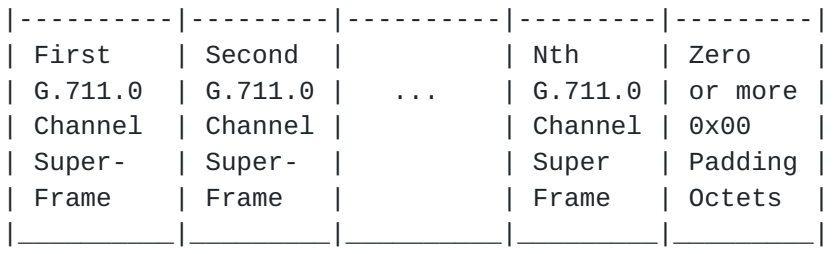

Figure 4

We note that although the individual superframes can be of different lengths in octets (and usually are), that the number of G.711 source symbols represented - in compressed form - in each channel superframe is identical (since all the channels represent the identically same time interval).

The G.711.0 decoder at the receiving end simply decodes the entire G.711.0 (multiple channel) payload into individual G.711 symbols. If M such G.711 symbols result and there were N channels, then the first M/N G.711 samples would be from the first channel, the second M/N G.711 samples would be from the second channel, and so on until the Nth set of G.711 samples are found. Similarly, if the number of channels was not known, but the payload "ptime" was known, one could infer (knowing the sampling rate) how many G.711 symbols each channel contained; then with this knowledge determine how many channels of data were contained in the payload. When SDP is used, the number of

channels is known because the optional parameter is a MUST when there is more than one channel negotiated (see Section 5.1). Additionally, when SDP is used the parameter ptime is a RECOMMENDED optional parameter. We note that if both parameters channels and ptime are known that one could provide a check for the other and the converse. Whichever algorithm is used to determine the number of channels, if the length of the source G.711 symbols in the payload (M) is not an integer multiple of the number of channels (N), then the packet SHOULD be discarded.

Lastly we note that although any padding for the multiple channel G.711.0 payload is RECOMMENDED to be placed at the end of the payload, the G.711.0 decoding algorithm described in Section 4.2.3 will successfully decode the payload in Figure 4 if the 0x00 padding octet is placed anywhere before or after any individual G.711.0 frame in the RTP payload. The number of padding octets introduced at any G.711.0 frame boundary therefore does not affect the number M of the source G.711 symbols produced. Thus the decision for padding MAY be made on a per-superframe basis.

5. Payload Format Parameters

This section defines the parameters that may be used to configure optional features in the G.711.0 RTP transmission.

The parameters defined here are a part of the media subtype registration for the G.711.0 codec. Mapping of the parameters into Session Description Protocol (SDP) RFC 4566 [RFC4566] is also provided for those applications that use SDP.

5.1. Media Type Registration

Type name: audio

Subtype name: G711-0

Required parameters:

clock rate: The RTP timestamp clock rate, which is equal to the sampling rate. The typical rate used with G.711 encoding is 8000, but other rates may be specified. The default rate is 8000.

complaw: This format specific parameter, specified on the "a=fmtp: line", indicates the companding law (A-law or mu-law) employed. This format specific parameter, as per RFC 4566 [RFC4566], is given unchanged to the media tool using this format. The case-insensitive values are "complaw=al" or "complaw=mu" are used for A-law and mu-law, respectively.

Optional parameters:

channels: See RFC 4566 [RFC4566] for definition. Specifies how many audio streams are represented in the G.711.0 payload and MUST be present if the number of channels is greater than one. This parameter defaults to 1 if not present (as per RFC 4566) and is typically a non-zero small-valued positive integer. It is expected that implementations that specify multiple channels will also define a mechanism to map the channels appropriately within their system design, otherwise the channel order specified in RFC 3551 [RFC3551] Section 4.1 will be assumed (e.g., left, right, center, ...). Similar to the usual interpretation in RFC 3551 [RFC3551], the number of channels SHALL be a non-zero positive integer.

maxptime: See RFC 4566 [RFC4566] for definition.

ptime: See RFC 4566 [RFC4566] for definition. The inclusion of "ptime" is RECOMMENDED and SHOULD be in the SDP unless there is an application specific reason not to include it (e.g., an application that has a variable ptime on a packet-by-packet basis). For constant ptime applications, it is considered good form to include "ptime" in the SDP for session diagnostic purposes. For the constant ptime multiple channel case described in Section 4.2.2, the inclusion of "ptime" can provide a desirable payload check.

Encoding considerations:

This media type is framed binary data (see Section 4.8 in RFC 6838 [RFC6838]) compressed as per ITU-T Rec. G.711.0.

Security considerations:

See Section 10.

Interoperability considerations: none

Published specification:

ITU-T Rec. G.711.0 and RFC XXXX.

[RFC Editor: please replace XXXXX with a reference to this RFC]

Applications that use this media type:

Although initially conceived for VoIP, the use of G.711.0, like G.711 before it, may find use within audio and video streaming

and/or conferencing applications for the audio portion of those applications.

Additional information:

The following applies to stored-file transfer methods:

Magic numbers: #!G7110A\n or #!G7110M\n (for A-law or MU-law encodings respectively, see Section 6).

File Extensions: None

Macintosh file type code: None

Object identifier or OIL: None

Person & email address to contact for further information:

Michael A. Ramalho <mramalho@cisco.com> or <mar42@cornell.edu>

Intended usage: COMMON

Restrictions on usage:

This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport within other framing protocols is not defined at this time.

Author: Michael A. Ramalho

Change controller:

IETF Payload working group delegated from the IESG.

5.2. Mapping to SDP Parameters

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP), which is commonly used to describe a RTP session. When SDP is used to specify sessions employing G.711.0, the mapping is as follows:

- The media type ("audio") goes in SDP "m=" as the media name.
- The media subtype ("G711-0") goes in SDP "a=rtpmap" as the encoding name.
- The required parameter "rate" also goes in "a=rtpmap" as the clock rate.

- The parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.

- Remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the media type string as a semicolon-separated list of parameter=value pairs.

5.3. Offer/Answer Considerations

The following considerations apply when using the SDP offer/answer RFC 3264 [RFC3264] mechanism to negotiate the "channels" attribute.

- If the offering endpoint specifies a value for the optional channels parameter greater than one and the answering endpoint both understands the parameter and cannot support that value requested, the answer MUST contain the optional channels parameter with the highest value it can support.

- If the offering endpoint specifies a value for the optional channels parameter the answer MUST contain the optional channels parameter unless the only value the answering endpoint can support is one, in which case the answer MAY contain the optional channels parameter with value of 1.

- If the offering endpoint specifies a value for the ptime parameter that the answering endpoint cannot support, the answer MUST contain the optional ptime parameter.

- If the offering endpoint specifies a value for the maxptime parameter that the answering endpoint cannot support, the answer MUST contain the optional maxptime parameter.

5.4. SDP Examples

The following examples illustrate how to signal G.711.0 via SDP.

5.4.1. SDP Example 1

```
m=audio RTP/AVP 98
a=rtpmap:98 G711-0/8000
a=fmtp:98 complaw=mu
```

In the above example the dynamic payload type 98 is mapped to G.711.0 via the "a=rtpmap" parameter. The mandatory "complaw" is on the "a=fmtp" parameter line. Note that neither optional parameters "ptime" nor "channels" is present; although it is generally good form to include "ptime" in the SDP if the session is a constant ptime session for diagnostic purposes.

5.4.2. SDP Example 2

The following example illustrates an offering endpoint requesting 2 channels, but the answering endpoint can only support (or render) one channel.

Offer:

```
m=audio RTP/AVP 98
a=rtpmap:98 G711-0/8000/2
a=ptime:20
a=fmtp:98 complaw=al
```

Answer:

```
m=audio RTP/AVP 98
a=rtpmap: 98 G711-0/8000/1
a=ptime: 20
a=fmtp:98 complaw=al
```

In this example the offer had an optional channels parameter. The answer must have the optional channels parameter also unless the value in the answer is one. Shown here is when the answer explicitly contains the channels parameter (it need not have and it would be interpreted as one channel). As mentioned previously, it is considered good form to include "ptime" in the SDP for session diagnostic purposes if the session is a constant ptime session.

6. G.711.0 Storage Mode Conventions and Definition

The G.711.0 storage mode definition in this section is similar to many other IETF codecs (e.g., iLBC, EVRC-NW) and is essentially a concatenation of individual G.711.0 frames.

We note that something must be stored for any G.711.0 frames that are not received at the receiving endpoint, no matter what the cause. In this section we describe two mechanisms, a "G.711.0 PLC Frame" and a "G.711.0 Erasure Frame". These G.711.0 PLC and G.711.0 Erasure Frames are described prior to the G.711.0 storage mode definition for clarity.

6.1. G.711.0 PLC Frame

When G.711 RTP payloads not received by a rendering endpoint a Packet Loss Concealment (PLC) mechanism is typically employed to "fill in" the missing G.711 symbols with something that is auditorially pleasing and thus the loss may be not noticed by a listener. Such a

PLC mechanism for G.711 is specified in ITU-T Rec. G.711 - Appendix 1 [G.711-AP1].

An natural extension when creating G.711.0 frames for storage environments is to employ such a PLC mechanism to create G.711 symbols for the span of time in which G.711.0 payloads were not received - and then to compress the resulting "G.711 PLC symbols" via G.711.0 compression. The G.711.0 frame(s) created by such a process are called "G.711.0 PLC Frames".

Since PLC mechanisms are designed to render missing audio data with the best fidelity and intelligibility, G.711.0 frames created via such processing is likely best for most recording situations (such as voicemail storage) unless there is a requirement not to fabricate (audio) data not actually received.

After such PLC G.711 symbols have been generated and then encoded by a G.711.0 encoder, the resulting frames may be stored in G.711.0 frame format. As a result, there is nothing to specify here - the G.711.0 PLC Frames are stored as if they were received by the receiving endpoint. In other words, PLC-generated G.711.0 frames appear as "normal" or "ordinary" G.711.0 frames in the storage mode file.

6.2. G.711.0 Erasure Frame

"Erasure Frames", or equivalently "Null Frames", have been designed for many frame-based codecs since G.711 was standardized. These null/erasure frames explicitly represent data from incoming audio that were either not received by the receiving system or represent data that a transmitting system decided not to send. Transmitting systems may choose not to send data for a variety of reasons (e.g., not enough wireless link capacity in radio-based systems) and can choose to send a "null frame" in lieu of the actual audio. It is also envisioned that erasure frames would be used in storage mode applications for specific archival purposes where there is a requirement not to fabricate audio data that was not actually received.

Thus, a G.711.0 erasure frame is a representation of the amount of time in G.711.0 frames that were not received or not encoded by the transmitting system.

Prior to defining a G.711.0 erasure frame it is beneficial to note what many G.711 RTP systems send when the endpoint is "muted". When muted, many of these systems will send an entire G.711 payload of either 0+ or 0- (i.e., one of the two levels closest to "analog zero" in either G.711 companding law). Next we note that a desirable

property for a G.711.0 erasure frame is for "non G.711.0 Erasure Frame aware" endpoints to be able to playback a G.711.0 erasure frame with the existing G.711.0 ITU-T reference code.

A G.711.0 Erasure Frame is defined as any G.711.0 frame for which the corresponding G.711 sample values are either the value 0++ or the value 0-- for the entirety of the G.711.0 frame. The levels of 0++ and 0-- are defined to be the two levels above or below analog zero, respectively. An entire frame of value 0++ or 0-- is expected to be extraordinarily rare when the frame was in fact generated by a natural signal, as analog inputs such as speech and music are zero-mean and are typically acoustically coupled to digital sampling systems. Note that the playback of a G.711.0 frame characterized as an erasure frame is auditorially equivalent to a muted signal (a very low value constant).

These G.711.0 erasure frames can be reasonably characterized as null or erasure frames while meeting the desired playback goal of being decoded by the G.711.0 ITU-T reference code. Thus, similarly to G.711 PLC frames, the G.711.0 erasure frames appear as "normal" or "ordinary" G.711.0 frames in the storage mode format.

6.3. G.711.0 Storage Mode Definition

The storage format is used for storing G.711.0 encoded frames. The format for the G.711.0 storage mode file defined by this RFC is shown below.

G.711.0 Storage Mode Format

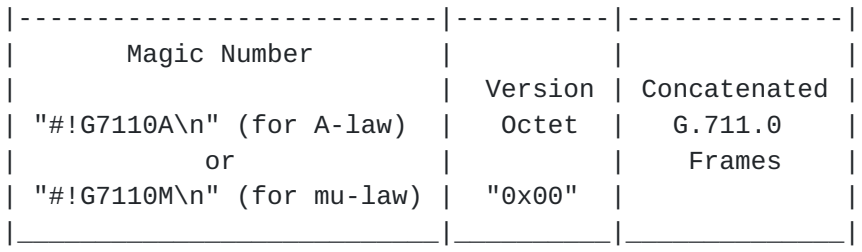

Figure 5

The storage mode file consists of a magic number and a version octet followed by the individual G.711.0 frames concatenated together.

The magic number for G.711.0 A-law corresponds to the ASCII character string "#!G7110A\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x30 0x41 0x0A". Likewise, the magic number for G.711.0 MU-law corresponds to

the ASCII character string "#!G7110M\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x4E 0x4D 0x0A".

The version number octet allows for the future specification of other G.711.0 storage mode formats. The specification of other storage mode formats may be desirable as G.711.0 frames are of variable length and a future format may include an indexing methodology that would enable playout far into a long G.711.0 recording without the necessity of decoding all the G.711.0 frames since the beginning of the recording. Other future format specification may include support for multiple channels, metadata and the like. For these reasons it was determined that a versioning strategy was desirable for the G.711.0 storage mode definition specified by this RFC. This RFC only specifies Version 0 and thus the value of "0x00" MUST be used for the storage mode defined by this RFC.

The G.711.0 codec data frames, including any necessary erasure or PLC frames, are stored in consecutive order concatenated together as shown in Section 4.2.2. As the Version 0 storage mode only supports a single channel, the RTP payload format supporting multiple channels defined in Section 4.2.4 is not supported in this storage mode definition.

To decode the individual G.711.0 frames, the algorithm presented in Section 4.2.2 may be used to decode the individual G.711.0 frames. If the version octet is determined not to be zero, the remainder of the payload MUST NOT be passed to the G.711.0 decoder, as the ITU-T G.711.0 reference decoder can only decode concatenated G.711.0 frames and has not been designed to decode elements in yet to be specified future storage mode formats.

7. Acknowledgements

There have been many people contributing to G.711.0 in the course of its development. The people listed here deserve special mention: Takehiro Moriya, Claude Lamblin, Herve Taddei, Simao Campos, Yusuke Hiwasaki, Jacek Stachurski, Lorin Netsch, Paul Coverdale, Patrick Luthi, Paul Barrett, Jari Hagqvist, Pengjun (Jeff) Huang, John Gibbs, Yutaka Kamamoto, and Csaba Kos. The review and oversight by the IETF Payload Working Group chairs Ali Begen and Roni Even during the development of this RFC is appreciated. Additionally, the careful review by Richard Barnes and extensive review by David Black and the rest of the IESG is likewise very much appreciated.

8. Contributors

The authors thank everyone who have contributed to this document. The people listed here deserve special mention: Ali Begen, Roni Even, and Hadriel Kaplan.

9. IANA Considerations

One media type (audio/G711-0) has been defined and requires IANA registration in the media types registry. See Section 5.1 for details.

10. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550], and in any appropriate RTP profile (for example RFC 3551 [RFC3551] or [RFC4585]). This implies that confidentiality of the media streams is achieved by encryption; for example, through the application of SRTP [RFC3711]. Because the data compression used with this payload format is applied end-to-end, any encryption needs to be performed after compression.

Note that the appropriate mechanism to ensure confidentiality and integrity of RTP packets and their payloads is very dependent on the application and on the transport and signaling protocols employed. Thus, although SRTP is given as an example above, other possible choices exist.

Note that end-to-end security with either authentication, integrity or confidentiality protection will prevent a network element not within the security context from performing media-aware operations other than discarding complete packets. To allow any (media-aware) intermediate network element to perform its operations, it is required to be a trusted entity which is included in the security context establishment.

G.711.0 has no known denial-of-service attacks due to decoding, as data posing as a desired G711.0 payload will be decoded into something (as per the decoding algorithm) with a finite amount of computation. This is due to the decompression algorithm having a finite worst-case processing path (no infinite computational loops are possible). We also note that the data read by the G.711.0 decoder is controlled by the length of the individual encoded G.711.0 frame(s) contained in the RTP payload. The decoding algorithm specified in Section 4.2.3 above ensures that the G.711.0 decoder will not read beyond the length of the internal buffer specified (which is in turn specified to be no greater than the largest

possible G.711.0 frame of 321 octets). Therefore a G.711.0 payload does not carry "active content" that could impose malicious side-effects upon the receiver.

G.711.0 is a variable bit rate (VBR) audio codec. There have been recent concerns with VBR speech codecs where a passive observer can identify phrases from a standard speech corpus by means of the lengths produced by the encoder even when the payload is encrypted [IEEE]. In this paper, it was determined that some code excited linear prediction (CELP) codecs would produce discrete packet lengths for some phonemes. And furthermore with the use of appropriately designed Hidden Markov Models (HMMs) that such a system could predict phrases with unexpected accuracy. One CELP codec studied, SPEEX, had the property that it produced 21 different packet lengths in its wideband mode and that these packet lengths probabilistically mapped to phonemes that a HMM system could be trained on. In this paper it was determined that a mitigation technique would be to pad the output of the encoder with random padding lengths to the effect: 1) that more discrete payload sizes would result, and 2) that the probabilistic mapping to phonemes would become less clear. As G.711 is not a speech model based codec, neither is G.711.0. A G.711.0 encoding, during talking periods, produces frames of varying frame lengths which are not likely to have a strong mapping to phonemes. Thus G.711.0 is not expected to have this same vulnerability. It should be noted that "silence" (only one value of G.711 in the entire G.711 input frame)" or "near silence" (only a few G.711 values) is easily detectable as G.711.0 frame lengths or one or a few octets. If one desires to mitigate for silence/non-silence detection, statistically variable padding should be added to G.711.0 frames that resulted in very small G.711.0 frames (less than about 20% of the symbols of the corresponding G.711 input frame). Methods of introducing padding in the G.711.0 payloads have been provided in the G.711.0 RTP payload definition in Section 4.2.2.

11. Congestion Control

The G.711 codec is a Constant Bit Rate (CBR) codec which does not have a means to regulate the bitrate. The G.711.0 lossless compression algorithm typically compresses the G.711 CBR stream into a lower bandwidth VBR stream. However, being lossless, it does not possess means of further reducing the bitrate beyond the G.711.0-based compression result. The G.711.0 RTP payloads can be made arbitrarily large by means of adding optional padding bytes (subject only to MTU limitations).

Therefore, there are no explicit ways to regulate the bit-rate of the transmissions outlined in this RTP Payload format except by means of modulating the number of optional padding bytes in the RTP payload.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[G.711.0] ITU-T G.711.0, , "Recommendation ITU-T G.711.0 - Lossless Compression of G.711 Pulse Code Modulation", September 2009.

[G.711] ITU-T G.711.0, , "Recommendation ITU-T G.711: Pulse Code Modulation (PCM) of Voice Frequencies", November 1988.

[G.711-AP1] ITU-T G.711 Appendix 1, , "Recommendation G.711 Appendix 1: A high quality low-complexity algorithm for packet loss concealment with G.711", September 1999.

[G.711-A1] ITU-T G.711 Amendment 1, , "Recommendation ITU-T G.711 Amendment 1 - Amendment 1: New Annex A on Lossless Encoding of PCM Frames", September 2009.

12.2. Informative References

[G.729] ITU-T G.729, , "Recommendation ITU-T G.729 - Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)", January 2007.

[G.722] ITU-T G.722, , "Recommendation ITU-T G.722 - 7 kHz audio-coding within 64 kbit/s", November 1988.

[ICASSP] N. Harada, , Y. Yamamoto, , T. Moriya, , Y. Hiwasaki, , M. A. Ramalho, , L. Netsch, , Y. Stachurski, , Miao Lei, , H. Taddei, , and Q. Fengyan, "Emerging ITU-T Standard G.711.0 - Lossless Compression of G.711 Pulse Code Modulation, International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010, ISBN 978-1-4244-4244-4295-9", March 2010.

[IEEE] C.V. Wright, , L. Ballard, , S.E. Coull, , F. Monrose, , and G.M. Masson, "Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations, IEEE Symposium on Security and Privacy, 2008, ISBN: 978-0-7695-3168-7", May 2008.

Authors' Addresses

Michael A. Ramalho (editor)
Cisco Systems, Inc.
6310 Watercrest Way Unit 203
Lakewood Ranch, FL 34202
USA

Phone: +1 919 476 2038
Email: mramalho@cisco.com

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709
USA

Phone: +1 919 476 2048
Email: paulej@packetizer.com

Noboru Harada
NTT Communications Science Labs.
3-1 Morinosato-Wakamiya
Atsugi, Kanagawa 243-0198
JAPAN

Phone: +81 46 240 3676
Email: harada.noboru@lab.ntt.co.jp

Muthu Arul Mozhi Perumal
Ericsson
Ferns Icon
Doddanekundi, Mahadevapura
Bangalore, Karnataka 560037
India

Phone: +91 9449288768
Email: muthu.arul@gmail.com

Lei Miao
Huawei Technologies Co. Ltd
Q22-2-A15R, Enviroment Protection Park
No. 156 Beiqing Road
HaiDian District
Beijing 100095
China

Phone: +86 1059728300
Email: lei.miao@huawei.com