RGL Codec (Version 1.0.0)
(G.711 Lossless Codec)

IETF draft:

http://www.ietf.org/internet–drafts/draft–ramalho–rgl–desc–01.txt

HTML version of draft:

http://www.winlab.rutgers.edu/~ramalho/rgl_codec_01.html

Whitepaper & code at:

www.vovida.org

Michael Ramalho

mramalho@cisco.com

RGL Codec – Use & Design Goals

Utility/Use:

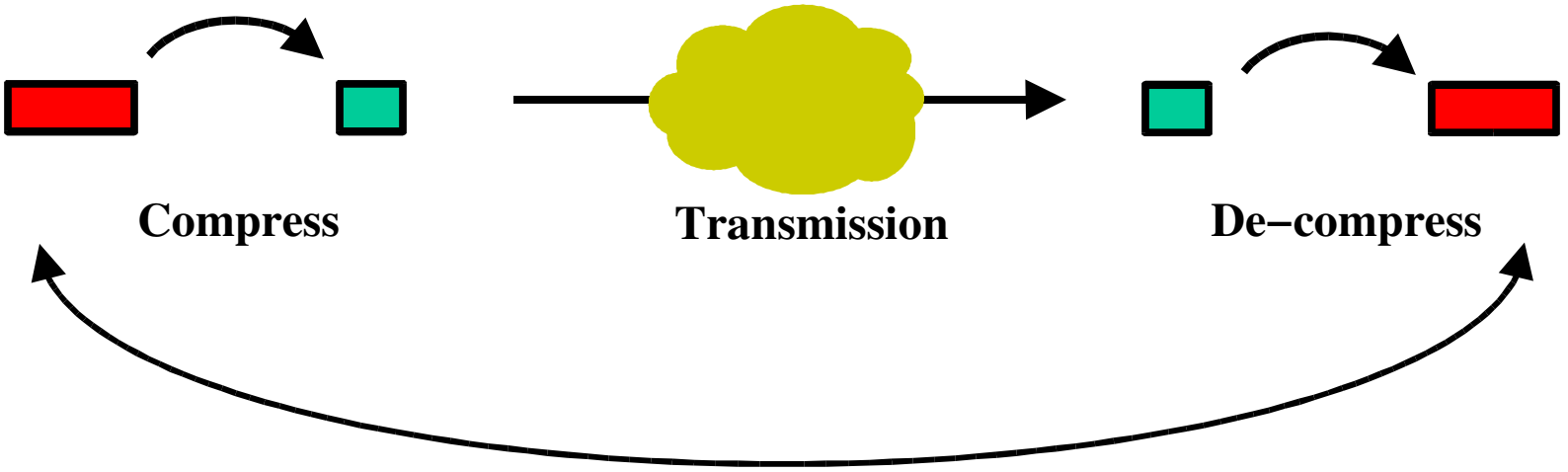
- End–to–end G.711 mandatory (e.g., V.90 modem pass–through in low–loss QoS environments).
- Applications where "instantaneous" real–time QoS bandwidth saved via compression is available for other elastic traffic.

Design Goals:

- G.711 Lossless Compression (both A–law and μ–law).
- Low complexity (on par with SRTP & DTMF decoders).
- If G.711 packet payload is "uncompressible", expansion is limited to one byte.
- Accommodate ***ANY*** G.711 payload (assumes zero–mean acoustical input sources – but can handle any payload)
- Compression of arbitrary length G.711 samples/frame (10 msec, 20 msec, ATM:AAL2/AAL5 sizes, etc.).

RGL Codec – Basic Idea

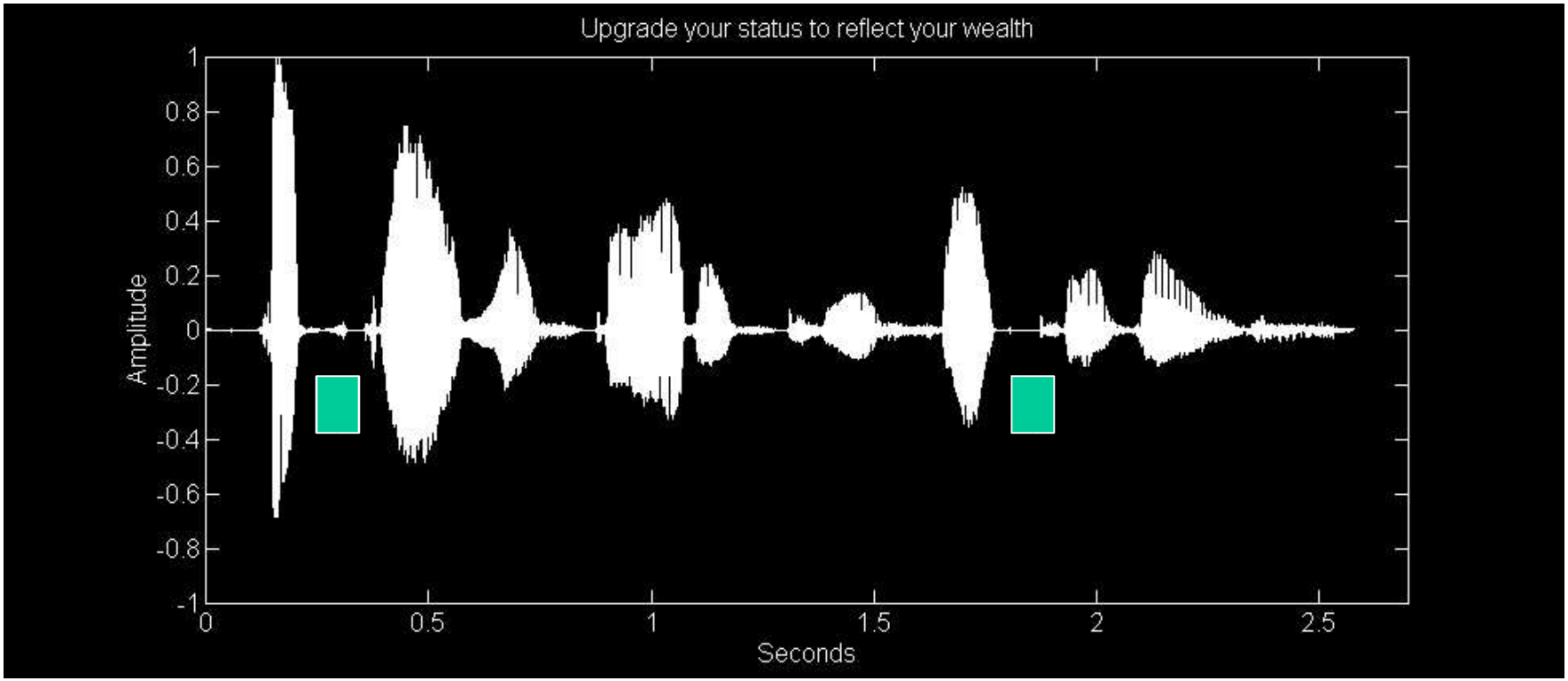
G.711 Lossless Codec



Lossless = Identical

A “ZIP–like compression” for G.711 payloads

RGL Codec – Methodology



= An approximate 10 msec segment

There are many 10 millisecond frames that exercise less than half the full input range – can save at least one bit for these segments!

(It's a brain–dead simple concept – no rocket science needed)

RGL Frame Format

Observations on RGL frames representing M samples of G.711:

- RGL frame can be from 1 to (M+1) bytes long. For all "eight bit per sample" encodings, the (M+1) byte RGL frame has a deterministic first byte (it is always {00011110}).
- Cannot determine number of samples in a received RGL frame (i.e., not sent as part of the RGL frame payload). This must be determined via SDP ("ptime" or the default "ptime" for codec) or sent explicitly in the corresponding RTP payload format (next draft to be discussed).
- A RGL decoder can successfully decode back to G.711 using only the RGL frame and knowledge of the number of G.711 samples in the frame. That is, the RGL frame is not "self describing".
- Version 1.0.0 RGL encoder reserves seven "first byte codes", which are never produced via the RGL compression process (0x3E, 0x5E, 0x7E, 0x9E, 0xBE, 0xDE and 0xFE). The proposed RTP payload format exploits the "reserved codes" in for use in RGL payload format definition.

RGL Compression Results

Talker Loudness	Background Noise Condition	Voice Activity Factor			
		35%	40%	45%	50%
Loud	Artificial Zero	68.4%	64.0%	59.7%	55.4%
	Near Zero	44.4%	41.9%	39.4%	36.9%
	Very Low	36.3%	34.4%	32.5%	30.7%
	Low	28.2%	26.9%	25.7%	24.4%
	Moderate	19.4%	19.4%	18.8%	18.2%
Nominal	Artificial Zero	72.8%	69.0%	65.3%	61.6%
	Near Zero	48.8%	46.9%	45.0%	43.2%
	Very Low	40.7%	39.4%	38.2%	36.9%
	Low	32.5%	31.9%	31.3%	30.7%
	Moderate	24.4%	24.4%	24.4%	24.4%

- **RGL codec has high compression during non–speech.**
- **Most compression gains occur during non–speech and are highly dependent upon background noise condition (VAD/SS not recommended).**

RGL Codec – Summary

- G.711 Lossless Compression (both A–law and μ–law). Identical fidelity to G.711 – no coding artifacts.
- Low complexity (0.16 MIPS encode, 0.14 MIPS decode).
- If G.711 packet payload is “uncompressible”, expansion is limited to one deterministic overhead byte ({00011110}).
- Accommodates ***ANY*** G.711 payload (assumes zero–mean acoustical input sources – but lossless under any payload).
- Compression of arbitrary length G.711 samples/frames.
- VAD/SS not recommended, as RGL has high compression during periods of non–speech (no silence suppression induced artifacts).
- Attractive for applications where transport G.711 is mandatory and “instantaneous” real–time QoS bandwidth saved via compression can be profitably used by applications using elastic transport protocols.

For source & documentation: www.vovida.org –> RGL

RTP Payload Format for RGL Codec

IETF draft:

http://www.ietf.org/internet–drafts/draft–ramalho–rgl–rtpformat–01.txt

HTML version of draft:

http://www.winlab.rutgers.edu/~ramalho/rgl_rtp_01.html

Whitepaper & code at:

www.vovida.org

Michael Ramalho

mramalho@cisco.com

RTP Payload Format for RGL Codec

Design Goals:

- Efficient format for one RGL frame per RTP payload.
- Efficient format for a common "two RGL frames" per RTP payload case (each with the identical number of samples per RGL frame). Common examples are two, 10 millisecond RGL frames (G.729 case) or two 15 millisecond frames (G.723 case).
- Efficient format for a common "three RGL frames" per RTP payload case (each with the identical number of samples per RGL frame). A common example is three, 10 millisecond RGL frames (G.723 case).
- A less efficient, but generic, format that allows for a multiplicity of RGL frames in the RTP payload (each with an arbitrary number of samples per RGL frame).

RTP Payload Format for RGL Codec

RGL Payload Format Methodology:

- The Version 1.0.0 RGL encoder reserves seven "first byte codes" (0x3E, 0x5E, 0x7E, 0x9E, 0xBE, 0xDE and 0xFE) that are never produced via the RGL compression process.
- To accommodate a TOC necessary for all but the "single RTP frame in payload" case, one of these codes will be the first byte of the TOC.
- The "single RGL frame per RTP payload" case is **inferred** by the absence of one of the "reserved codes" as the first byte of the RTP payload.

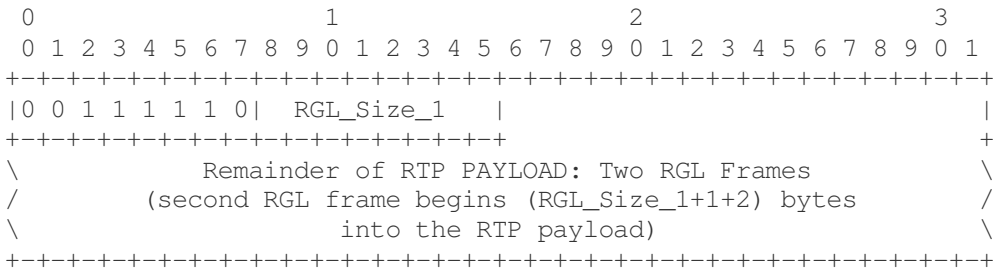
RTP Payload Format for RGL Codec

Type One: Single RGL frame in RTP Payload

- The entire RGL frame is placed in the RTP payload.
- The first byte in the RTP payload is therefore **NOT** one of the reserved seven "first byte codes" (0x3E, 0x5E, 0x7E, 0x9E, 0xBE, 0xDE and 0xFE). The RTP decoder will thereby determine the entire payload contains only one RGL frame.
- The RGL decoder knows the number of G.711 samples contained in the RGL frame via the SDP "ptime" parameter (or the default "ptime" for RGL, which is 20 milliseconds).
- If one desires to transmit the number of samples in the RTP payload explicitly, the "Type Four" encoding should be used instead.

RTP Payload Format for RGL Codec

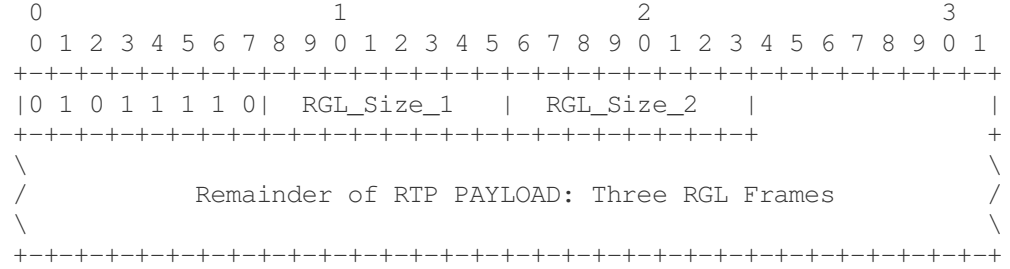
Type Two: Two RGL frame in RTP Payload
(w/ identical number of samples in each RGL frame)



- The TOC for this case is a simple one, the “reserved code” 0x3E followed by the unsigned char “RGL_Size_1”. The two RGL frames follow the TOC.
- The first RGL frame begins immediately after the RGL_Size_1 and is exactly [RGL_Size_1+1] bytes long.
- The second RGL frame begins immediately after the first RGL frame.
- The number of G.711 samples contained in the RGL frames is determined via the “ptime” parameter (default or otherwise).
- If one desires to transmit the number of samples in the RTP payload explicitly, the “Type Four” encoding should be used instead.

RTP Payload Format for RGL Codec

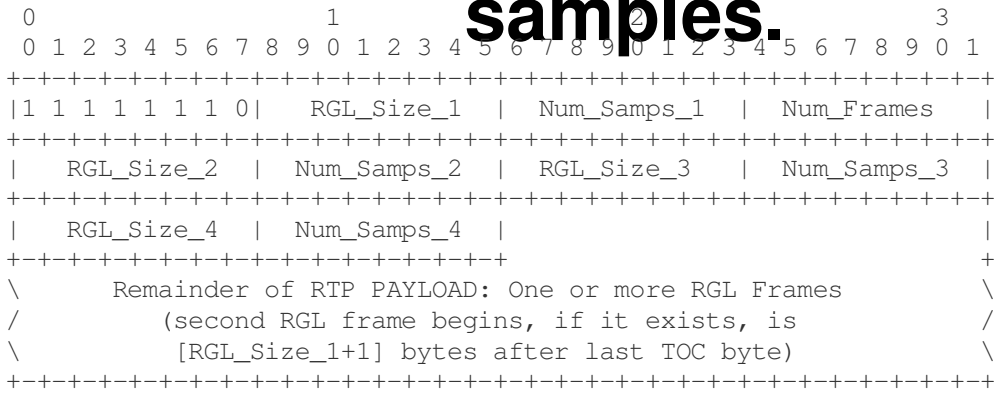
Type Three: Three RGL frame in RTP Payload (w/ identical number of samples in each RGL frame)



- The TOC for this case is the “reserved code” 0x5E followed by RGL_Size_1 and RGL_Size_2. The two RGL frames follow the TOC.
- The first RGL frame begins immediately after the RGL_Size_2 and is exactly [RGL_Size_1+1] bytes long.
- The second RGL frame begins immediately after the first RGL frame and is exactly [RGL_Size_2+1] bytes long. The third RGL frame is after the second.
- The number of G.711 samples contained in the RGL frames is determined via the “ptime” parameter (default or otherwise).
- If one desires to transmit the number of samples in the RTP payload explicitly, the “Type Four” encoding should be used instead.

RTP Payload Format for RGL Codec

Type Four: Arbitrary number of RGL frames in RTP Payload, each with an arbitrary number of samples.



- The TOC is 0xFE followed by RGL_Size_1, Num_Samps_1, Num_Frames followed by zero or more {RGL_Size_j, Num_Samps_j} tuples dependent on the number of RGL frames in the payload.
- The number of RGL frames in the payload is exactly Num_Frames.
- RGL frame j is exactly [RGL_Size_j+1] bytes long and has compressed exactly [Num_Samps_j] G.711 samples.
- The first RGL frame begins immediately after the TOC and each successive RGL frame follows afterwards.

RTP Payload Format for RGL Codec

Type Four: Arbitrary number of RGL frames in RTP Payload, each with an arbitrary number of samples.

SDP Issues:

• The number of samples contained in any RGL frame in the payload is specified by the corresponding Num_Samps_j parameter and overrides any "ptime" parameter specified by SDP.

• If SDP (or other media negotiation mechanism) is used, the sum of all the Num_Samp_j parameters SHOULD be consistent with the number of samples specified via "ptime" (assuming a given sampling rate, "ptime" essentially specifies the number of samples in the payload).

•The number of RGL frames in the payload should be an integer number (no fractional RGL frames).

Other Issues:

•Should I develop a "bulk/storage" mode?

RTP Payload Format for RGL Codec

IETF draft:

http://www.ietf.org/internet–drafts/draft–ramalho–rgl–rtpformat–01.txt

HTML version of draft:

http://www.winlab.rutgers.edu/~ramalho/rgl_rtp_01.html

Whitepaper & code at:

www.vovida.org

Michael Ramalho

mramalho@cisco.com