Opus: The Swiss-Army Knife of Audio Codecs

Jean-Marc Valin,
Koen Vos,
Gregory Maxwell, and
Timothy B. Terriberry

The Xiph.Org Foundation & The Mozilla Corporation

Outline

- **Introduction and Motivation**
- Opus Design
  - SILK
  - CELT
- Conclusion

Lossy Audio Codecs

- Two common types:
  - Speech/communication (G.72x, GSM, AMR, Speex)
    - Low delay (15-30 ms)
    - Low sampling rate (8 kHz to 16 kHz): limited fidelity
    - No support for music
  - General purpose (MP3, AAC, Vorbis)
    - High sampling rates (44.1 kHz or higher)
    - "CD-quality" music
    - High-delay (> 100 ms)
  - We want both: high fidelity with *very* low delay

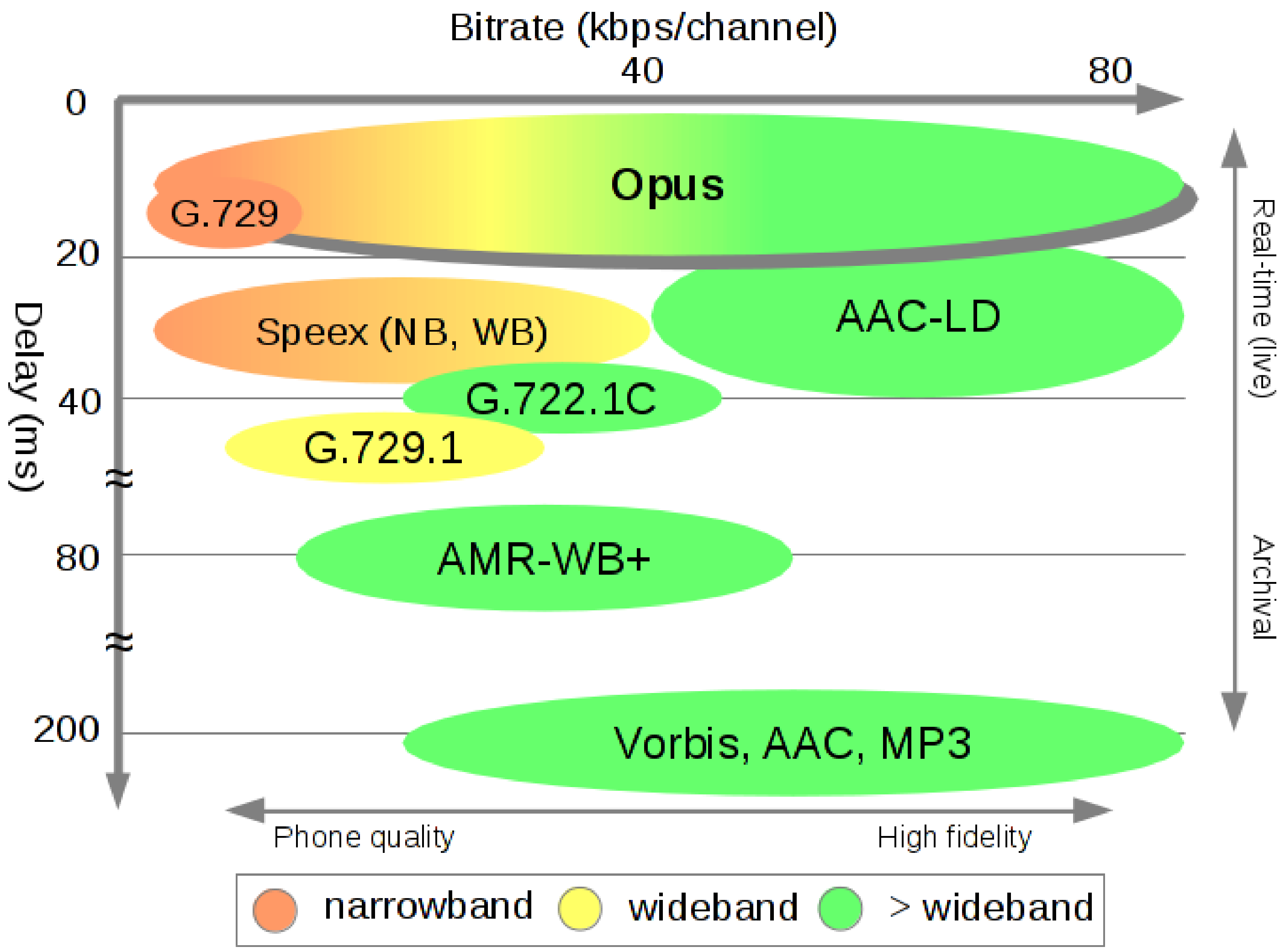
Coding Latency

- Low delay is critical to live interaction
  - Prevents collisions during conversation
  - Reduce need for echo cancellation
    - Good for small, embedded devices without much CPU
  - Higher sense of presence
  - Allows synchronization for live music
    - Need less than 25 ms *total* delay (Carôt 2006)
    - Equivalent to sitting 8 m apart (farther requires a conductor)
- Lower delay in the codec increases range
  - 1 ms = 200 km in fiber

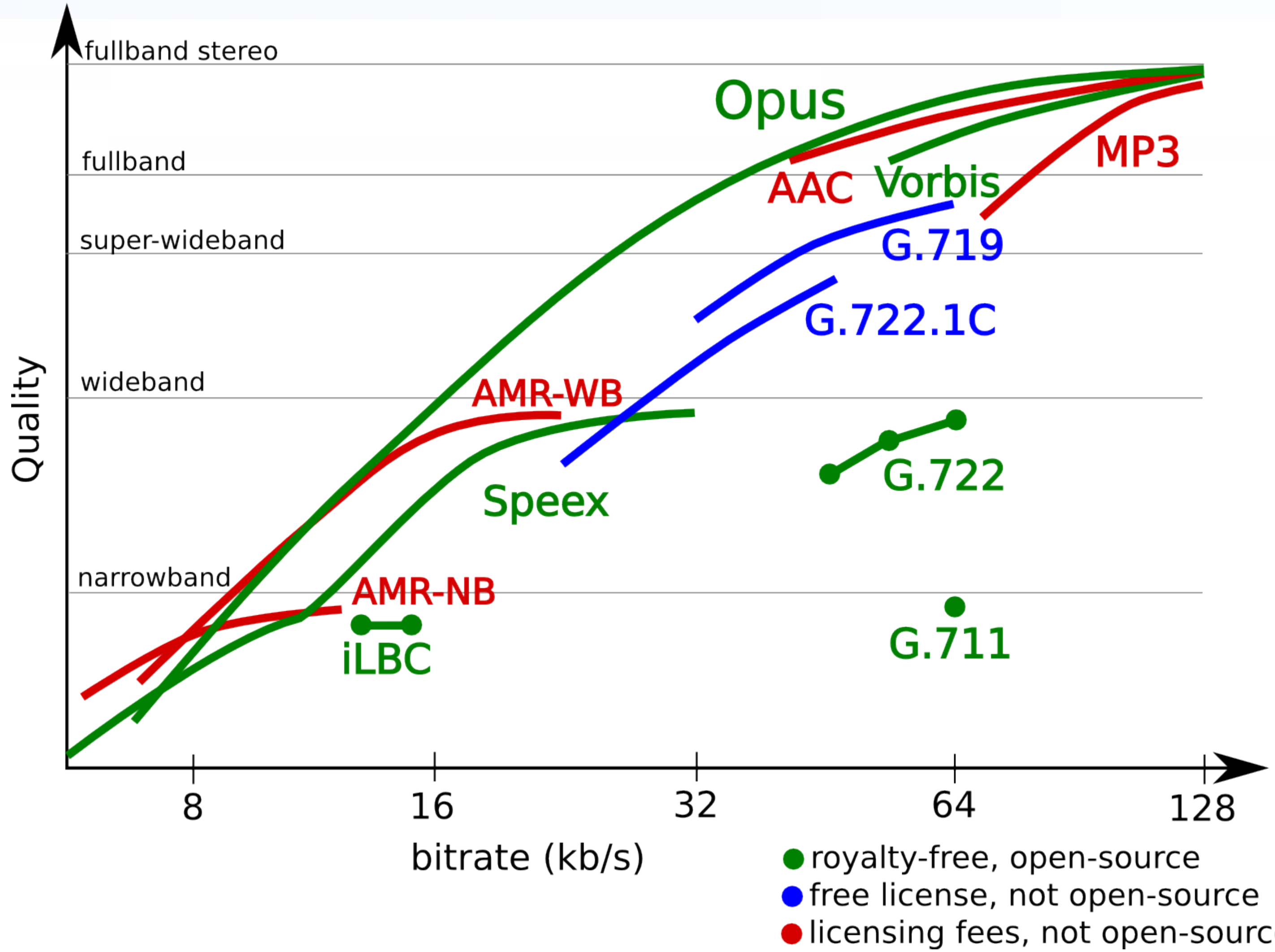
High delay (~250 ms)

Low delay (~15 ms)

Opus vs. the Competition: Latency



Opus vs. the Competition: Quality



Opus Features

- Sampling rate: 8...48 kHz (narrowband to fullband)
- Bitrates: 6...510 kbps
- Frame sizes: 2.5...20 ms
- Mono and stereo support
- Speech and music support
- Seamless switching between all of the above
- Combine multiple streams for up to 255 channels
- <u>It just works for everything</u>

Adaptive sweep: 8...64 kbps

Outline

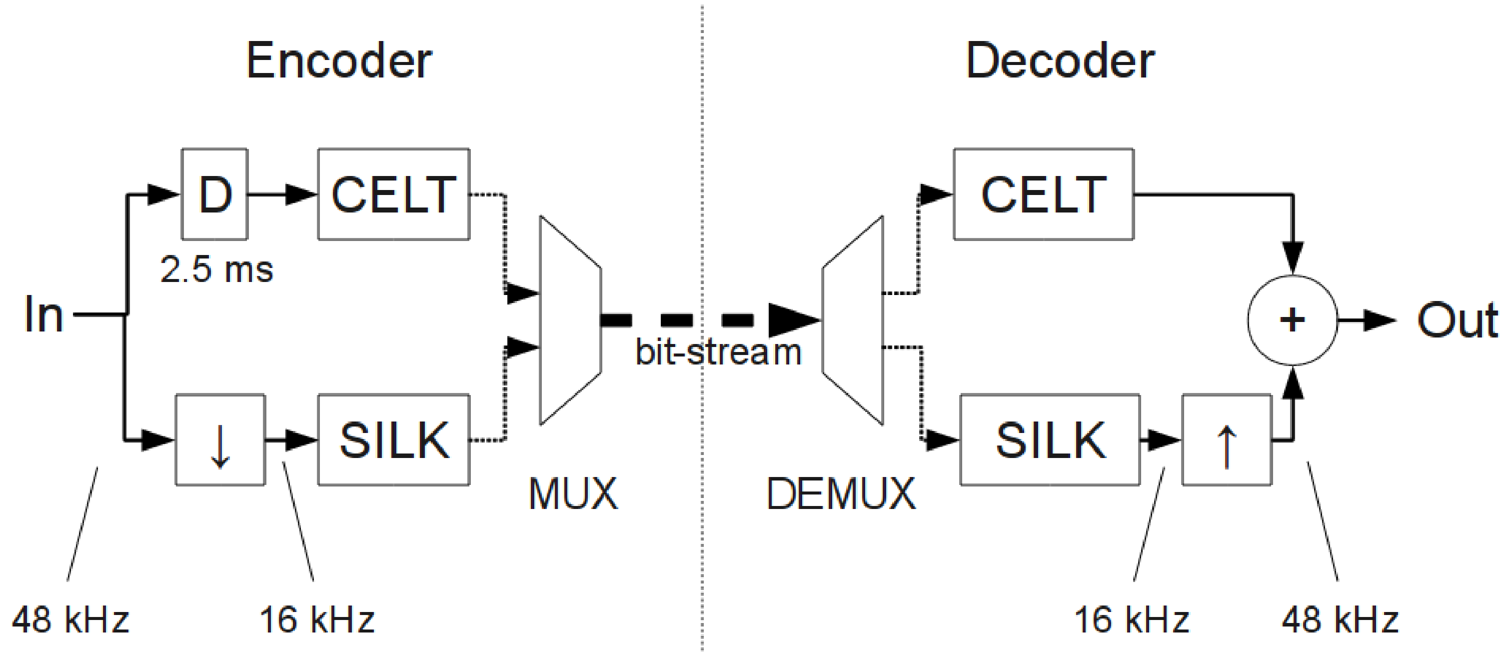
- Introduction
- **Opus Design**
  - SILK
  - CELT
- Conclusion

Opus Characteristics

- Standardized by the IETF (RFC 6716)
  - First free, state-of-the-art audio codec standardized
- Built out of two separate codecs
  - SILK: a *linear prediction* (speech) codec
    - In-development by Skype (now Microsoft) since Jan. 2007
  - CELT: an *MDCT* (music) codec
    - In-development by Xiph since November 2007
  - Both were modified a *lot* to form Opus
    - Standardization saw contributions from Mozilla, Microsoft (Skype), Xiph, Broadcom, Octasic, Google, etc.

Opus Operating Modes

- **SILK-only**: Narrowband (NB), Mediumband (MB) or Wideband (WB) speech
- **Hybrid**: Super-wideband (SWB) or Fullband (FB) speech
- **CELT-only**: NB to FB music



Outline

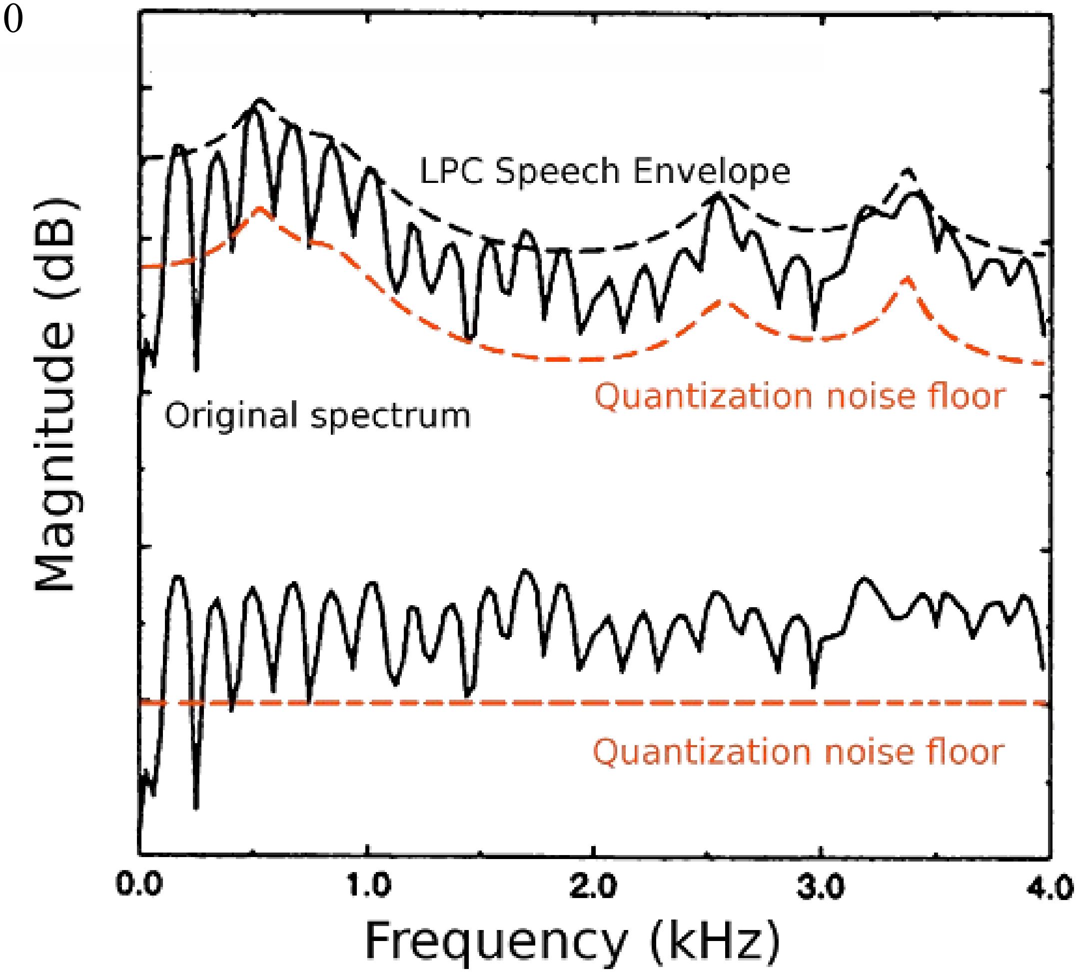
- Introduction
- Opus Design
  - SILK
    - Linear Prediction
    - Short-term Prediction (LPCs)
    - Long-term Prediction (LTP)
  - CELT
- Conclusion

SILK

- Linear prediction
  - Short-term prediction via a linear IIR filter
    - 10 or 16 coefficients (for NB or MB / WB respectively)
    - Good for speech: filter coefficients directly related to cross-sectional area of human vocal tract
  - Long-term prediction via a “pitch” filter
    - Good for “periodic” signals from 55.6 Hz to 500 Hz
- Variable bitrate
  - Quantization level controls rate indirectly
  - Range (arithmetic) coding with fixed probabilities

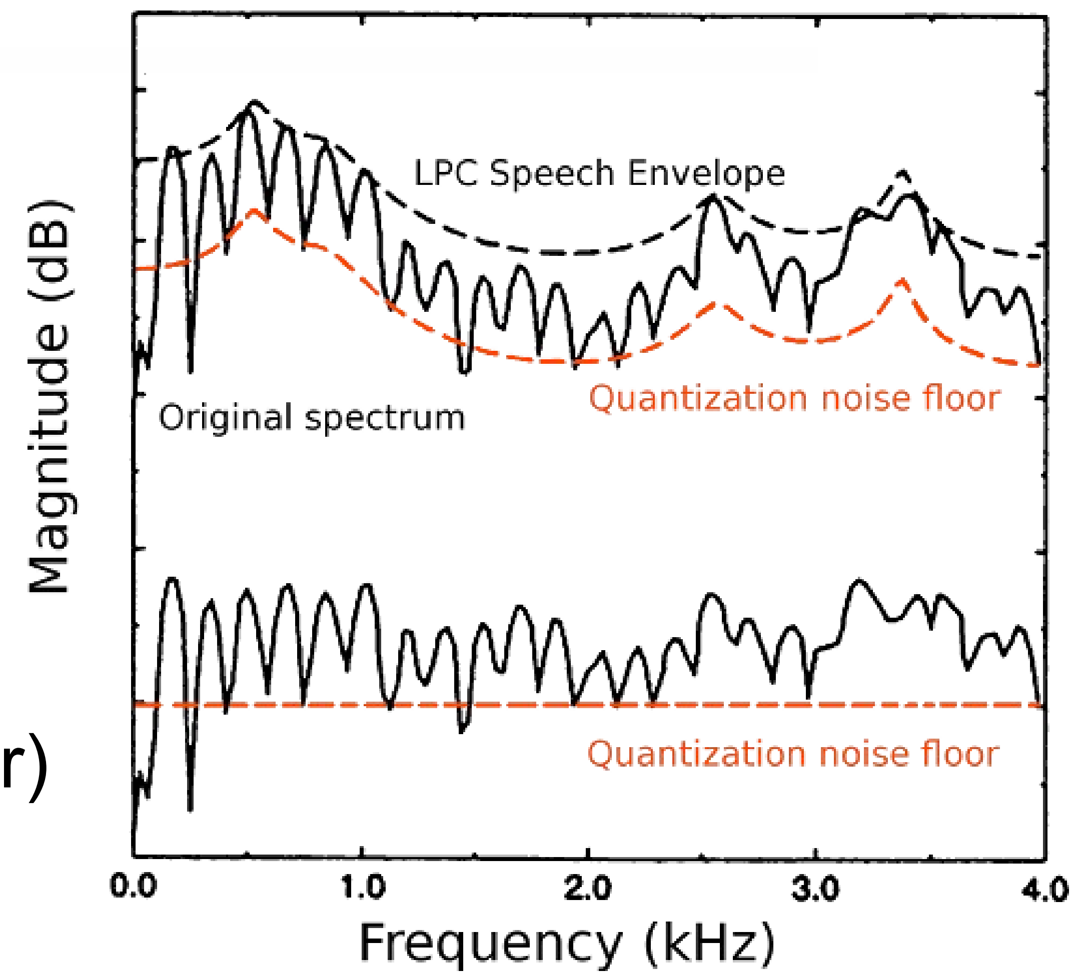
Linear Prediction

- IIR filter: $y[i]=x[i]+\sum_{k=0}^{D-1} a[k]y[i-k-1]$
- Analysis "whitens" a signal
- Quantization (lossy compression) adds noise
- Synthesis "shapes" the noise the same as the spectrum



Linear Prediction

- SILK: different analysis and synthesis filters
- De-emphasizes spectral valleys
  - Distortion least noticible there
  - Reduces entropy (distance between signal and noise floor)
    - Uses fewer bits



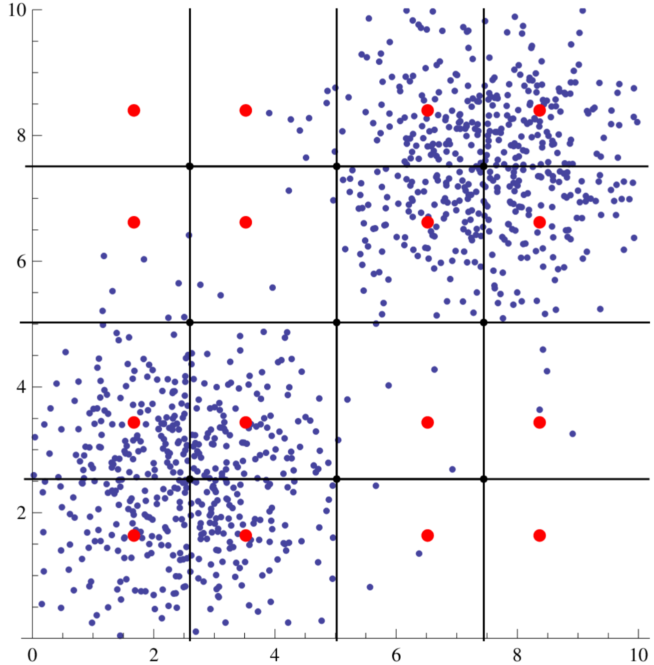
LPC Coefficients

- The filter *a*[*k*] needs to be quantized and transmitted
  - Quantizing the filter coefficients directly is bad
    - Drastically changes the frequency response of the filter
- Convert to “line spectral frequencies” (LSFs)
  - Split filter into two polynomials with roots on the unit circle (Itakura 1975)
    - Each root represents a frequency (0...π)
    - Math at http://en.wikipedia.org/wiki/Line_spectral_pairs
  - SILK quantizes LSFs using *vector quantization* (VQ) + scalar quantization

Vector Quantization

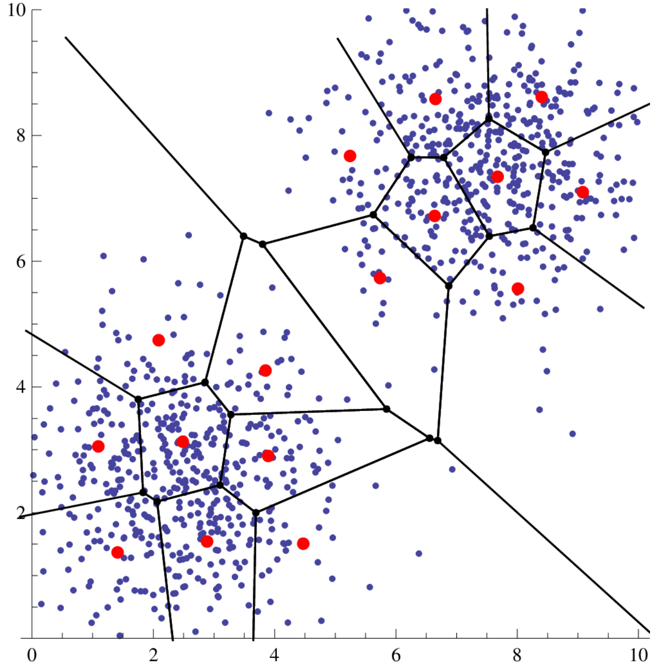
- Approximates a multidimensional distribution with a finite number of codewords (vectors)

Scalar Quantization (2 bits/dim)



RMS error = 0.89

Vector Quantization (2 bits/dim)

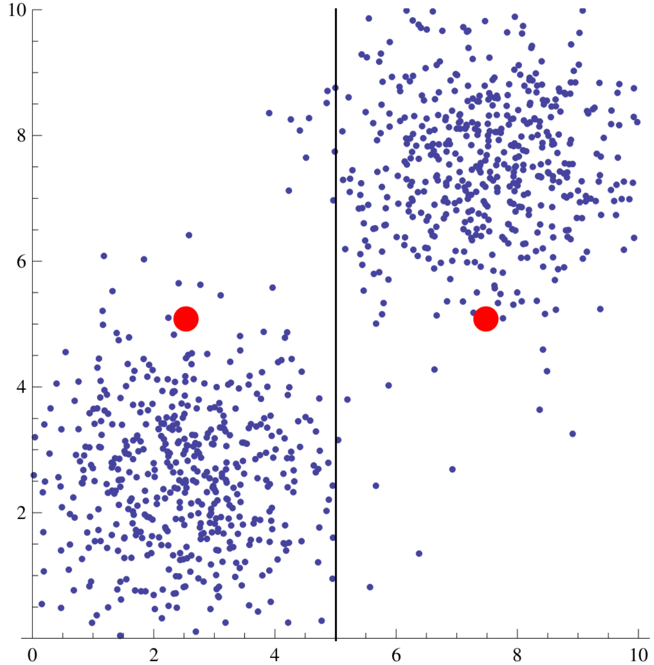


RMS error = 0.71
(20% better)

Vector Quantization

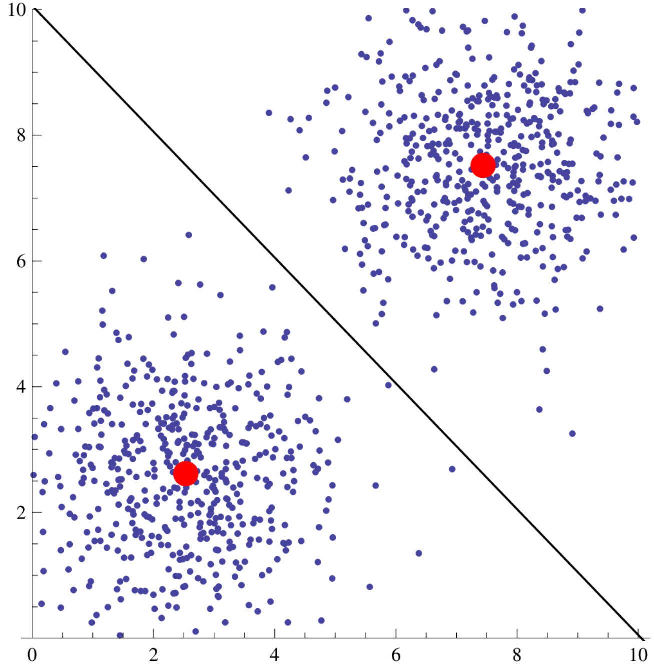
- Easily scales to less than 1 bit per dimension (Opus uses VQ with up to 176 dims)

Scalar Quantization (0.5 bits/dim)



RMS error = 2.93

Vector Quantization (0.5 bits/dim)



RMS error = 1.63
(44% better)

Quantizing LSFs: Stage 1

- Use a trained, 32-entry VQ codebook
  - Just search a big table for the best entry
  - 4.27 (NB) to 4.49 (WB) bits on average
- Good quality: less than 1 dB *spectral distortion* (SD)

$$\frac{1}{2\pi}\int_{-\pi}^{\pi}\left[10\log\left(S(\omega)\right)-10\log\left(\hat{S}(\omega)\right)\right]^2 d\omega$$

- We have 10 or 16 LSFs arranged arbitrarily on a circle (ignoring order): 32 entries is not enough

Quantizing LSFs: Stage 2

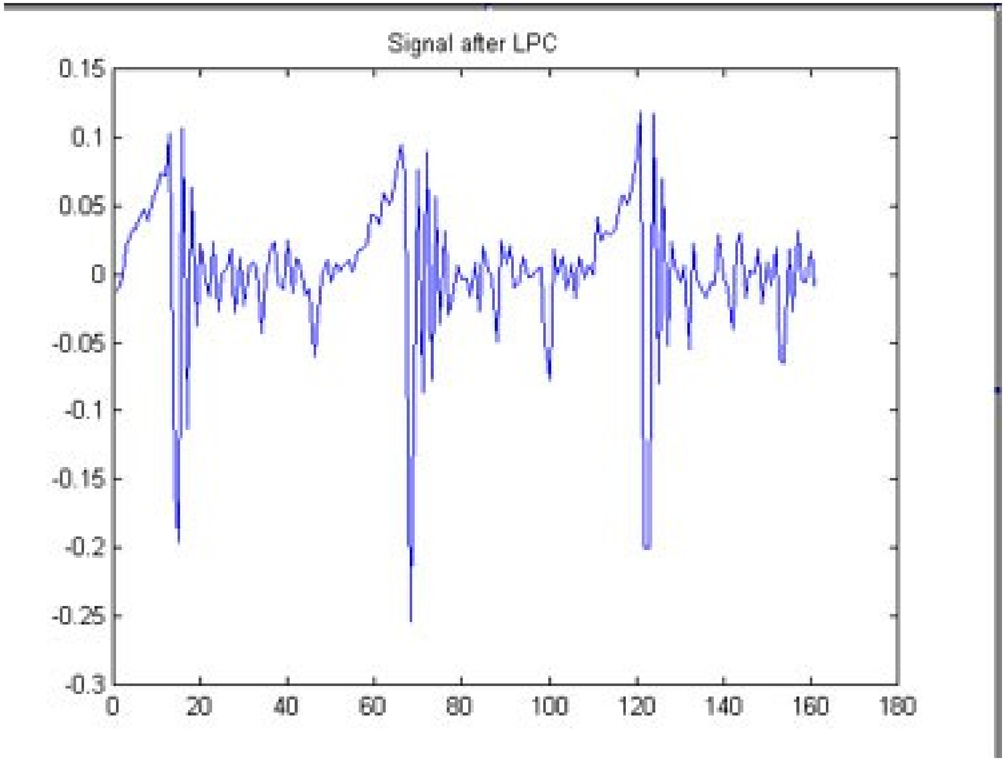
- Scalar quantization of error from stage 1
  - Also uses additional first-order prediction of error
- Error in LSFs has a non-uniform effect on SD
  - LSFs bunched close together more important
- SILK: Use LSFs from stage 1 to compute approximate weights (Laroia 1991)

$$w[k] = \frac{1}{c[k] - c[k-1]} + \frac{1}{c[k+1] - c[k]}$$

- Weights determine scalar quantization step size

Long-Term Prediction

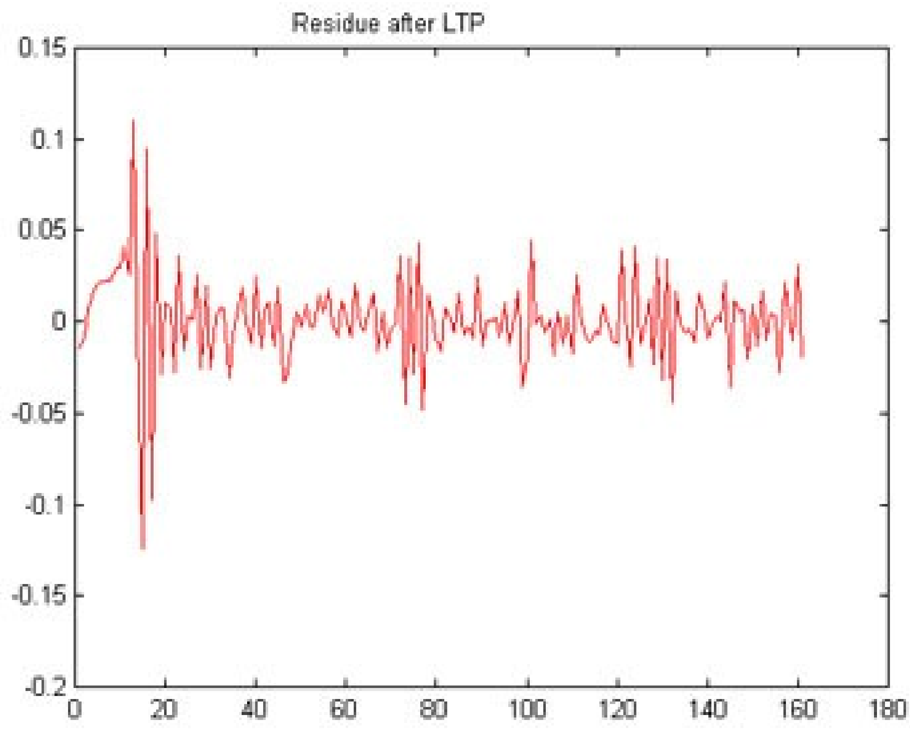
- LPC residual not really white (still periodic)



Picture blatantly stolen from
http://health.tau.ac.il/Communication Disorders/noam/noam_audio/adit_kfir/html/lpc3.htm

Long-Term Prediction

- Use *long-term* IIR: $x[i] = e[i] + \sum_{k=0}^{4} b[k]\, x[i-L-k-1]$
- *L* is the *pitch lag* (period)
- *b* signaled with another trained VQ codebook



Original After LPC After LTP

Picture and sounds blatantly stolen from
http://health.tau.ac.il/Communication Disorders/noam/noam_audio/adit_kfir/html/lpc3.htm

Handling Packet Loss

- LTP uses decoded signal from previous frames (up to 18 ms back)
  - Packet loss causes mis-prediction in future frames
- SILK: Artificially scale down previous frames
  - Uses less prediction (more bits) for the first period
    - But *only* affects the first pitch period
  - Amount depends on packet loss: signaled in bitstream (1.5 bits on average)

Outline

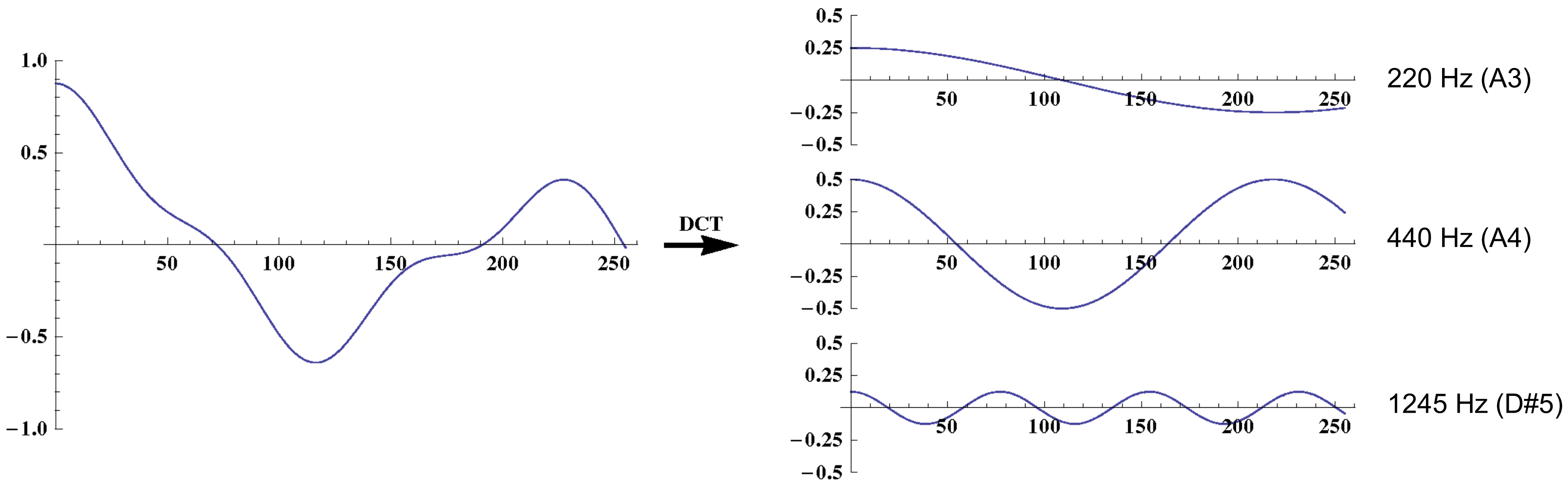
- Introduction
- Opus Design
  - SILK
  - **CELT**
    - "Lapped Transform"
    - "Constrained Energy"
    - Coding Band Shape
    - Psychoacoustics
- Conclusion

CELT: "Constrained Energy Lapped Transform"

- Transform codec (MDCT, like MP3, Vorbis)
  - Short windows → poor frequency resolution
- *Explicitly* code energy of each band of the signal
  - Coarse shape of sound preserved no matter what
- Code remaining details using algebraic VQ
- Useful roughly 40 kbps and above
  - Not good for low bitrate speech

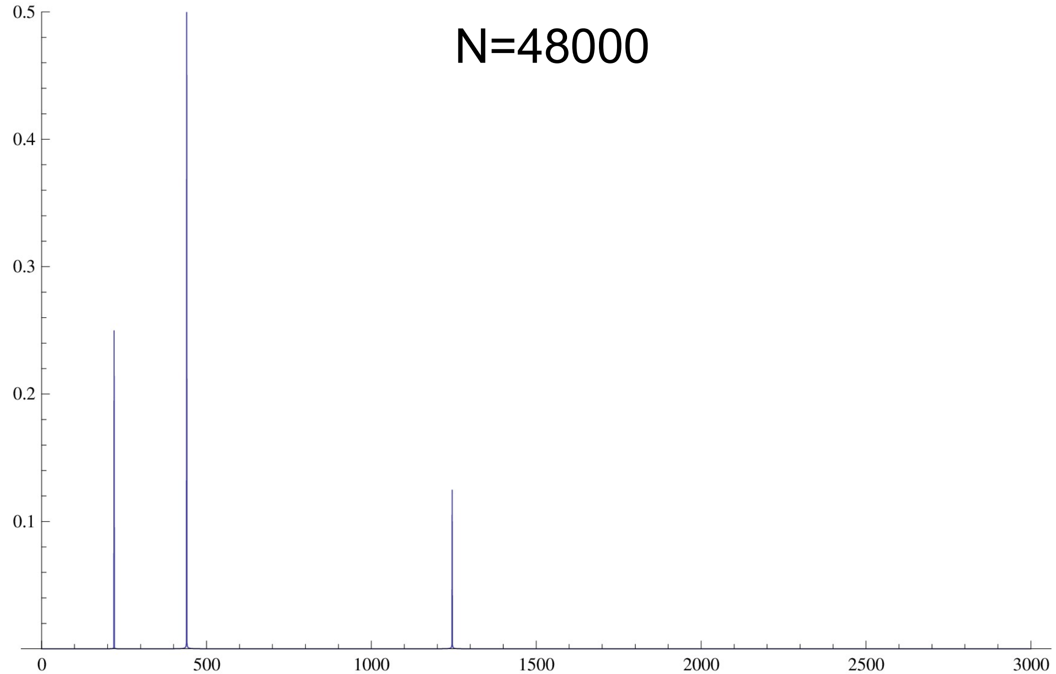
"Lapped Transform"
Time-Frequency Duality

- *Any* signal can be represented as a weighted sum of cosine curves with different frequencies
- The Discrete Cosine Transform (DCT) computes the weights for each frequency



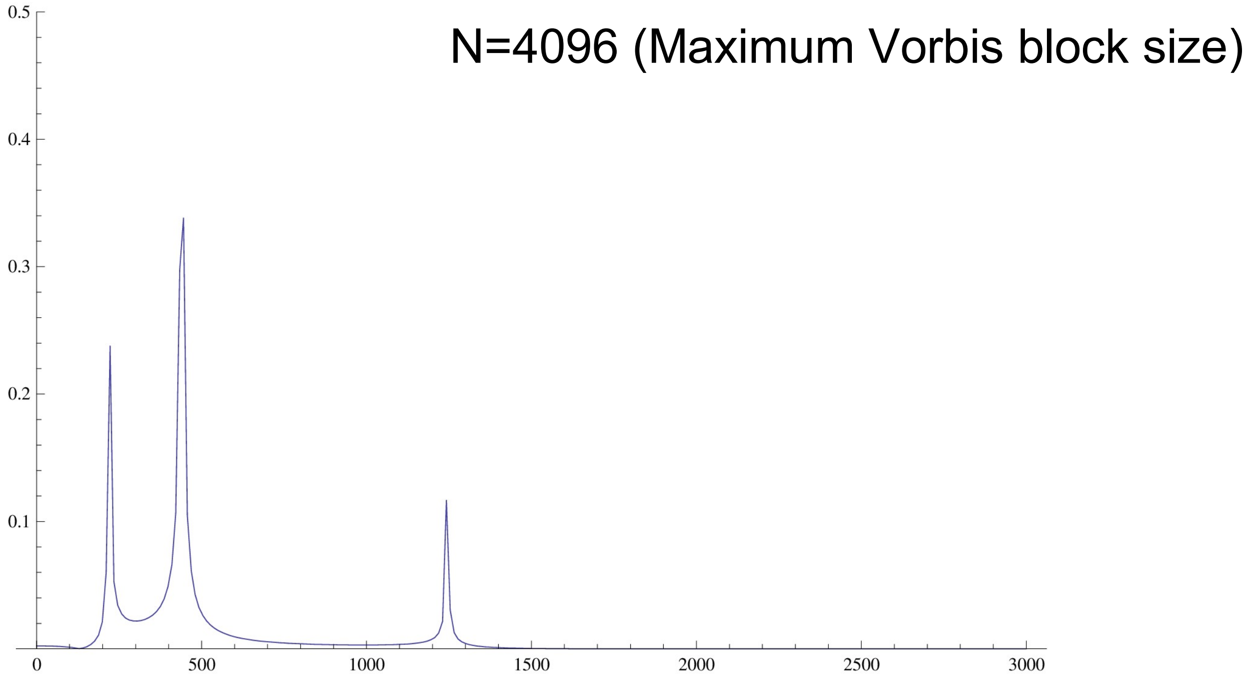
"Lapped Transform" Discrete Cosine Transform

- The "Discrete" in DCT means we're restricted to a finite number of frequencies
  - As the transform size gets smaller, energy "leaks" into nearby frequencies (harder to compress)



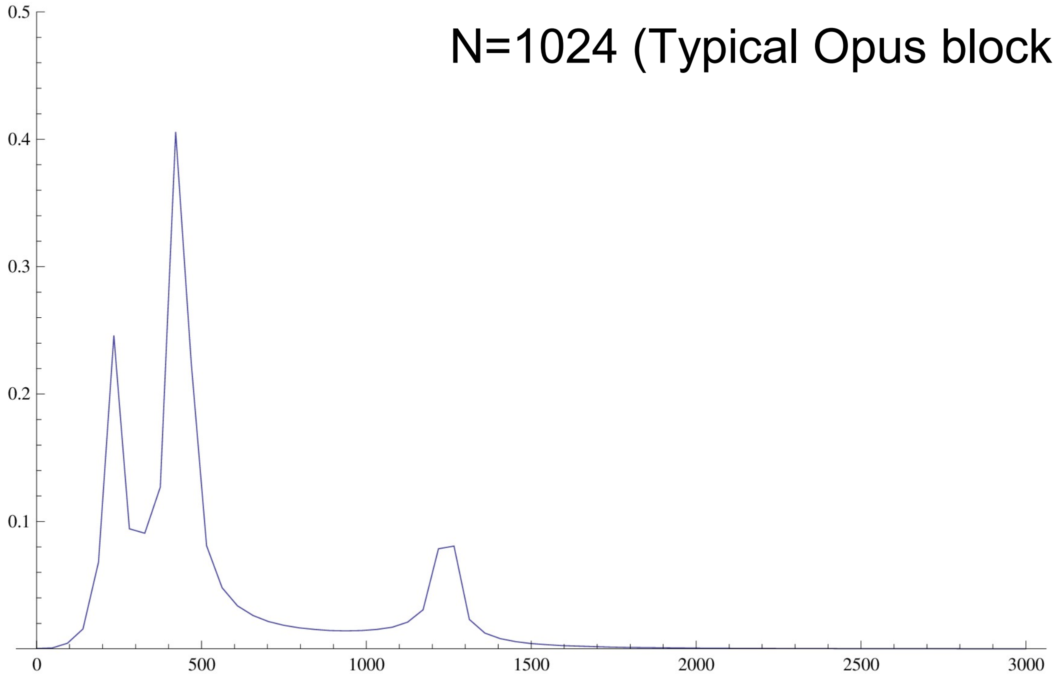
"Lapped Transform"
Discrete Cosine Transform

- The "Discrete" in DCT means we're restricted to a finite number of frequencies
  - As the transform size gets smaller, energy "leaks" into nearby frequencies (harder to compress)



"Lapped Transform"
Discrete Cosine Transform

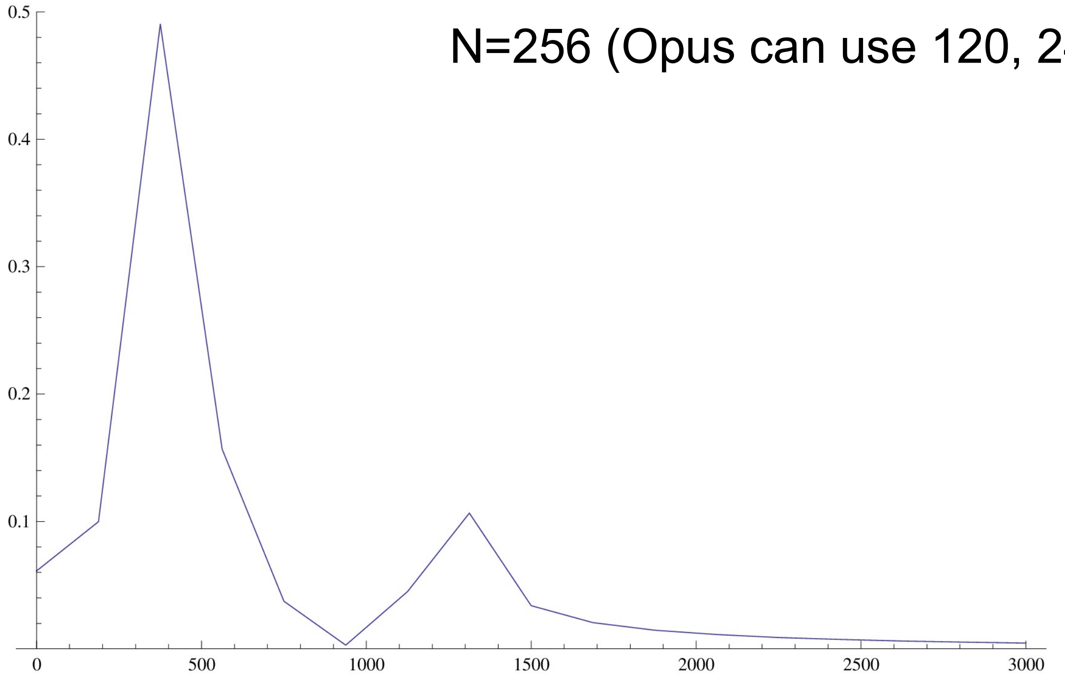
- The "Discrete" in DCT means we're restricted to a finite number of frequencies
  - As the transform size gets smaller, energy "leaks" into nearby frequencies (harder to compress)



N=1024 (Typical Opus block size = 960)

"Lapped Transform" Discrete Cosine Transform

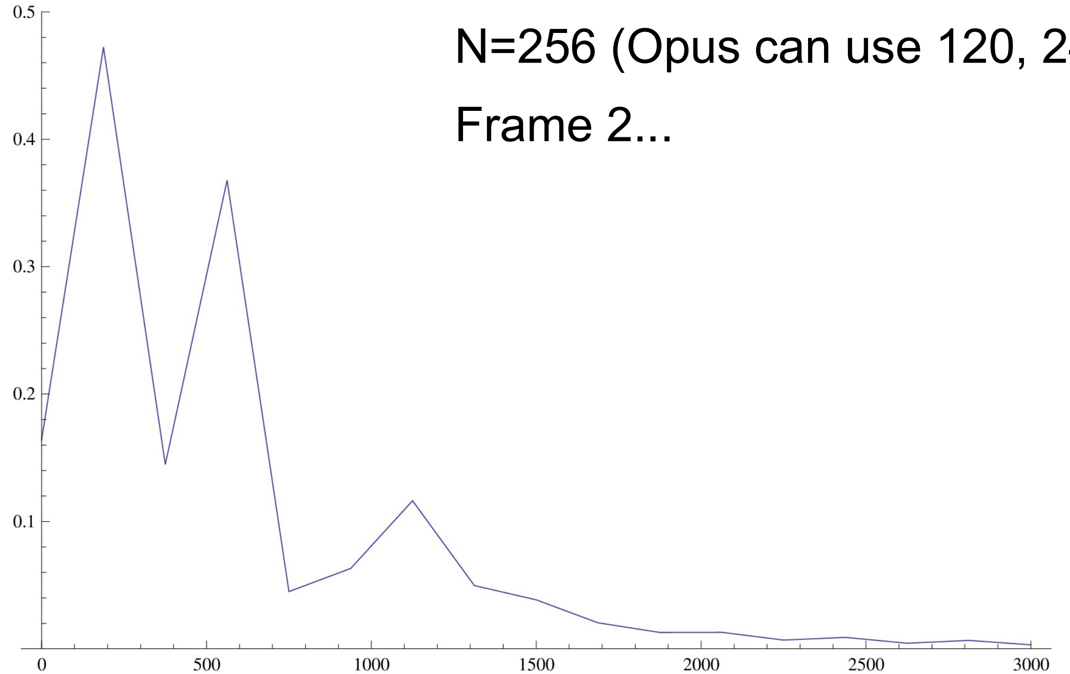
- The "Discrete" in DCT means we're restricted to a finite number of frequencies
  - As the transform size gets smaller, energy "leaks" into nearby frequencies (harder to compress)



N=256 (Opus can use 120, 240, 480 960)

"Lapped Transform" Discrete Cosine Transform

- The "Discrete" in DCT means we're restricted to a finite number of frequencies
  - As the transform size gets smaller, energy "leaks" into nearby frequencies (unstable over time)

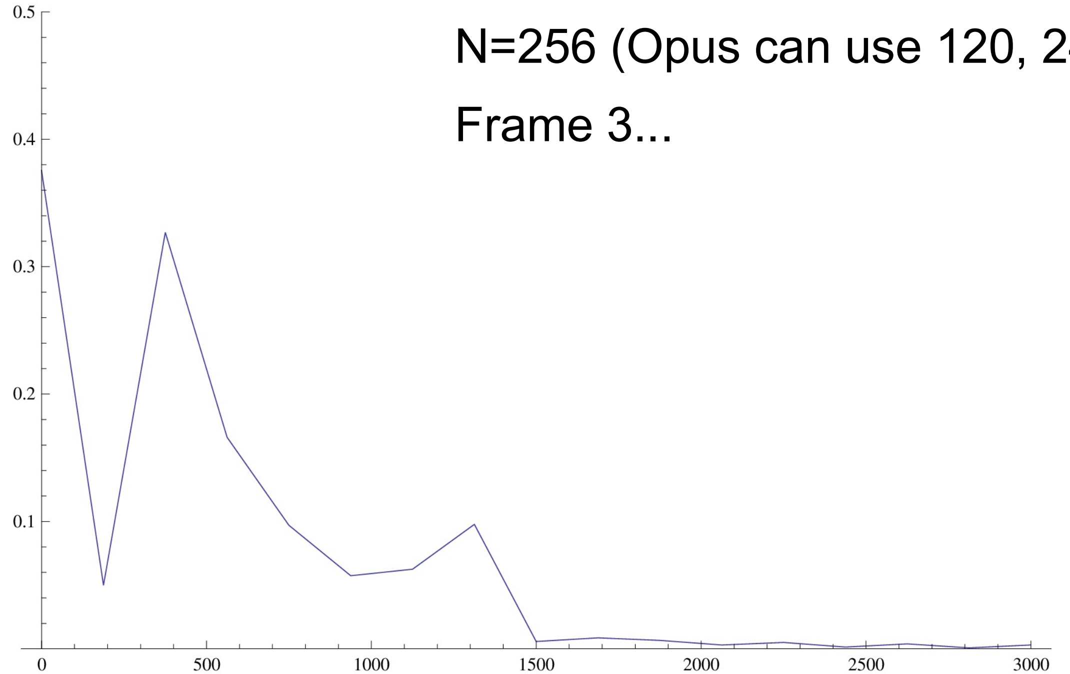


N=256 (Opus can use 120, 240, 480, 960)

Frame 2...

"Lapped Transform" Discrete Cosine Transform

- The "Discrete" in DCT means we're restricted to a finite number of frequencies
  - As the transform size gets smaller, energy "leaks" into nearby frequencies (unstable over time)

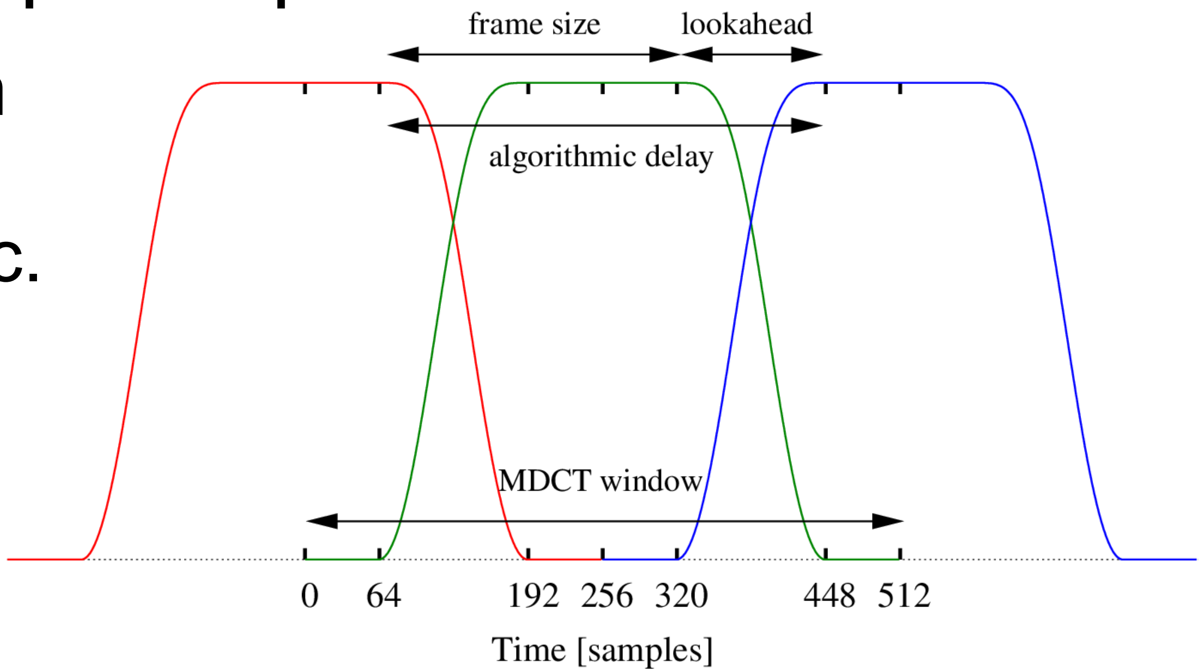


N=256 (Opus can use 120, 240, 480, 960)

Frame 3...

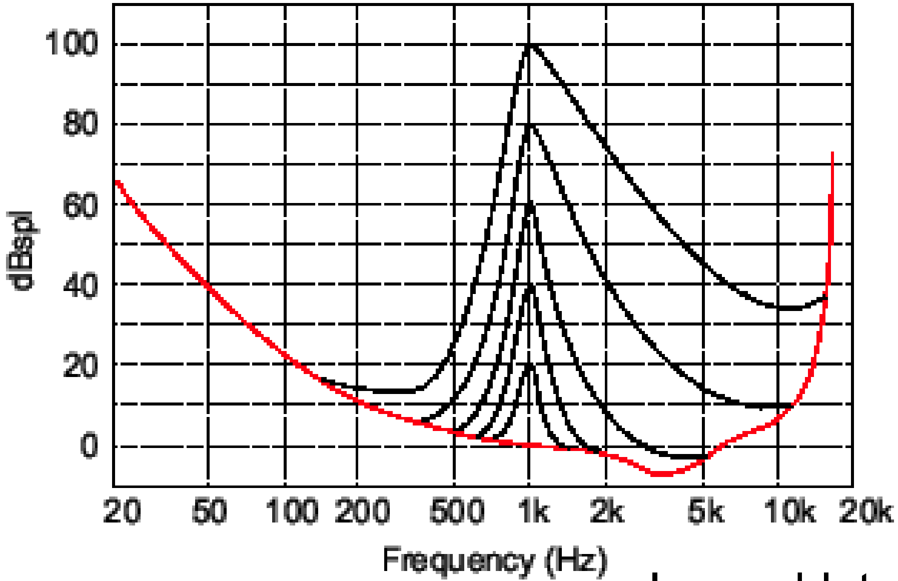
"Lapped Transform"
Modified DCT

- The normal DCT causes coding artifacts (sharp discontinuities) between blocks, easily audible
- The "Modified" DCT (MDCT) uses a decaying window to overlap multiple blocks
  - Same transform used in MP3, Vorbis, AAC, etc.
  - But with much smaller blocks, less overlap



"Constrained Energy" Critical Bands

- The human ear can hear about 25 distinct "critical bands" in the frequency domain
  - Psychoacoustic masking within a band is much stronger than between bands

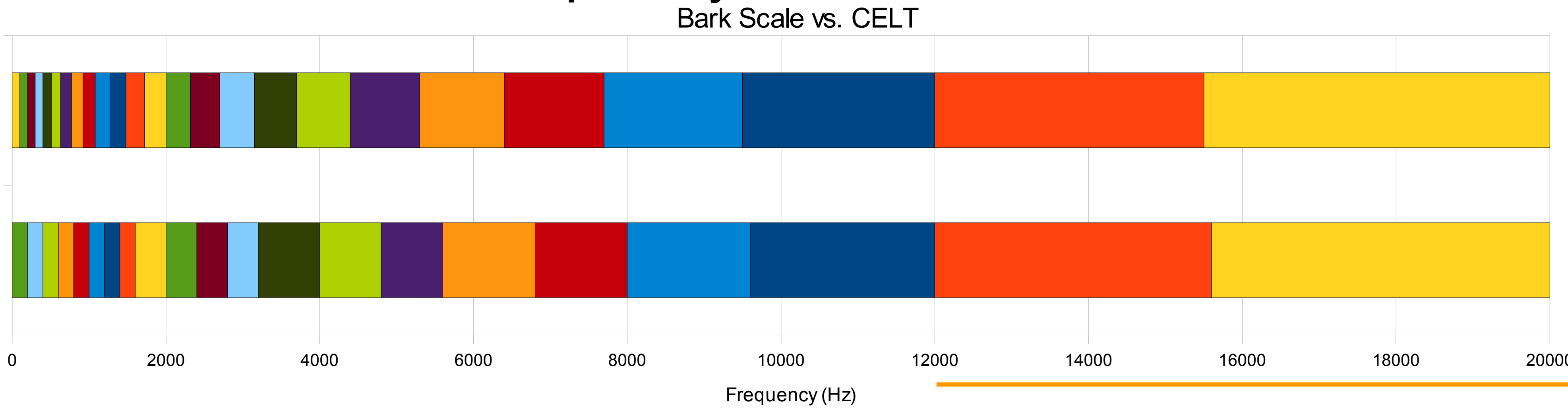


Threshold of detection in the presence of masker at 1kHz with a bandwidth of 1 critical band and various levels.

Image blatantly stolen from http://www.tonmeister.ca/main/textbook/node331.html

"Constrained Energy" Critical Bands

- Group MDCT coefficients into bands approximating the critical bands (Bark scale)
  - Band layout the same for all frame sizes
    - Need at least 1 coefficient for 120 sample frames
    - Corresponds to 8 coefficients for 960 sample frames
  - Insufficient frequency resolution for all the bands



"Constrained Energy"
Coding Band Energy

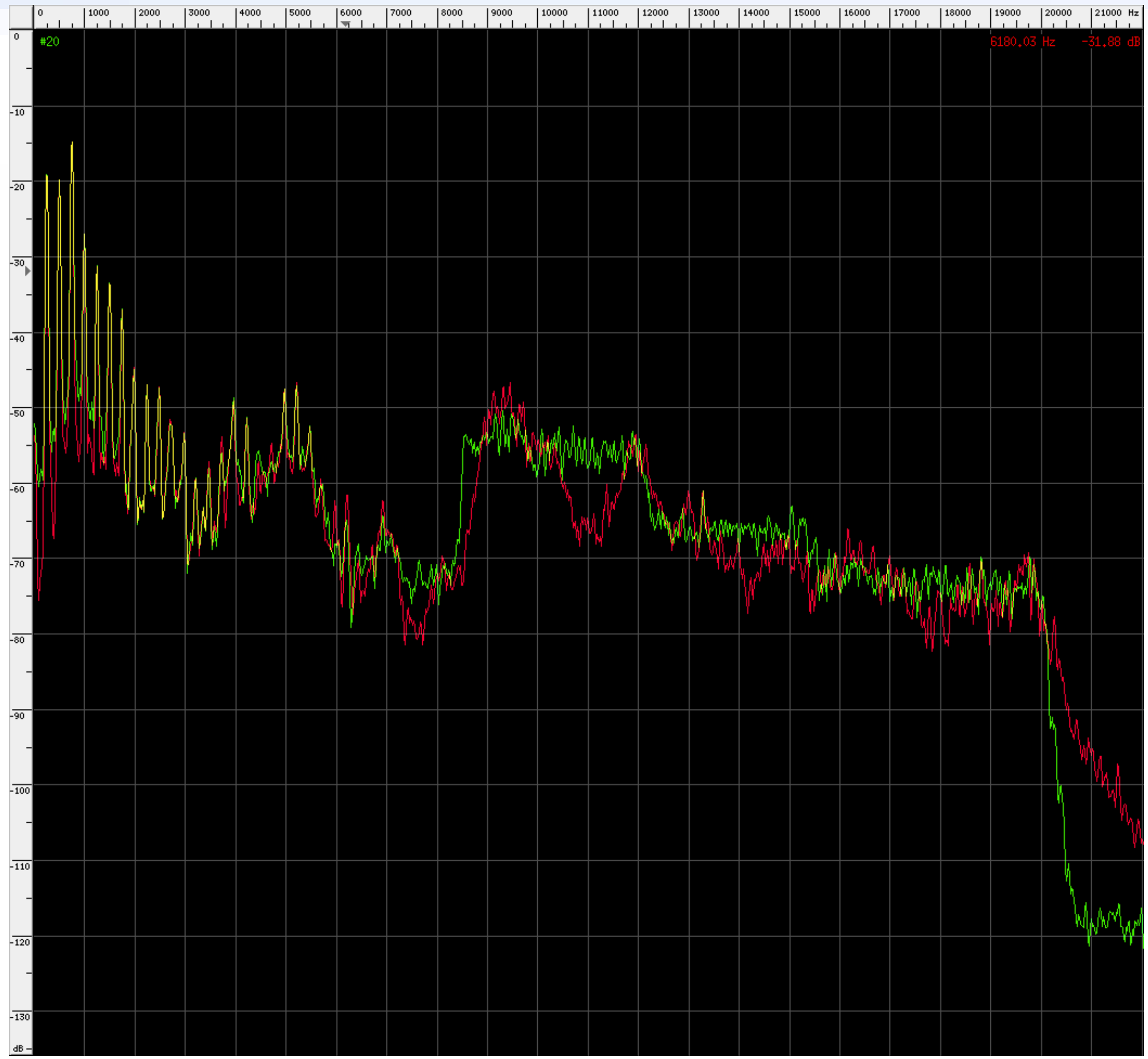
- Most important psychoacoustic lesson learned from Vorbis:

Preserve the energy in each band

- Vorbis does this implicitly with its "floor curve"
- CELT codes the energy explicitly
  - Coarse energy (6 dB resolution), predicted from previous frame and from previous band
  - Fine energy, improves resolution where we have available bits, not predicted

"Constrained Energy"
Coding Band Energy

- CELT (green) vs original (red)
  - Notice the quantization between 8.5 kHz and 12 kHz
  - Speech is intelligible using coarse energy alone (~9 kbps for 5.3 ms frame sizes)



Coding Band Shape

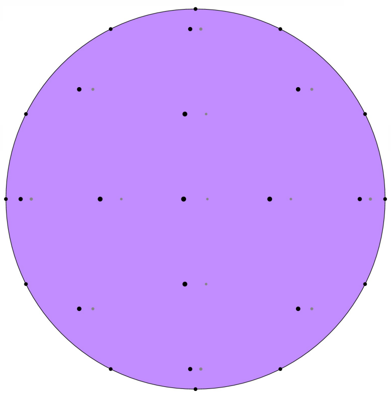
- After normalizing, each band is represented by an *N*-dimensional unit vector
  - Point on an *N*-dimensional sphere
  - Describes "shape" of energy within the band
- CELT uses *algebraic* vector quantization
  - Have lots of codebooks (# dims, bitrates)
  - Very large codebooks (exponential in # of dims)
    - 50 dims at 0.6 bits/dim is over 1 billion codebook entries
  - But we’re coding uniformly distributed unit vectors

Coding Band Shape
Algebraic Vector Quantization

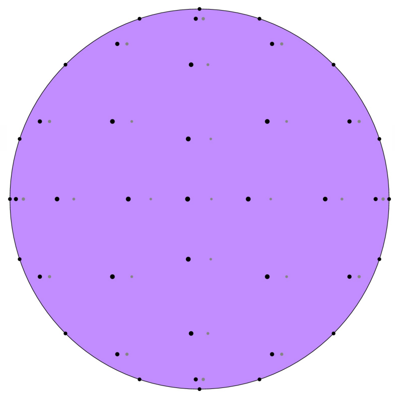
- Use a regularly structured, algebraic codebook: Pyramid Vector Quantization (Fischer, 1986)
  - We want evenly distributed points on a sphere
    - Don't know how to do that for arbitrary dimension
  - Use evenly distributed points on a pyramid instead
- For *N* dimensional vector, allocate *K* "pulses"
- Codebook: normalized vectors with integer coordinates whose magnitudes sum to *K*

$$S(N,K) = \{\frac{\mathbf{y}}{\|\mathbf{y}\|} \in \mathbb{Z}^N : \sum_{i=1}^{N} |y_i| = K\}$$

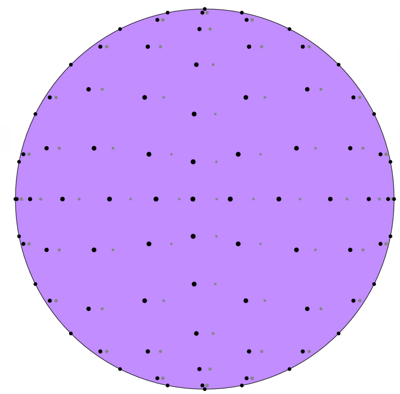
Coding Band Shape
N=3 at Various Rates



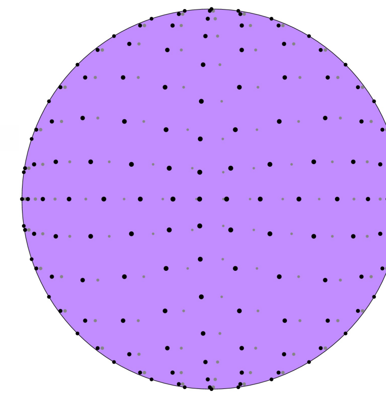
5.25 bits (K=3)



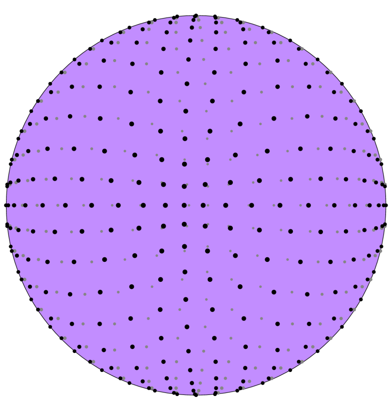
6.04 bits (K=4)



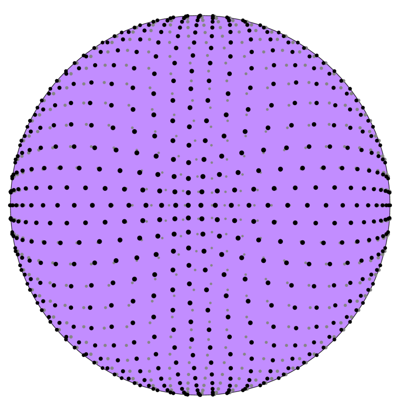
7.19 bits (K=6)



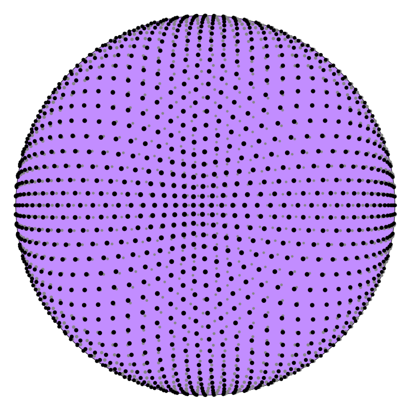
8.01 bits (K=8)



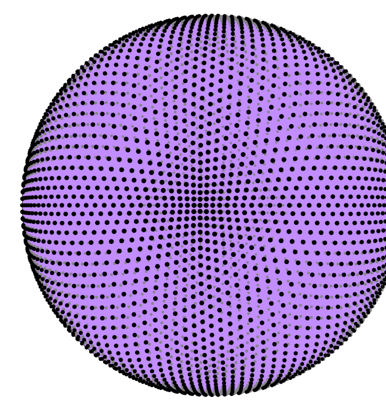
8.92 bits (K=11)



10.00 bits (K=16)



11.05 bits (K=23)



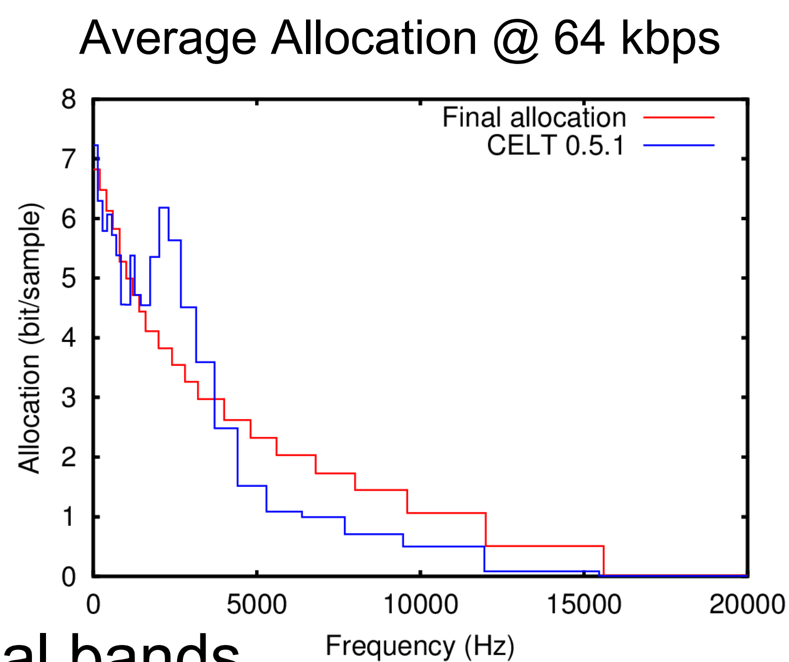
12.00 bits (K=32)

Coding Band Shape
Pyramid Vector Quantization

- PVQ codebook has a fast enumeration algorithm
  - Converts between vector and integer codebook index
  - $O(N+K)$ (lookup table, muls) or simpler $O(NK)$ (adds)
  - Latter great for embedded processors, often faster
- Fast codebook search algorithm: $O(N \cdot \min(N,K))$
  - Divide by L_1 norm, round down: at least K-N pulses
  - Place remaining pulses (at most N) one at a time
- Codebooks larger than 32 bits
  - Split the vector in half and code each half separately

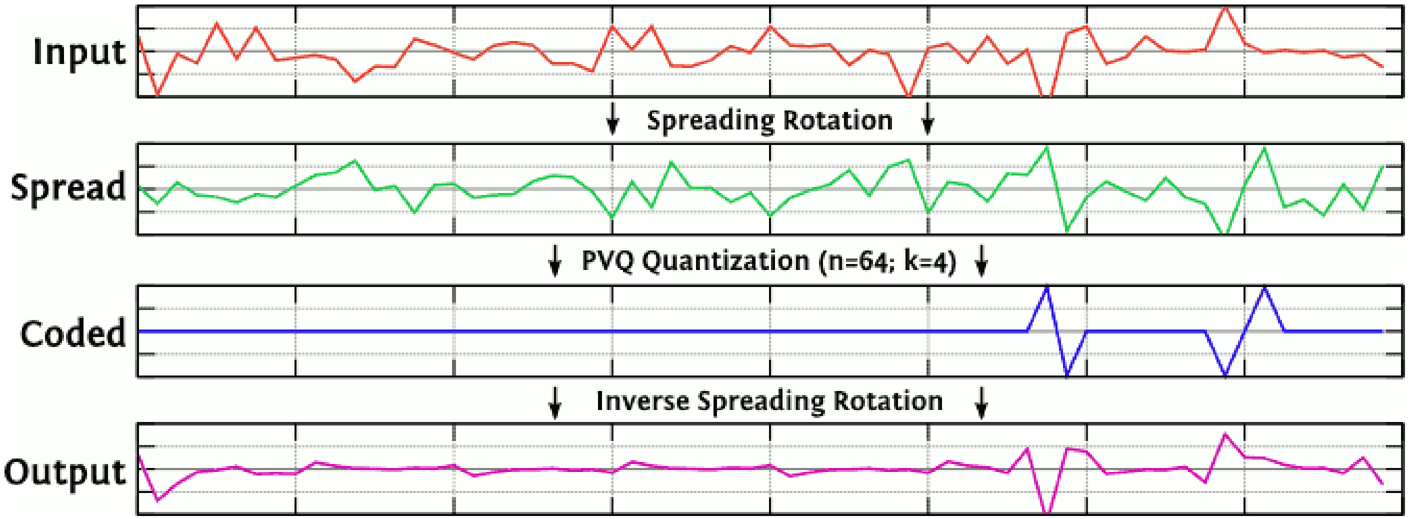
Psychoacoustics Rate Allocation

- Encoder decides final bitrate early on
  - Right after coarse energy and side information
  - Can change from packet to packet, to adapt to network conditions
- Allocation between bands mostly static
  - Roughly constant *signal-to-mask ratio*
  - Two knobs available:
    - Boost: Gives more bits to individual bands
    - Tilt: shifts bits from LF to HF



Psychoacoustics
Avoiding Birdie Artifacts

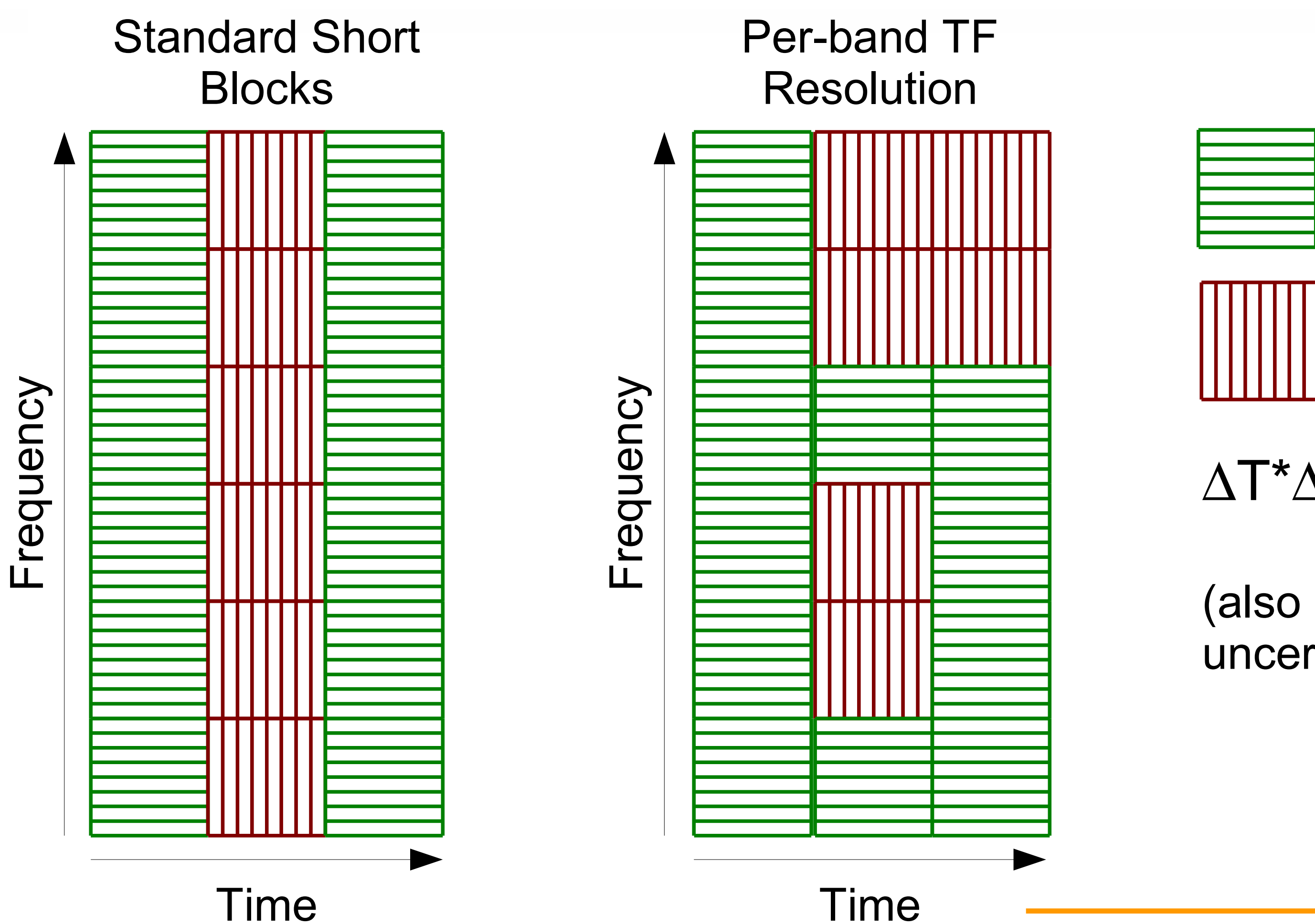
- Small *K* → sparse spectrum after quantization
  - Produces tonal “tweets” in the HF
- CELT: Use pre-rotation and post-rotation to spread the spectrum (make it “rougher”)
  - Completely automatic (no per-band signaling)



Psychoacoustics
Transients (pre-echo)

- Quant. error spreads over whole MDCT window
  - Can hear noise before an attack: pre-echo
- Split a frame into smaller MDCT windows (“short blocks”)
  - Interleave results and code as normal
    - Still code one energy value per band for all MDCTs
- Simultaneous tones and transients?
  - CELT: Use adaptive time-frequency resolution

Psychoacoustics
Time-Frequency Resolution



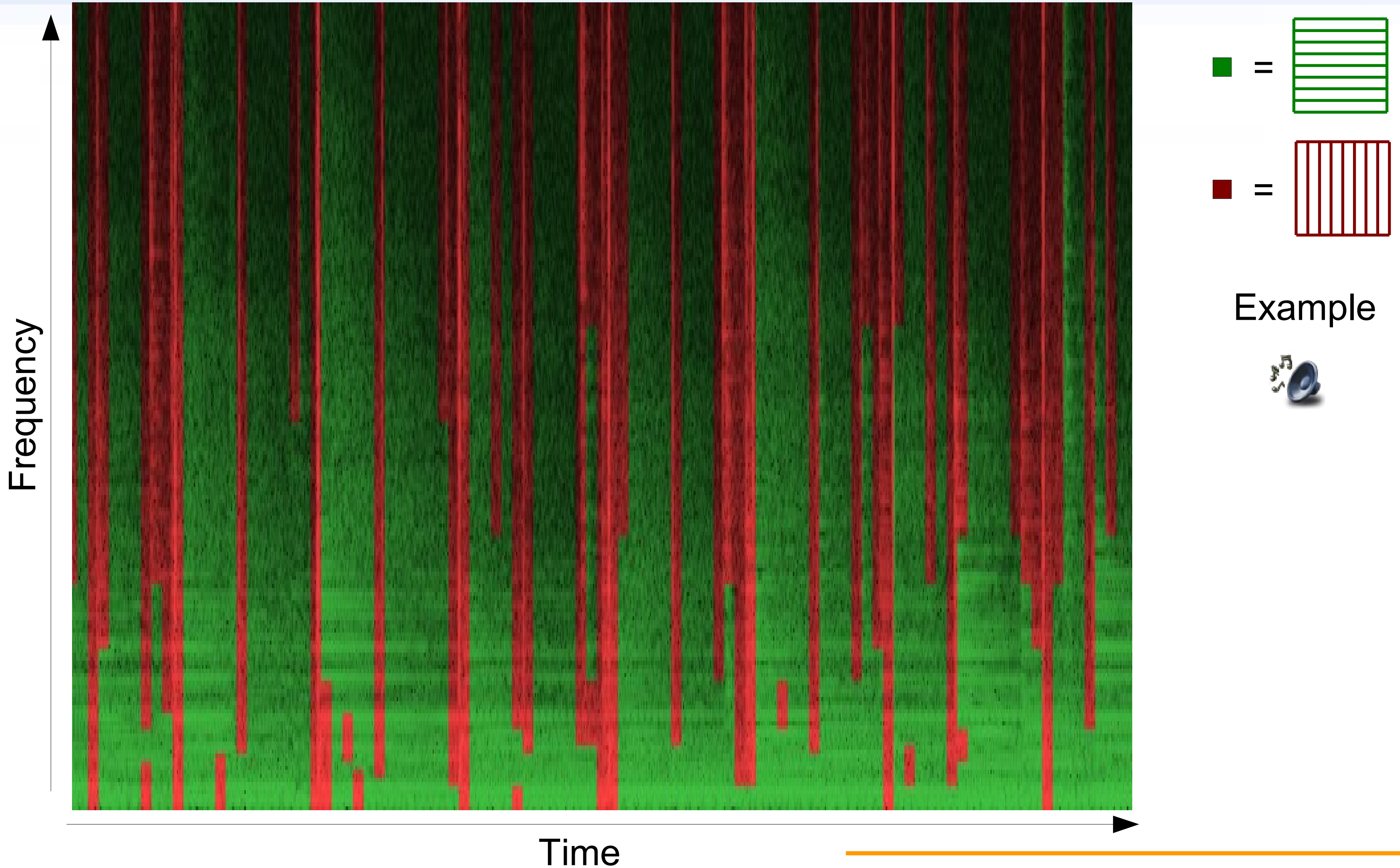
Good frequency resolution

Good time resolution

$\Delta T^{*}\Delta f \geq$ constant

(also known as Heisenberg's uncertainty principle)

Psychoacoustics
T-F Resolution Example

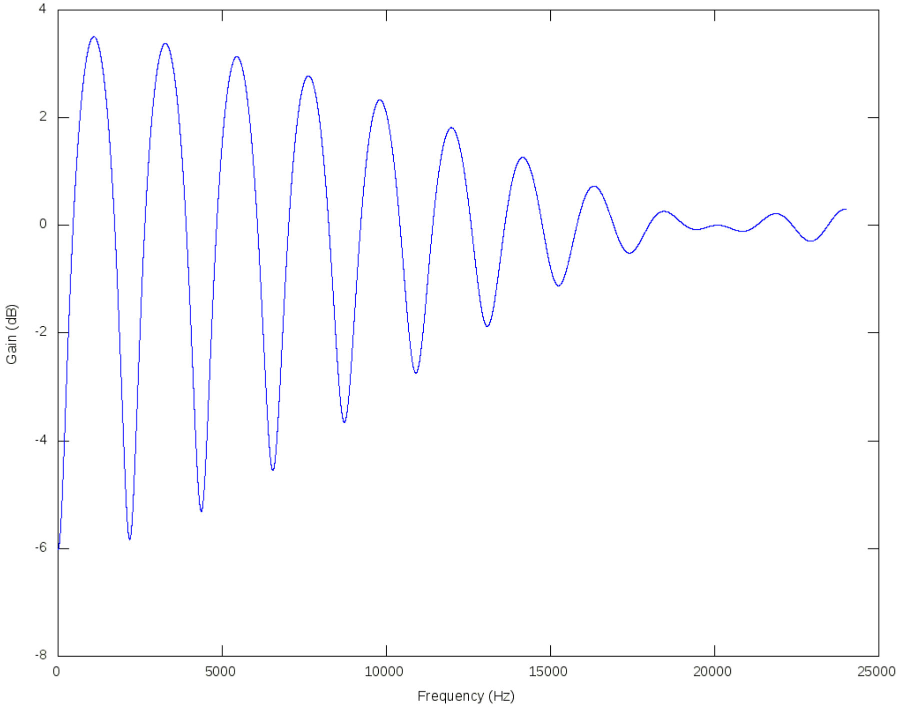


Example

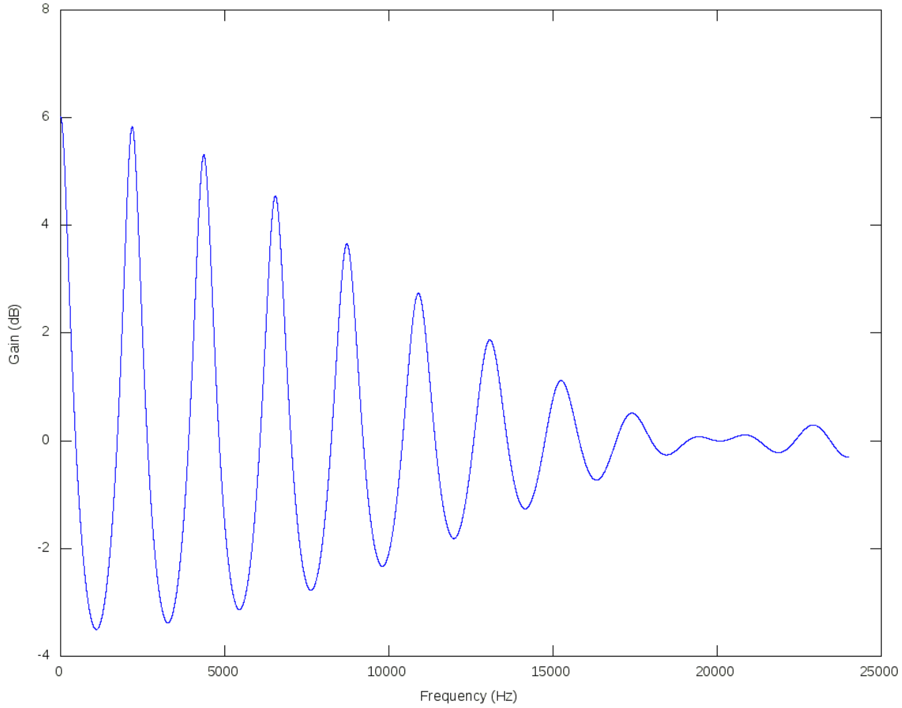
Psychoacoustics
Pitch Prefilter/Postfilter

- Shapes quant. noise (like SILK's LPC filter), but for harmonic signals (like SILK's LTP filter)
  - Contributed by Broadcom

Prefilter



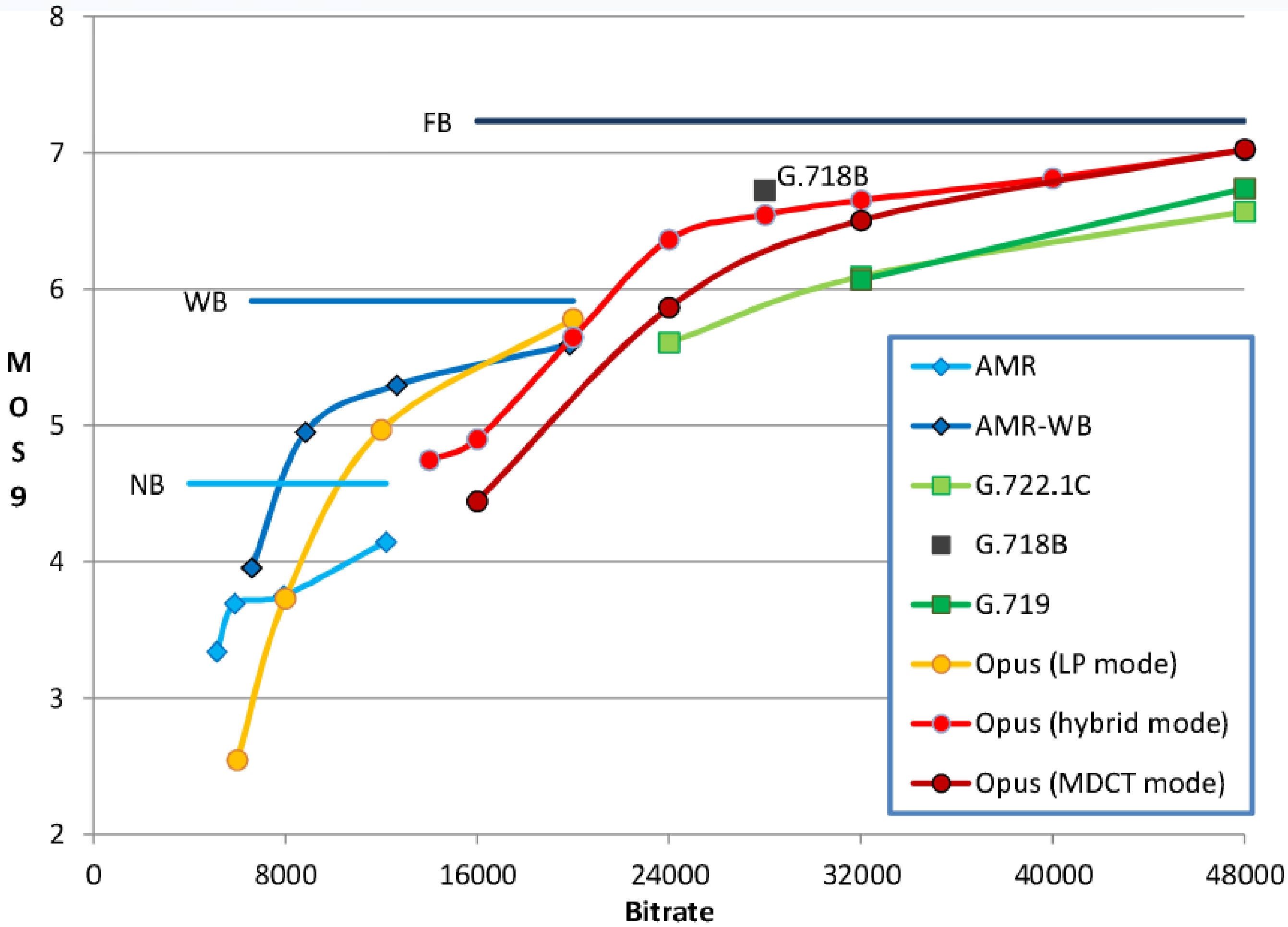
Postfilter



Outline

- Introduction
- Opus Design
  - SILK
  - CELT
- **Conclusion**

Opus Speech Quality



Anssi Rämö, Henri Toukomaa, "Voice Quality Characterization of IETF Opus Codec", *Proc. Interspeech*, 2011.

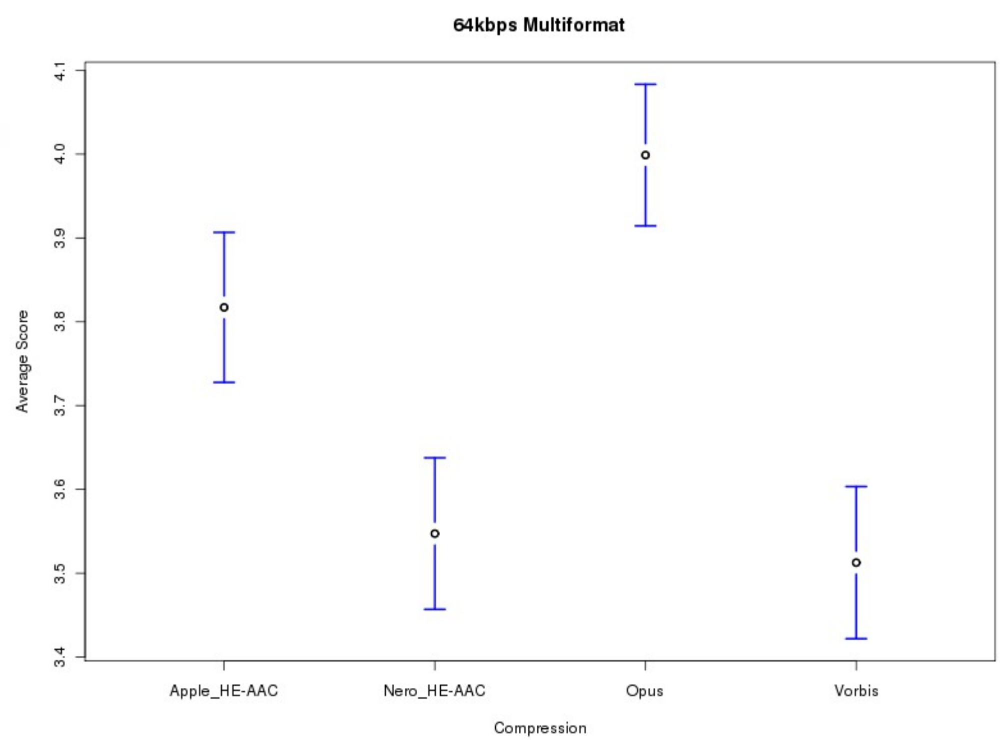
See IETF proceedings for more listening test results:

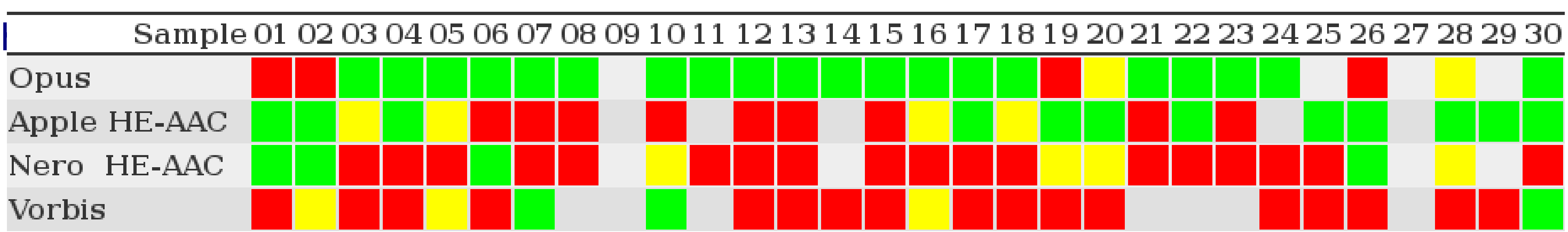
http://www.ietf.org/proceedings/82/slides/codec-1.pdf

http://www.ietf.org/proceedings/80/slides/codec-5.pdf

Opus Music Quality

- 64 kb/s stereo music ABC/HR listening test by Hydrogen Audio





Opus in GStreamer

- Current support in gst-plugins-bad
  - Contributed by oggkoggk (Vincent Penquerc'h, Collabora)
  - Backported to 0.10.23
    - Some PLC/FEC/header parsing fixes in 0.10.24
  - Also available in 0.11.1
- Both Opus in Ogg and RTP supported
  - Properly handles multichannel, seeking w/pre-roll, pre-skip, sample-accurate cutting, output gain, etc.

(Advanced) GStreamer Integration Issues

- Sample rate (playback vs. file output)
  - Opus files all 48 kHz internally
    - No negotation failure for interactive use
    - Impact of resampling smaller than that of lossy compression
  - Should play back directly at this rate if possible (no additional resampling)
  - But users expect decoding to .wav to have the same sample rate as the input they encoded
    - Can GStreamer look downstream in the graph to distinguish these cases?

(Advanced) GStreamer Integration Issues

- Packet loss concealment / jitter buffer issues
  - Variable frame size means you need to decide how much concealed audio to generate
    - Currently uses last frame size, should use timestamps
  - Packet loss rate
    - Informs encoder decisions
      - Inter/Intra energy coding, built-in FEC,etc.
    - Should also use to enable built-in FEC in decoder
  - Late packet recovery
    - Clone decoder before PLC/FEC decode
    - Replay with actual packet once it arrives, so future packets decode correctly

Questions?