FR/HR/EFR voice codecs in Osmocom RAN

Presented by:

Mother Mychaela N. Falconia, operator of Themyscira Wireless GSM network
based on Osmocom CNI components.

High Priestess of Telecommunications,
Women's Republic of Themyscira

Summary of GSM speech codecs

GSM speech codecs in the order of invention:

FRv1: 260 bits per 20 ms frame, 13.0 kbit/s
HRv1: 112 bits per 20 ms frame, 5.6 kbit/s
EFR: 244 bits per 20 ms frame, 12.2 kbit/s

AMR: not part of this presentation

HR was invented in an effort to increase cell capacity, but EFR's sole improvement over FRv1 is voice quality, created for no other reason.

Every GSM MS must support FRv1, all others are optional in any combination.

Why care about non-AMR codecs?

A network that aims to provide service to Vintage Mobile Phones must support FRv1 at the minimum - it is the only codec all phones are required to support.

There are many vintage phones that support EFR but not AMR, in addition to required FRv1. Higher voice quality, their original selling point, is achieved only when the network supports EFR.

If significant effort is to be expended to support FRv1 and EFR to the finest level of quality, AMR support can be deprioritized - EFR is same voice quality as the highest mode of AMR.

Themyscira Wireless currently operates with EFR as the preferred codec and FRv1 as fallback; AMR is disabled in OsmoBSC codec-list config. (And no TCH/H timeslots, only TCH/F.)

DTX with FR/HR/EFR codecs

All 3 codecs support DTX, in both UL and DL, but:

* DTXd is possible only on multicarrier cells (not on C0)
* OTOH, DTXu is always allowed and almost always desired
* Therefore, a high-quality GSM retronetworking operation needs to support the combination of DTXu without DTXd

Enabling any form of DTX means allowing SID frames to occur:

A SID frame is a specially modified codec frame, encodes CN (comfort noise) parameters.

Classic libgsm for FRv1 does not support SID in input!

SID classification complication: 3rd state of invalid SID

Classifying each received frame as either (good) speech or (valid) SID is not enough, there is also the 3rd state of invalid SID.

If a SID frame is corrupted, having it interpreted as speech (because the SID codeword isn't detected) would be bad – hence the need for invalid SID class.

The threshold between invalid SID and speech is a compromise between corrupted SID escaping the check and speech frames getting misclassified as invalid SID.

FR & EFR specs: if at least 80 bits out of the 95-bit SID field are set to indicate SID, but it isn't a valid SID (too many bit errors), then it is invalid SID.

More on valid vs invalid SID

SID classified as valid: CN (comfort noise) params taken from this frame;

SID classified as invalid: tells Rx that CN should be started or continued, but no usable params in this frame;

The threshold between valid and invalid SID is also a compromise: undesirable to reject minimally corrupted frames with still-usable CN params, but also undesirable to have badly corrupted CN params accepted as valid.

FR & EFR specs: one bit error in SID field is still a valid SID, two or more bit errors make it invalid SID.

SID detector problem for HR

The specs prescribe exact bit counting rules for FR & EFR, but not for HR!

In the original architecture the SID detector was closely coupled
with the GSM 05.03 channel decoder;

SID field bit counting needs to be done before the voiced/unvoiced bit
reordering selection in the channel decoder - otherwise invalid SID will
be missed and (badly) misinterpreted as unvoiced speech if the two
SID codeword bits that fall onto mode bits are corrupted.

If we (Osmocom) wish to support HRv1 codec decently and properly, we as
the community will have to come up with our own thresholds for:

1) the boundary between invalid SID and non-SID speech;
2) the boundary between valid and invalid SID.

There is a _non-normative example_ provided in the ETSI C code in GSM 06.06,
but it is weird, refers to a mystery BCI flag that was dropped from the final
published HR codec specs...

Architecture of standard speech decoders

Per the specs, the complete speech decoder block for each of the 3 codecs begins with a subblock called Rx DTX handler - and this block is mandatory whether you run with DTX enabled or not! Handling of bad or entirely missing speech frames is part of the duties of the Rx DTX handler block, and of course there will always be some BFIs and frame gaps in input. And when DTX is enabled, SIDs are naturally handled by the Rx DTX handler.

- For FRv1 the Rx DTX handler is a modular piece, a front end to the basic GSM 06.10 speech decoder. The only FLOSS implementation I know of is the one I wrote, released as Themyscira libgsmfrp.

- For HR and EFR the Rx DTX handler has been an integral part of the speech decoder (reference source published by ETSI) from day 1.

Particularly with FRv1 codec, skipping the required Rx DTX handler and using only a raw GSM 06.10 library (libgsm) in tools like gapk is an ever-present source of misunderstanding and misdiagnosis of speech/voice/audio problems among developers who haven't spent months studying this material like I had to!

Transport within RAN

Classic E1 Abis: GSM 08.60 (FR & EFR) and 08.61 (HR)

Out-of-band flags for UL frame stream:

BFI, SID, TAF (all 3 codecs)
UFI (HR only)

BFI = Bad Frame Indicator
UFI = Unreliable Frame Indicator
TAF = Time Alignment Flag (SACCH alignment marker for Rx DTX handler)

Note 1: frame data bits are still sent when BFI=1!

Note 2: SID ternary classification is done by the BTS and indicated out-of-band, not reconstructed from frame payload by the recipient! Doesn't matter for FR & EFR, but matters a lot for HR!

DL frame stream: good speech expected in every frame without DTXd,
or a mixture of good speech and valid SID with DTXd. No BFIs or invalid SID!

TFO and TrFO

TFO = Tandem-Free Operation
TrFO = Transcoder-Free Operation

In both cases codec frames from leg A UL go to leg B DL without tandem transcoding. But what about BFIs in the stream from UL? And what if leg A UL did DTXu but there is no DTXd on leg B DL? A single-carrier cell means no DTXd!

TS 28.062 section C.3.2.1.1 sets the rules for turning leg A UL into leg B DL; this spec was written for TRAUs with TFO, but I see no reason why the same logic shouldn't carry over to TrFO systems such as a self-contained Osmocom network.

These C3211 rules essentially amount to a parameter-level (operating on codec frame bits) ECU plus CNG (comfort noise generator) implemented in the TFO or TrFO path.

C3211 rules in practice

Spec language: "subject to manufacturer dependent future improvements and is not part of this recommendation."

They tell us what SHALL be done, but not how to actually do it algorithmically!

I would LOVE to get my hands on a real TRAU (remotely over OCTOI would be just fine), experiment and see what they actually do!

If I had to do it: easy for FRv1, a bit harder but hopefully still doable for HR, but very off-putting-ly difficult for EFR.

The whole point of EFR is to give better voice quality than FRv1, it was created to be a "luxury" codec - so let’s not ruin it with a quirk-hack, low-effort implementation of C3211 rules for TrFO!

What I did in OsmoBTS TrFO path

- Weed out invalid SID like the TFO spec says;

- Reposition valid SID frames to where they need to be relative to the SACCH multiframe.

Logical DTXd without physical DTXd: transmit inverted CRC3, so the Rx DTX handler in the MS ends up in the same state as if Alice and Bob had their call on an E1-based network with TRAUs, TFO and DTXd on each DL leg.

Bad frame gaps outside of DTXu pauses: instead of applying ECU in the TrFO path like the TFO spec says, transmit inverted CRC3 and let the Rx DTX handler in the MS do the heavy lifting.

Result: best possible outcome for EFR, better than what a parameter-level ECU+CNG combo would produce if we tried to implement TS 28.062 section C.3.2.1.1 rules to the letter.

RTP encodings for IP-based RAN

RTP payload format for FRv1 was invented by early VoIP people without any consideration given to IP-based GSM RAN implementors.

RTP payload format for EFR was invented by the TIPHON group (not Groupe speciale mobile!) in ETSI by essentially copying what IETF people did for FRv1, thus the same problem of not having been designed for GSM RAN remains.

All out-of-band metadata bits of GSM 08.60 & 08.61 are lost in RTP!

BFI in UL is indicated by sending no RTP packet at all (apparent industry standard practice) or sending zero-length RTP payload (rtp continuous-streaming vty option in OsmoBTS).

Setting all 260 bits of FR or all 244 bits of EFR to 0 is not a valid BFI marker, it is a bogon! Only recently fixed in osmo-bts-trx...

Problem with RTP-based GSM RAN using IETF/TIPHON formats

Compared to GSM 08.60, the industry standard RTP format for FR & EFR (specified in both RFC 3551 and TS 101 318) has two functional regressions:

- No ability to indicate BFI along with data bits;

- TAF bit is lost.

These functional regressions from GSM 08.60 are bad when the RTP stream goes to a network edge transcoder implemented in the spirit of retronetworking, especially if that transcoder also implements in-band TFO in its G.711 output.

TFO frames are slightly modified TRAU-UL frames, so imagine this chain:

E1 Abis -> OsmoMGW -> RTP -> Network edge TC -> TFO inside G.711

RTP transport effectively mutilates TRAU-UL frames...

Solution: TW-TS-001 enhanced RTP format

Themyscira Wireless Technical Specification TW-TS-001

Enhanced RTP transport of FR and EFR codec frames in an IP-based GSM RAN

An extended RTP payload format specifically for GSM RAN, modeled after GSM 08.60

TRAU-like Extension Header (TEH) octet prepended before basic RTP payload

OsmoBTS implementation: falconia/rtp_traulike branch

Two patches on that branch:

* First patch makes OsmoBTS accept TW-TS-001 packets: simply strip off TEH octet like we do with RFC 5993 ToC;

* Second patch adds vty option to emit this TW-TS-001 format.

What about HR codec?

The problems addressed by TW-TS-001 for FR & EFR are not already solved for HR by RFC 5993 format. The following defects still remain:

* No way to indicate BFI along with data bits
* No way to represent "invalid SID" output from SID-aware GSM 05.03 decoder
* TAF bit lost just like in FR & EFR standard RTP formats
* UFI bit (exists for HR only in GSM specs) is likewise dropped

Themyscira Wireless does not use HR codec, but I may produce a TW-TS-002 spec for HR that will define an extension to RFC 5993 format:

* Two more frame types (FT field in ToC octet) for BFI-with-data and Invalid SID;
* Use reserved bits in the ToC octet to carry UFI, TAF and DTXd bits like TW-TS-001 does.

This putative TW-TS-002 spec will also define a storage format, similar to how *.amr file format is based on RTP octet-aligned format for AMR, and this format will be used if and when Themyscira GSM codec libraries and utilities toolkit gets extended to support HR.

More on HR in RTP: TS 101 318 vs RFC 5993

Unless we further extend that ToC octet with our own TS, RFC 5993 as it stands does not provide any new capabilities over more basic TS 101 318 format:

* RFC 5993 can explicitly indicate a missing frame (FT=7), but we already have zero-length RTP payloads that achieve the same effect in a codec-independent way.

* FT=0 vs FT=2 indicates SID status, but:

  - no representation is provided for Invalid SID;

  - my reading of the RFC tells me that even for valid SID (FT=2) the SID field must be all 1s - thus nothing is gained over the fallback method of having the receiver check for the SID codeword.

Neither of the two standard formats is friendly to a BTS implementor who seeks to do what ETSI envisioned, with proper ternary SID classification tightly coupled to the GSM 05.03 channel decoder.

Remaining bogon in osmo-bts-trx: ECU call in the UL path

The call to ECU in the UL path of osmo-bts-trx is unnecessary, counter to the specs and should be considered a bug.

In the traditional GSM architecture a BTS never applies an ECU to its UL output, instead it emits BFIs when it received a bad frame or nothing at all from the air.

If one interprets TFO rules as also applying to TrFO, TS 28.062 section C.3.2.1.1 calls for an ECU in the path from leg A UL (RTP input in OsmoBTS) to leg B DL (internal DL path in OsmoBTS) - but this rule would be super-difficult to implement for EFR, and we already have a better-performing alternative implementation which is the same for all 3 non-AMR codecs.

Therefore, there is no justification at all for an ECU call anywhere in OsmoBTS!

Mistaken belief about uplink ECU in proprietary BTS PHYs

There is a myth that sysmoBTS PHY applies some ECU of its own to its UL output. It does not - at least not for FRv1 or EFR, no easy test setup on my end for HRv1 or AMR.

Actual experimental observations: when GSM MS exercises DTXu and sysmoBTS PHY is receiving radio noise, this PHY sends zero-length payloads (indicating BFI) to the ARM Linux part of sysmoBTS - not any kind of ECU output.

Occasionally the CRC3 check happens to succeed despite the PHY receiving radio noise (1/8 probability), and during these times the PHY sends garbage payloads presented as valid speech - the link quality check in l1sap has to filter them out. This quirk serves as further proof that there is no secret ECU in the DSP.

Early osmo-bts-trx developers came to this mistaken belief because they got noticeably worse speech quality with their early osmo-bts-trx work than what sysmoBTS produces.

Real reason for bad audio on early osmo-bts-trx: the UL code indicated BFI conditions with a bogon in RTP output, and when the receiving end interpreted this bogon instead of invoking the spec-defined bad frame handler, unpleasant sounds were the result. Applying ECU in the BTS UL output path masked this bug.

Internal UL ECU application in OsmoBTS is very inconsistent

The misfeature exists only in osmo-bts-trx version and not in any others;

Even within osmo-bts-trx it is inconsistent: because the ECU call happens before the link quality check in l1sap, sometimes the RTP output will be this ECU output, and othertimes it will be standard BFI produced by the l1sap layer.

Proposed solution to OsmoBTS ECU problem

My first-choice preference is to remove the ECU call from OsmoBTS altogether - see proposed patch on falconia/ecu-ectomy branch. But will such a patch be acceptable _in principle_, aside from code implementation details to be worked out in code review?

If that first-choice option is not acceptable to the community, the alternative is to move the ECU from osmo-bts-trx model code to the common layer, making it available on all models, and make it a vty option.

But do we really need this ECU, and why?

Voice testing tools: gapk shortcomings

It is my understanding that many Osmocom developers use gapk-based processing chains for voice testing, either as RTP source and sink on the network side or with trxcon as part of test MS in virtual Um test setups.

Bad news: gapk is broken, and I am not smart enough to fix it.

The scope of what gapk aims to do seems overly ambitious to me, and I am unsure if it is possible to make a fully correct implementation with such wide ambition.

gapk is fundamentally designed as an anything-to-anything converter, but this design leaves no room for any use-case-specific out-of-band flags: BFI flag for all codecs, or HR-specific UFI flag, or HR-specific out-of-band SID classification, let alone fine details like TAF.

In order for a GSM voice testing tool (developer-oriented) to be properly correct, it needs to be endowed with support for and understanding of these out-of-band flags, and of most practical relevance, these flags need to be passed to the proper Rx DTX handler, which is either built into the main body of the speech decoder (HR and EFR) or implemented as a front-end preprocessor (FRv1). But I don’t know how to implement these ideas within gapk architecture.

Proposal to Osmocom community: a narrower-scope tool maybe?

A super-general, super-versatile tool like gapk is great, but if making it perform correctly in real-life use cases is too difficult of a problem...

What about a more limited-scope, more specialized tool?

Seeking input from the community: what are your real (used everyday, not hypothetical) use cases for voice testing with gapk? Given some specific use cases, perhaps I can produce an alternative tool, based on public domain Themyscira GSM codec libraries, that does whatever you currently do with gapk, but with a correct Rx DTX handler (of which the BFI handler and ECU are an integral part) at the speech decoder input.

Summary of open questions

- Should we (Osmocom community) invest any more effort in HRv1 codec? Does anyone actually need or use it?

- Community thoughts on merging vty-optional support for TW-TS-001 for FR & EFR.

- Removing the bogus UL ECU from osmo-bts-trx, or alternatively moving it to model-independent common layer and making it optional.

- What to do about gapk architectural limitations? The community deserves a proper voice testing tool that heeds out-of-band metadata flags and invokes proper Rx DTX handlers for all 3 codecs covered here.

... and I still want to experiment with a real historical TRAU - how to get a hold of one, or get remote access to one?

If I make an international call in G.711 codec to a country that has a legacy GSM network with TRAUs, can I get a transparent-enough G.711 channel all the way through, so I can talk in-band TFO to that TRAU?