GSM networks that require transcoding to G.711

or alternative title:

Matching the quality of commercial GSM networks in FLOSS

How can we, FLOSS community centered around Osmocom, build new GSM networks that replicate the full functionality _and quality_ of extant commercial ones?

(Hint: extant commercial GSM networks in 1st and 2nd worlds transcode to G.711, and we need to do the same in FLOSS.)

Presented by:

Mother Mychaela N. Falconia,
High Priestess of Telecommunications,
Women's Republic of Themyscira

How extant commercial GSM networks perform (1st & 2nd worlds)

Speech codec support:

Commercial GSM networks in (at least) USA, Mexico and Russia implements this fallback chain for codecs:

1st choice: AMR
2nd choice: EFR (if the MS does not support AMR)
last resort: FRv1 (all type-approved MS must support)

Phone battery saver:

All commercial GSM networks command the MS to turn on DTX in the uplink (DTXu), allowing handsets to conserve battery when the user is not talking.

Take-away points for FLOSS implementations:

* A network that does not support EFR (always uses FRv1, or worse, operates with a policy that rejects non-AMR-capable phones) is defective in comparison.

* A network that operates with DTXu disabled is likewise deficient compared to extant commercial ones.

Speech transcoding requirement

To have a chance at matching extant commercial ones, our GSM network MUST have robust speech transcoding capabilities!

The lingua franca of any-to-any public phone network interconnection is G.711, *not* AMR or GSM speech codecs!

Only two practical choices:

Option 1: do all transcoding yourself, and talk G.711 to outside world;

Option 2: rely on your VoIP-to-PSTN connectivity provider to do the transcoding for you - i.e., outsource the transcoding task.

Some VoIP providers support "GSM" codec (actually meaning bare 06.10 subset of FRv1, *without* GSM-FR DTX functions) in addition to G.711 and G.729 - but none support GSM-EFR.

If the network is to operate primarily with EFR, falling back to FRv1 only for the very oldest of phones, it makes more sense to transcode both FRv1 and EFR to G.711 under your own control, than outsource FRv1 transcoding to your VoIP provider.

Additional observations in commercial GSM networks

In-band homing feature of EFR: mandatory in the specs, correctly implemented in practice by both T-Mobile USA and Telcel Mexico.

In-band homing feature of FRv1: optional per the specs, but the legacy GSM network of T-Mobile USA does support it. (Not tested in Mexico yet.)

Homing in GSM uplink direction: you send special decoder homing frames (DHFs) from your MS, the network transcoder resets to spec-defined home state and outputs constant 0xFE (mu-law) or 0xD5 (A-law) PCM samples.

Homing in GSM downlink direction: you send a long stream of 0xFE or 0xD5 from IP-PSTN side, the network transcoder resets and emits a stream of DHFs on GSM radio downlink.

A FLOSS GSM network that fails to replicate _this_ feature would also be deficient!

Pre-existing state of FLOSS (before Themyscira)

Using classic libgsm to implement FRv1 codec:

* No support for SID frames – thus you have to disable DTXu in OsmoBSC config and shorten handset battery life.

* API exported by the library does not permit state reset – hence no ability to implement the in-band homing feature even externally.

Using OpenCORE AMR library to implement EFR:

* EFR matches 12k2 mode of AMR only in the absence of DTX; AMR libraries can’t grok EFR SID frames – back to the problem of having to disable DTXu.

* Same problem as in libgsm with lack of state reset functions.

Additional problems if one uses osmo-sip-connector

If the GSM network is Osmocom CNI while outside-world interface software (IP-PSTN etc) is some other FLOSS, where is the interface point between the two?

So far the official answer in Osmocom is the internal SIP leg (separate and distinct from external SIP interfaces to PSTN etc) created with osmo-sip-connector. But this interface is a straightjacket:

* The paradigm of SIP is very different from Q.931 and GSM CC. If you have to convert to SIP anyway to connect to outside world, then not much loss in using similar SIP internally, but forcing internal calls (within your own network) to go GSM-SIP-GSM is outright evil.

* Try passing CSD calls through SIP bottleneck - a square peg in a round hole.

DTMF problem:

* In GSM there is Start DTMF message when you press the button, and Stop DTMF when you lift your finger.

* SIP design requires osmo-sip-connector to send a fixed duration upfront in the INFO method - thus actual DTMF duration is lost.

Clash of paradigms: SDP or 3GPP worldview

Paradigm currently adopted in Osmocom
(presented at OsmoDevCon 2024 by Neels):

* Osmocom network interconnects to outside world in SIP+SDP
* AMR and other codecs exposed externally (no transcoding)
* SDP is the native/preferred language, AoIP interface seen as problem-maker

Alternative paradigm advocated by yours truly:

* Outside world interconnection happens only in G.711
* All GSM codecs, AMR or otherwise, fully hidden from the outside world
* The bridge between inside and outside worlds is the transcoder to G.711

* RTP payload types, AMR mode sets, SDP - all that mess that received so much work in recent years - becomes completely internal & private to the GSM (or GSM+UMTS) network - we can do whatever we like (whatever is easiest) without having to coordinate with anyone!

Solution adopted by Themyscira Wireless

What works currently:

* Using OsmoMSC from 2023-02 release of OsmoCNI (prior to SDP-fication patches);
* MNCC is the interface point between Osmocom and ThemWi.

New version I am working on:

* AoIP interface is the new boundary point between Osmocom and ThemWi;
* ThemWi-MSC will be a divergent network element based on OsmoMSC.

Only OsmoMSC is being forked in this approach, not the rest of OsmoCNI!

* All Osmocom BSS components are still mainline;

* OsmoHLR is still mainline, including the path to SMSC via GSUP;

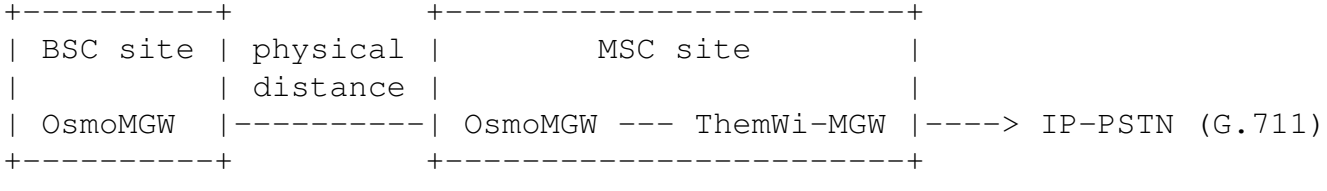
* ThemWi-MSC will still use mainline versions of all Osmocom libraries
  that are used by OsmoMSC to talk to other network elements.

But of course even a partial fork of only one Osmocom component is still bad...

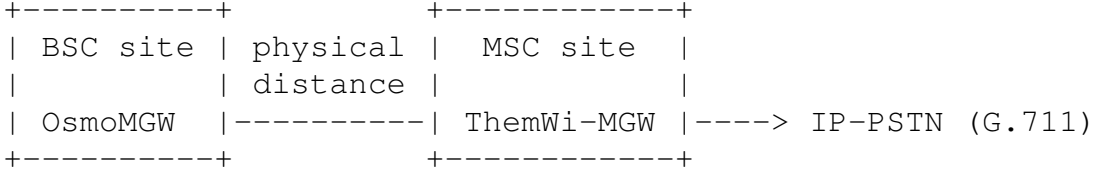
Changes desired in ThemWi-MSC

Why am I looking at the highly unpleasant idea (both technically and socially) of forking OsmoMSC? What changes do I seek in the MSC component that stand no chance of being mergeable into mainline OsmoMSC?

The big one: get rid of extra OsmoMGW hop. Instead of this:



I want this:



Hypothetical alternative: bring transcoding into Osmocom

Would it be possible to bring transcoding into OsmoMGW? Keep the mid-2023 Neels-pioneered paradigm of OsmoMSC talking SDP in MNCC, but extend that model with the possibility of MNCC asking for G.711, with OsmoMSC then instructing its associated OsmoMGW to transcode - would such approach be possible?

The first major difficulty I see: what about codec libraries?

Themyscira Wireless maintains and publishes a suite of GSM codec libraries covering the full set, but:

* The EFR library (the most important one in practice) is derived from ETSI source.

* It is my understanding that those who have chosen the path of obedience to copyrights will see this work as non-free, thus making unacceptable the idea of any Osmocom component being built on top of these libraries.

OpenCORE AMR library is _also_ based on original 3GPP source, indirectly through PacketVideo - but PV/Google obtained special permission from 3GPP Org Partners to redistribute the result under Apache license. I have no idea how to replicate that feat with the older codecs of interest to me.

Osmocom+ThemWi combination for users

IANAL, but even more fundamentally: not only am I not a lawyer, but more profoundly, I have absolutely no desire to ever become or act in any way like one. Caring whether something is legal or illegal is "acting like a lawyer", and part of things I refuse to do.

However, from the perspective of an end user / retronetworking operator: if you merely download gsm-codec-lib-latest.tar.bz2 from my site, then download and compile ThemWi applications built with these libraries, and then use them together with Osmocom CNI components, _you_ are not doing any redistribution of ETSI-copyrighted code in violation of that copyright, only _I_ am doing that by producing and publishing my derivative works - so _you_ are not breaking any laws by being a _user_ of my work, are you?

Hence for those who care about practicalities more than legalities, the combination of Osmocom CNI plus Themyscira Wireless components is a practically-free, published-source implementation of a GSM network that can match the quality of T-Mobile, Telcel and other big commercial ones - a feat which no one else in any FOSS or FOSS-like project has done before!

Recent development: OS#6448 merged into Osmocom mainline

With OS#6448 patches merged (thanks Harald!), Osmocom BSS can now emit enhanced RTP format of TW-TS-001 (FR & EFR), and maybe later TW-TS-002 (HR) as well, if we find time/resources for OS#6036 and/or OS#6496.

Shortcoming: in the present time, this new capability of Osmocom BSS can be exercised only by running a non-mainline patched version of OsmoMSC, to send the necessary BSSMAP extension IE (TW-TS-003) in the Assignment Request message.

Planned use of this capability in Themyscira: with the BSS emitting TW-TS-001 extended RTP format, ThemWi-MGW will be able to embed proper GSM 08.62 TFO frames in its outgoing G.711 PCM sample stream - subject of a future presentation. ThemWi codec libraries for FR and EFR also perform better when the input to the decoder has all of the metadata flags of GSM 08.60 TRAU-UL frames or TW-TS-001.

Thoughts for Osmocom community: if anyone is interested in using the new TW-TS-00[1-3] capability of Osmocom BSS in more of a "pure Osmocom" environment without ThemWi software, let's discuss how we can make it happen. In particular, how do you envision it being represented in MNCC-SDP of current mainline OsmoMSC?

Discussion time:

Have others in Osmocom previously looked into the problem of speech transcoding, supporting all GSM codecs? If so, what was your take-away?

From my PoV, I am perfectly happy to continue maintaining and publishing GSM speech transcoding software (GSM codec libraries package and ThemWi-MGW built on top of it) entirely on my own (unlike so many other people, I have no fears in this regard), and leave it to end users / retronetworking operators to combine ThemWi components with OsmoCNI in their deployments.

But if someone would like to suggest other ways, I am all ears - let's discuss pros and cons of each method!

Finally, question from previous slide: is there any interest in Osmocom in extending TW-TS-001 & TW-TS-002 support such that it could work with mainline OsmoMSC via MNCC-SDP?

EOF