TFO in GSM and 3GPP networks

TFO = Tandem-Free Operation

R98 and R99: GSM 08.62 -- GSM only, FR/HR/EFR only
3GPP Rel4 and later: TS 28.062 -- adds AMR in GSM and AMR-only 3G/UMTS

TFO is an _in-band_ protocol inside a 64 kbit/s G.711 PCM sample stream that allows two different GSM (or 3G) networks to have a mobile-to-mobile call without double transcoding - what a masterful feat!

Consider this situation:

* I operate a GSM network in San Diego;

* Someone else operates a different GSM network in Free Keene;

* Each network connects to PSTN and transcodes to G.711 PCMU or PCMA for all outside connections;

* Without TFO, any call between networks would be forever limited to double transcoding - but TFO protocol allows us to achieve Tandem-Free Operation.

TFO vs TrFO: let's keep them straight

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

TFO is an in-band protocol inside a 64 kbit/s G.711 PCM sample stream - hence if you are not transcoding from GSM/3GPP codecs to G.711, then you cannot meaningfully talk about TFO.

TrFO is a different architectural approach to network deployment in which compressed speech from call leg A passes directly to call leg B without ever hitting a transcoder. Mainline Osmocom CNI does TrFO, not TFO!

TrFO cannot exist in classic TDM-based GSM architecture where each call leg passes through a TRAU before it hits the nearest MSC switch fabric.

3GPP specs for TrFO are part of OoBTC and BICC world: they work everything out in the context of 3G, but coyly avoid any mention of how those ideas would work with a GSM RAN. It's the worst for AMR: given an E1 BTS that implements AMR as specified in TS 48.060 & 48.061, I don't see any way how it could be interfaced to TrFO instead of TFO. 3GPP really threw GSM under the bus here!

Implementations of TFO are very scarce!

Reading 3GPP specs is good fun, but where are practical implementations?

Sadly, it appears that extant commercial GSM networks no longer support TFO. I can make 64 kbit/s calls between a SIP-to-PSTN access provider and legacy GSM networks of T-Mobile USA and Telcel Mexico, but no TFO_REQ messages appear; sending such in-band messages elicits no response.

Implementations of TFO for FR/HR/EFR are known to exist in TRAUs - got one here in our lab! Did anyone ever implement such non-AMR TFO in any "modern" IP-based transcoding MGW equipment? Maaaayyyybe, but seems unlikely.

Implementations of TFO for AMR: scattered references suggest that some people did implement it. But where? No idea...

Nokia TCSM2 supports AMR, but it _seems_ to not support TFO for AMR. I say "seems" because it is very difficult to test without an E1 BTS that also supports AMR with TFO: in the case of AMR, TFO logic extends across Abis into the BTS! Plus the docs for Nokia's later TCSM3i explicitly say "TFO for AMR is not supported." :-(

TFO development timeline

TFO for FR/HR/EFR introduced in R98: all 3 codecs were already well-established then, hence the spec treats them equally.

AMR was also introduced into GSM in R98 – but not supported for TFO!

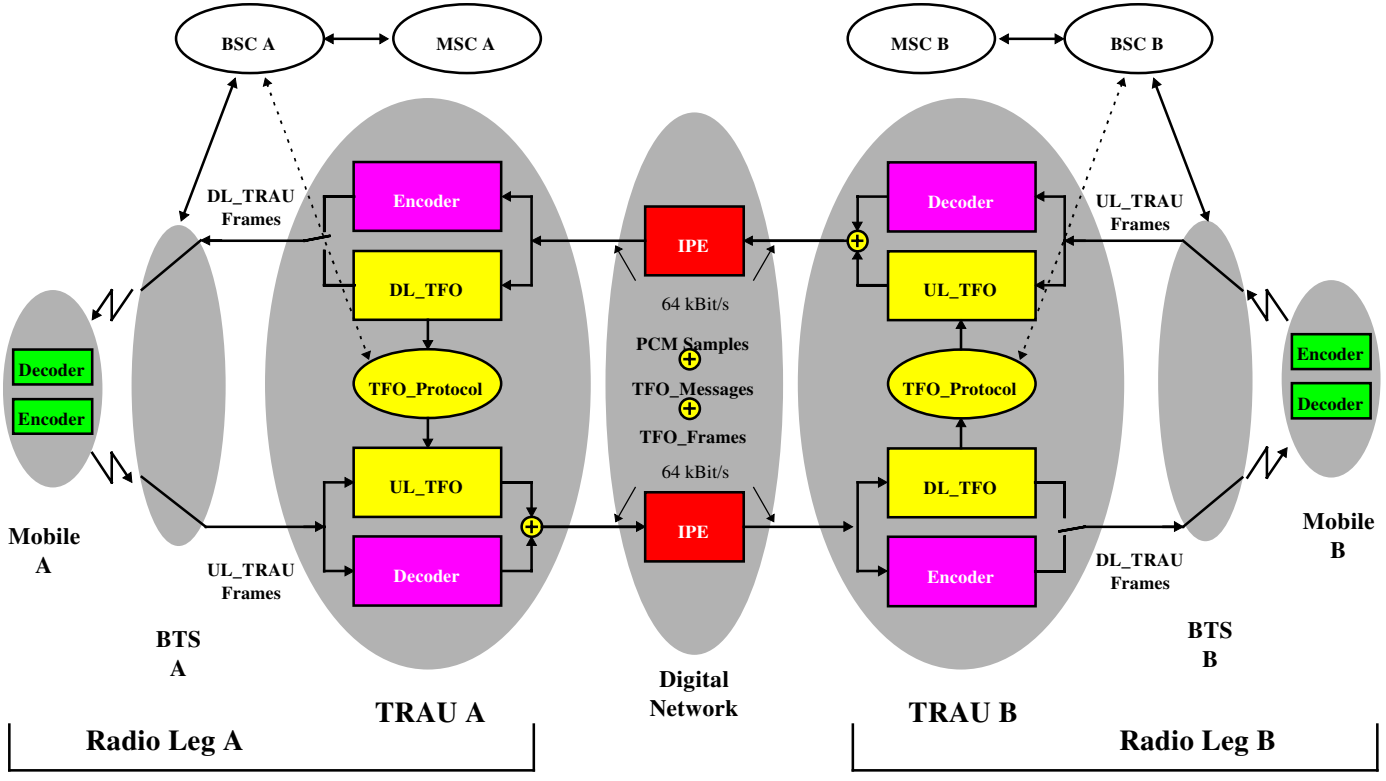
R99 introduces 3G/UMTS, speech is AMR only – but still no possibility of TFO. :(

Rel4 formally introduces TFO for AMR, for both GSM (with TRAUs) and 3G. The spec is renamed from GSM 08.62 to TS 28.062, majorly restructured, sections moved around. But it is _insanely_ complex!

TrFO for 3G was also introduced in Rel4 – but as I already said, 3GPP definition of TrFO is of no help to GSM, neither with traditional codecs nor with AMR.

In this presentation, I adopt forward chronological approach: let’s cover classic non-AMR TFO first, then revisit AMR.

TFO diagram from classic GSM 08.62



TFO for FR/HR/EFR: key principles

Prior to TFO establishment, each TRAU sends TFO messages, hoping to receive a response - then gives up after about 5 s. These in-band messages steal the lsb of every 16th PCM sample - 0.5 kbit/s rate.

If TFO_REQ messages sent out by a TRAU elicit a response, a negotiation protocol ensues - it allows TFO to proceed only if both ends are GSM and use the same codec. Digital transparency of the 64 kbit/s path is also checked.

When TFO is established, TFO frames are sent instead:

* FR & EFR: stealing two lsbs of _every_ PCM sample to make a 16 kbit/s channel;
* HR: stealing one lsb of every PCM sample to make an 8 kbit/s channel.

The remaining bits of every 64 kbit/s octet are still the upper bits of PCM samples that normally go there: hence the speech decoder in each TRAU has to keep running.

What are these TFO frames that replace the 1 or 2 lsbs of each PCM sample in the active state? Answer: they are TRAU-UL frames with minimal modification!

TFO frames and their significance for GSM RAN architecture

In most people's conventional understanding of GSM, TRAU frames are an entirely private matter inside one network implementation: they pass between a legacy E1 BTS and either a vendor-matching TRAU or OsmoMGW taking its place, and disappear from view afterward. With native IP BTS no TRAU frames exist at all.

But if a network implements TFO, then its internal TRAU-UL frames (UL output from BTS) are exposed to the world of public inter-operator interconnection! TFO is designed to work between different operators, even in different countries on different continents - thus once you decide to speak TFO, you are really speaking a standardized protocol, and you have no control over what kind of implementation runs on the other end - just like with public Internet protocols.

Enabling TFO implies a willingness to expose some of your innards! The party on the other end of the call, and anyone sniffing the call, can see where every SACCH-aligned 24th frame lies (TAF bit), they can see when Alice's UL radio leg experiences errors (BFI in TRAU-UL frames), they can see when Alice's phone exercised DTXu...

This quality also makes TFO a lot more difficult to implement if your internal RAN architecture is not based on TRAU frames, if it uses RTP instead! That's why I had to develop TW-TS-001 enhanced RTP transport: TRAU-like RTP. :)

TFO transform from UL to DL: general principle

Speech flow from Alice to Bob in TFO:

* TRAU-A takes its TRAU-UL input from BTS-A and re-emits that input directly into its 64 kbit/s output stream, embedded into the lsbs of PCM output.

* TRAU-B (could be across an ocean!) extracts TFO frames (very slightly modified TRAU-UL frames) from the 64 kbit/s stream and prepares TRAU-DL frame stream for call leg B.

* Just one big problem: the UL leg from BTS-A can have BFI frame gaps and DTXu pauses; the DL leg to BTS-B must be pristine, with nothing but a perfectly good speech frame in every 20 ms position without fail!

The UL-to-DL transform (or TFO transform as I call it now) has to perform bad frame handling and comfort noise insertion on codec parameter level.

The spec says "shall" followed by "subject to manufacturer dependent future improvements and is not part of this recommendation." Just lovely!

TFO transform from UL to DL: difficulty of the problem

FR codec: the non-bit-exact Rx DTX handler (ECU + comfort noise generator) that must be inserted between radio Rx output and the input to the basic GSM 06.10 decoder, fairly easy to implement based on GSM 06.11, 06.12 and 06.31 specs, also works equally well as a TFO transform - but did historical TRAU implementations do anything more fancy? CN interpolation perhaps?

HR codec: the official endpoint decoder is now bit-exact _including_ comfort noise, unlike FR! The Rx DTX handler in the reference source is _partially_ modularized... But questions abound: does the TFO transform have to regenerate _interpolated_ CN parameters, or can one get away without such? Is it acceptable for TFO transform to drop UFI conditions that don't turn into BFI?

EFR codec: the official endpoint decoder is not only bit-exact, but fully monolithic, no longer containing a separate Rx DTX front end. Bad frame handling and CN generation are now implemented in the _guts_ of the decoder!

All of these mysteries have been nagging at me ever since I read the spec and realized the need for this transform in both TFO and TrFO - and sent me on a quest to acquire my own TFO-capable TRAU in order to find some answers.

TFO implementation in Nokia TCSM2

Nokia TCSM2 supports TFO as a "premium" feature, meaning that not every customer got it. Not sure if they made different TRCO main fw builds or if it was some kind of license key scheme - but thank heavens that our TRCO from shields-e came with this feature included!

If TFO feature is included, the operator can enable or disable it as desired via the management interface. Because I run my TCSM2 without any BSC, the management interface I use is the RS-232 port on TRCO.

When a TRAU channel has TFO administratively enabled, it behaves like I expected from reading the spec:

* TFO message output begins when the channel is active from both sides: TRAU-UL frames coming from BSS, something other than 0x54 idle coming from MSC.

* The TRAU emits 3x TFO_FILL and then repeats TFO_REQ 35 times, lasting about 5 s in total.

* If I bridge two active TRAU channels (two E1 timeslots) on the A side of the TCSM, I get successful TFO establishment between them.

TFO protocol implementation

TFO protocol is very complex! A from-scratch implementation, working solely from the spec, is certainly possible - but without interoperability testing against someone else's implementation, there would be very little assurance of correctness - hence the search for an existing historical implementation that led me to this Nokia TRAU.

Themyscira FLOSS implementation of TFO is a project for later; all current experiments with TFO on Nokia TCSM2 were done by bringing up two TRAU channels on our TCSM2 and cross-bridging them together on the A interface. (A simple test program opens two E1 timeslots on A and cross-routes byte traffic between them.)

Analyzing Nokia's implementation of TFO by looking at captured A interface byte traffic between cross-connected TRAU channels:

- Mostly matches my reading of GSM 08.62 spec.

- Only one discrepancy found so far: when a TCSM2 TRAU channel emits TFO_REQ_L or TFO_ACK_L, List_Ind bit in SIG_LUC word is set to 0, even though the long codec list is included. Per the spec this bit should be 1 - bug in Nokia's implementation?

TFO transform from UL to DL: Nokia TCSM2 implementation

Key observation from TCSM2 experiments: this TRAU really does implement an "honest-to-god" TFO transform for all 3 codecs, without "cheating".

A "cheating" scheme would have been to emit special idle frames or some other hack on TRAU-DL during BFI frame gaps on UL, and/or have the BTS implement a fake "logical DTXd" in the absence of physical DTXd, transmitting induced BFI frames on radio DL in both cases. But nope, no such cheating tricks in Nokia's BSS implementation: the TRAU does exactly what the TFO spec calls for, a real TFO transform.

Detailed notes on how this TRAU implements the TFO transform for all 3 codecs:

https://www.freecalypso.org/hg/gsm-net-reveng/file/tip/doc/TFO-xform/FRv1
https://www.freecalypso.org/hg/gsm-net-reveng/file/tip/doc/TFO-xform/HRv1
https://www.freecalypso.org/hg/gsm-net-reveng/file/tip/doc/TFO-xform/EFR

Plus some theoretical introduction:

https://www.freecalypso.org/hg/gsm-net-reveng/file/tip/doc/TFO-xform/Theory

Doing TrFO for FR/HR/EFR: significance for Osmocom CNI

Osmocom CNI does TrFO instead of TFO. But guess what: with non-AMR codecs, the same UL to DL transform which I now call TFO transform is also needed for TrFO! And here I mean the informal, Osmocom-implemented kind of TrFO, rather than 3GPP kind that only covers 3G and not GSM.

With native OsmoBTS we can "cheat" and avoid this transform: transmit inverted CRC (inducing BFI in the MS receiver) during gaps in the incoming TrFO stream, and apply "logical" DTXd (same induced BFIs between SID updates) in the absence of physical DTXd. But the same trick does not work with E1 BTS: those expect a valid TRAU-DL frame every 20 ms, rain or shine!

OsmoMGW-E1 takes the place of a TRAU, sending TRAU-DL frames to the BTS. Right now it emits a dummy frame whenever the incoming RTP stream gaps or carries a BFI - but this dummy frame is wrong! There is no other constant filler that would be any better - instead the correct solution is to invoke what I now call the TFO transform. IOW, the correct transform from incoming RTP to TRAU-DL output from MGW is exactly the same as the transform from incoming TFO frames to the same TRAU-DL output! RTP packet loss should be treated like BFIs.

TFO for AMR

We just covered "classic" TFO for FR/HR/EFR - how does AMR differ?

With classic GSM codecs and _if_ the BSS is implemented with TRAUs, then TFO is fully confined to the TRAU: BTS and BSC remain oblivious to it.

Not so for AMR! TS 28.062 presents _several different ways_ how the logic of TFO can be split between the TRAU, the BTS and the BSC - yikes! Does it mean that each BSS vendor looking to implement this feature had to choose on their own which approach to take, further killing the idea of interoperable Abis?

TFO on the 64 kbit/s side is still intended to be a globally interoperable interface, though! With AMR, it is now supposedly interoperable between AMR-FR and AMR-HR in GSM, and between GSM-AMR and UMTS - very admirable ambition indeed - but the complexity is insane!

TFO for AMR: a brief tour of the complexity

TFO-AMR partners exchange (via in-band messages) information about active codec sets, supported codec sets and their capabilities in terms of changing their configuration.

The spec is written with an expectation that different cells in a network may use different AMR mode sets because of different terrain, resulting in AMR mode set changing on handovers. With TFO, such changes on one side affect the remote side too!

AMR-FR implementations using 16 kbit/s TRAU frames to support modes up to 12k2 must also support dynamically switching to 8 kbit/s frames if the far end is AMR-HR - and the spec leaves it as an option whether the BTS switches its frame format or if the TRAU has to convert on the fly...

AMR transport in RTP is a little bit like IuUP - or at least closer to IuUP than to TRAU frames - and the spec for TFO-AMR in 3G talks about converting frame formats... And there is the nastiness of having to recover CMR/CMI phase...

TFO-AMR in classic E1 gear: some thoughts

I do NOT plan (at least not currently) to search for another TRAU that supports TFO with AMR. In my current understanding, doing so would be pointless: there won't be a platform to experiment with and learn from unless not only the TRAU supports TFO-AMR, but also the matching same-vendor BTS, and the required config messages from the BSC would have to be known.

However, if someone has an E1 BTS with AMR capability and they would like to experiment with it, that path would be more promising:

* Extend my e1-fake-trau program to emit TRAU-DL frames for AMR, to keep the BTS happy (keep it from declaring Remote Transcoder Failure) despite the lack of AMR support in OsmoMGW-E1.

* Put a TCH timeslot into AMR mode with CHANnel ACTIVate, and include a MultiRate Control IE with TFO enabled.

* See what TRAU-UL frames that BTS puts out, let me examine them, go from there!

If someone in the community determines that a given E1 BTS model supports not only AMR, but also TFO-AMR, then maybe we can implement the TRAU part in sw.

AMR with E1 BTS: can we just force TrFO onto the BTS?

As already covered, Osmocom CNI way of doing GSM is not TFO, but an informal kind of TrFO. If someone tried to extend OsmoMGW-E1 to support AMR, it would likewise still be TrFO, not the same as TFO.

In this light, could we ignore the lack of TFO-AMR support in E1 BTS? Can we just take the bits from incoming RTP (RFC 4867 for AMR) and shove them directly into TRAU-DL frames toward the BTS?

Without experimentation, we don't know if the BTS would accept such forced TrFO in the absence of formally negotiated TFO!

Remember the UL-to-DL transform for FR/HR/EFR? It doesn't exist for AMR - instead when the BTS and the TRAU have negotiated TFO, the raw frame stream from Alice (UL Rx) is passed to the DL of BTS-B, and the BTS has to do the logic of transmitting BFIs. The inverted CRC trick is formally blessed for AMR: TS 26.093 section A.5.1.2.3. :-)

But if a given E1 BTS does not support TFO-AMR, what would it do when faced with TRAU-DL frame types which the spec allows only in TFO and not otherwise? BFI frame gaps are in this category, and so is DTXu without DTXd.

TFO-AMR: more general wondering

Aside from the question of what E1 BTS models, if any, implement TFO-AMR in cooperation with their matching same-vendor TRAU, the more general question is: where else was TFO-AMR implemented, in what kind(s) of equipment that is not necessarily TDM-based GSM, and where in the world was it deployed in terms of commercial networks?

Are there still any networks (2G or 3G or ???) left anywhere in the world with which one could establish a TFO-AMR call over global PSTN?

Given the effort that would be required to implement TFO-AMR (way messier than the already-complex FR/HR/EFR TFO), and given the difficulty of testing such implementation for correctness, it would be very worthwhile to know if there is anyone out there to interoperate with...

With FR/HR/EFR TFO we also don't really know if there are any live networks remaining with which we could interoperate - but at least we can develop a known-correct FLOSS implementation by testing against Nokia TCSM2 in our lab, and if we never find an extant commercial network to do TFO with, we'll have FR/HR/EFR TFO between different community GSM networks. But with TFO-AMR we have the extra difficulty of not having anything we could test against, while the complexity is through the roof. :-(

Open questions: Ericsson RBS6k and AMR

E1 BTS of most practical interest: RBS6k DUG20 combined with RUS or RRUS

I have both DUG20 and RUS01, made around 2014, decommissioned from T-Mobile USA, and I hope to get around to powering up and playing with this equipment some time this year (2025).

Does it support AMR? I can only assume it does, since T-Mobile and others operated networks with this gear well past the point of AMR world domination - but it needs to be tested. Note the lack of existing non-broken support in Osmocom for anything other than FRv1 on E1 BTS!

Does it support TFO-AMR? I saw some mentions in Ericsson docs suggesting that it doesn't...

Whether or not RBS6k DUG20 supports TFO-AMR, how did various big networks implement what is essentially TrFO? If they implement 3GPP AoIP at their A interface while Abis remains based around TRAU frames of TS 48.060 (Ericsson TFP seems to carry these TRAU frames over IP without change in semantics), how would they convert from RFC 4867 DL input at AoIP interface to TRAU-DL going to the BTS? Perhaps by foregoing strict obedience to TS 48.060 and applying that spec more loosely?