OsmoBTS - Feature #1837

autogenerate dtx_dl_amr_fsm.*

11/08/2016 11:49 AM - msuraev

Status:	New	**Start date:**	11/08/2016
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:			
Target version:			
Spec Reference:			

Description

We have FSM defined as c code in src/common/dtx_dl_arm_fsm.* and also as DOT description in osmo-gsm-manuals/OsmoBTS/dtx.dot Maintaining both in sync is tricky and error-prone. The code is very regular so it could be rather easy generated (for example from python script using pydot) from the same dtx.dot file we use for documentation. This way it would be automatically in-sync and we will have single source of truth when it comes to DTX FSM implementation. This would also facilitate future use of static analysis tools as .dot file is easier to manipulate than c sources. We already use similar process to generate convolutional codes in libosmocore.

History

#1 - 11/09/2016 09:44 AM - laforge

- Assignee set to msuraev

#2 - 11/09/2016 09:45 AM - laforge

- Priority changed from Normal to Low

#3 - 12/01/2016 05:16 PM - msuraev

Note: the same script could be used for other FSMs.

#4 - 12/08/2016 07:08 PM - neels

- Subject changed from autogenerate dtx_dl_arm_fsm. to autogenerate dtx_dl_amr_fsm.**

Ok, when I look at dtx_dl_amr_fsm.c, the actions for each state are trivial
in that each incoming event simply transitions to the corresponding state.

However, when you look at the VLR fsm code, auto-generation is not as simple:

- things have to be verified
- some events have no clear 1:1 mapping to a state
- arbitrary actions need to be taken, like start other fsms, send messages etc.

Generate this:

https://git.osmocom.org/openbsc/tree/openbsc/src/libvlr/vlr_lu_fsm.c?h=neels/vlr
https://git.osmocom.org/openbsc/tree/openbsc/src/libvlr/vlr_auth_fsm.c?h=neels/vlr

So I doubt that fsm generation from dot will impact beyond the dtx_dl_amr_fsm.
There though it might be quite useful, given that you expect it to change a lot.

#5 - 12/09/2016 09:20 AM - msuraev

It's just matter of splitting state transition logic (completely described by .dot file) from action logic (application-specific) which I think is a good practice anyway. The former should be autogenerated and include calls to functions like ...fsm_state_XXX_evt_YYY() which will contain action logic. Btw, events should not have any mapping to states - those are orthogonal concepts, do not mix them. States are nodes of the graph representing FSM, events are the external signals which cause transitions between those nodes.

#6 - 12/09/2016 02:00 PM - neels

> Btw, events should not have any mapping to states - those are orthogonal concepts, do not mix them.

I wouldn't call it orthogonal: each state allows a limited set of events to transition to a limited set of other states.
I assume you refer to "some events have no clear 1:1 mapping to a state"; by that I meant:
in the AMR FSM, when state X receives event foo, it always transitions to exactly state Y. But,
for example, in the http://kleinekatze.de/quooXai5/vlr_auth_fsm.dot.png FSM, the START event can
cause a transition to either the WAIT_RESP or the NEEDS_AUTH_WAIT_AI state, which depends on the
decision made in "arbitrary" C code that is also part of the FSM logic.

But let's not get carried away in theoretical discussions.

I would agree that the osmo_fsm code could be made less generalistic, to benefit
automatic generation of graphs or vice versa. Let your patches speak? :)
When writing, it would be good to incorporate the complex VLR FSMs as well.

#7 - 12/09/2016 03:25 PM - msuraev

neels wrote:

> for example, in the http://kleinekatze.de/quooXai5/vlr_auth_fsm.dot.png FSM, the START event can
> cause a transition to either the WAIT_RESP or the NEEDS_AUTH_WAIT_AI state, which depends on the
> decision made in "arbitrary" C code that is also part of the FSM logic.

I think this should be rewritten to remove such ambiguities - for example by splitting one state into couple of separate states. I hope to get to this once autogeneration for DTX FSM is in place.

#8 - 12/12/2016 11:34 AM - laforge

On Fri, Dec 09, 2016 at 09:20:50AM +0000, msuraev [REDMINE] wrote:

> It's just matter of splitting state transition logic (completely
> described by .dot file) from action logic (application-specific) which
> I think is a good practice anyway. The former should be autogenerated
> and include calls to functions like ...fsm_state_XXX_evt_YYY() which
> will contain action logic.

I know that approach, but I feel like it leads to an explosion of
functions, and in the end to less readable code. The fact that all
events that can occur in a state (excluding the allstate events) are
handled at one given location in the code is a "design feature".

I'm not saying you should give up the idea, I'm just raising my
concerns. In the end, it probably depends on comparing some examples in
the generated vs. hand written approach.

--
- Harald Welte <laforge@gnumonks.org> http://laforge.gnumonks.org/
==========
"Privacy in residential applications is a desirable marketing option."
(ETSI EN 300 175-7 Ch. A6)

#9 - 12/12/2016 11:34 AM - laforge

On Fri, Dec 09, 2016 at 03:25:35PM +0000, msuraev [REDMINE] wrote:

> neels wrote:
>
> > for example, in the http://kleinekatze.de/quooXai5/vlr_auth_fsm.dot.png FSM, the START event can
> > cause a transition to either the WAIT_RESP or the NEEDS_AUTH_WAIT_AI state, which depends on the
> > decision made in "arbitrary" C code that is also part of the FSM logic.
>
> I think this should be rewritten to remove such ambiguities

I don't think this is the way to go. In reality, you always have such
situations. the osmo_fsm code was written to be able to reflect the
state machines (with their events and states) as they are specified in
e.g. 3GPP specs. At that point, your aim is to have the same states and
events like the spec - not to split up into even more states, creating a
non-trivial mapping between the spec and your osmo_fsm based

implementation.
--
- Harald Welte <laforge@gnumonks.org> http://laforge.gnumonks.org/
==========
"Privacy in residential applications is a desirable marketing option."
(ETSI EN 300 175-7 Ch. A6)

#10 - 12/27/2016 12:49 PM - msuraev

laforge wrote:

> I don't think this is the way to go. In reality, you always have such
> situations. the osmo_fsm code was written to be able to reflect the
> state machines (with their events and states) as they are specified in
> e.g. 3GPP specs.

Well, in this case we just have to ask user to supply function which will make this selection resolving the ambiguity. Anyway, let's take care about such implementation details when we'll come around to actually implementing it.

#11 - 03/01/2018 11:17 PM - laforge

- Assignee deleted (msuraev)