OsmoGSMTester - Feature #2230

osmo-gsm-tester: arfcn as resource

05/04/2017 10:18 PM - neels

Status:	New	**Start date:**	05/04/2017
Priority:	Normal	**Due date:**	
Assignee:	osmo-gsm-tester	**% Done:**	10%
Category:			
Target version:			
Spec Reference:			

Description

So far we have ARFCN resources configured, but they aren't actually resolved and reserved yet.

- Add arfcn to the suites/sms/suite.conf
- implement reservation of an arfcn according to the picked BTS band

A problem to overcome could be that a reserved ARFCN doesn't match the BTS band. This may need to be resolved in the scenario files selecting a BTS. i.e. on top of picking a BTS type, a matching ARFCN band needs to be selected. To resolve this directly from the selected BTS implies hardcoded magic, which is also possible, but it would be better if that can be avoided.

Related issues:

Related to OsmoGSMTester - Bug #2245: osmo-bts cfg: fix band string	**New**	**05/07/2017**

History

#1 - 05/04/2017 10:22 PM - neels

fyi, the arfcn so far comes from the defaults.conf.

Maybe it does make sense after all to have some magic for the arfcn, to imply automatically that a BTS also needs an ARFCN of matching band...

#2 - 05/07/2017 12:51 AM - neels

- Related to Bug #2245: osmo-bts cfg: fix band string added

#3 - 05/14/2017 11:11 AM - laforge

- Assignee deleted (Osmocom CNI Developers)

#4 - 05/15/2017 12:58 PM - laforge

- Assignee set to osmo-gsm-tester

#5 - 08/28/2017 04:44 PM - pespin

- Status changed from New to Feedback

- Assignee changed from osmo-gsm-tester to pespin

- % Done changed from 0 to 90

Patch available for review here: https://gerrit.osmocom.org/#/c/3731/1

#6 - 09/07/2017 03:23 PM - pespin

After discussions with Neels, the good way to solve this seems a bit complex and we are postponing this for a while until there's more real/urgent need for this.

Right now, we do the following:

1. grab state lock
2. intersect free "resources" dictionary/set with "scenarios" dictionary/set.
3. Then, 1 permutation/possibility of this intersection of resources is used and reserved for the test.
4. free the lock

The summary is that specific logic needs to be added while having the state file lock grabbed which does a second step of resolution through all the possibles permutations to find the one which validates the following requirements and then reserve that specific ARFCN dynamically. This step has actually 2 sub-steps:

1. all objects share a given band
2. an ARFCN resource is free for that band.

suite.conf must specify how many ARFCN resources are required.

- If no band and arfcn are specified, the complete process with the 2 steps above is run.
- If band is specified but no arfcn is specified, the the code validates that resources (bts and modem) support the same band specified by checking the "bands" attribute. Sub-step 2 is then run to try to allocate an ARFCN for that band.
- If band and arfcn is specified, only try to allocate that arfcn (and fail if not possible, same for all cases explained above).

==========
See an example here:

```
free resources: {
modem: { ... , bands: [ x, y ] }, # let's call this modem "A"
modem: { ... , bands: [ z ] },    # let's call this modem "B"
modem: { ... , bands: [ x, z ] },     # let's call this modem "C"
bts; { ..., bands [ x, z ],     # let's call this BTS "D"
 # no free arfcn for band X, 2 free arfcns for band Z
arfcn: { arfcn: 50, band = z },
arfcn: { arfcn: 60, band = z },
}

scenario:
{ # no requirements
}

suite.conf:
{
modem:
 times: 2
bts:
 times: 1
}
```

1. Current code (already existing) creates a subset "intersect" by intersecting free resources with scenario (in this case the subset is the same as free resources because scenario is empty).
2. With this "intersect" set, we create a set list of permutations based on different objects of type "bts" and "modem" which fullfill the requirements for suite.conf, and we iterate over it: [A+B+D], [A+C+D], [B+C+D]
3. [A+B+D] --> No common band between them, skip
4. [A+C+D] --> Common band "X" between them, but there's no free arfcn resource for band X, so we skip too
5. [B+C+D] --> Common band "Z", and we successfully allocate firs tarfcn available for that band: 50

Then this resource is during the test fetched through suite.py and we pass that to BTS and modem (in ofono implmentation we just don't use it as we don't need it).

==========

Similar process can be applied for "ciphers" attribute for instance, which has the issue, although it's not that important because only first requirement is needed in this case (because there's no limited cipher resources) and because all modems so far support a5 0 (which we are using by default).

We could extend this by somehow specifying "air groups", which would basically allow to specify which/how many modems and bts need to share a band. For instance, "this 2 modems and this bts should use same frequency" and "this other modem with htis BTS should use another one".

#7 - 09/15/2017 09:46 AM - pespin

- Status changed from Feedback to New

- Assignee changed from pespin to osmo-gsm-tester

- % Done changed from 90 to 10

#8 - 12/14/2017 04:08 PM - pespin

There's an extra requirement we didn't think of when writing the description above:

- TRX of a given BTS cannot use immediately consecutive ARFCNs. For instance, if a BTS has 2 TRX, it cannot use arfcn 868+869, but needs to pick for instance 868+870.