Each VLR requesting auth tuples should use a distinct IND bucket for 3G auth. Currently we do that by assigning a different IND to each connected client. Since MSC and SGSN for a given subscriber are two distinct GSUP peers, they will end up using distinct INDs.

However, in a distributed setup, a roaming subscriber has one direct GSUP peer proxying for both remote MSC and SGSN. That means as soon as a subscriber roams to a different site, the site's MSC and SGSN compete for the same IND bucket and waste SQNs rapidly.

So instead of using the local client as IND, we need to keep a list of VLR names and assign a different IND to each. Then we use the GSUP source_name as indicator, which reflects the actual remote VLR's name. Hence MSC and SGSN will be distinct INDs also for remote sites.

This could be done in-mem like now, but would also be better to persist in the database.

In #4305, there is a discussion about IND collisions across remote sites. The discussion goes about how often per day a subscriber is likely to pass between sites that have identical INDs set. When thinking about MSC vs SGSN INDs, however, an IND collision would be far worse, because each and every CS and PS Complete-Layer-3 would waste SQNs and load osmo-hlr to generate auth all the time.

So it would make sense to have a built-in scheme that definitely and always avoids CS vs. PS IND collisions. We could simply assign even-numbered INDs to CS and odd INDs to PS.

Related to OsmoMSC - Feature #4318: osmo-msc should indicate CS-domain in GSUP SendAuthInfo-Request added
We could simply assign even-numbered INDs to CS and odd INDs to PS.

This part is not as trivial as I thought at first. It is of course easy to make sure that a CS IND is always even and a PS IND is always odd, but what also plays into this is that the source_name also differs for MSC vs. SGSN.

Example: site A has 'MSC-A' and 'SGSN-A'.

My first idea would be to keep one table of known source_name entities, and apply the CS/PS bit after that. This plays out as:

<table>
<thead>
<tr>
<th>idx</th>
<th>source_name</th>
<th>resulting IND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'MSC-A'</td>
<td>(1 &lt;&lt; 1) = 2</td>
</tr>
<tr>
<td>2</td>
<td>'SGSN-A'</td>
<td>(2 &lt;&lt; 1) + 1 = 5</td>
</tr>
</tbody>
</table>

I get 2 and 5, where I would have liked to get 0 and 1. It is not actually a problem that it starts with 2, so I am fine with getting 2 and 3. But since SGSN-A has table index 2, I get 2 and 5. Imagine every site always showing up with MSC and SGSN in sequence, then we would have

<table>
<thead>
<tr>
<th>idx</th>
<th>source_name</th>
<th>resulting IND ((idx &lt;&lt; 1) + ps) % 32</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MSC-A</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>SGSN-A</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>MSC-B</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>SGSN-B</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>MSC-C</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>SGSN-C</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>MSC-D</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>SGSN-D</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>MSC-E</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>SGSN-E</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>MSC-F</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>SGSN-F</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>MSC-G</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>SGSN-G</td>
<td>29</td>
</tr>
<tr>
<td>15</td>
<td>MSC-H</td>
<td>30</td>
</tr>
<tr>
<td>16</td>
<td>SGSN-H</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>MSC-I</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>SGSN-I</td>
<td>5</td>
</tr>
</tbody>
</table>

i.e. the cumulative effect of SGSN getting only every other table index + the separation makes us use only half the IND space by skipping every other IND, and actually upon IND number space wrapping, makes us wrap exactly back to the already used IND instead of those in-between: we use 2 and 6 for MSC, but never 0 or 4.

I would still like to separate CS and PS INDs for certain, though, yet not rely on the MSCs and SGSNs to always turn up nicely in sequence. So I need separate tables for CS and PS.
found a way to have both in one table with distinct indexes for each cn_domain, with the benefit that the cn_domain is just a number, and arbitrary other cn_domains could be invented without needing to change the db scheme.

(We could use a separate cn_domain (OSMO_GSUP_CN_DOMAIN_NONE == 0) for those SendAuthInfo requests that failed to send a CN domain at all. The benefit would be nil though -- I'm not going to <<1 another bit to assign actually half of all IND space for omitted CN domain indicators. Ok, nice feat but not useful unless we actually have more valid CN domains.))

CREATE TABLE ind (  
    cn_domain INTEGER NOT NULL,  
    -- 3G auth IND bucket to be used for this VLR, where IND = (idx << 1) + is_ps  
    ind INTEGER NOT NULL,  
    -- VLR identification, usually the GSUP source_name  
    vlr TEXT NOT NULL,  
    PRIMARY KEY(ind),  
    UNIQUE (cn_domain, vlr)  
);  
insert into ind (cn_domain, ind, vlr) select $cn_domain, count(ind)*2+$cn_domain-1, $vlr from ind where cn_domain = $cn_domain;

meh, that looks nice because it directly shows the actually used IND, but leads to problems adding new entries as soon as one of them is removed from the middle. Separate tables may be necessary after all, unless I find something nifty to also cover that.

Another aspect is that we might like an IND to become re-used if it was deleted, instead of first wrapping around the entire number space. So unless some SQL magic shows up that converges in finite time, we'll actually need to read all IND entries into RAM and handle all those constraints in C instead.
This does exactly what we want:

```
CREATE TABLE ind (
    cn_domain INTEGER NOT NULL,
    -- 3G auth IND bucket to be used for this VLR, where IND = (idx << 1) + cn_domain -1
    ind INTEGER PRIMARY KEY,
    -- VLR identification, usually the GSUP source_name
    vlr TEXT NOT NULL,
    UNIQUE (cn_domain, vlr)
);

insert into ind (cn_domain, ind, vlr)
SELECT $cn_domain,
CASE WHEN EXISTS(SELECT NULL from ind where cn_domain = $cn_domain LIMIT 1)
THEN
    SELECT ((ind + 2) - ((ind + 2)%2) + ($cn_domain-1))
FROM    ind as mo WHERE NOT EXISTS (SELECT NULL FROM ind as mi
    WHERE cn_domain = $cn_domain and mi.ind = ((mo.ind + 2) - ((mo.ind + 2)%2) + $cn_domain-1)) OR
DER BY ind LIMIT 1
    ELSE ($cn_domain-1)
END ind
, $vlr;
```

Patch is implemented and passes C tests. Next up is test in a live environment and verify that ttcn3 suite still passes.

```
#10 - 12/12/2019 06:52 PM - neels

Patch is implemented and passes C tests. Next up is test in a live environment and verify that ttcn3 suite still passes.

#11 - 12/13/2019 12:40 PM - neels

Turns out that bit-shifting separation isn't such a great idea after all. See https://osmocom.org/issues/4318#note-7

We do need some way or other of getting adjacent sites into distinct IND pools, all of:
- CS and PS of the same site
- 4G and 23G of the same site
- physically adjacent separate sites.
By now I think my bit shifting idea is not a good one at all, because it addresses only one of the above points. We need a solution for all of them.

Instead, I will roll back that complexity and use a simple incrementing IND across all VLR names.

#12 - 04/28/2020 02:48 PM - neels

https://gerrit.osmocom.org/c/osmo-hlr/+/16764
waiting for merge

#13 - 05/06/2020 12:53 AM - neels

- Status changed from In Progress to Resolved
- % Done changed from 90 to 100

merged

#14 - 05/06/2020 12:54 AM - neels

- Status changed from Resolved to In Progress
- % Done changed from 100 to 90
whoops sorry, this one is still waiting on gerrit