Distributed GSM - Feature #4305
GSUP proxy: cache auth tuples to re-use / fall back to simpler auth methods
12/04/2019 01:56 PM - neels

Status: Stalled
Priority: Low
Assignee: 
Category: 
Target version: 

Start date: 12/04/2019
Due date: 
% Done: 0%

Description
if a link goes away, we'd like to still allow continued operation for some time.

Goals:
- we don't want to replicate the auth keys themselves; just ask for tuples from the home HLR.
- for a later attach, assume the link to home hlr is broken.
- we have cached the previous auth tuples and attempt to re-use them.
- Maybe we have asked for N more tuples on auth tuple request.
- mileage sqn: use a separate SQN IND bucket for each site, so going back to a previous site will be fine with SQN number jumps.
- if that doesn't work, fall back to 2G auth if possible.
- if that doesn't work, fall back to no auth (for some time/during link failure, for continued service)

Related Issues:
Precedes Distributed GSM - Feature #4306: GSUP proxy cache: store data persis...

History
#1 - 12/04/2019 01:58 PM - neels
- Priority changed from Normal to High

#2 - 12/04/2019 04:09 PM - neels

#3 - 12/10/2019 03:55 PM - neels

thoughts about auth tuples:
- whenever a VLR does a SendAuthInfo-Request, we ask the home HLR for auth tuples.
- we request more auth tuples than the VLR requests.
- when number of unused auth tuples left reaches a low watermark, we ask the home HLR for more.
- Numbers: maybe ask for 20 auth tuples whenever less than 10 are available? (configurable)
- Milenage: each site should use a different SQN IND bucket.
  - Configurable? Site admins assign one CS and PS IND bucket to each site manually?
    - drawback: typical IND bitlen of 5 allows 32 buckets, i.e. 16 sites with CS and PS each. The 17th site causes collision with one specific other site.
    - (Could be chosen so that colliding sites are far apart)
    - To have more sites requires larger IND bitlen which means reconfiguring every SIM card
    - Impl: Currently only the home HLR controls IND bucket choice. The site requesting auth tuples does not indicate a bucket, it just gets tuples. Choosing different buckets per VLR is entirely up to the home HLR.
    - Determine ad-hoc? Each new requesting site gets the next IND bucket?
      - need only buckets for number of sites a given subscriber actually visits
      - drawback: storing each site's IND bitlen needs DB enhancement. (Might be a good idea anyway though)
      - either way nr of buckets can be insufficient for say a traveling nurse visiting each and every site,
  - maybe need some sort of sane recovery from IND bucket collision?
    - maybe it is acceptable to just resync on collision (and waste SQNs), which means the home HLR needs to be reachable for that
    - SQNs would be wasted only if a subscriber often travels between two sites with the same IND bucket.
    - If the subscriber usually visits 5 sites, but once a year visits 12 other sites as well, the problem is actually very limited in effect.
    - A problem would also only manifest when the home HLR were unreachable and a sync is not possible.
  - maybe recommend operators to choose an IND bitlen of 8 from the start, which raises the IND bitlen problems only with the (256/2 + 1=) 129th site -- seems "impossible" that any subscriber visits 129 sites more than once a year.

#4 - 12/10/2019 08:50 PM - laforge

Hi,
On Tue, Dec 10, 2019 at 03:55:26PM +0000, neels [REDMINE] wrote:

- maybe recommend operators to choose an IND bitlen of 8 from the start, which raises the IND bitlen problems only with the \( (256/2 + 1) = 129 \) 129th site -- seems "impossible" that any subscriber visits 129 sites more than once a year.

This should definitely be tested. I could imagine some problems with SIM cards, as they need to store all those counters in one file. It's 6 bytes per bucket, so at \( 2^{8} \) it would be 1.5 kilobytes in a single file on the card. That may work, but at least for SIM cards that's an unusually large file.

I can try to create such a card, in case you want to test it.

Also, even if it works with the (future) sysmoSIM-SJA2, that of course doesn't mean that it will work with many other cards / OSs / vendors.

#5 - 12/11/2019 11:05 PM - neels

laforge wrote:

- maybe recommend operators to choose an IND bitlen of 8 from the start, which raises the IND bitlen problems only with the \( (256/2 + 1) = 129 \) 129th site -- seems "impossible" that any subscriber visits 129 sites more than once a year.

This should definitely be tested. I could imagine some problems with SIM cards, as they need to store all those counters in one file. It's 6 bytes per bucket, so at \( 2^{8} \) it would be 1.5 kilobytes in a single file on the card. That may work, but at least for SIM cards that's an unusually large file.

I guess figuring out a sane strategy for re-using buckets across sites is the best way for now. Even visiting more than 16 sites every day seems uncommon in practice. I mean, 8 bits and 128 sites is practically "infinite", 16 sounds like not so many, but fair enough.

I can try to create such a card, in case you want to test it.

Isn't it "just" some milenage parameter fu? (We can get back to this when we've progressed a bit, so far it's just brainstorming.)

I guess the strongest way to re-use IND buckets is to actually store in the HLR DB which vlr_name gets which IND bucket, for each subscriber.

Another way that needs less database storage (and does less user tracking) would be to per-HLR globally assign an IND bucket number to each source_name that shows up. That could by coincidence choose an IND collision between adjacent sites, which would then affect all subscribers.

"adjacent" meant in the sense of common usage patterns, not necessarily physically adjacent (but usually also physically adjacent).

I would assign incrementing numbers to each new vlr_name, and then modulo it with the individual subscriber's IND bucket size, so that if anyone ever chooses a larger IND bucket, they would immediately benefit from that without resetting the IND bucket assignments.

Thinking in practical probability, with such round robin of IND buckets and a person at-home here moving along remote sites adjacent to each other, it seems unlikely to end up with same IND buckets for adjacent sites. It would have to be exactly \((n^{16})\)-1 new sites showing up, exactly in-between two adjacent sites first showing up. Possible but seems improbable. In other words, if subscribers follow their common usage patterns, adjacent sites will tend to show up in order, so that they would tend to get different IND buckets; this doesn't hold true though if a multiple of 16 subscribers do that concurrently. :P Rough ballpark estimate for sites with hundreds of active subscribers right from the start would still be \(< 6\% \) \((100/16)\) probability of assigning adjacent sites the same IND bucket.

For scenarios where initially very few people roam across sites to establish common "adjacency" concurrently, the collision probability should be near zero.

Thinking further, osmo-hlr could technically also keep track of which sites with identical IND buckets often see resync, or could keep simple counters of site adjacency (globally count subscribers moving from which site to which other site), and could swap IND buckets around to optimize. Or, an admin could manually modify the HLR db table of assigned IND buckets to adjust for common usage / physical proximity.

Changing the IND bucket that one site uses for a remote site can be done any time, which might trigger a lot of resyncs for a short period for users at-home at the site where the config was changed, but it is a situation that would settle without human action required in a reasonable time frame.

After this brainstorming I think we can just store an IND bucket per vlr_name globally per-site, and we don't really need to store it per-subscriber. Storing it per-subscriber could much more safely avoid SQN resync, but
- collisions only begin to matter for more than 16 sites.
- the database tables storing separate IND per subscriber and per vlr_name could grow pretty large.
- it would also be a privacy concern in the sense of revealing exactly where each individual subscriber has ever been. These combined seem to me not worth the trouble when held against a worst-case <6% chance of collisions plus the possibility of heuristic and/or manually configurable remedy for unfortunate IND bucket assignments.

#6 - 12/11/2019 11:52 PM - neels

The worst case of assigning IND per source_name globally at each HLR:
One or very few subscribers have a usage pattern of visiting two sites every day, but no-one else has a similar usage pattern, and globally it doesn't show as a problem / current choice seems optimal. These few subscribers would then resync every day and rapidly waste SQNs.
In that light it might actually seem desirable to rather manage INDs per-subscriber?

A heuristic against this could be to per-subscriber count number of Location Updating per number of tuples requested. If the numbers diverge from the site's average, it could give a stronger trigger to re-arrange the IND assignments.

How far would the numbers diverge? Normally, I would have about four periodic LU every hour and maybe 5 calls/SMS per day, so I would have slightly more tuples than LU, but roughly the same. If I add to this scenario moving across an IND collision and back once a day, and if an HLR proxy requests 20 tuples each time, I would add to normal 24*4 + 5 = ~100 tuples another 20 wasted tuples. It doesn't actually seem like a big deal at all.

It really starts mattering if a subscriber usually is at the boundary between two sites and often does LU back and forth between two IND-colliding sites. For every move there and back we'd waste another 20 SQN. But at the same time, if two sites are physically adjacent, it would also make sense to globally assign non-colliding IND for those sites globally, and it is unlikely that it would go unnoticed.

So after all I think we should implement one global IND assignment for vlr_names showing up, per HLR db and not per subscriber. If we end up seeing problems with this, it is still fairly easy to add another feature that enables per-subscriber IND assignments (could even be enabled on a per-subscriber basis).

The database tables could be designed in a way that optionally allows tagging IND entries with an individual subscriber from the start. Then again, all it would need is adding one column to the IND table, which would be easy to do in a db schema upgrade.

Ok, enough brainstorming for Milenage now.

#7 - 12/12/2019 07:40 AM - laforge

On Wed, Dec 11, 2019 at 11:05:08PM +0000, neels [REDMINE] wrote:

Isn't it "just" some milenage parameter fu?

Well, the card obviously needs to allocate persistent memory for each of those IND buckets. I think it was six bytes for each of them. That in turn means you need to create a larger file during personalization time, and I don't think any other usage on SIM/USIM cards currently allocates files that are ~ 1.5 kByte in size.

File sizes beyond 256 bytes also means that you cannot read or write them in a single "READ BINARY" or "UPDATE BINARY" command, where you have 16bit of index but only 8 bit of length...

In theory it would work. But it's one of those things where I'd not be surprised if nobody actually ever has tested it, particularly not on each different card/OS model out there.

#8 - 04/28/2020 02:37 PM - neels

- Status changed from New to Stalled
- Priority changed from High to Low

There are a number of quite elaborate patches on a private branch:
Full design (only partly implemented) of two FSMs handling incoming and outgoing GSUP requests in the HLR proxy. The non-caching patches currently up for merge is able to simply feed GSUP messages thru between home HLR and local site MSC.
Adding an auth tuple cache requires terminating the GSUP messages at the HLR proxy, and separately negotiating GSUP towards the home HLR. So, a design for this is finished, but when starting to implement it, it became apparent that the effort is too substantial.

The "parked" design and code sits in osmo-hlr.git on branch neels/dgsm-tuple-cache

#9 - 04/28/2020 02:39 PM - neels

- Precedes Feature #4306: GSUP proxy cache: store data persistently added