OsmoSGSN - Feature #1580

IP header compression

02/23/2016 03:42 PM - laforge

Status:	Closed	**Start date:**	02/23/2016
Priority:	High	**Due date:**	
Assignee:	dexter	**% Done:**	100%
Category:			
Target version:			
Spec Reference:			

Description

3GPP specifies optional IP header compression. We should offer it at least to those phones that support it in order to save scarce air-interface bandwidth!

Related issues:

Related to OsmoSGSN - Feature #1655: KPIs for OsmoSGSN	**Stalled**	**03/11/2016**
Blocks OsmoSGSN - Bug #1794: support random IV for GEA (via XID)	**Stalled**	**08/09/2016**

History

#2 - 04/28/2016 07:27 PM - laforge

- Assignee set to spaar

#3 - 05/31/2016 03:52 PM - lynxis

- Related to Feature #1655: KPIs for OsmoSGSN added

#4 - 06/22/2016 10:14 PM - laforge

- Assignee changed from spaar to msuraev

#5 - 06/22/2016 10:21 PM - laforge

- Priority changed from Normal to High

#6 - 07/01/2016 10:37 AM - laforge

- Assignee changed from msuraev to dexter

#7 - 07/01/2016 11:05 AM - laforge

Relevant spec references:

- http://www.etsi.org/deliver/etsi_ts/144000_144099/144065/13.00.00_60/ts_144065v130000p.pdf for SNDCP, which refers to three possible header compression schemes
  - RFC1144 old-style van jacobsen header compression
  - RFC2507 (IPHC) https://tools.ietf.org/html/rfc2507
    - no foss implementations?
    - wireshark only as part of ppp?
  - RFC3095 (ROHC) https://tools.ietf.org/html/rfc3095
    - https://en.wikipedia.org/wiki/Robust_Header_Compression
    - https://rohc-lib.org/ LGPL implementation, actively developed/maintained
    - wireshark seems to support ROCH (maybe not hooked up to LLC/SNDCP yet?) as well as the ROHC negotiation parameters part of SNDCP-XID.

#8 - 07/01/2016 11:06 AM - laforge

so one of the first steps is to figure out which of the three header compression schemes to implement.

ROHC is probably the most powerful and efficient scheme, and there's a full library implementation so one doesn't have to worry about details of the algorithm. However, it is unclear how many phones actually support ROHC. If support was scarce, implementing one of the older algorithms might have higher priority.

#9 - 07/07/2016 07:56 AM - dexter

- % Done changed from 0 to 10

Status update: It turned out that a functioning XID parameter negotiation is mandatory to make use of compression. We will have to implement the XID negotiation first. I am working on this, I already have some experimental code that sends a XID and I am able to see the response from the phone. However, it now needs some elaboration and cleanup. It also has to be automated, the XID message has to be sent right after the PDP-Context establishment.

#10 - 07/11/2016 08:14 AM - dexter

Status update: We are now able to send XID requests to the phone and get a response. Currently I am working on cleaning up my implementation and make it programmatic. We also need to implement some functionality to compile the SNDCP-XID, which is then send as a parameter-field withe the normal XID.

#11 - 07/11/2016 04:30 PM - dexter

- % Done changed from 10 to 20

Status update: The XID now gets issued programmatic from gprs_gmm.c:gsm48_tx_gsm_act_pdp_acc() to the right TLLI/SAPI. (I am not sure if that is the right place) I also implemented some routines to generate the XID from a prepared LLIST, I also have the parser ready that turns the XID bytestream into an LLIST. Before moving on with code that generates the intersection of applicable XIDs I think its best trying to issue some compression parameters. Implementing the message format described in ETSI TS 144 065, Figure 10 should not be that hard.

#12 - 07/12/2016 05:12 PM - dexter

Status update: Made some progress with the SNDCP-XID today. We now have encoding functions for the ROHC header compression parameters and an encoding function that warps the encoded parameters in an protocol control information compression field. The resulting bytestream has to be wrapped together with some header data (version etc...) into the final field that is finally sent out as a regular SNDCP parameter field.

#13 - 07/13/2016 04:10 PM - dexter

- % Done changed from 20 to 30

Status update: I went once more through the code for the SNDCP-XID I wrote yesterday. Now its generating the complete nested TLV structure from a list of structs. The result can be added as LL-XID parameter type 11. Setting of the XID in gprs_gmm.c was not very nice. The spec says that the SNDCP-XID has to be set of by the SNDCP user. This would be gtp in our case. I found a suitable place in gprs_libgtp.c. Like in ETSI TS 144 065, Figure 11 it now sets of the SNDCP-XID, which then sets of an LL-XID at the LLC layer. Currently it does not generate any compression parameters yet. I also will have to figure out how to handle the confirmations (here I might need some advise)

#14 - 07/14/2016 05:00 PM - dexter

Status update: We are now able to generate SNDCP-XID messages for all three header compression schemes. We decided to implement the parameter fields for the two older schemes because we want some alternatives to experiment with in case we have problems. The current situation is that we are able to send an SDNCP message, we also get an answer but the phone always seems to reject the parameters we send (We say: All SAPIs are applicable, Phone says: No SAPIs are applicable). I tryed with all three comression types, same result for all of them. Before we move on we need to experiment a bit. I suspect that just selecting all SAPIs is not a good idea, maybe other parameters are wrong too. A trace from productive network would be very helpful.

#15 - 07/17/2016 10:10 PM - dexter

Status update: Unfortunately we don not have any luck with the compression parameters yet. The code now only selects the NSAPI which is currently in use. So it does not pick all NSAPIS anymore. In Wireshark I can confirm that the NSAPI selection works perfectly fine. There was also some misunderstanding with the PCOMP values, which is now fixed. I am currently trying with rfc1144 header compression. This is the simplest of the three. It only has one (!) parameter. Even there my compression scheme selection gets rejected. It always echos Applicable NSAPIs with all zero, which means by the standard that the compression is rejected and the compression entity has to be deleted. I also checked on T-Mobile. As far as I can see they do not offer any compression.

On the internet I found a very promising PCAP trace here: http://pcapr.net/view/tyson.key/2009/9/5/15/xid_rohc.txt.pcap.html Unfortunately it seems to be impossible to get an accound on pcapr.

#16 - 07/18/2016 08:03 AM - msuraev

- Blocks Bug #1582: GEA Encryption is missing added

#17 - 07/18/2016 04:23 PM - dexter

Status update: We found out that at least for the k800i phone we have here the header compression can be activated in the APN settings. If activated, the phone sends a XID by ittself and asks the GGSN for a compression entity. Apperently it seems to be common sense that, if compression should be used, the phone has to care about the XID. I am a little bit confused by the fact that I can activate compression via the APN settings. I suspect that compression has to be explicitly allowed in the APN settings in order to be used. However, with the current Code I can not answer SNDCP-XID requests from the phone. I started writing the decoding functions for SNDCP-XID messages. The code is almost finished, only ROHC parameters are still missing. Will push my stuff tomorrow when I am done with that.

#18 - 07/19/2016 04:59 PM - dexter

We are now also able to pars SNDCP-XID bytestreams. Finishing up the code took me longer than I thought. Dissecting XID messages is rather painful with its nested structures and optional fields (P-Bit!). I also learned that multiple compression fields can be appended one after another, there is no need to put each of it into a separate TLV field. The code works with the testvectors I picked from wireshark. Tomorrow I will test it on the ggsn code.

#19 - 07/20/2016 11:30 AM - dexter

Status update: There is some progress with the SNDCP-XID: "<0013> gprs_sndcp.c:551 We don't support compression yet" - Looks like compression is truned on. Just wanted to let you know.

#20 - 07/20/2016 08:25 PM - dexter

Status update: The SNDCP-XID code got some bugfixing here and there and now works find. We are now able to answer phone originated SNDCP-XID indications correctly. Currently I reject any request for compression, which causes the network to function normally again. When I accept the compression I can see "We don't support compression yet" in the log and some of the packets in wireshark have a PCOMP value set. This confirms that the compression in the phone got activated. I am still wondering if compression is something that explicitly has to be allowed within the APN settings. However, the k800i turned on its compression mode. In this case its RFC1144 only, but we finally have something to play with.

I am currently having a look at the RFC1144 compression (http://lxr.free-electrons.com/source/drivers/net/slip/slhc.c), I managed to compile it, I can also allocate and free compression states. However. There is stuff to fix, it uses functions that are only available in the kernel. I havend found userspace alternatives for all of them yet.

#21 - 07/21/2016 04:22 PM - dexter

Status update: Experimented a bit with slhc.c to see if it does something useful. I managed to compress a packet, but I can't decompress it yet. Its also not easy to test it standalone because it requires actual TCP/IP traffic with ACKs and so on to work. I think the best way to test it is to just compile it into the sgsn and compress and decompress all traffic in-place. I have isolated the position in the code where the packets are passing by so I can start some experiments tomorrow.

#22 - 07/22/2016 04:36 PM - dexter

- % Done changed from 30 to 40

Status update: I managed to run SLHC in a test environment (compressing/decompressing in same place as mentioned in the last update) After a lot of fixing and trying it now runs without errors. I can in the log how the packets get compressed and decompressed. I think we can risk to try it with a phone on monday.

#23 - 07/24/2016 08:54 PM - laforge

Hi Dexter,

in which branch are you keeping your patches? For features that take multiple weeks of implementation, it is very useful if others can follow your work, either by looking at the commitlog or by explicitly reviewing the branch.

Also, as a side note, the ticket is at 40% but the state is still "new?"

#24 - 07/25/2016 08:17 AM - dexter

- Status changed from New to In Progress

- % Done changed from 40 to 50

Hi Harald,

I am on dexter/draft (http://git.osmocom.org/openbsc/?h=dexter/draft). I did not want to push directly in the review branch because I did not want to cause problems.

We now have the building blocks for this task complete. We can generate and dissect XID and and SNDCP-XID messages and we have Header compression code that works in a test environment. I would say we are at 80% now, but I can not say how long it would take to complete the task,

there are too many variables so I think its safe to say we are at 50%.

#25 - 07/25/2016 04:59 PM - dexter

Status update: I had a lot of trouble today. RFC1144 says that the first byte (Protocol version + Length = 45) of the IP packet shall be used to distinguish between TYPE_IP, UNCOMPRESSED_TCP or COMPRESSED_TCP. Thats how the receiving party can tell if it is a transparent packet (TYPE_IP) that is passed through without any modification or processing, if it is an uncompressed packet (UNCOMPRESSED_TCP) which has to fed into the compressor to learn the header information, or if its a compressed packet (COMPRESSED_TCP) where the header has to be restored. So far the theory.

By accepting the compression parameters the mobile phone sends I am able to turn on 1144 header compression. The phone then performs a DNS query (which has PCOMP=0) and then starts to send packets where PCOMP is set to 1. So compression is turned on. The packet still begins with 45 and has the appearance of a TYPE_IP packet, which is fine on the first look.

Unfortunately ETSI TS 144 065 implements a slightly modified variant. Since the packet type is already distinguished by the PCOMP flags, there is no need to patch the first byte of the IP packet as well. It is just left at 0x45.

So the packet we observe in the PCOMP=1 fields are in reality UNCOMPRESSED_TCP not an TYPE_IP packet as I thought. Since the UNCOMPRESSED_TCP uses the IP protocol fields as a connection ID, those packets can not be forwared unmodified otherwise the connection will die quickly. In an experiment I overwrote all IP protocol fields with unknown protocols with 0x06. This fixes the packets and the connection works again (but only as long as only UNCOMPRESSED_TCP packets are transmitted).

#26 - 07/28/2016 08:51 AM - dexter

- % Done changed from 50 to 60

Status updtate: There is some good news! After a lot of trying and testing the RFC1144 header compression I was able to set up a compressed connection with my K800i the first time. It actually seems to work fine. I now have to clean a few things up now and we also need an implementation for properly store and manage the compression entities.

#27 - 07/28/2016 04:55 PM - dexter

Status update: Started to implement the compression entity handling. The SNDCP-XID code has to be able to update/delete compression entites and the compressor has to be able to query the current compression state, the PCOMP-Indexes and so on... Its half way implemented now, but not yet active.

#28 - 07/29/2016 04:06 PM - dexter

Status update: The code to handle the compression entities is now done. The compression state is now completely handled. Also cleaned up the debug log messages. It now uses proper LOGP messages.

The rfc1144 compressor works fine, but I have the feeling that it does not run at its intended performance. The count of compressed packets that are transmitted is a bit low. When I look at the trace it looks good to me. I see some compressed packets going out, then a packet with an ACK flag set comes back. We definetly have to have an second look at it. But for now I am ok with it. I will now focus more on the protocol details. We still have some work to do to properly perform the xid.

#29 - 08/03/2016 05:03 PM - dexter

- % Done changed from 60 to 70

Status update: I am currently working on the network side XID. Its halfway done and there is good news. When I send (passive) an SNDCP-XID with rfc1144 parameters to my blackberry 9000, I get my NSAPI accepted. I could not fully try it out yet because I do not create the comression entity yet and on opt of that the blackbarry 9000 still has problems to access the internt. But for now it seems to me that compression in general seems to be a very exclusive feature. Also interesting: When I turn of header compression on my K800i, I do not get the parameters accepted. The K800i only seems to support compression when it is explicitly enabled.

#30 - 08/04/2016 04:58 PM - dexter

Status update: Most of code is now in the review branch (i think gprs_sndcp.c is the only one that is still missing, this will follow tomorrow). Also finished the networks side XID. It now sends a XID to the phone. When it receives the answer it checks on it and creates the compression entities if necessary.

#31 - 08/09/2016 02:22 PM - msuraev

- Blocks Bug #1794: support random IV for GEA (via XID) added

#32 - 08/25/2016 10:35 AM - dexter

- % Done changed from 70 to 90

After a longer time without updates, a quick heads up where we actually are:

- All code related to TCP/IP header compression is currently in the review process.
- The VTY has been updated, the user can select if the network should do header compression. By default header compression is always disabled.
- Since RFC1144 does not work effectively when TCP/IP option 8 (timestamps) is active this option has to be stripped
- The tests with the k800i show that the implementation is working, but as soon as 2 users are active, the connection breaks down for at least one subscriber. The problem has been investigated by checking all IP-Header and TCP-Checksums of all packets that enter and leave gprs_sndcp.c. All packets passing by check good, there are zero damaged packets! There is also no sign that the compression states are mixed up, nor is there a problem with mixed up pcomp values. I doubt that the problem is located in the sndcp layer or in the compressor but I am on it and we will try to pinpoint the problem.

#33 - 08/27/2016 09:15 AM - laforge

On Thu, Aug 25, 2016 at 10:35:57AM +0000, dexter [REDMINE] wrote:

> *The tests with the k800i show that the implementation is working, but as soon as 2 users are active, the connection breaks down for at least one subscriber.

If you suspect the problem is is somehow related to the PCU (which I doubt), you could test with a nanoBTS and its internal proprietary PCU and compare the results with those from OsmoPCU.

--
- 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)

#34 - 08/30/2016 05:05 PM - laforge

- Blocks deleted (Bug #1582: GEA Encryption is missing)

#35 - 09/19/2016 03:52 PM - dexter

- Rebased changes to current origin/master
- Optimized compression handling (no more memcpy() or talloc_zero_size() when compression is disabled)

#36 - 10/07/2016 07:54 AM - dexter

- Status changed from In Progress to Resolved

- % Done changed from 90 to 100

#37 - 10/11/2016 10:04 AM - laforge

- Status changed from Resolved to Closed