Contents

Introduction

This article covers understanding and troubleshooting Evolved General Packet Radio Service Tunneling Protocol for Control Plane (EGTPC) / GTPC (3rd Generation Partnership Project Technical Specification 29.274), GPRS Tunnelling protocol User Plane (GTPv1-U) (3GPP TS 29.281), and most relevant to this article, Restoration Procedures (3GPP TS 23.007) issues , which revolve around such concepts as restart counters, path failures detection, and echo requests/responses. It includes a list of relevant commands and what to look for in the output, along with explanation of related configurables. This is certainly an area for which a lot of questions are raised. Note that troubleshooting specific call control issues is in itself a very extensive area of interest, but this article only mentions how to capture statistics for call control - anything beyond that (i.e. call setup issues) is not covered. Really this article focuses mainly on troubleshooting and verifying the peer connections, without which there would be no call control. While written for operators, designers / implementers could also benefit from the content here.

Examples all use fake (changed) IP addresses from a real operator's network.

Background

Each side of a GTP connection (one of S5 (**Serving Gateway** (SGW) - **Packet Data Network Gateway** (PGW)), S11 (**Multimedia Management Entity** (MME) - SGW), S4 (**Serving GPRS Service Node** (SGSN) - **Gateway GPRS Service Node** (GGSN)) maintains a restart counter that is designed to alert the peer if the originating node has restarted so that that peer can take the appropriate action. While a node can use a unique counter for each peer it talks to, the design tends to be to use the SAME counter for all the peers, since when the node restarts, the counter needs to be incremented for all peers, and so tracking a separate value on a peer basis serves no extra benefit since they all need to increase by one anyway. In fact using the same restart counter for all peers is helpful in troubleshooting a large network where all the peers can be checked to make sure that the restart counters received have the same (single) value for the node those peers are talking to.

The restart counter CAN be included in all types of control messages, while the restart counter MUST be included in all echo requests and responses. As soon as a restart counter change occurs on a node, the various peers are informed about it via the next control messages sent to the respective peers. When a restart occurs, per specification, the receiving side has the option of temporarily maintaining the calls it has for a short while or in dropping them, the latter which is typically what is the configured behavior.

In the case of using echo requests, the method is to send the messages on a configured interval, and if no response is received after a configured number of retries (each after a configured timeout), the path is considered to be down and the appropriate action taken. Configuration allows for echos to be optionally sent, and this would apply to the peer as well, whose approach doesn't necessarily have to be the same, especially when the peer could be from a different vendor and/or

different carrier. In fact with regards to Echo responses, a node is only obligated to respond to an Echo request - whether it receives echos or not and with what frequency is not significant - it just needs to send an echo response for each request, and to take action if the restart counter received changes. Of course as already mentioned, IF it is also sending echo requests and doesn't receive timely responses, it needs to take action for that scenario as well.

Note that path failure detection can also be done over the GTP-U (user plane) connection via Echo, but ONLY via timeout, since there is no restart counter included in GTP-U messages like there is for GTP-C as previously discussed. GTP-U may or may not be using the same path as by control messages. If it is the same path, then GTP-U detection may not be as valuable since GTP-C detection should be sufficient.

In summary, here are the ways that path failure detection can be done:

- Restart Counter Change (via control msg Req/Rsp or Echo)
- GTPC Echo Req Time out
- GTPU Echo Req Time out
- Control Req Message Time-out (usually not configured)

Relevant Commands

In order to troubleshoot any kind of path issues, familiarize with the following commands. Following is an explanation of the relevant details of each.

- **show [egtpc-service | gtpu-service] all**
- **show egtpc peers** [egtp-service <service>]
- **egtpc test echo gtp-version {1 | 2} src-address <service address> peer-address <peer address>**
- **show egtpc statistics** [[[(sgw-address | pgw-address | mme-address) <address>] [demux-only]] [debug-only] [verbose] | path-failure-reasons]
- **show gtpu statistics [peer-address <peer>]**
- **show gtpc statistics** [ggsn-service <service>] [smgr-instance <instance>]
- **show demux-mgr statistics <egtpinmgr | egtpegmgr | gtpumgr> all** (SGW-egress | GGSN/ PGW,/SGW ingress | GTPU)
- **EGTPCPathFail[Clear], EGTPUPathFail[Clear] SNMP traps**
- **show session disconnect-reasons** - **gtpc-path-failure, gtpu-path-failure, path-failure**

Other commands for counting subscribers related to GTP:

show sub [summary] ggsn-service <service>

show sub [summary] gtpu-service <service>

show egtpc sessions [egtp-service <service>]

The basic command to check the status and configuration of all EGTP services is **"show egtpc-service"**. Most of the output should be what is already known via the configuration, as well as the status should be STARTED. The unique piece of information gleaned is the "Restart Counter" which cannot be found anywhere else. Related command **"show gtpu-service all"** returns configurables for the GTPU service(s) that are associated with the EGTPC services.

"show egtpc peers" is very useful in identifying all of the peers, including statuses, restart counters, and current/max subscriber counts.

- Note that this command covers both SGW and SGSN peers, whereas in the past there was a separate command for SGSN/GGSN.
- Since this command includes all peer types, the peer type can be filtered on via the service name (Even though the keyword egtpc-service deceptively also takes a GGSN service name).
- If there are no subscribers with the peer, then the status is Inactive and GTPC Echo disabled.

In that state, for EGTPC there MAY be some peers for which the Restart counter is reported, while most may be 0. This occurs for situations where the peer is actively sending echo requests to which ASR5K should be responding. The restart counter can be determined by running the EGTPC test command (discussed next), though in the inactive state, the counter will not be updated in "show egtpc peers" even though it was received. Meanwhile for GGSN, the restart counter is never forgotten.

Note a couple peers below that have counts much higher than the other peers. These are for SGWs that are physically located in the same site or the sister site of the PGW to which the MME funnels most calls.

Note that running this command on an Inter-Chassis Service Redundancy (ICSR) standby node is only relevant for active peers, in which case the numbers should be close, for example in the following pair:

PGW> show egtpc peers ... 20:51:52 (ACTIVE)

	Peer	RC	# sessions	Max Sess	Peer
AESKS	192.0.2.54	2	31	443	ADSKS
AESKS	192.0.2.24	3	117	123	ADNKS
AESKS	192.0.2.36	15	4223	6889	ADSKS
AESKS	192.0.2.44	7	119	331	ADSKS
AESKS	192.0.2.68	34	6174	9798	ADSKS
AESKS	192.0.2.69	7	2311	5027	ADSKS
AESKS	192.0.2.88	17	3914	8880	ADSKS
AESKS	192.0.2.164	16	833705	1298070	ADSKS
AESKS	192.0.2.184	3	1438418	2153317	ADSKS
AESKS	192.0.2.193	21	11447	26577	ADSKS
AESKS	192.0.2.195	5	48	162	ADSKS
AESKS	192.0.2.226	21	1962	3931	ADSKS
AESKS	192.0.2.227	6	4465	9332	ADSKS

PGW-ICSR> show egtpc peers ... 20:51:51 (STANDBY)

	Peer	RC	# sessions	Max Sess	Peer
AESKS	192.0.2.54	2	31	443	ADSKS
AESKS	192.0.2.24	3	117	123	ADNKS
AESKS	192.0.2.36	15	4223	6889	ADSKS
AESKS	192.0.2.44	7	119	331	ADSKS
AESKS	192.0.2.68	34	6171	9797	ADSKS
AESKS	192.0.2.69	7	2311	5027	ADSKS
AESKS	192.0.2.88	17	3913	8874	ADSKS
AESKS	192.0.2.164	16	832037	1294412	ADSKS
AESKS	192.0.2.184	3	1435461	2147280	ADSKS
AESKS	192.0.2.193	21	11443	26568	ADSKS
AESKS	192.0.2.195	5	48	162	ADSKS
AESKS	192.0.2.226	21	1962	3931	ADSKS
AESKS	192.0.2.227	6	4466	9332	ADSKS

"egtpc test echo" is used to check a specific peer to see if it is reachable or not and must be run in the context where the service is defined.

- Use gtp-version 1 for GGSN-SGSN connections
- Using ping command is not a valid test, though if it is successful, it is known that there is some level of reachability.
- The peer restart counter (Recovery) is displayed.
- Running this command will increment the Tx echo request / Rx Echo response counters in "show egtpc statistics"
- The restart counter of the peer is reported after Recovery keyword (11 in example below)
- Warning: though this should not happen in normal circumstances, if a test echo is performed

on a peer and it times out for whatever reason (if all is working then it shouldn't time out), all the calls would e dropped (even though things were working just fine up to the point of doing the test).

"show egtpc statistics" is for EGTPC (v2) and reports on success/failures of many EGTPC control messages (Request/Response, Tx/Rx) including Create Session, Modify Bearer, Delete Session, Downlink Data Notify, Release Access Bearer, Create Bearer, Update Bearer, Modify Bearer, Suspend, etc., and the associated responses. It includes all path management counters for Echo Request/Response Tx/Rx, and based on the timers, the growth of these timers can be predicted and used to troubleshoot if not incrementing as expected.

The following example shows echos and responses being initiated and responded to from both sides. Note that the SGW has been configured to send once per minute while the PGW has been configured for once every three minutes (both sides don't need to have the same setting) and so every three minutes there are twice the number of packets (6:16:33, 6:19:33). Keep in mind that although both sides are on the same "cycle" in this example, they do not have to be and can run independent of eachother. The restart counters has been captured in a column as well. Note the Restart counter of 11 from the SGW's echo response matches the value from its response to the PGW's test echo from above (the test was done at different time than the packet capture but for the same peer pair).

No.	Time	Source	Destination	Restart Counter	Info
1	06:15:33	SGW	PGW	11	Echo Request
2	06:15:33	PGW	SGW	18	Echo Response
3	06:16:33	SGW	PGW	11	Echo Request
4	06:16:33	PGW	SGW	18	Echo Response
5	06:16:33	PGW	SGW	18	Echo Request
6	06:16:33	SGW	PGW	11	Echo Response
7	06:17:33	SGW	PGW	11	Echo Request
8	06:17:33	PGW	SGW	18	Echo Response
9	06:18:33	SGW	PGW	11	Echo Request
10	06:18:33	PGW	SGW	18	Echo Response
11	06:19:33	SGW	PGW	11	Echo Request
12	06:19:33	PGW	SGW	18	Echo Response
13	06:19:33	PGW	SGW	18	Echo Request
14	06:19:33	SGW	PGW	11	Echo Response
15	06:20:33	SGW	PGW	11	Echo Request
16	06:20:33	PGW	SGW	18	Echo Response
17	06:21:33	SGW	PGW	11	Echo Request
18	06:21:33	PGW	SGW	18	Echo Response

The **demux-only** option captures all the path management counters (echo requests/responses) as well as control messages that never reach a sessmgr, which is generally Create Session Request/Response. Demux process etpinmgr residing on the Demux card is responsible for handling all echo request/response and initial handling of all call control messages before distributing them to the appropriate sessmgrs (randomly for new calls, and specifically for existing calls anchored on a specific sessmgr). The **sessmgr-only** option conversely only counts messages that get handled by sessmgrs, which does NOT include path management counters. Not specifying either qualifiers is the default of counting all values.

"show gtpc statistics" is the analogous command for SGSN/GGSN services (GTPC v1) (CPC, Update PDP Context, etc.)

"show gtpu statistics" reports stats on the user-plane traffic handled by the GTP-U service(s) that are associated with EGTPC service(s). One interesting counter is Error Indication Tx/Rx which

would be sent when the receiving node has no record of the subscriber that should be associated with the Terminal Endpoint Identifier (TEID) of the packet in question which could happen for a number of reasons (The bulkstats schema/variable is GTPU1/ err-ind-tx). If the MME/SGW has not been informed that the PGW no longer has a binding to the subscriber, then calls may need to be manually cleared on the MME/SGW (or wait for them to timeout) to fully resolve such an issue of orphaned calls.

The peer specific versions of the above stat commands are necessary when troubleshooting specific peers, which often is the case.

Path Failure detection

The **EGTPCPathFail** trap is the key for knowing when a path failure has occurred. Amongst some basic values that one would already assume, it reports the number of calls dropped, the old and new restart counters, and the reason for the failure. In addition, the session disconnect reason **gtpc-path-failure** will be incremented.

The **EGTPCPathFailClear** trap will indicate the restart counter along with the reason for the path clearing. The most recent StarOS v17 is better worded to say "clear reason" instead of "failure reason" (it cleared, not failed!) to avoid confusion.

The following are path failure counts that are reported from **"show egtpc stat path-failure-reasons"**

Notes:

- Control message restart counter at sessmgr could be any type of control message (not all listed here)
- Changed restart counter from the SGSN via the next Create PDP Context Request after that change **(**
- Changed restart counter from the SGSN via the next Update PDP Context Request after that change (which would be a handoff from another SGSN) **(upc-restart-counter-change)**
- Path clear failure reason sessions-add is a deceiving term - it is not a failure but a success, and it indicates that a new session has come in to establish the connection
- If the values reported for restart counters and session counts for the PathFail and PathClear traps do not seem to line up, open a ticket with Cisco for further explanation.

It is important to understand how EGTPC/GTPC and to a lesser degree EGTPU/GTPU (as user-plane failure detection is not always configured) path failure detection is implemented to monitor the connection between a GGSN or PGW and the various peer SGSN and SGWs. Normally the connections within a service providers network are fairly stable, in which case path failures are infrequent. On the flip side, experience has shown that GTPC paths amongst roaming SGSNs connecting to GGSNs may not be so stable, and so with poor connections the result is path failures happening at a fairly frequent basis.

The following config covers timers and transmission frequency for control and echo messages which can be used to detect path failures. Same concept applies to GGSN, SGW, MME, and EGTP services:

ggsn-service GGSN1

retransmission-timeout 20	For GTP control messages (versus the echo messages)
max-retransmissions 5	for BOTH GTP control and echo messages – how many times to retransmit bef path failure detection trigger
echo-interval 120	An echo is sent every 2 minutes (120 seconds).
echo-retransmission-timeout 5	For Echos, if no response, it retransmits every 5 seconds
path-failure detection-policy gtp echo	Path failure detection is possible via gtp echo (versus not using this as a metho detecting path failure (but still sending echos)).

associate gtpu-service **GGSN1_gtpu** bind ipv4-address 209.165.201.2	Associates the named GTPU service GGSN1_gtpu with this GGSN service The user and control plane bind IP addresses are the same in the following exa configs, but may not necessarily be configured as such with other service provi implementations.

Notes:

- Given time 0 being when an Echo request cycle starts (every 2 minutes), if there are no responses after a TOTAL of 30 seconds (6 attempts), the path fails (0 + 5sec + (5 * 5sec))
- Process egtpinmgr (gtpcmgr deprecated since v14) running on the demux card is responsible for processing the echo packets
- “show gtp statistics”, section “Path Management Messages” has statistics for the echo requests/responses for the control path in BOTH directions:

Another method of path detection is on the user plane instead of the control plane, and this is governed by the following config.

Notes:

- “no echo-interval” disables sending of echo requests for user plane and therefore CLI “path-failure detection-policy gtp echo” has no relevance – NO Path failure detection is done on GTPU
- GGSN still does respond to echo requests from the SGSN though
- “show gtpu stat” and “show demux-mgr stat gtpumgr” capture echo stats

Various Examples of path fail traps and explanations

Usually the peer restart counter will increase by one when there is a restart. In this case, a bug in PGW caused it to override the restart counter to the value of the RC of another peer (first trap), and so when the next packet arrived over the connection, it treated it as a restart (second trap) thinking that the RC had changed. The path then clears on any successive packet (third trap). The point here is not to know about a specific bug but rather to try and be aware of any counter changes that cannot be explained.

This example is from a GGSN where the peer SGSN has changed its RC from 66 to 67 via a Create PDP Context Request:

This example is from a PGW where the peer ePDG has changed its RC from 69 to 70 via a Create Session Request:

In this example, the output from various PGW nodes all connected to the same SGW show that the SGW stopped becoming reachable from the PGWs per an EGTPCPathFail trap at the various PGWs within a short time frame of eachother. Note that all show the same old restart counter as was discussed earlier that normally a node uses the same counter for all its peers. In the case of no response, the new restart counter displayed is 9, but the new counter might better be printed as undefined since the connection is down as of when the trap is triggered and so having a restart counter doesn't really make sense.

The following example shows a path failure and same new restart counter as in the previous example, but then a half hour later shows when the peer comes back up again and the counter has increased by one from 15 to 16. Over time the count increases for that peer:

```
Wed Jul 22 05:07:08 2015 Internal trap notification 1112 (EGTPCPathFail) context XGWin, service
EGTP1, interface type pgw-ingress, self address 209.165.201.13,  peer address 198.51.100.162,
peer old restart counter 15,  peer new restart counter 15,  peer session count  34, failure
reason no-response-from-peer
Wed Jul 22 05:38:00 2015 Internal trap notification 1113 (EGTPCPathFailClear) context XGWin,
service EGTP1, interface type pgw-ingress, self address 209.165.201.13, peer address
198.51.100.162, peer restart counter 16,   peer session count  1, clear reason sessions-update

[local]PGW> show egtpc peers
Wednesday July 22 10:04:30 UTC 2015

|||||  Service          Restart--------+  No. of
|||||  ID               Counter        |  restarts
|||||   |                              |   |   Current       Max
vvvvv   v              Peer Address    v   v   sessions    sessions
-----  --- ------------------------------------ --- --- -----------
AESKS   5               198.51.100.162  16   1        1157        2186

[XGWin]PGW> egtpc test echo gtp-version 2 src-address 209.165.201.13 peer-address 198.51.100.162

Wednesday July 22 10:52:41 UTC 2015
EGTPC test echo
---------------
Peer: 198.51.100.162                          Tx/Rx:  1/1  RTT(ms): 51   (COMPLETE) Recovery: 16
(0x10)
```

This example shows a repeating path failure due to **CPC restart** for 3 specific SGSN peers to the GGSN. The output below shows just one of the peers 209.165.200.225, but they all have the same issue. Repeated run of "show egtpc peers" and "show gtpc stat sgsn-address" also shows the alternating values. "show gtpc stat" confirms no issue with connectivity with respect to echo requests and responses, which in this case are only being initiated by the GGSN (acceptable as discussed earlier). One sidebar point to be made here is that there are often multiple pieces of data that can be collected that all corroborate to the same conclusion - one can choose to collect the data that is the most conclusive with the minimal amount of effort.

```
show snmp trap history verbose | grep -E "209.165.201.31|209.165.200.225|209.165.200.246"

Mon Jun 22 22:20:51 2015 Internal trap notification 1112 (EGTPCPathFail) context XGWin, service
GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225, peer
old restart counter 129,  peer new restart counter 132,  peer session count  3, failure reason
cpc-restart-counter-change
Mon Jun 22 22:20:51 2015 Internal trap notification 1113 (EGTPCPathFailClear) context XGWin,
service GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225,
peer restart counter 132,   peer session count  4, clear reason sessions-add
Mon Jun 22 22:20:55 2015 Internal trap notification 1112 (EGTPCPathFail) context XGWin, service
GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225, peer
old restart counter 132,  peer new restart counter 129,  peer session count  3, failure reason
cpc-restart-counter-change
Mon Jun 22 22:20:55 2015 Internal trap notification 1113 (EGTPCPathFailClear) context XGWin,
service GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225,
peer restart counter 129,   peer session count  4, clear reason sessions-add
Mon Jun 22 22:20:58 2015 Internal trap notification 1112 (EGTPCPathFail) context XGWin, service
GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225, peer
old restart counter 129,  peer new restart counter 132,  peer session count  3, failure reason
cpc-restart-counter-change
Mon Jun 22 22:20:58 2015 Internal trap notification 1113 (EGTPCPathFailClear) context XGWin,
service GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225,
peer restart counter 132,   peer session count  4, clear reason sessions-add
Mon Jun 22 22:21:02 2015 Internal trap notification 1112 (EGTPCPathFail) context XGWin, service
GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225, peer
old restart counter 132,  peer new restart counter 129,  peer session count  2, failure reason
cpc-restart-counter-change
Mon Jun 22 22:21:02 2015 Internal trap notification 1113 (EGTPCPathFailClear) context XGWin,
service GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225,
```

```
peer restart counter 129,   peer session count  3, clear reason sessions-add
Mon Jun 22 22:21:03 2015 Internal trap notification 1112 (EGTPCPathFail) context XGWin, service
GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225, peer
old restart counter 129,  peer new restart counter 132,  peer session count  4, failure reason
echo-rsp-restart-counter-change
Mon Jun 22 22:21:03 2015 Internal trap notification 1113 (EGTPCPathFailClear) context XGWin,
service GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225,
peer restart counter 132,   peer session count  5, clear reason sessions-add
Mon Jun 22 22:21:06 2015 Internal trap notification 1112 (EGTPCPathFail) context XGWin, service
GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225, peer
old restart counter 132,  peer new restart counter 129,  peer session count  2, failure reason
cpc-restart-counter-change
Mon Jun 22 22:21:06 2015 Internal trap notification 1113 (EGTPCPathFailClear) context XGWin,
service GGSN1, interface type ggsn, self address 209.165.202.129,  peer address 209.165.200.225,
peer restart counter 129,   peer session count  3, clear reason sessions-add

[XGWin]GGSN> show egtpc peers | grep 209.165.200.225
|||||  Service                      Restart No. of
|||||  ID                           Counter restarts
|||||   |                             |     |        Current Max
vvvvv   v             Peer Address   |     |       sessions    sessions
| | |
v v v
Monday June 22 23:09:26 UTC 2015
AESKG   6             209.165.200.225 129 86339           2          57

[XGWin]GGSN> show egtpc peers | grep 209.165.200.225
Monday June 22 23:09:28 UTC 2015
AESKG   6             209.165.200.225 129 86341           3          57

[XGWin]GGSN> show egtpc peers | grep 209.165.200.225
Monday June 22 23:09:29 UTC 2015
AESKG   6             209.165.200.225 132 86342           7          57

[XGWin]GGSN> show egtpc peers | grep 209.165.200.225
Monday June 22 23:09:30 UTC 2015
AESKG   6             209.165.200.225 132 86344           9          57

[XGWin]GGSN> show egtpc peers | grep 209.165.200.225
Monday June 22 23:09:31 UTC 2015
AESKG   6             209.165.200.225 129 86345          11          57

[XGWin]GGSN> show egtpc peers | grep 209.165.200.225
Monday June 22 23:09:32 UTC 2015
AESKG   6             209.165.200.225 129 86345           7          57

[XGWin]GGSN> show gtpc statistics sgsn-address 209.165.200.225
Monday June 22 23:07:36 UTC 2015
SGSN Address:    209.165.200.225    Status:          Active
Total Restarts:  86287             Restart Counter: 129

Session Stats:
 Total Current:                 2   S6b Assume Positive:             0
 ...

Path Management Messages:
 Echo Request RX:               0   Echo Response TX:                0
 Echo Request TX:           22919   Echo Response RX:            22917
```

Looking at other peers in the subnet, they all show a RC = 132 all the time, while the broken peers alternate between 129 and 132 as just shown, which points towards these SGSN peers as potentially passing the wrong RC in the broken scenarios:

```
                            Restart
Counter
```

```
AESKG 6 209.165.200.225 129 85115 8          57  <==
IDNKG   6        209.165.200.226 132 21 0 3 AESKG 6 209.165.200.227 132 60 47 92 AESKG 6
209.165.200.228 132 53 24 59 AESKG 6 209.165.200.229 132 69 92 126 AESKG 6 209.165.200.232 132
66 21 30 AESKG 6 209.165.200.236 132 43 8 21 AESKG 6 209.165.200.238 132 49 18 32 AESKG 6
209.165.200.239 132 39 1 6 AENKG 6 209.165.200.240 132 17 1 5 AESKG 6 209.165.200.241 132 67 6
43 AESKG 6 209.165.200.242 132 56 22 29 AENKG 6 209.165.200.243 132 40 2 9 AENKG 6
209.165.200.246 129 17713 3          24  <==
AESKG   6        209.165.200.247 132  64          29          45
AESKG   6        209.165.200.250 132  62          42          56
```

Recall from earlier, another approach to issues like this is to check other GGSNs (or PGWs for LTE) to see if they have the same issue with those seemingly problematic SGSN peers, and in this case they do also. For GGSN3 the first run of the command showed values 132 for both peers but another run showed 129 for one of the peers. Multiple runs (not shown here for brevity) would show all GGSNs bouncing between the two RCs for those peers over time:

```
                              Restart
Counter
AESKG 6 209.165.200.225 129 85115 8          57  <==
IDNKG   6        209.165.200.226 132 21 0 3 AESKG 6 209.165.200.227 132 60 47 92 AESKG 6
209.165.200.228 132 53 24 59 AESKG 6 209.165.200.229 132 69 92 126 AESKG 6 209.165.200.232 132
66 21 30 AESKG 6 209.165.200.236 132 43 8 21 AESKG 6 209.165.200.238 132 49 18 32 AESKG 6
209.165.200.239 132 39 1 6 AENKG 6 209.165.200.240 132 17 1 5 AESKG 6 209.165.200.241 132 67 6
43 AESKG 6 209.165.200.242 132 56 22 29 AENKG 6 209.165.200.243 132 40 2 9 AENKG 6
209.165.200.246 129 17713 3          24  <==
AESKG   6        209.165.200.247 132  64          29          45
AESKG   6        209.165.200.250 132  62          42          56
```

Finally to confirm the issue is the SGSN changing the RC, monitor subscriber menu option **by SGSN peer** shows changing RCs. When it does change, all calls would be disconnected with **path-failure** as the disconnect reason. A screenshot of a packet capture also shows this:

```
[XGWin]GGSN> mon sub
Monday June 22 23:34:22 UTC 2015

      y) By SGSN IP Address

      IP Address: [209.165.200.225   ]


INBOUND>>>>>  23:35:32:688 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x1DB8 (7608)
...
                     IMSI: 300420078559902 Recovery: 0x81 (129)
...

***CONTROL*** 23:36:00:363 Eventid:10285
  Disconnect Reason: path-failure


INBOUND>>>>>  23:36:48:414 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x3E2D (15917)
...
                      IMSI: 300420125984926
                  Recovery: 0x84 (132)
...

INBOUND>>>>>  23:37:28:337 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x3517 (13591)
```

```
...
                        IMSI: 300420094205377
                    Recovery: 0x84 (132)
... Monday June 22 2015 INBOUND>>>>> 23:37:40:559 Eventid:47000(3) GTPC Rx PDU, from
209.165.200.225:2123 to 209.165.202.129:2123 (176) TEID: 0x00000000, Message type:
GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10) Sequence Number:: 0x4E47 (20039) ... IMSI: 300420194755472
Recovery: 0x81 (129)
... ***CONTROL*** 23:37:40:755 Eventid:10285 Disconnect Reason: path-failure
```

No.	Time	Source	Destination	Recovery	Info
42	17:50:03	SGSN	GGSN	129	Create PDP context request
63	17:50:03	SGSN	GGSN	129	Create PDP context request
68	17:50:04	SGSN	GGSN	129	Create PDP context request
69	17:50:04	SGSN	GGSN	129	Create PDP context request
70	17:50:04	SGSN	GGSN	132	Create PDP context request
94	17:50:06	SGSN	GGSN	129	Create PDP context request
157	17:50:11	SGSN	GGSN	132	Create PDP context request
164	17:50:12	SGSN	GGSN	132	Create PDP context request
166	17:50:12	SGSN	GGSN	129	Create PDP context request
167	17:50:12	SGSN	GGSN	132	Create PDP context request
169	17:50:13	SGSN	GGSN	132	Create PDP context request
171	17:50:13	SGSN	GGSN	132	Create PDP context request
195	17:50:14	SGSN	GGSN	129	Create PDP context request
209	17:50:16	SGSN	GGSN	132	Create PDP context request
249	17:50:17	SGSN	GGSN	129	Create PDP context request
397	17:50:18	SGSN	GGSN	132	Create PDP context request
398	17:50:18	SGSN	GGSN	129	Create PDP context request
436	17:50:20	SGSN	GGSN	129	Create PDP context request
490	17:50:22	SGSN	GGSN	129	Create PDP context request
495	17:50:23	SGSN	GGSN	129	Create PDP context request
688	17:50:26	SGSN	GGSN	132	Create PDP context request
756	17:50:27	SGSN	GGSN	132	Create PDP context request
772	17:50:30	SGSN	GGSN	132	Create PDP context request
774	17:50:31	SGSN	GGSN	132	Create PDP context request
779	17:50:31	SGSN	GGSN	132	Create PDP context request
855	17:50:32	SGSN	GGSN	129	Create PDP context request

This example shows a path failure where the problem was actually in the path and not the PGW or SGW. As a result, the RC remains the same when path re-establishes. In the second peer example the clear is a result of echo request establishment:

```
[XGWin]GGSN> mon sub
Monday June 22 23:34:22 UTC 2015

      y) By SGSN IP Address

      IP Address: [209.165.200.225   ]


INBOUND>>>>>  23:35:32:688 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x1DB8 (7608)
...
                      IMSI: 300420078559902 Recovery: 0x81 (129)
...

***CONTROL*** 23:36:00:363 Eventid:10285
```

```
  Disconnect Reason: path-failure


INBOUND>>>>>  23:36:48:414 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x3E2D (15917)
...
                        IMSI: 300420125984926
                    Recovery: 0x84 (132)
...

INBOUND>>>>>  23:37:28:337 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x3517 (13591)
...
                        IMSI: 300420094205377
                    Recovery: 0x84 (132)
... Monday June 22 2015 INBOUND>>>>> 23:37:40:559 Eventid:47000(3) GTPC Rx PDU, from
209.165.200.225:2123 to 209.165.202.129:2123 (176) TEID: 0x00000000, Message type:
GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10) Sequence Number:: 0x4E47 (20039) ... IMSI: 300420194755472
Recovery: 0x81 (129)
... ***CONTROL*** 23:37:40:755 Eventid:10285 Disconnect Reason: path-failure
```

This example shows another anomaly where repeatedly after the old restart counter has reset to 2, it somehow ends up at 0 again when the path fail clears. This turned out to be a bug. Again the lesson here is if something appears odd, it should be reported.

```
[XGWin]GGSN> mon sub
Monday June 22 23:34:22 UTC 2015

      y) By SGSN IP Address

      IP Address: [209.165.200.225   ]


INBOUND>>>>>  23:35:32:688 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x1DB8 (7608)
...
                      IMSI: 300420078559902 Recovery: 0x81 (129)
...

***CONTROL*** 23:36:00:363 Eventid:10285
  Disconnect Reason: path-failure


INBOUND>>>>>  23:36:48:414 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x3E2D (15917)
...
                        IMSI: 300420125984926
                    Recovery: 0x84 (132)
...

INBOUND>>>>>  23:37:28:337 Eventid:47000(3)
GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176)
TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10)
Sequence Number:: 0x3517 (13591)
...
                        IMSI: 300420094205377
```

Recovery: 0x84 (132)

... Monday June 22 2015 INBOUND>>>>> 23:37:40:559 Eventid:47000(3) GTPC Rx PDU, from 209.165.200.225:2123 to 209.165.202.129:2123 (176) TEID: 0x00000000, Message type: GTP_CREATE_PDP_CONTEXT_REQ_MSG (0x10) Sequence Number:: 0x4E47 (20039) ... IMSI: 300420194755472

Recovery: 0x81 (129)

... ***CONTROL*** 23:37:40:755 Eventid:10285 Disconnect Reason: path-failure