batman-adv - Bug #413

B.A.T.M.A.N. V aggregation causes originator loops

07/23/2020 01:40 PM - Sven Eckelmann

Status:	In Progress	**Start date:**	07/23/2020
Priority:	Normal	**Due date:**	
Assignee:	Linus Lüssing	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

It was reported that some installations with enabled OGMv2 aggregation can cause the own originator to appear in the originator table via a different node:

```
My originator messages look wrong as I can see my host originator messages along with all the neig
bor nodes:

root@OpenWrt:/etc/config# batctl o -n
[B.A.T.M.A.N. adv 2020.1-openwrt-2, MainIF/MAC: mesh0/00:30:1a:4e:b8:26 (bat0/f2:07:f1:5f:e0:78 BA
TMAN_V)]
   Originator        last-seen ( throughput)  Nexthop           [outgoingIF]
 * 00:30:1a:4e:b8:18    0.570s (       86.7) 00:30:1a:4e:b8:2e [     mesh0]
   00:30:1a:4e:b8:18    0.570s (       21.6) 00:30:1a:4e:b8:18 [     mesh0]
 * 00:30:1a:4e:b8:2e    1.510s (      212.6) 00:30:1a:4e:b8:2e [     mesh0]
   00:30:1a:4e:b8:2e    1.510s (       38.9) 00:30:1a:4e:b8:18 [     mesh0]
   00:30:1a:4e:b8:26    1.510s (       38.9) 00:30:1a:4e:b8:18 [     mesh0]
 * 00:30:1a:4e:b8:26    1.510s (      108.9) 00:30:1a:4e:b8:2e [     mesh0]

root@OpenWrt:/etc/config# batctl n -n
[B.A.T.M.A.N. adv 2020.1-openwrt-2, MainIF/MAC: mesh0/00:30:1a:4e:b8:26 (bat0/f2:07:f1:5f:e0:78 BA
TMAN_V)]
IF             Neighbor               last-seen
00:30:1a:4e:b8:2e    0.490s (      179.0) [     mesh0]
00:30:1a:4e:b8:18    0.380s (       79.2) [     mesh0]
```

Disabling aggregation seems to solve this problem.

See also:

- https://lists.open-mesh.org/mailman3/hyperkitty/list/b.a.t.m.a.n@lists.open-mesh.org/thread/V6P773QWN54CYATHOBORRTE4VBM7T5WD/
- https://twitter.com/FreifunkMUC/status/1285605110157062150
- https://twitter.com/FreifunkMUC/status/1285864677138931713

History

#1 - 07/23/2020 01:43 PM - Sven Eckelmann

There was also following in the IRC channel:

```
[2020-06-17 12:29:46] <nLi> marec: Yes, I revied the code and found that aggreated OGM does not honor the Prel
iminary Check for OGMv2. -Own Message Check: If the originator address of the OGM is our own the message must
be silently dropped as this OGM originated from this node.
[2020-06-17 12:45:54] <nLi> marec: For now i disabled aggregated OGM for my deployment, which resulted in the
expected behaviour. I.E the node is no longer listing itself in the orginiator table.
[2020-06-17 12:55:31] <nLi> marec: I suppose that the the "Own Message Check" needs to be implemented in eithe
r the while loop in batadv_v_ogm_packet_recv or batadv_v_ogm_process for each OGM that was aggregated, any tho
ughts on your side?
```

#2 - 07/23/2020 03:24 PM - Sven Eckelmann

- Status changed from New to In Progress

- Description updated

This is the proof of concept patch from Linus: https://git.open-mesh.org/batman-adv.git/commit/0115502eab54a80f2c05884efce6ee164ed3cd9f

```
batman-adv: Fix own OGM check in aggregated OGMs

The own OGM check is currently misplaced and can lead to the following
issues:

For one thing we might receive an aggregated OGM from a neighbor node
which has our own OGM in the first place. We would then not only skip
our own OGM but erroneously also any other, following OGM in the
aggregate.

For another, we might receive an OGM aggregate which has our own OGM in
a place other then the first one. Then we would wrongly not skip this
OGM, leading to populating the orginator and gateway table with ourself.

Fixes: 667996ebeab ("batman-adv: OGMv2 - implement originators logic")
Signed-off-by: Linus LÃ¼ssing <linus.luessing@c0d3.blue>
---
 net/batman-adv/bat_v_ogm.c | 11 ++++++-----
 1 file changed, 6 insertions(+), 5 deletions(-)

diff --git a/net/batman-adv/bat_v_ogm.c b/net/batman-adv/bat_v_ogm.c
index 0f8495b9..731810ee 100644
--- a/net/batman-adv/bat_v_ogm.c
+++ b/net/batman-adv/bat_v_ogm.c
@@ -881,6 +881,12 @@ static void batadv_v_ogm_process(const struct sk_buff *skb, int ogm_offset,
            ntohl(ogm_packet->seqno), ogm_throughput, ogm_packet->ttl,
            ogm_packet->version, ntohs(ogm_packet->tvlv_len));

+    if (batadv_is_my_mac(bat_priv, ogm_packet->orig))
+        batadv_dbg(BATADV_DBG_BATMAN, bat_priv,
+               "Drop packet: originator packet from ourself\n");
+        return;
+    }
+
     /* If the throughput metric is 0, immediately drop the packet. No need
      * to create orig_node / neigh_node for an unusable route.
      */
@@ -1008,11 +1014,6 @@ int batadv_v_ogm_packet_recv(struct sk_buff *skb,
     if (batadv_is_my_mac(bat_priv, ethhdr->h_source))
         goto free_skb;

-    ogm_packet = (struct batadv_ogm2_packet *)skb->data;
-
-    if (batadv_is_my_mac(bat_priv, ogm_packet->orig))
-        goto free_skb;
-
     batadv_inc_counter(bat_priv, BATADV_CNT_MGMT_RX);
     batadv_add_counter(bat_priv, BATADV_CNT_MGMT_RX_BYTES,
                skb->len + ETH_HLEN);
```

#3 - 07/23/2020 03:31 PM - Sven Eckelmann

Btw. I would apply this patch :)