Technical Disclosure Commons

Defensive Publications Series

June 19, 2019

HEARTBEAT MECHANISM FOR SYNCHRONIZATION IN LOW-POWER AND LOSSY NETWORKS

Lele Zhang

Chuanwei Li

Huimin She

Feiliang Wang

HEARTBEAT MECHANISM FOR SYNCHRONIZATION IN LOW-POWER AND LOSSY NETWORKS

AUTHORS:
Lele Zhang
Chuanwei Li
Huimin She
Feiliang Wang

ABSTRACT

Described herein are techniques to enable nodes to determine whether the Border Router (BR) is alive. These techniques leverage existing traffic, thereby avoiding wasting bandwidth resources such as periodic asynchronous messages to maintain Wireless Mesh Network (WMN) connectivity.

DETAILED DESCRIPTION

Smart grids require highly stable, reliable, and easily deployable communication technologies to support various types of electrical services and applications. The Connected Grid Mesh (CG-Mesh) network may provide Advantaged Metering Infrastructure (AMI) and Distributed Automation (DA) devices for industrial users in Wireless Automatic Meter Reading (WMAR) marketing. The CG-Mesh can support thousands of nodes to establish a Wireless Mesh Network (WMN) in a Personal Area Network (PAN) area. It is typically built as a tree-based topology according to Routing Protocol for Low power and Lossy Networks (LLNs) (RPL) as described in Internet Engineering Task Force (IETF) Request for Comments (RFC) 6550.

As illustrated in Figure 1 below, many nodes in the CG-Mesh are not connected to the Border Router (BR) directly. Most nodes can only communicate directly with their parent node. For example, if the red node breaks down, the purple node may require a long time to determine that it has detached from the PAN. Commonly, the transmission probability of a node is very sparse. Therefore, it is often too late for a terminal node to determine that it has detached.

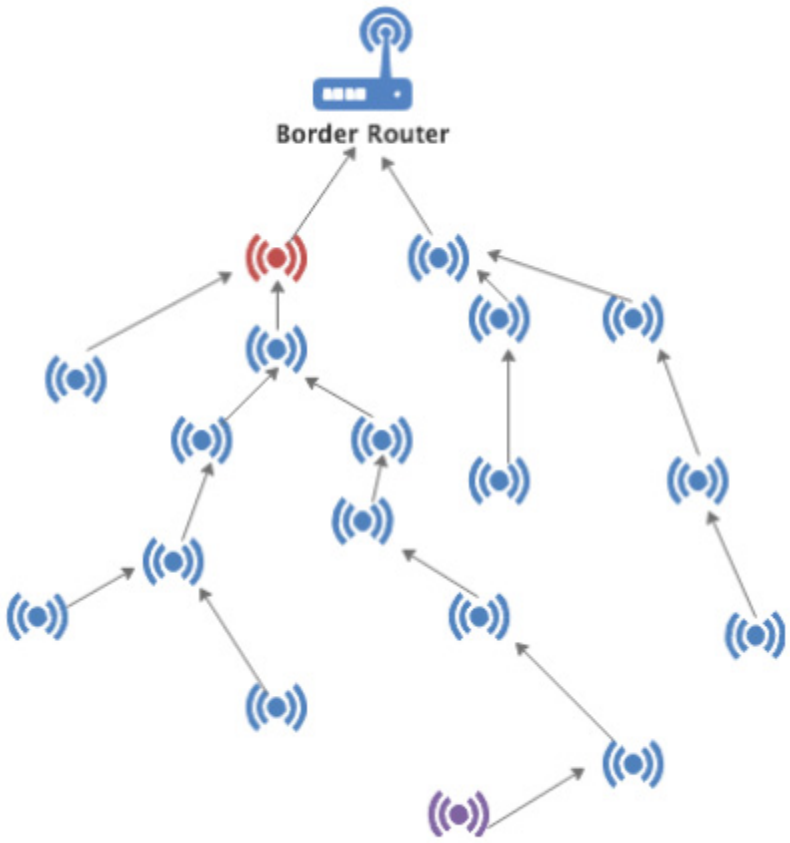

Figure 1

As shown in Table 1 below, the latency of the detection is cumulated by hops. For example, the purple node in Figure 1 requires more time than its parent or grandparent to detect detachment.

hop number	detection moment
1	10 minutes later
2	20 minutes later
...	...
n	10*n minutes later

Table 1

In order to avoid this problem, a heartbeat mechanism is currently adopted in the CG-Mesh. The BR propagates heartbeat information periodically within the scope of the PAN (e.g., four minutes per round). Once a node determines that it cannot update a

heartbeat for more than two rounds, it detaches from the PAN and attempts to rejoin. Thus each node has a fixed detection time which is at most twice the period of the heartbeat.

As illustrated in Figure 2 below, the current method is workable but adds too much redundant traffic to the network. As the heartbeat messages are expected to be received as often as possible, they are wrapped in asynchronous messages. Based on Institute of Electrical and Electronics Engineers (IEEE) 802.15.4, an asynchronous message needs to be spread in all channels and therefore requires a long time for propagation. Assuming a node works on a U.S. band with 50kbps, an asynchronous message may take more than four seconds, and thus each node devotes 1.8% (assuming four minutes per round) of its bandwidth maintaining the heartbeat. This not only wastes bandwidth resource, but also creates undesirable traffic storming among nodes.

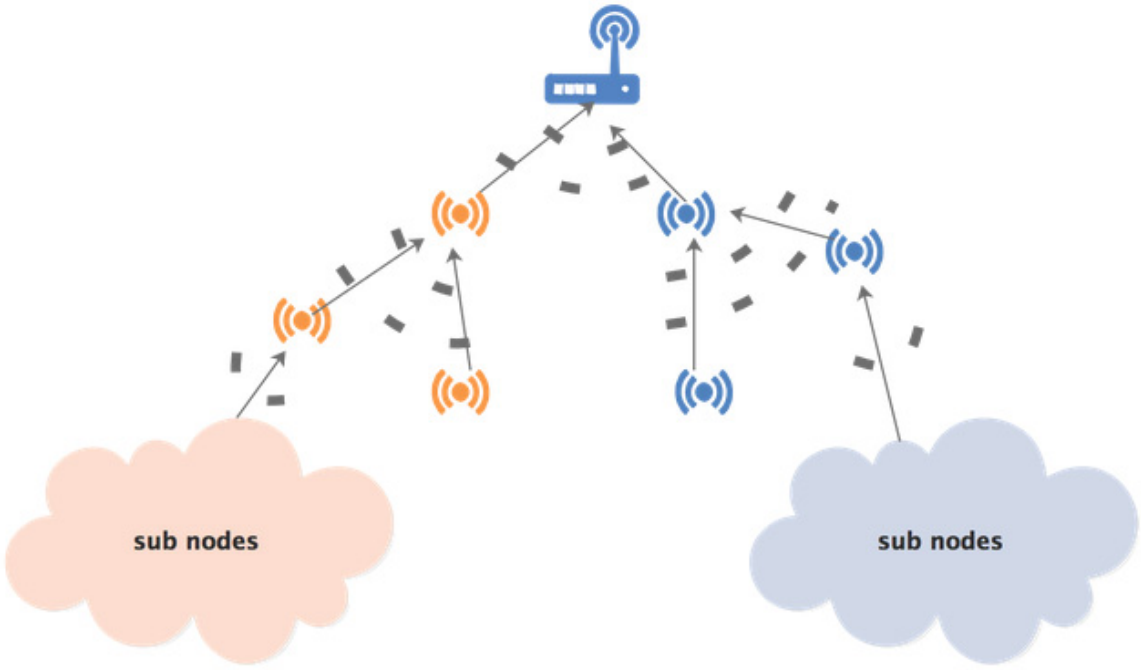

Figure 2

Accordingly, described are techniques to permit each node to detect connectivity with the BR easily and quickly while avoiding creation of periodic traffic. In one example, one node may determine that the BR is alive as long as it receives a packet from the BR.

As illustrated in Figure 3 below, as a member in the CG-Mesh, each node has an opportunity to communicate with the BR to exchange data. For example, due to the inherent instability of LLNs, the terminal node often switches its parent node depending

on link metrics (e.g., expected transmission count). The terminal node may then send a Destination Advertisement Object (DAO) frame to the BR to submit an update, and the BR responds with an ACK for that DAO. If the node receives the ACK successfully, it may determine that the BR is alive. Thus, as long as the red node receives the packets directly from the BR, the node may determine that the BR is alive.

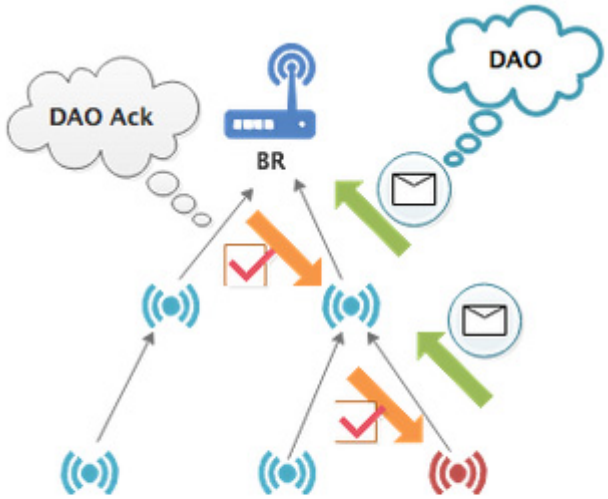

Figure 3

In another example, if one node receives a source routing packet but the destination is not the node, that node may nonetheless determine that the BR is alive. As illustrated in Figure 4, the BR wants to send a source routing packet to the green node via the red node. Although the red node knows this packet is not destined for the red node, the red node may nevertheless determine that the BR is alive based on the BR sending the packet.

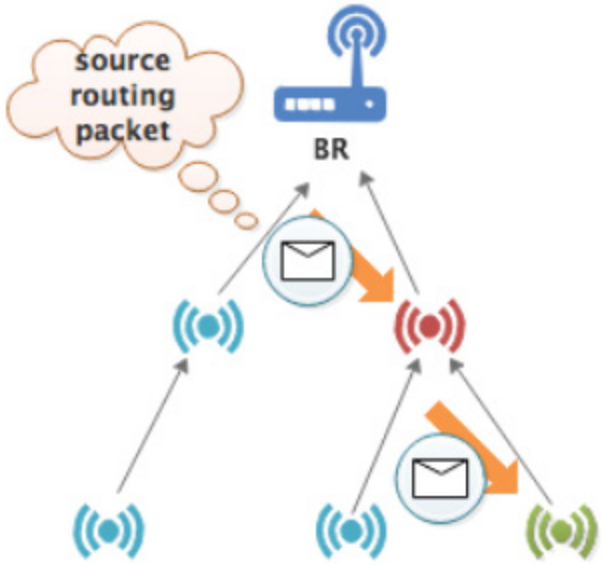

Figure 4

As illustrated in Figure 5 below, in a CG-Mesh implementation, the broadcast packets may only be generated by the BR. The other nodes simply forward the broadcast packets as they are received. Thus, the nodes may determine that the BR is alive if they receive a broadcast packet, such as firmware upgrading packets or Destination-Oriented Directed Acyclic Graph (DODAG) Information Object (DIO) packets with new versions.

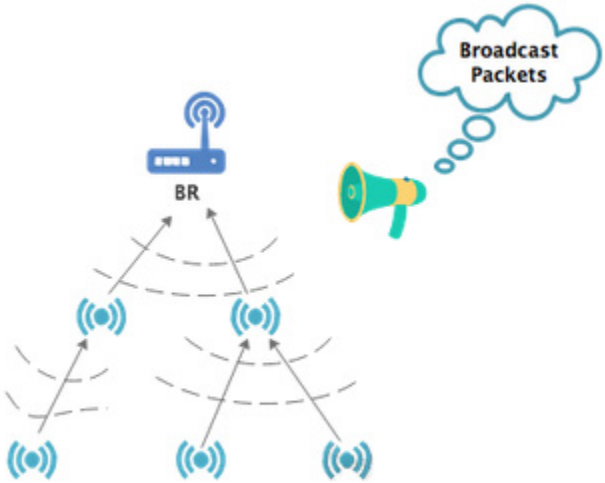

Figure 5

In another example, a node may determine that the BR is not alive if the aforementioned packets do not arrive in a sufficient amount of time. In a LLN like CG-Mesh, data is typically exchange infrequently. As such, some nodes may not receive any

packets for long periods of time for a variety of reasons. Some rules are provided herein to address such situations.

As illustrated in Figure 6 below, a constant time may be used as a threshold for connectivity between the BR and the node (e.g., 30 or 60 minutes). This threshold may be referred to as PAN_TIMEOUT. Each node may have a timer to count ticks since the last updated event (i.e., received any of the three aforementioned types of packets). Once it receives a new packet, the timer may be refreshed. If the timer exceeds half of PAN_TIMEOUT, it may request an update from its neighbors. As long as a node receives a request, it may reset a trickle timer to arrange a response schedule, which may send back the requester with its own counter number. A new information element (IE) may be added to carry the counter value, which may be integrated in a sync beacon. If one node receives a request for the sync beacon, it may respond with this IE. Other neighbors that have received a sync beacon with a smaller value may suppress transmission of the sync beacon and update its own counter record. Otherwise, it may spread its sync beacon in the neighborhood according to the trickle timer schedule in the neighborhood. If the timer value is larger than PAN_TIMEOUT, the node may determine that the BR is not alive and perform detach operations such as looking for a new parent to rejoin.

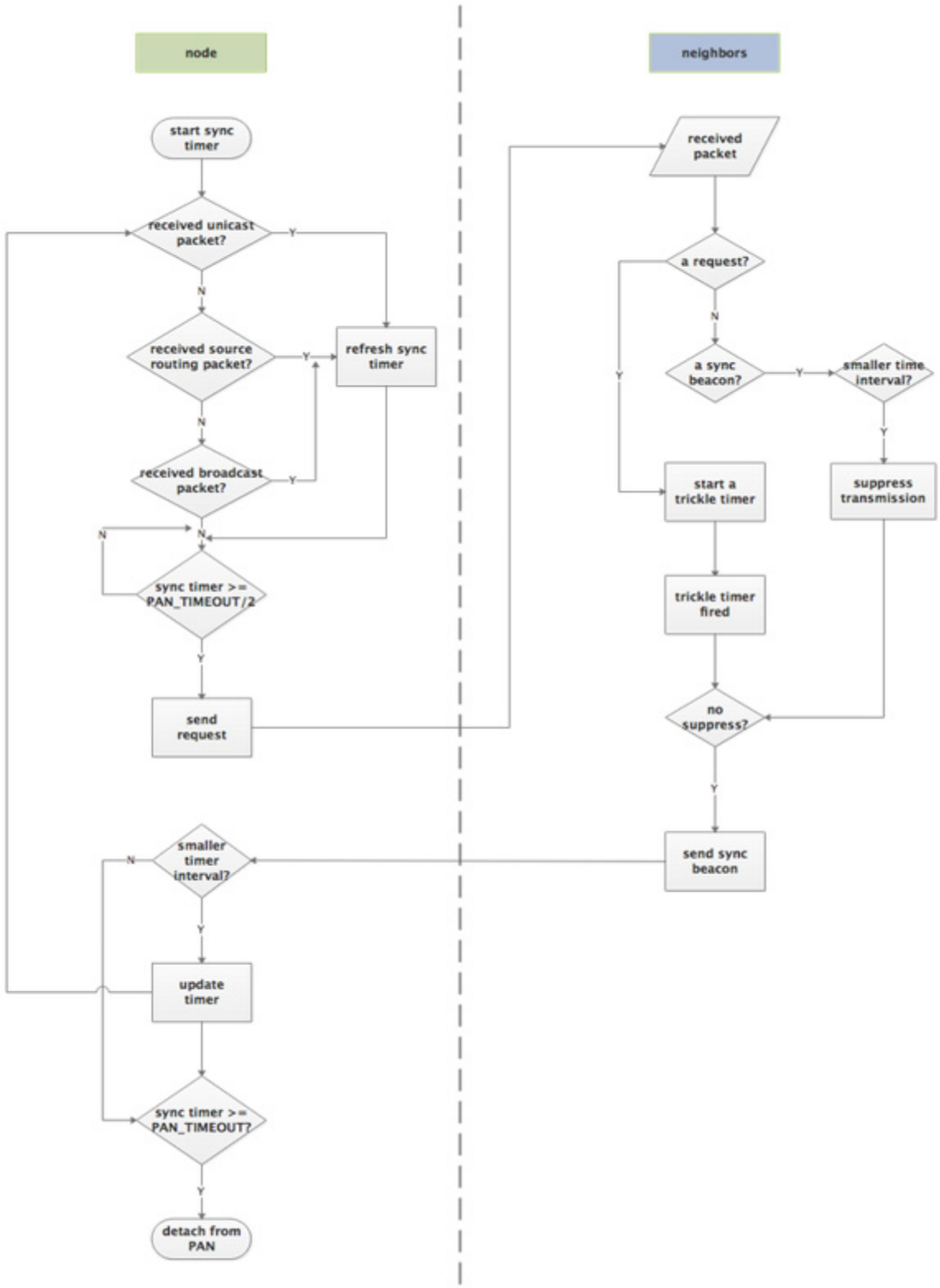
node
neighbors
start sync timer
received unicast packet?
Y
N
received source routing packet?
Y
refresh sync timer
N
received broadcast packet?
Y
N
N
sync timer >= PAN_TIMEOUT/2
Y
send request
received packet
a request?
N
Y
a sync beacon?
Y
smaller time interval?
Y
start a trickle timer
suppress transmission
trickle timer fired
no suppress?
Y
send sync beacon
smaller timer interval?
N
Y
update timer
sync timer >= PAN_TIMEOUT?
Y
detach from PAN

Figure 6

For the whole WMN, the latest update event is infectious. Other nodes gradually sync to neighbors with the same smallest counter value. If all nodes do not receive the updates for a long time (e.g., PAN_TIMEOUT/2), they may try to request neighbors for the latest updates according to a different schedule. However, the nodes near the BR may always receive the update, because the BR is required to respond to these kind of requests with its current time. Thus, all other nodes under this BR may at last synchronize with the update hop by hop.

There are two ways to keep nodes "alive" in LLNs: active mode and passive mode. In active mode, maintenance messages are periodically transmitted regardless of whether the messages proceed from node to BR (e.g., trigger DAO transmission via a DAO trickle timer) or from BR to node (e.g., trigger DIO transmission via a DIO trickle timer). Either action increases routine throughput in LLNs. Thus, the goal is to reduce the frequency of these transmissions but still have sensitive awareness for each node to determine whether it is attached in a PAN. In passive mode, the node determines from normal traffic that the BR is alive. The node may analyze all the packets that it receives by some simple but useful rules.

It may be undesirable to use DAO and DIO mechanisms in RPL too often in WMN for maintenance, in order to provide more time for regular traffic (e.g., application traffic generated by users). As such, the node may send the DAO actively only if it changes the preferred parent node rather than a trickle timer firing.

Additionally, it may be desirable to minimize the frequency of DIO messages (e.g., thirty minutes or one hour per round). That node may determine that the BR is alive faster than the DIO interval (e.g., eight minutes or less). This requirement creates a paradox: shortening the intervals of DIO or DAO is useful, but neither is expected due to increasing traffic.

The theoretical basis for determining whether the BR is alive is that one of the neighbors always has at least a smaller sync age (less than a threshold) for each node in the WMN. This can be proven through mathematical induction. Sync age is a metric to measure how much time has elapsed since the BR "alive" information was last obtained. In current CG-Mesh WMN, a trickle timer called "sync beacon" may be used to spread sync information periodically around its neighborhood. As described herein, "sync age" is

injected into the regular sync beacon to notify all entities in the neighborhood. If the sync age of one node exceeds the threshold (e.g., PAN_TIMEOUT/2), it may request an update from neighbors having a smaller sync age.

The known conditions are as follows.

1. n is defined as the hop number for a node, so p(n) presents a node with hop n. For example, p(0) denotes BR, p(1) denotes hop 1 and p(n) denotes hop n.
2. The trickle timer of a request sync beacon has an initial offset, which is I = *a*hop + b*sync_age + jitter*, where a and b are parameters to adjust the weight (e.g., 10, 20, etc.).

The following is a proof which shows that one of the neighbors always has at least a smaller sync age (less than threshold) for each node in WMN.

1. When n = 1, p(1) has a neighbor who always has smaller sync age, which is p(0) (i.e., this is BR), so this proposition is always true.
2. Assume when n = k, this proposition is true, which means p(k) always has a neighbor p(k-1) with smaller sync age.
3. When n = k+1, p(k+1) request a refreshment of the sync age from p(k) when its sync age is larger than a threshold. If p(k)'s sync age is smaller than the threshold, then the proposition is true. Otherwise, p(k) will pull an update from p(k-1) before p(k+1) sends the request to itself, because p(k)'s trickle timer will be fired prior to p(k+1). Step 2) proves that p(k-1) has a smaller sync age than p(k) (less than a threshold).

   Therefore, this proposition is true.

While siblings and preferred parents have a desired sync age with greater likelihood, the techniques described herein are intended to address passive mode to allow nodes to determine the "alive" status of the BR. This scenario is therefore not unrealistic.

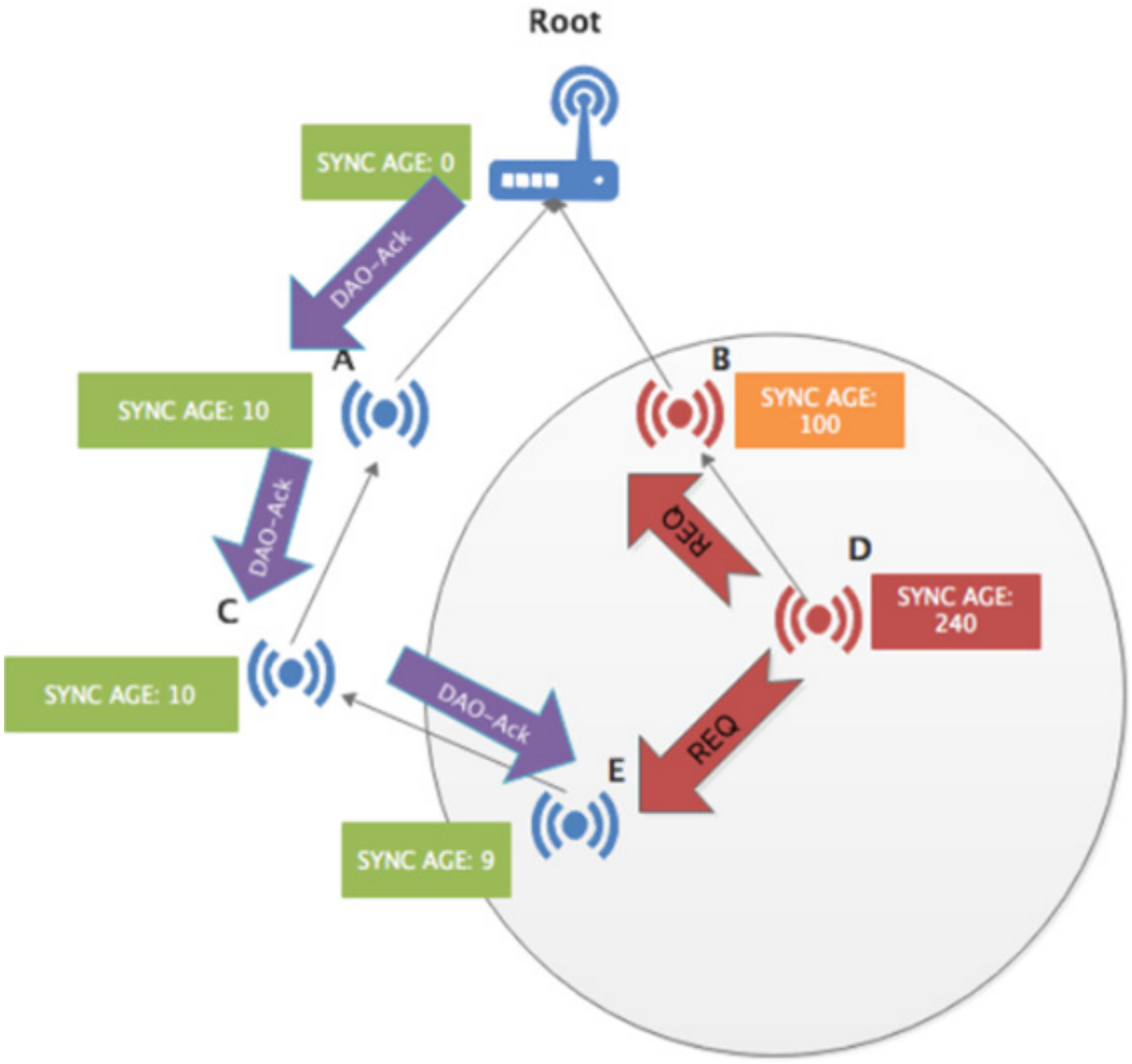

Figure 7

As illustrated in Figure 7 above, nodes B (i.e., p(1)) and E (i.e., p(3)) are neighbors of node D (i.e., p(2)). Node B is the preferred parent of node D, and node E is reachable to node D but not its child. At one point in time, node E has sync age of nine because it received a DAO-Ack nine seconds before. Node B's sync age is 100, and node D's sync age is 240, which exceeds the threshold. Accordingly, node D needs to pull an update from the neighbors.

If it pulls an update from preferred parent node B, the sync age will be 100. This could work, but 140 seconds later it would need to pull again if there is no notification with a smaller sync age from its neighbors. On the other hand, if it propagates the request within its neighborhood, node E may contribute a smaller sync age (nine) to node D, which may delay the next request for node D if needed. Otherwise, if node B can hear the notification

from node E, it may sync up with sync age nine as well, which may also delay the frequency of the request for node B.

Therefore, techniques described herein spread the request throughout the neighborhood area rather than to the preferred parent. This is more efficient, and reduces traffic among neighbors. Request messages are not forwarded. This is localized in a small scale (e.g., no more than hop 1) and therefore does not lead to storming. Once a node has received a request from its neighbor, it may trigger a trickle timer to send a sync beacon with its sync age. The trickle timer may have an offset which depends on its sync age and depth in the RPL tree. The node which has a smaller sync age may intend to be triggered first with high probability, and may suppress other neighbors' transmissions in the meantime. The other neighbors, and not the requester, may update its sync age as long as it finds that the sync age in the received beacon frame is smaller than its own sync age. This may also save traffic.

In summary, described herein are techniques to enable nodes to determine whether the BR is alive. These techniques leverage existing traffic, thereby avoiding wasting bandwidth resources such as periodic asynchronous messages to maintain WMN connectivity.