May 10, 2018

HARMONIOUS MULTICAST RETRANSMISSION FOR LOW-POWER AND LOSSY NETWORKS FIRMWARE UPGRADING

Charlie Chen
Cisco Systems, Inc.

Li Yang
Cisco Systems, Inc.

Chuanwei Li
Cisco Systems, Inc.

Feiliang Wang
Cisco Systems, Inc.

HARMONIOUS MULTICAST RETRANSMISSION FOR LOW-POWER AND LOSSY NETWORKS FIRMWARE UPGRADING

AUTHORS:
Charlie Chen
Li Yang
Chuanwei Li
Feiliang Wang

CISCO SYSTEMS, INC.

ABSTRACT

Techniques are described herein for composing multicast retransmissions harmoniously by prioritizing transmission of new messages and regulating retransmission of old messages. Only the same multicast messages are forwarded simultaneously such that loss due to collision can be significantly reduced. By measuring the intervals of new messages and counting the duplicates, retransmissions are reasonably curbed with awareness of input rate and medium usage. This prevents the "domino effect" on a crowded channel when loss occurs. Moreover, self-silence mechanisms allow regular nodes to release channel resources for critical forwarders. Multicast retransmission is provided with guaranteed delivery rate, which is imperative for firmware upgrading in Low-Power and Lossy Networks (LLNs).

DETAILED DESCRIPTION

To perform large-scale firmware upgrading in Low-Power and Lossy Networks (LLNs), multicast is deemed a promising starting point to disseminate firmware blocks among mesh nodes. Minimizing the makespan of this upgrading procedure accelerates service recovery. To achieve this goal, many solutions have been proposed to enhance multicast delivery performance. Most of these solutions have so far been focused on source rate control, retransmission, or suppression schemes. Multicast Protocol for LLNs (MPL) uses proactive or on-demand multicast retransmission to combat packet loss. Meanwhile, trickle behavior is employed in MPL to suppress unnecessary retransmissions.

However, multicast in LLNs suffers packet loss due to crowded broadcast channels. Though rate control can help alleviate a mild channel jam, it could also result in low

utilization and fail to take effect when multicast retransmission exists. Since the medium would again be packed with retransmitted packets, loss rate due to collision remains high, even if Clear Channel Assessment (CCA) is enabled.

Figure 1 below illustrates a first example in which a new multicast and multicast retransmission occur concurrently. As shown, node A is broadcasting a new multicast packet while node C is retransmitting an old multicast packet. Node B would probably not detect either packet, and would need to issue a request for retransmission.

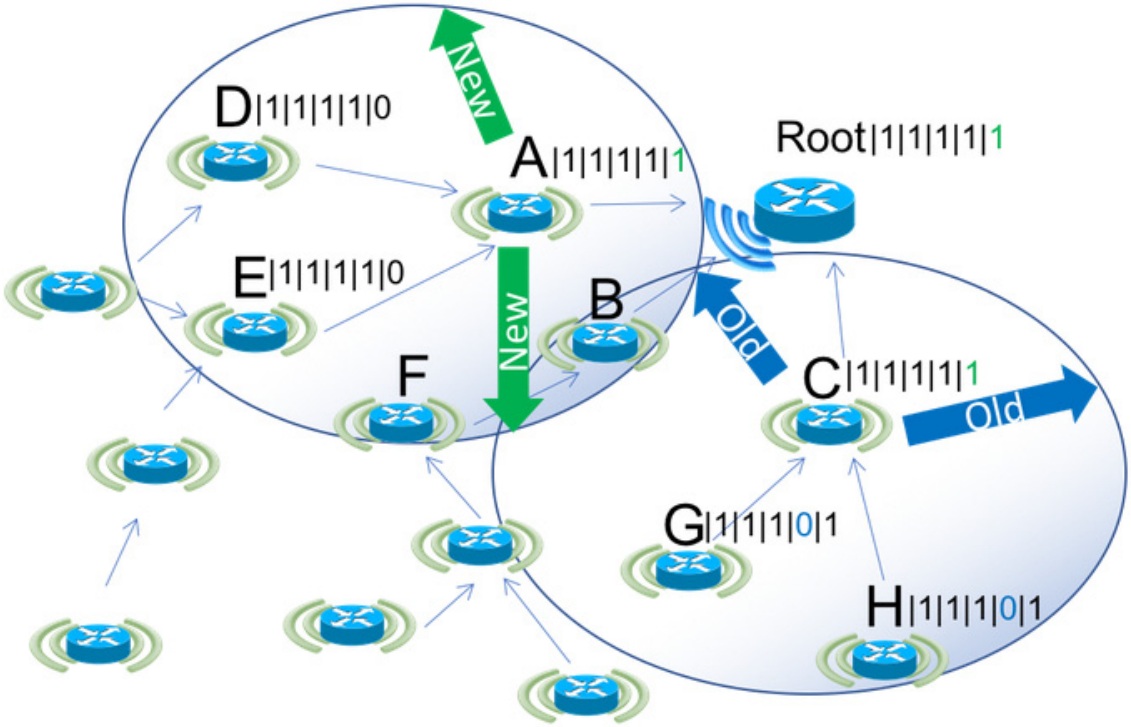

Figure 1

Figure 2 below illustrates how multicast retransmission can impact the propagation of new multicast messages. As shown, in multi-hop LLNs, it is difficult to propagate new multicast messages to deeper layers if retransmissions occur in a random manner.

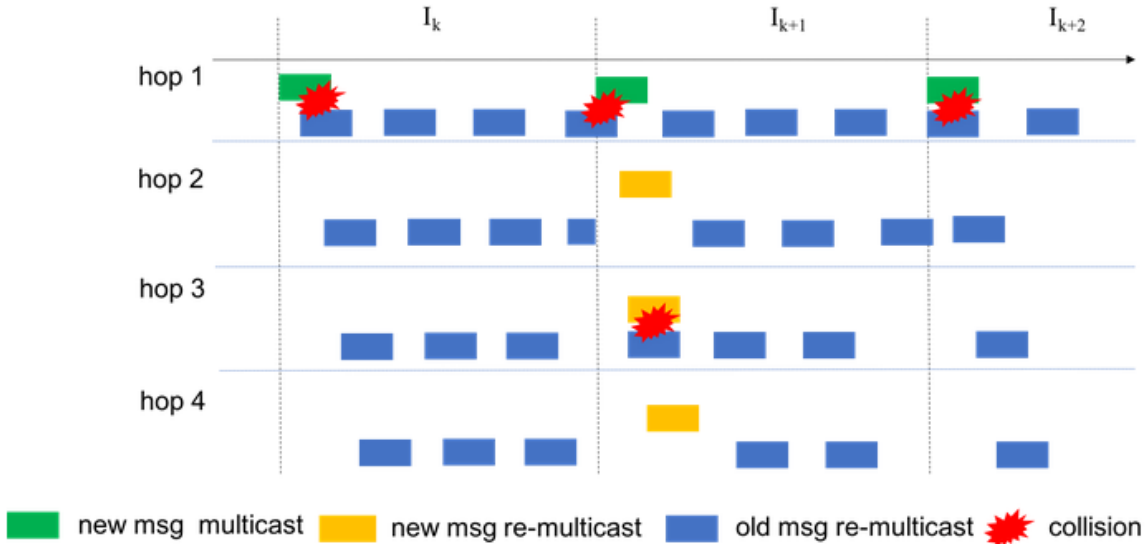

Figure 2

A simple suppress mechanism (e.g., incrementing Trickle *c* when a packet with the same sequence received) often leads to the halt of retransmission requested by other nodes. This highlights the scalability of multicast schemes for firmware upgrading in LLNs. Therefore, optimizing multicast retransmission together with awareness of input rate and medium usage is critical for improving delivery performance.

Accordingly, a multicast retransmission mechanism is provided to further boost LLN firmware upgrading. This mechanism regulates multicast retransmission behavior in LLNs for firmware upgrading. Previously**,** there was no differentiation between transmission of new multicast messages and retransmission of old multicast messages in LLNs. New and old multicast messages could coexist simultaneously, which can cause additional packet loss. Thus, the delivery rate of multicast with retransmission ($P_{retrans}$) easily fluctuates with external factors such as input rate and network topology, even lower than that of multicast without retransmission ($P_{no\_retrans}$). By contrast, as described herein, multicast transmissions can be composed in a harmonious way so that $P_{retrans} >= P_{no\_retrans}$ at an arbitrary input rate. This guarantees the benefit of multicast retransmission to accelerate firmware upgrading.

As illustrated in Figure 3 below, new multicast messages always have high priority for propagating through each hop. Retransmissions are arranged efficiently such that collisions due to retransmission are greatly reduced. Meanwhile, the number of requests as well as those of retransmissions during each interval are well-bounded.

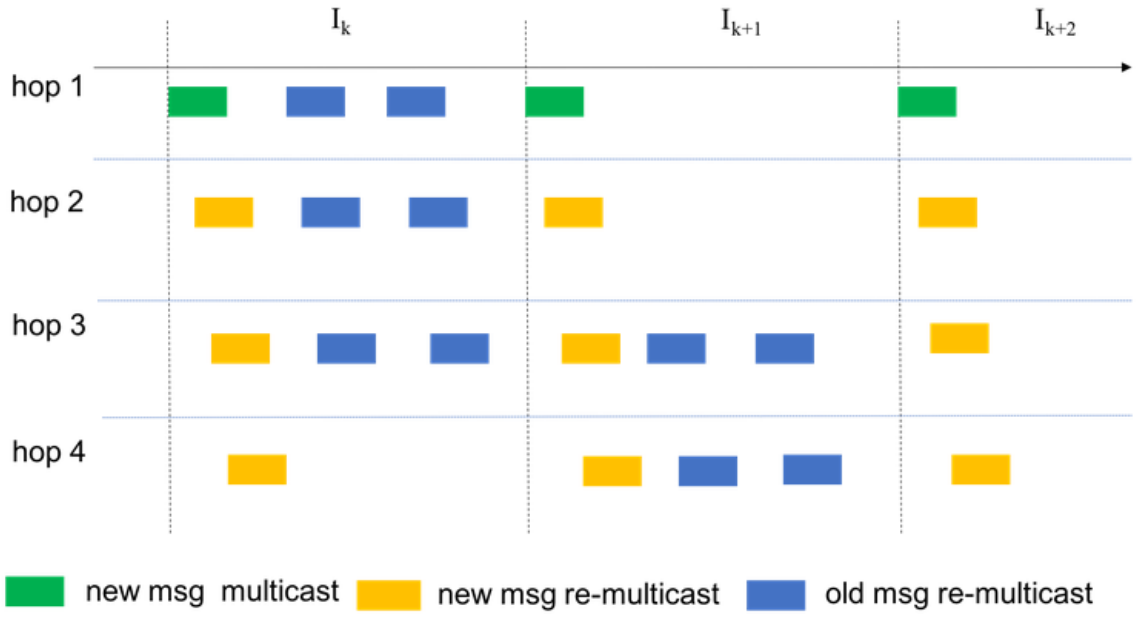

Figure 3

At least four operations may assist with this retransmission arrangement. The first operation relates to trickle delay. Usually the trickle timer uses an incrementing-c operation to delay or skip a transmission once. But the duration before the next transmission occurs long after doubling, which is inflexible for the transmission arrangement. Thus, a new operation *trickle_delay_fire*(*delta*) is introduced, where *delta* is a minimum delay before trickle fires again. Specifically, if *time_to_fire* < *delta,* increment *time_to_fire* by *delta*. Otherwise, *time_to_fire* stays the same.

The second operation relates to measurement of the interval of new multicast messages. When retransmissions follow a rhythm of new multicast messages, it is necessary to know the number of old multicast messages that can be filled between two consecutive new messages. Therefore, each node needs to measure the interval of new multicast messages, where Exponentially Weighted Moving Average (EWMA) can be used. With this average value, retransmission times can be reasonably estimated. The shorter the interval, the fewer retransmissions should be scheduled.

The third operation relates to a counter of duplicated new multicast messages. After a new multicast message is injected, it is to be forwarded by all nodes. Thus, each node can hear duplicated new messages from the other nodes. The counter of duplicated new messages during each interval indicates the density of the surrounding nodes. Here, the counter records the maximum value of all intervals. The larger the counter, the denser the surrounding nodes. Based on the counter, a node can adjust its possibility of sending a retransmission request since the other nodes may request the same missing packets. Thus, both request and retransmission can be reduced.

The fourth operation relates to the time of the last received request for retransmission. When a node receives a request for retransmission, the node's multicasts are required by at least one requester. It should then forward any retransmission received. However, if no request was received recently, a node should only forward the new multicast message in order to alleviate the channel jam. This value is recorded by each node to assist the retransmission determination.

These operations are used for the retransmission arrangement and performed in a distributed manner. The first operation prioritizes the transmission of new multicast messages by delaying retransmission of old multicast messages. The second operation allows each node to be aware of input rate in order to limit requests as well as retransmissions, which are further reduced in the third operation by considering the density of neighbors. The fourth operation helps a node to decide multicast forwarding in an efficient way.

Previously, basic multicast steps for firmware upgrading required a Network Management System (NMS) to periodically push firmware blocks to a Personal Area Network (PAN) through a Field Area Router (FAR), which encapsulated those blocks into MPL data messages with a bitmap. FAR and RPL nodes would use the proactive operation of MPL data messages to transmit blocks from the NMS, which carried its own updated bitmap. Each time a node received a new MPL data message, it updated its bitmap and then compared the bitmap inside the message with its own bitmap. The node unicasted its control message with latest bitmap to its preferred parent node if necessary. Each time a node received an MPL control message from its child node, it compared the bitmap with

its own bitmap and attempted to retransmit the old messages that were missed by its child node.

The techniques described herein may integrate additional operations. First, each time a node receives a new MPL data message, it uses a trickle delay operation to postpone all scheduled retransmissions by a short period (e.g., *data_message_imin*) that allows new data messages to first be transmitted. The average value of the interval is updated using the latest interval duration to estimate the maximum retransmission requests it can send during this interval. The maximum value of duplicated new multicast messages received is updated. The counter is reset to zero and duplicates are counted for a new interval.

Second, if a bitmap hole is found compared to the new data message's bitmap, it is determined whether the duration since the time of the last received request for retransmission exceeds a certain threshold. If so, the new message is not forwarded. Otherwise, the new message is forwarded. It is also determined whether the sent requests exceed the limit in this interval. If so, the transmission of the control message is skipped. Otherwise, the maximum value of duplicated new multicast messages is mapped to a probability. The larger the maximum value, the smaller the probability. This probability mapping can be pushed to nodes along with other configurations. Then, it is determined whether to send this control message according to the probability.

Third, each time a node receives a MPL control message, the time of the last received request is updated for retransmission. Fourth, each time a node receives an old MPL data message, it is determined whether the message fills any of its bitmap hole. If not, the message is dropped without forwarding. Otherwise, it is determined whether the duration since the time of the last received request for retransmission exceeds certain threshold. If so, the old message is not forwarded. Otherwise, the old message is forwarded.

With respect to the first operation, as illustrated in Figure 4 below, firmware blocks are propagated using MPL data messages with a bitmap through the root to RPL nodes. Node A and node C have received the latest data message. Node D and node H have bitmap holes, and might have already sent requests for retransmissions to node A and node C, respectively. Thus, node A and node C use trickle delay operation to make sure that there are no retransmissions before the transmission of this new message. They update the

average value of the interval and reset the counter of duplicated new multicast messages. As their MPL data message trickles fire, they forward the message to their children.

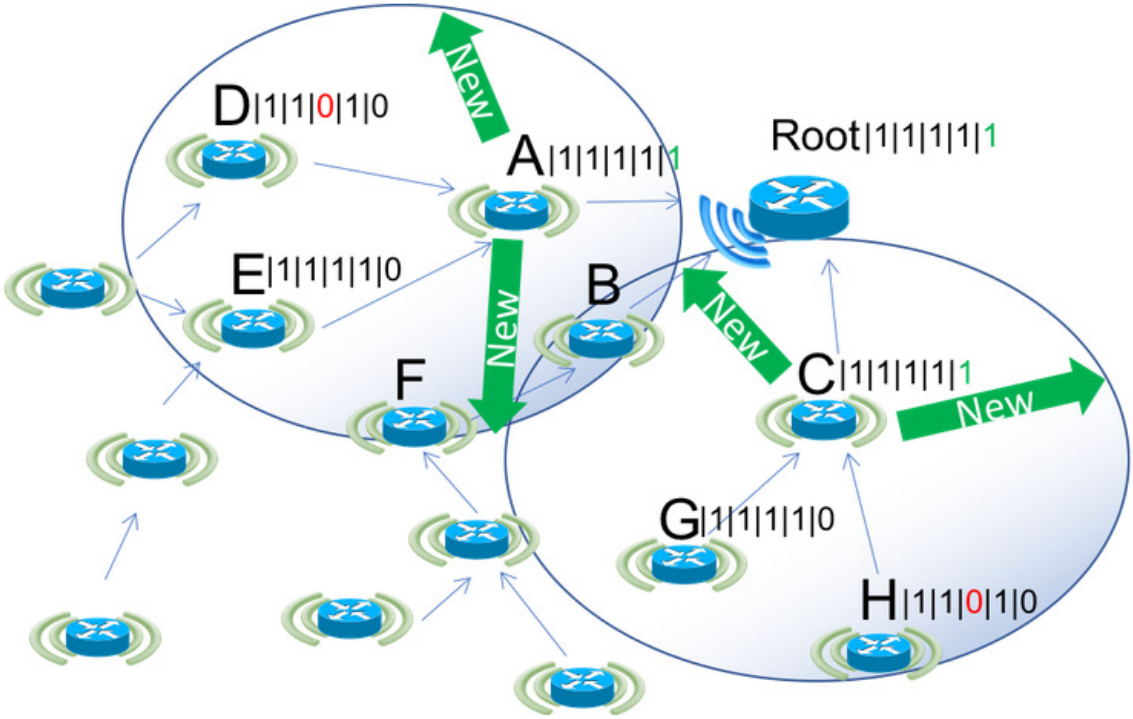

Figure 4

With respect to the second operation, as illustrated in Figure 5 below, children nodes such as node E and node G receive the new multicast message, repeat the first, second, and third operations, and then forward the message. However, for node D and node H, they have bitmap holes and have not been requested for retransmission recently. According to the new steps, they do not forward the new received multicast message. This self-silence behavior reduces unnecessary retransmissions and saves channel resources.

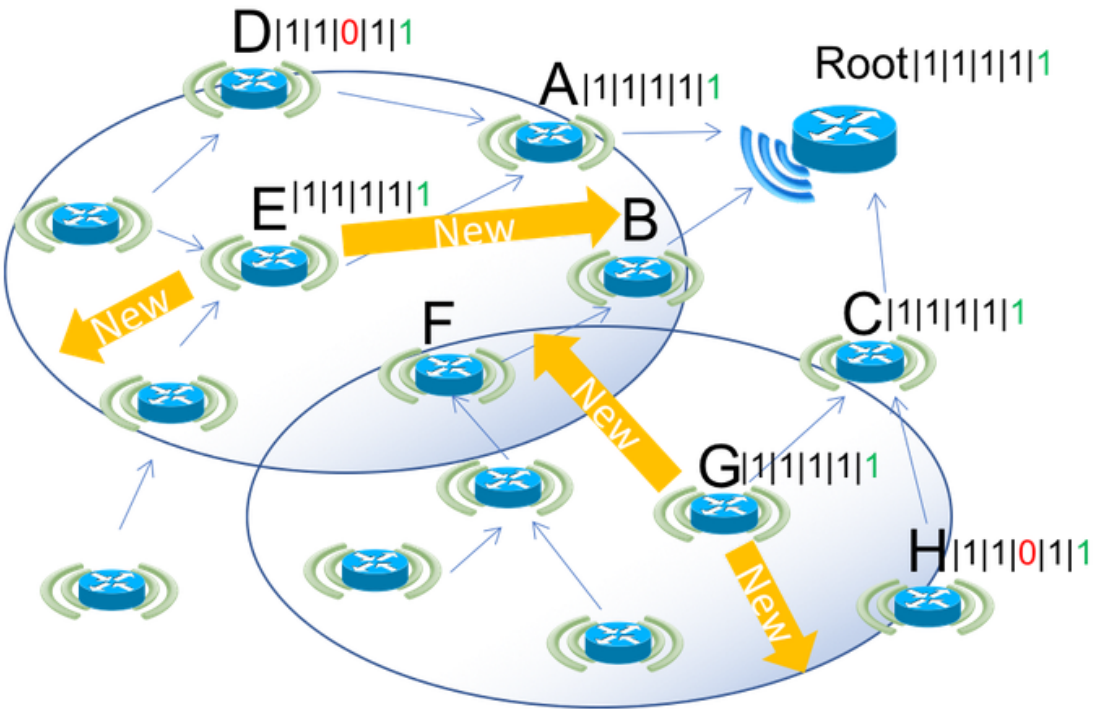

Figure 5

With respect to the third operation, as illustrated in Figure 6 below, as firmware upgrading proceeds, one message is missed by node D, node E, node G, and node H. Previously, the nodes would send out four MPL control messages to request for retransmission. But now, as described herein, each node first checks whether the request limit is reached and then decides whether to send the request using a random method based on the maximum duplicated new messages received. In this way, only node D and node H decide to send the unicast requests to node A and node C, respectively.

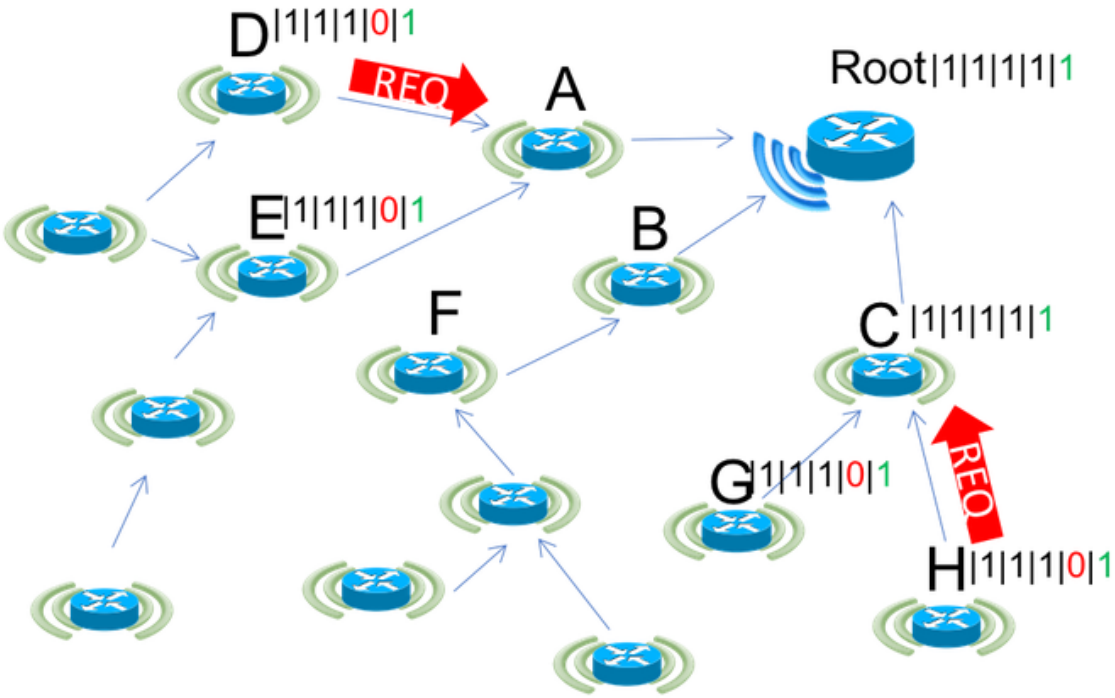

Figure 6

With respect to the fourth operation, as illustrated in Figure 7, when node A and node C receive the requests, they update the time of the last received request for retransmission and reset the trickle of the requested message. After the retransmission occurs, node D, node E, node G, and node H receive the missed old message. Then they examine whether they have been requested for retransmission recently before forwarding the old message.

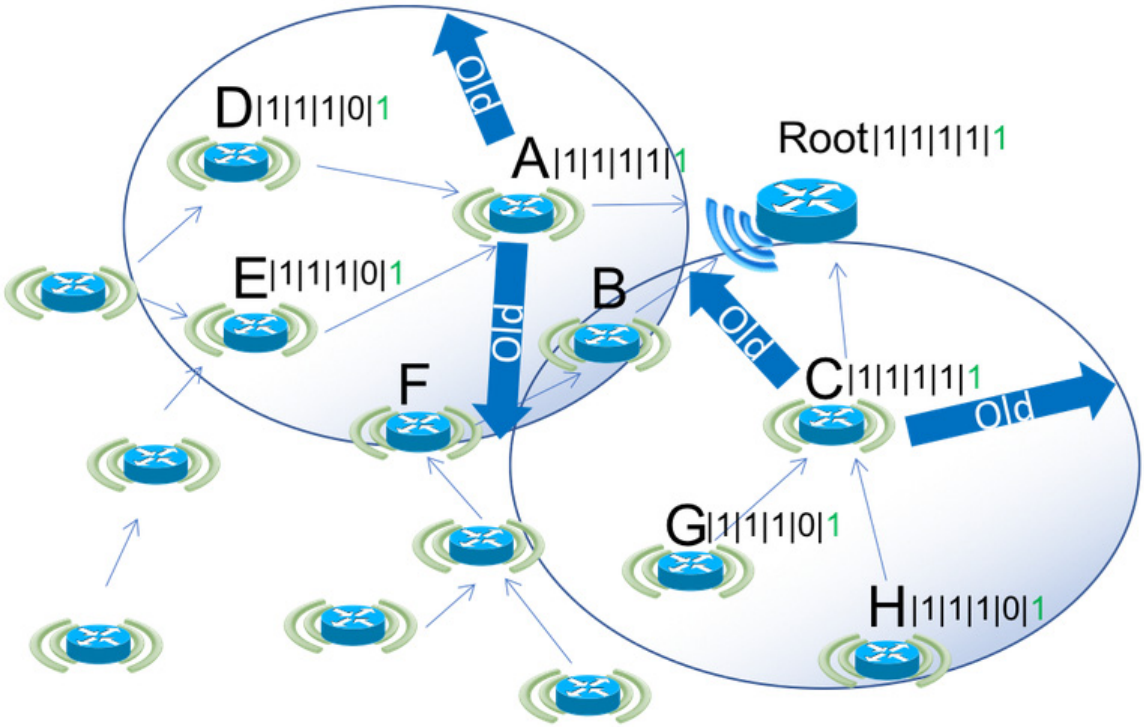

Figure 7

In summary, techniques are described herein for composing multicasts harmoniously by prioritizing transmission of new messages and regulating retransmission of old messages. Only the same multicast messages are forwarded simultaneously such that loss due to collision can be significantly reduced. By measuring the intervals of new messages and counting the duplicates, retransmissions are reasonably curbed with awareness of input rate and medium usage. This prevents the "domino effect" on a crowded channel when loss occurs. Moreover, self-silence mechanisms allow regular nodes to release channel resources for critical forwarders. Multicast retransmission is provided with guaranteed delivery rate, which is imperative for firmware upgrading in LLNs.