November 30, 2018

OPTIMIZING UPWARD LOAD BALANCING FOR LOSSY AND LOW-POWER NETWORKS

Lele Zhang

Yajun Xia

Yinfang Wang

Feiliang Wang

OPTIMIZING UPWARD LOAD BALANCING FOR LOSSY AND LOW-POWER NETWORKS

AUTHORS:
Lele Zhang
Yajun Xia
Yinfang Wang
Feiliang Wang

ABSTRACT

Presented herein are techniques for better load balancing of uplink flows at an overloaded node. The techniques presented herein leverage the capacity of neighboring nodes in order to share the traffic from the overloaded node. As a result, after several rounds of sharing, the probability of congestion decreases exponentially.

DETAILED DESCRIPTION

Connected Grid Mesh (CG-Mesh) networks are composed of up to millions of wireless nodes and can support various applications, such as Advanced Metering Infrastructure, Distribution Automation, *etc*. CG-Mesh networks follow the RPL (RFC6550) protocol to form a tree based route, where each node has a preferred parent and several candidates per Excepted Transmission Count (ETX). In Lossy and Low-power Networks (LLNs), each node is able to independently generate a small quantity of packets. However, because a CG-Mesh network is a converged network, the upward throughput for a node is the sum of all its children's traffic and its own generated packets.

Figure 1, below, is a schematic diagram illustrating an example because CG-Mesh network.

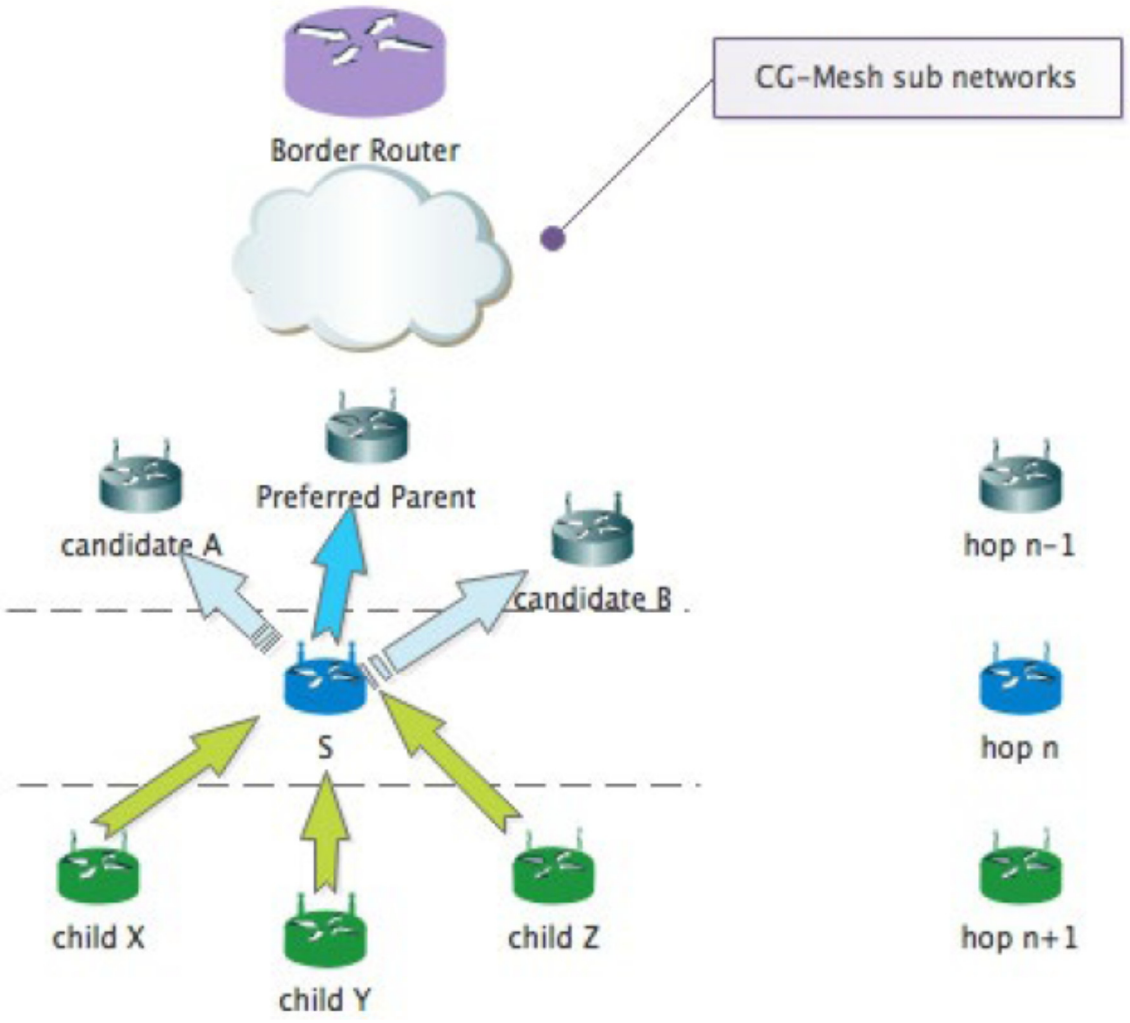

Figure 1

In Figure 1, blue node S has several children nodes (e.g., nodes X, Y, and Z). At the same time, node S has a preferred parent node, as well as two candidate nodes. As such, the upward throughput for node S contains all of the packets generated by nodes S, X, Y and Z. In some cases, this throughput is too large for the preferred parent node, thus the ETX will be increased in response. Once node S detects one candidate's ETX is lower than that of the current preferred parent, node S chooses this candidate device as its new preferred parent node. All traffic of node S will turn to the new parent until the ETX is larger than another candidate.

Therefore, CG-Mesh networks, such as the example CG-Mesh network of Figure 1, often encounter several problems following an upward torrent. These problems include:

1. ETX incrementation depends on packet drop probabilities. Because ETX updates are slow to occur, there must be a large number of invalid retransmission packets before a node changes its parent node. This is significant waste for leveraging network bandwidth.
2. Pendulum effect. Because node S has a large number of packets to upload, congestion with its preferred parent node is likely to occur. Thus, node S will switch among its potential parents.
3. Node S has a lot of traffic to transmit, thus it attempts to out-compete it its sibling nodes. If a sibling nodes is out-competed by node S, then its ETX increases in response. As a result, the congestion spreads.

Presented herein are techniques to optimize the load balancing of upward traffic in CG-Mesh networks, thereby reducing the congestion risk and making the mesh network topology more stable. More particularly, the techniques presented herein apply a strategy that is akin to bill sharing. For example, a group of people having a dinner in a restaurant and each person could only afford to spend 50 dollars ($50). So, if one person's bill exceeds $50, he or she will split some to a preferred neighbor. If the remaining part still exceeds $50, he or she will continue splitting, and the same for the neighbors. As described below, the techniques presented herein are sometimes referred to as including two phases, referred to as "Phase 1" and "Phase 2." A description of Phase 1, below, is followed by a description of Phase 2.

Phase 1

Each node of a CG-Mesh netwrk may have multiple sub-nodes as its children, thus the node could count the amount of packets from each child. The techniques presented herein propose to place these packets into respective queues according to source child nodes. There is a rule to generate 3 kinds of queues:

1. The queue, which has longest length, is be marked as Qmax.
2. The other queues are marked as Qnormal.

3. A special queue is marked as Qtransfer. Detailed usage of Qtransfer will be described below.

Figure 2, below, illustrates Qnormal, Qmax, and Qtransfer.

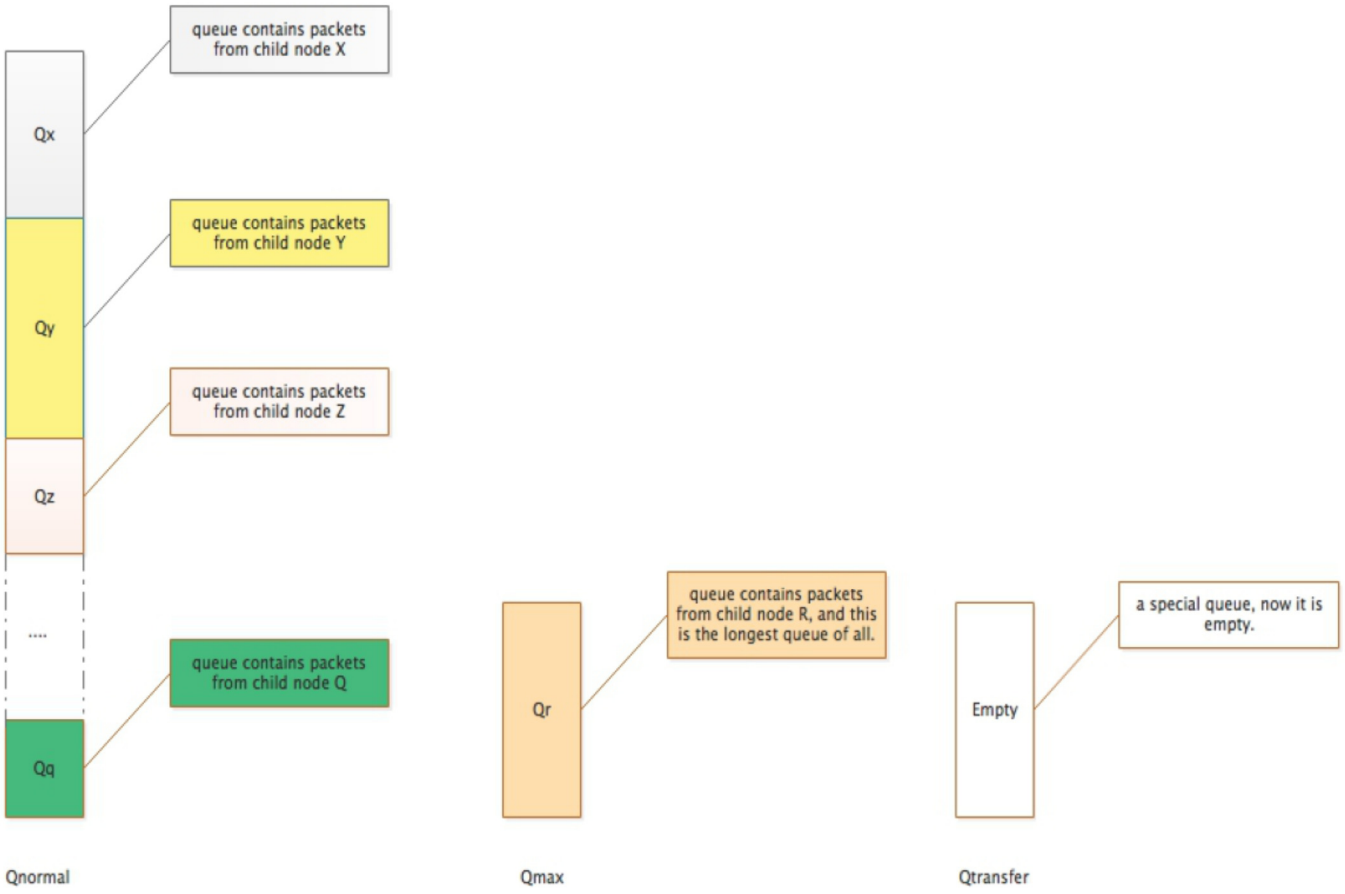

Figure 2

There are some known methods to detect congestion, including free buffer depth, retransmission amount and CCA detection failures. In the example of Figure 1, once node S is determined to be experiencing congestion, a method to generate $Q_{transter}$ is indicated as below:

1. Node S marks Q_{max} as $Q_{transfer}$.
2. Node S selects the maximum number of queue from Q_{normal}, and then sets it as new Q_{max}.

Thereby, as shown in the following Figure 3, node S now generates all three kinds of queues.

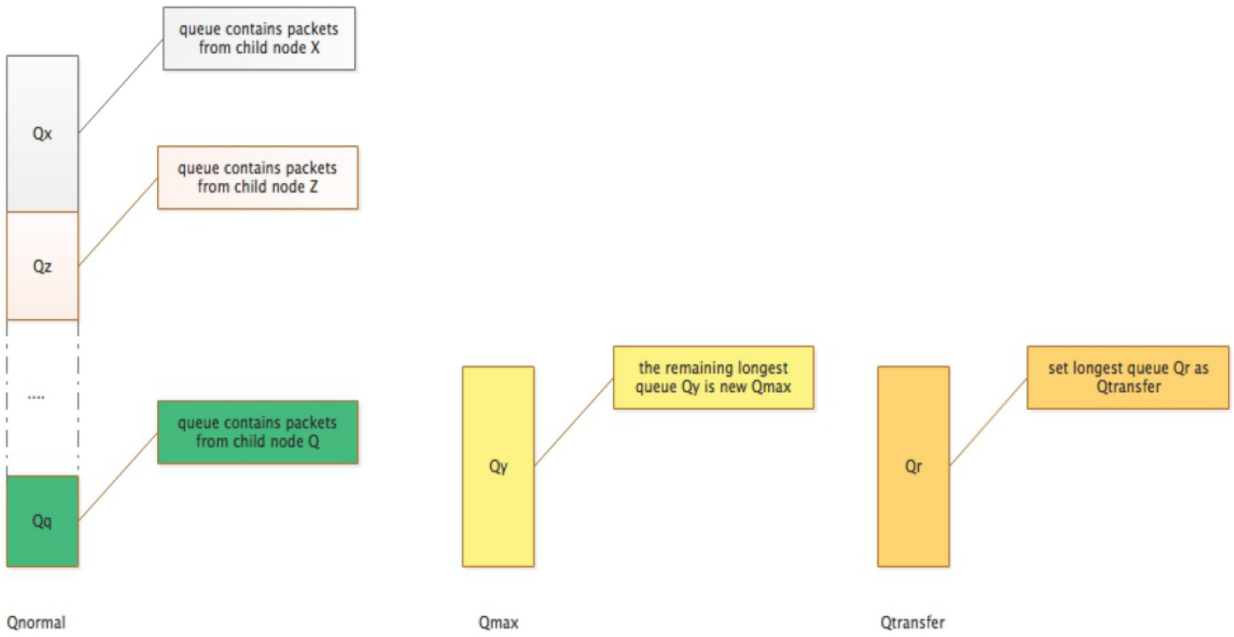

Figure 3

If node S is still congested after it sends $Q_{transfer}$ packets to the backup parent, more packets are required to be move to $Q_{transfer}$. Thus, as shown in the following Figure 4, node S merges current Q_{max} into $Q_{transfer}$, and then fetches the longest length queue from Q_{normal} as new Q_{max}. Node S performs this loop until the congestion disappears.

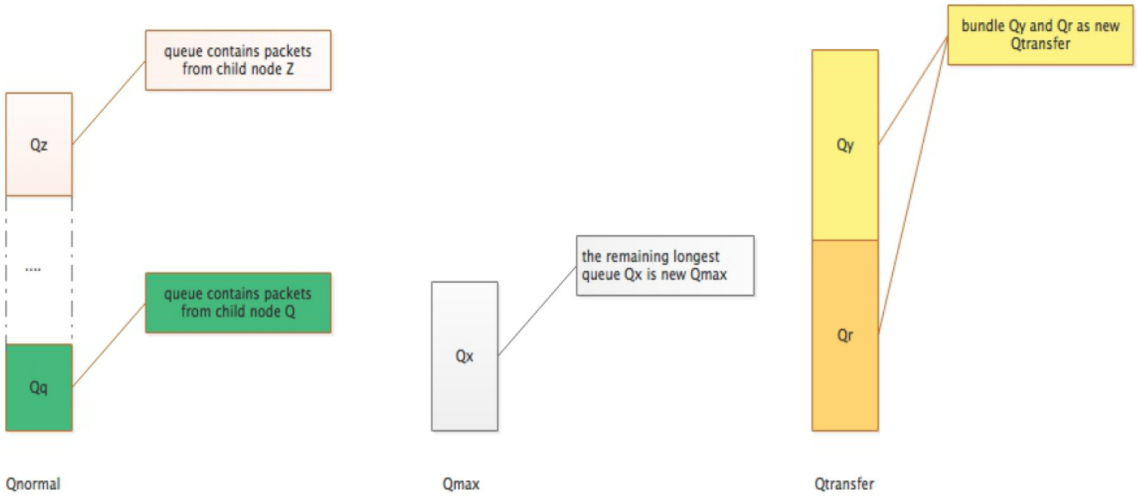

Figure 4

In one example, it is assumed that there is no congestion on candidate node B before node S shares some packets there with. Once node S starts to share $Q_{transfer}$ with node B, node B may experience congestion due to additional throughput from node S. At this point, the same solution as described above is applied at node B, where node B needs to generate Q_{max}, Q_{normal} and $Q_{transfer}$ by itself, and then share partial flows with its candidate parent node.

In summary of Phase 1, as described above, a node adds a backup upward route to share overloaded flow. Additionally, an overloaded node splits partial traffic to another candidate parent node.

Phase 2

In Phase 2, Node S sends packets of Q_{max} and Q_{normal} to a preferred parent and sends packets of $Q_{transfer}$ to a candidate parent whose ETX is lower than other candidates, as shown below in Figure 5.

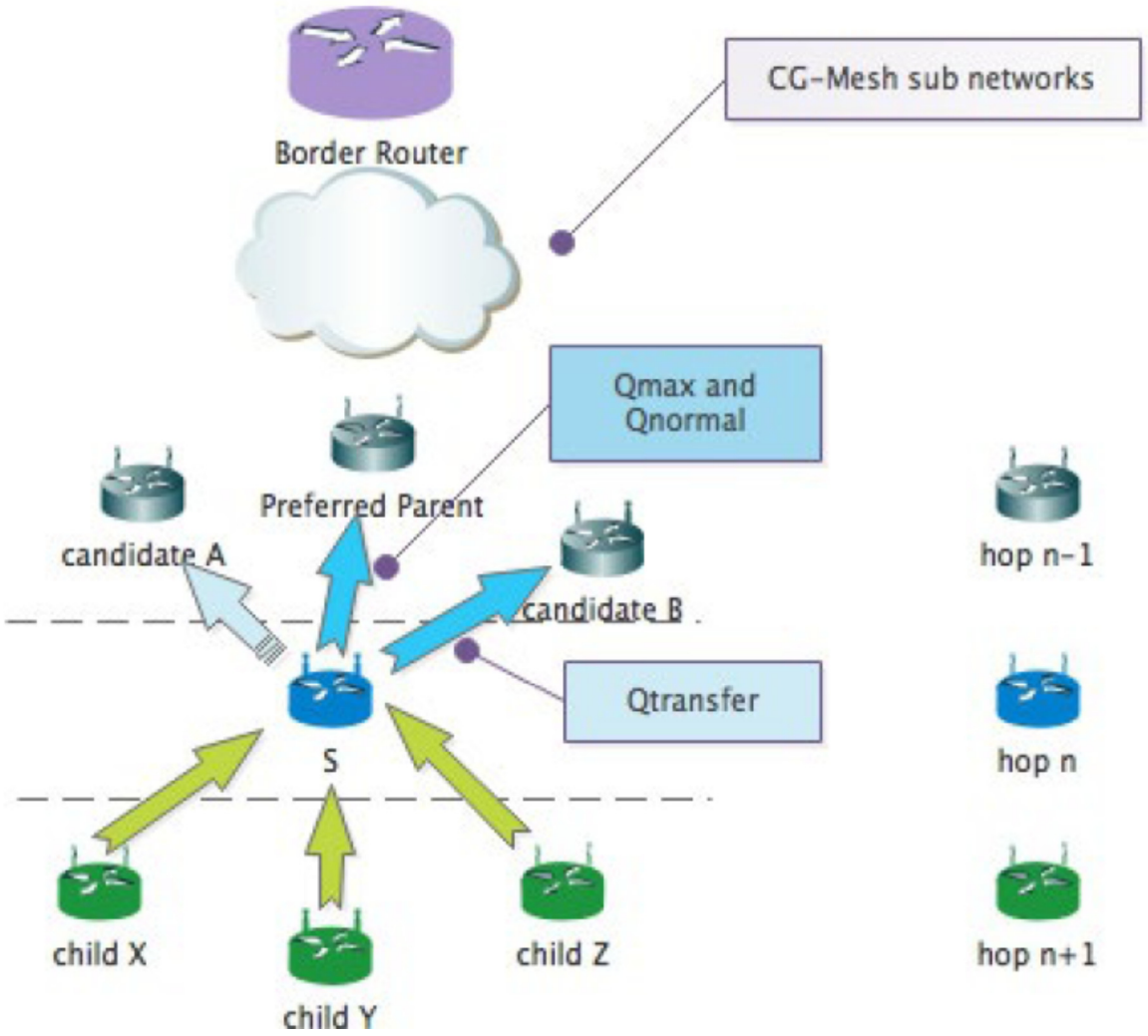

Figure 5

Because node S has two upward selections, the node S may decide which queue should be transmitted at a given moment by:

1. Node S could calculate the ratio of $Q_{transfer}$ to (Q_{max} plus Q_{normal}).
   r = length($Q_{transfer}$) * 100 / (length(Q_{max}) + length(Q_{normal}))
2. Before node S starts a transmission, it first generates a pseudo-random number (e.g., hash(mac address)), such as p = pseudo_random_generate(seed) % 100.
3. Node S compares "p" with "r". If p <= r, node S will transmit packets in $Q_{transfer}$ to candidate B. Otherwise, if p > r, node S will transmit packets of both Q_{max} and Q_{normal} to the preferred parent.

In summary of Phase 2, as described above, an overloaded node chooses which parent to which it should transmit at a given moment. This techniques presented herein could smooth the transmission probability reasonably for better throughput on uplink path. The techniques also enable an overloaded node to alternatively make use of different parents, which provides more free available time slots for each parent node (e.g., when Node S turns to parent A, parent B could be connected by other children, and vice versa).

The techniques presented herein efficiently share upward traffic flows. For example, assuming the average congestion probability for one node is p, if it shifts part of its traffic to a neighbor, the congestion probability will turn to near p^2, and then p^3, ..., p^n, and so on. The more neighbors that share the traffic flow, the faster the congestion disappears, as shown below in Figure 6.

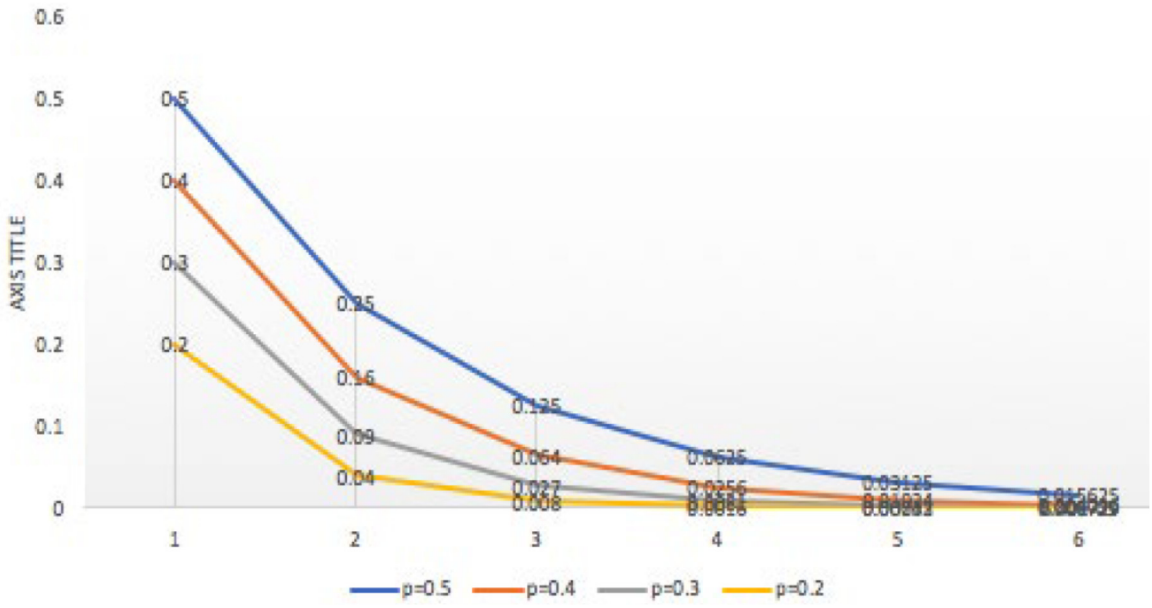

Figure 6

According to above strategy, if node S which has heavy traffic, it may randomly or pseudo-randomly send its upward flows to different parents. Thus, when node S selects one of the parents, the other node has more free slots for its other children to access. This method reduces the probability of collision among the neighbors of node S as well.

In general, the techniques presented herein leverage frequency hopping features in CG-Mesh networks to facilitate a method for packet distribution between parents to share traffic. Other neighbor nodes could reuse the residual time slots when a node turns to another alternative parent. The techniques presented herein also use the longest child queues to avoid out-of-order problems, which also leaves more time slots open for other sibling nodes. The techniques presented herein also facilitate a method to share traffic hop by hop, which could rapidly reduce the congestion risk exponentially. Finally, the techniques presented herein also propose a random mechanism to decide when to distribute packets. This random transmission helps to avoid radio collisions.