August 07, 2019

RANDOM FORESTS BASED ON DYNAMIC NETWORK PARAMETERS IN LOW POWER AND LOSSY NETWORKS

Li Zhao

Chuanwei Li

Lele Zhang

Yinfang Wang

RANDOM FORESTS BASED ON DYNAMIC NETWORK PARAMETERS IN LOW POWER AND LOSSY NETWORKS

AUTHORS:
Li Zhao
Chuanwei Li
Lele Zhang
Yinfang Wang

ABSTRACT

Techniques are described herein for a random forest - based mechanism to select network parameters in Low-power and Lossy Networks (LLNs). These techniques may simplify configuration, accelerate network formation and network convergency, and reduce collisions.

DETAILED DESCRIPTION

A Connected Grid Mesh (CG-Mesh) may be used for Internet of Things (IoT) applications, such as smart grids in Advanced Metering Infrastructure (AMI) networks and Distribution Automation (DA) gateways. The Wireless Smart Utility Networks (Wi-SUN) alliance promotes interoperable wireless standards-based solutions for the IoT.

As shown in Figure 1 below, nodes in mesh networks (e.g., CG-Mesh/Wi-SUN) participate in the following stages to join a Personal Area Network (PAN):

- Network discovery stage
- Mesh access control stage (e.g., security authentication)
- Route discovery stage
- Internet Protocol version 6 (IPv6) address assignment stage (e.g., using Dynamic Host Configuration Protocol version 6 (DHCPv6))
- Route registration stage (e.g., using Routing Protocol for Low-power and Lossy Networks (LLNs) (RPL))
- Field Network Director (FND) registration stage

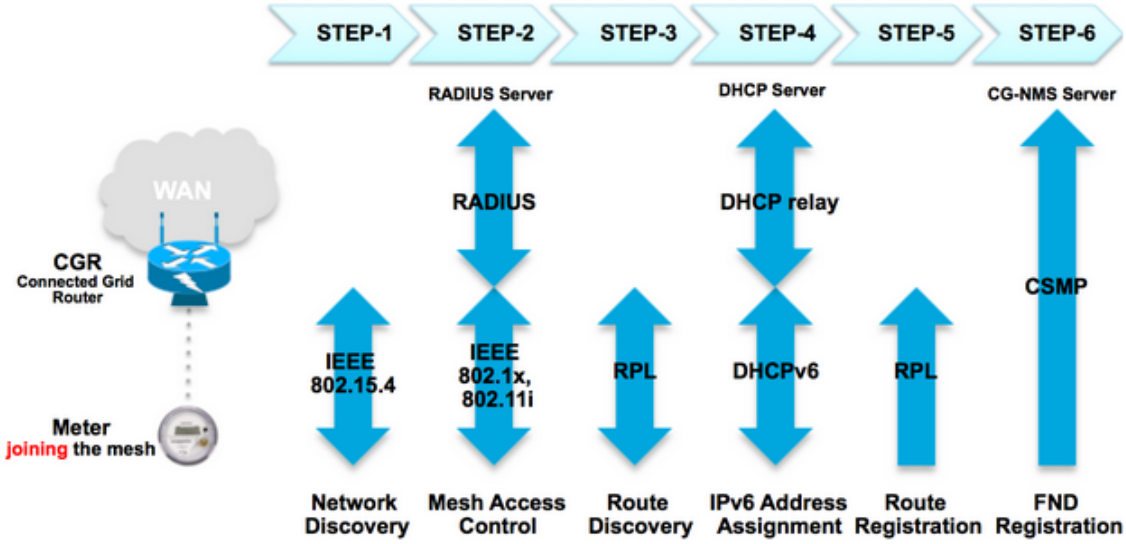

Figure 1 Network Formation

In order to accelerate network joining and network convergency while reducing collisions, CG-Mesh defines three network scales for different network scenarios: small scale, middle scale, and large scale. Currently, network scale selection is pre-provisioned and based on the node number for target deployment in a PAN. Moreover, as shown in Table 1 below, different network parameters correspond to different network scales.

Table 1. Different Network Parameter Suits of Frames in Different Network Stage

Network Stage	Frame Type In Network	Network Parameter Suits (**Small Scale** [min, max])	Network Parameter Suits (**Middle Scale** [min, max])	Network Parameter Suits (**Large Scale** [min, max])
Discovering network	Discover Beacon Frame	[10, 60] seconds	[15, 60] seconds	[60, 900] seconds
	Sync Beacon Frame	[30, 60] seconds	[15, 60] seconds	[60, 900] seconds
Security authentication	EAPOL Key Frame	[60, 300] seconds	[60, 300] seconds	[300, 3600] seconds
IPv6 address assignment	Solicit Frame	[3, 3] seconds	[60, 3600] seconds	[60, 3600] seconds
Route registration	DIO Frame	[32, 1048] seconds	[1048, 16777] seconds	[1048, 16777] seconds
FND Registration	Coap Frame	[60, 300] seconds	[300, 3600] seconds	[300, 3600] seconds
Joining Timing		[195, 1771] seconds	[1498, 24397] seconds	[1828, 179600] seconds

For example, the sending time of discover/synchronize beacon messages during the network discovery phase in small scale networks is earlier than that of middle and large scale networks. As shown in Table 1 above, the minimum sending time intervals are 10 seconds in small scale networks, 15 seconds in middle scale networks, and 60 seconds in large scale networks, and the maximum sending time intervals are 60 seconds in small scale networks, 60 seconds in middle scale networks, and 900 seconds in large scale networks. A similar mechanism also applies to the other stages listed above. The earlier a

node sends a message in the current network stage, the faster that node can enter into the following stage to join a PAN quickly.

As a result, the minimum joining times are 195 seconds in small scale networks, 1498 seconds in middle scale networks, and 1828 seconds in large scale networks, as shown in Table 1 above. The maximum joining times are 1771 seconds in small scale networks, 24,397 seconds in middle scale networks, and 179,600 seconds in large scale networks.

One problem with standard methods is that in current CG-Mesh deployments, the scale parameters are pre-provisioned in flash based on the target network, and therefore cannot match the runtime network environment. For example, if the target deployment network is a large scale network, each node is fixedly preconfigured with large scale parameters in the factory, and it takes a minimum join timing of 1828 seconds to join in the network. The CG-Mesh network is often a multi-hop network (e.g., up to 20 hops or more), and the minimum join timing of the deepest node can be 10 hours and more. The preconfigured large scale nodes can thus lead to longer joining times and network convergency, even during the initial stage of network formation where there are only a few mesh nodes that exist in the network.

In another example, there are two PANs: PAN1 and PAN2. PAN 1 is preconfigured as a small scale network, and PAN2 is preconfigured as a middle scale network. Some nodes can cross between the two PANs. When the wireless environment varies, the nodes in PAN1 may try to join PAN2. That leads to the network congestion because PAN2 is preconfigured as a middle scale network. As a result, the migrating nodes may not join the PAN2 network. Furthermore, nodes that are already a part of PAN2 may be dropped from PAN2.

Another problem is that during manufacture, each node is preconfigured with one scale by an external tool. When the preconfigured node is placed into an unmatched scale configuration in field deployment, it can require extraneous human resources to reprogram the node.

Accordingly, a mechanism is described herein to select network parameters using Machine Learning (ML). Decision Tree (DT) is a common ML tool, and does not depend on professional knowledge. It leverages appropriate attributes as metrics to divide tuples

into the appropriate classes. Unlike other popular ML algorithms, DT is easy to use and does not require a strong computing resource. As such, DT may be implemented on a small embedded system platform such as a CG-Mesh node. Random forests can grow many classification trees. To classify a new object from an input vector, the input vector may be put down each of the trees in the forest. The forest may choose the classification having the most votes (over all the trees in the forest). Random forests may be established on the cloud or a central server based on massive runtime or virtual network information of the mesh networks.

As described herein, a DT mechanism may be used to determine which network parameters should be chosen in a runtime environment. The correct point should be split in order to build a suitable DT for the model. Furthermore, the appropriate threshold should be determined for accurately identifying classes from mass tuples. In terms of refining respective split-sets, three general situations are described as follows:

1. Attributes are discrete and do not fit in binary decision tree models, and each partition is a branch.
2. Attributes are discrete and fit in binary decision tree models. This enables obtaining two branches from one partition, one representing attributes that belong to a subset and the other representing attributes that do not.
3. Attributes are continuous values. The threshold should be determined as split-point, and two branches may be generated, one representing attributes greater than the split point and the other representing attributes less than or equal to the split point.

Therefore, the DT helps select the correct metrics and split-points for attributes. A variety of algorithms (e.g., Iterative Dichotomiser 3 (ID3), C4.5, Classification And Regression Tree (CART), etc.) may be used.

In another example, a set of elements may be defined as attributes for DT. These metrics may include the following factors:

- Topology information (e.g., the depth of the node, which may bundle with the problem regarding route path or parent selection)
- Stable neighbor number
- Stable neighbor information (e.g., signal conditions from neighbors such as bidirectional Received Signal Strength Indicator (RSSI) and Link Quality Indicator

(LQI), quantity of visible neighbors, and path cost like Expected Transmission Count (ETX) or Rank.)

- Stable neighbor ratio (e.g., stable neighbor number / amount of visible neighbors)
- Bidirectional frames statistics (e.g., a number of frames that the node receives or transmits, data packets drop ratio, control packets drop ratio, etc.). Different kinds of frames may represent different meanings. For instance, discovery beacon requests / discovery beacons may be associated with a problem regarding the beginning stages of joining a PAN, and synchronize beacon requests / synchronize beacons may indicate that the node has difficulty locating an available parent node.
- Modulation (e.g., 2 – Frequency Key Shifting (2FSK) or Orthogonal Frequency-Division Multiplexing (OFDM))
- Symbol ratio (e.g., 50 kbps or 150 kbps)
- Deployment place (e.g., indoor or outdoor, wild field, campus, etc.)
- Frequency band (e.g., the U.S. adopts 900 MHz, India uses 800 MHz, etc.)

Figure 2 below illustrates a DT built using one or more of the aforementioned metrics. As shown, split-points are represented by the gray ovals and the endpoints are represented by rectangles in red/green/blue. If one node has a depth of two, and its modulation is OFDM, it will choose the small scale. If the modulation is 2FSK, the node will choose the middle scale.

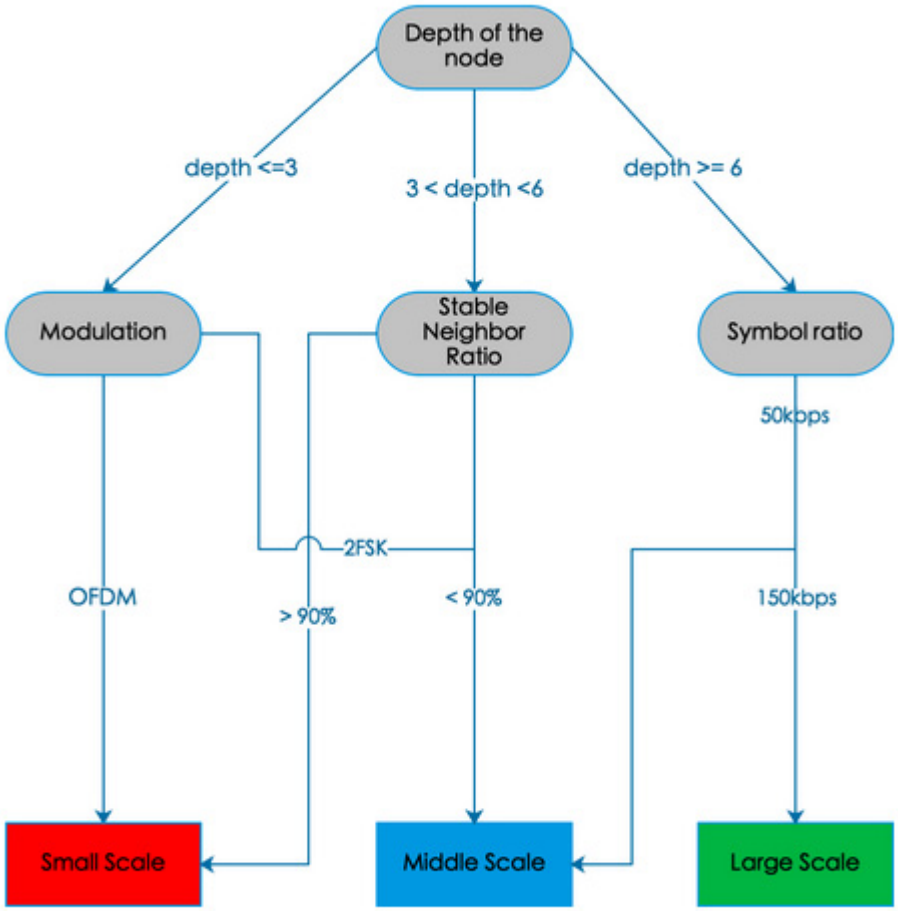

Figure 2

All split-points have a determinate threshold as the criteria of the classification. To obtain these thresholds, the concepts of entropy and gain are often used. To this end, the ID3 or C4.5 algorithms may be employed.

Random forests may be used to address the over-fitting problem. In particular, this solution may use multiple DTs to obtain stronger robustness and reduce loss as much as possible. Single DT solutions may help meet the over-fitting problem, and improved methods may also be used (e.g., Random forest, Adaboost, Gradient Boosted Decision Tree, etc.). Random forest, which is used to rank the importance of variables in a regression or classification problem in a natural way, is one popular solution to overfitting in DT implementation.

As illustrated in Figure 3 below, random forests may generate three DTs. Rather than choosing the scale parameter directly, each DT has respective probabilities.

In one specific example, a node has the following conditions:

- depth: 3
- modulation: OFDM
- deployment place: outdoor
- stable neighbor number: 80
- stable neighbor ratio: 97%
- data packets drop ratio: 1%

Based on these conditions, DT1 votes small scale, DT2 votes large scale, and DT3 votes small scale. Accordingly, the final scale is small.

Figure 3

Hundreds of DTs may be built for improved accuracy fitting with this method. This solution is better than a single DT and has good performance in overcoming the overfitting problem.

In a further example, the default scale may be large to ensure that the node joins mesh network successfully. After joining the network, the node may adjust the scale parameter according to the voting results of the random forests. Meanwhile, the random forest algorithm is part of the node firmware. The network administrator may update the algorithm on the node when the algorithm is evolved in the cloud or a central server. This may be easier than updating the entire firmware.

In summary, techniques are described herein for a random forest - based mechanism to select network parameters in LLNs. An ML method regarding random forests may be used to select the optimal network parameters in a mesh network. These techniques may simplify configuration, accelerate network formation and network convergency, and reduce collisions.