Supplemental Material of

Merged Path: Data Dissemination for Large Scale Multiple Mobile Sinks WSN

Ammar Hawbani, Xingfu Wang, Omar Busaileh, Ammar Qarariyah, Liang Zhao, Mohammed A. A. Al-qaness, and S. H. Alsamhi

This document is a supporting content for the manuscript (Merged Path: Data Dissemination for Large Scale Multiple Mobile Sinks WSN), given the limited number of pages allowed by the publisher. We provided more details about the algorithms and the simulation results.

Index Terms: Wireless sensor networks, multiple mobile sinks, data routing, merged path.

1. Clustering and Bifurcating

1.1 Clustering Algorithm:

The goal of this algorithm is to divide the sinks into groups and to find the branches where the packet to should be traveled. Thus, this algorithm returns a list of branches. The **Branch** object contains **Cluster (**line #157**)**, **Start** and **End** points of the cluster. Cluster contains a list of sinks of the branch. Start point is the start point of the branch while End point is the end of the branch. Before explaining how to find the branches, we will first explain how the clustering algorithm runs. The high-tier node sends back the locations of the sinks at the network to the source node that has data to send. Thus, the clustering algorithm is performed at current source node (the current bifurcation point). Thus the inputs for the clustering algorithm are **bk_sinks** (the locations sinks in the field which responded by the high-tier node), the **b_k** (current bifurcation point), and the **clustingThreshould**. The source code for the **Clustering** algorithm is on the link . The object (`Angle` `line#24`) contains various of functions. The main function [`GetAngle` `line #35`] is to return the cosine similarity between two sinks(namely, `sink1 and sink2`) considering the current (`bifurcation`) point. The clustering algorithm starts at line #250. It runs in three steps. First, (`FindAngles` `line #468`) to find the cosine similarities of all nodes by calling the (`GetAngle` `line #35`) . All similarities are stored in an object named (`AngleSimlirityEdge` `line #96`). The second step is to remove the similarity edges which are greater than the threshold value (`FindMinusScors` `line #553`). The third step is to find the clusters (`ComputeClusters1` line #292). This function runs as explained in the main manuscript.

1.2 Bifurcation

The bifurcation point is exactly where the merged path should be branched to several sub-branches, and explicitly where the clustering process is computed. The source code of bifurcating is on the link. It contains the (`Branch` `line #15`) object the contains a **Cluster** (line #157). The main function in in the (`Bifurcation object`) **is the (**`FindBranches` `line 32`**)** that returns the branches at the current bifurcation point. This algorithms woks as follows. The source node divided the sinks into clusters as explained in the previous section (`ComputeClusters1` line #292). The main process here is to find the start and end of each branch. See the details code on the link.

2. Diagonal Virtual Line Construction

The construction process for the overlay structure is explained in the link (`DiagonalVirtualLine`). As explained in the manuscript, DVL defines two reference points for the diagonal for example (`Point(0, 0); and Point(NetworkSquareSideLength,NetworkSquareSideLength);`). After identifying these two points, the protocol starts by broadcast (`DVLConstructionMessage`). It has the following functions: `GeneratePacket`, `SendPacket`, `RecivePacket`, `SelectNextHop`.

3. Merged Path