Project Report

On

Source Routing for Download Data Traffic

Wireless Sensor Network 2016-2017

Mehetab Alam Khan

Student ID: 188957

Project Goal

The aim of this project work is to implement a multi-hop Source Routing protocol for a wireless sensor network. The deliverable of the project is a routing layer (a nesC component) that supports two traffic patterns: many-to-one, allowing network nodes to send data packets up to the sink (root) and one-to-many, which is the exact opposite.

Source Routing

In Source Routing algorithm, each data packet contains complete routing information to reach the destination. These routes are computed by the source node (the root in our case) and piggybacked in the packet header. In case of source routing the intermediate nodes do not need to maintain routing tables in order to forward the packets, since the packets themselves already contain all necessary information.

So in order to implement the source routing algorithm, we need a many-to-one communication module. We use the tree building and many-to-one communication module so that the motes communicate with the root and the root knows the connectivity within the topology. This is done by each non-route node by sending the data its own address and its parents address in an upward direction to the root node. The root on receiving the packet can update a routing table where it will have all the connectivity information about the nodes within the topology.

The routing table is further used by the one-to-many communication module to send data to the target node.

Many-to-One Traffic Pattern (*MyMany2OneC*)

The mechanism consists of a network of sensor nodes where every non-root node is able to send data to the root node through one or more hops. This could be achieved by the help of a tree spanning algorithm where each non-route node registers its parent node locally and use this information to send data in an upward direction to the root node. After the spanning tree has been built, each node has the information about its parent node. This information is used by the node where application data originates to send this data to its parent node. An intermediate node, upon reception of this data, forwards it to its own parent node. In this way data forwarding continues till it has reached the root node.

Implementation

command void send(M2OAppData psData)*

- If it’s a non-route node, create a network packet by adding hop and data source related information to the application packet. Send this packet to the parent node.

- If it’s a root node, print error message.

event void receive(am_addr_t from, M2OAppData psData)*

- Upon reception of data, each node signals an event to the application telling the information about the actual source of data and content of data.
- Intermediate nodes further transmits this data in an upward direction to the parent node (towards the root node).

Spanning Tree (MytreeC)

The tree-building process is initiated by the root node through broadcast messages to neighbouring nodes. Upon reception of these broadcast messages, the non-route nodes start broadcasting to their corresponding neighbouring nodes. Each time a non-route node receives a broadcast message from another node in the network, its parent ID value is refreshed. Only the signals with RSS value greater than -90 is taken into consideration.

Implementation

command void buildTree(void)

- Start building tree (in application) after 1 seconds of start-up for non-root nodes (NODE_UPTIME) and 2 seconds of start-up for root nodes (ROOT_UPTIME). This is to ensure that all the non-route nodes are up and running when the route node sends the broadcast beacons.
- Upon reception of a broadcast message, check if the message is new (higher sequence number) or its metric (hop count) is better (lower hop count) than previously received message. If so, then update the parent information of the current node with the sender ID of the current message.
- Upon reception of a broadcast message, each node will increment the hop count value by 1 in the above-received broadcast message and broadcast the packet to its own neighbouring nodes.
- Build the tree (in application) every 2 minutes (REBUILD_PERIOD).
- Routing table needed for Source Routing protocol is also created in this module by sending the parent information of each node to the root node using many-to-one interface.

One-to-Many Traffic Pattern (*MyOne2ManyC*)

In one-to-many communication, the route node should be able to send the data to the other nodes within the topology. Each packet of data carries within itself, along with the application data, the header information that is required for the data to reach the target. This header information is retrieved from the source routing table previously implemented. When an intermediate node receives this data, it modifies the packet heard and send the data in a downward direction towards the target node.

Implementation

command void send(uint8_t u8DestId, O2MAppData psO2MData);*

- If the current node is the root node, create the packet header by attaching IDs of all the intermediate nodes and the target nodes to the application data. Maximum path length (MAX_PATH_LENGTH) used in the current implementation is 5.
- If the current node is not the root node, then print an error.

event void receive(am_addr_t to, O2MAppData psData);*

- Upon reception, if the current node is not the root node, signal an event to the application telling it the target node ID and the application data content of the message. If the current node is not the target node, then modify the packet header by nullifying the entry related to the current node ID and then forward the modified packet in a downward direction as per the packet contents.
- If the current node is the root node, print an error.

Conclusion

The above implementation was tested using the Cooja Simulator with maximum number of node (MAX_NODES) 15.