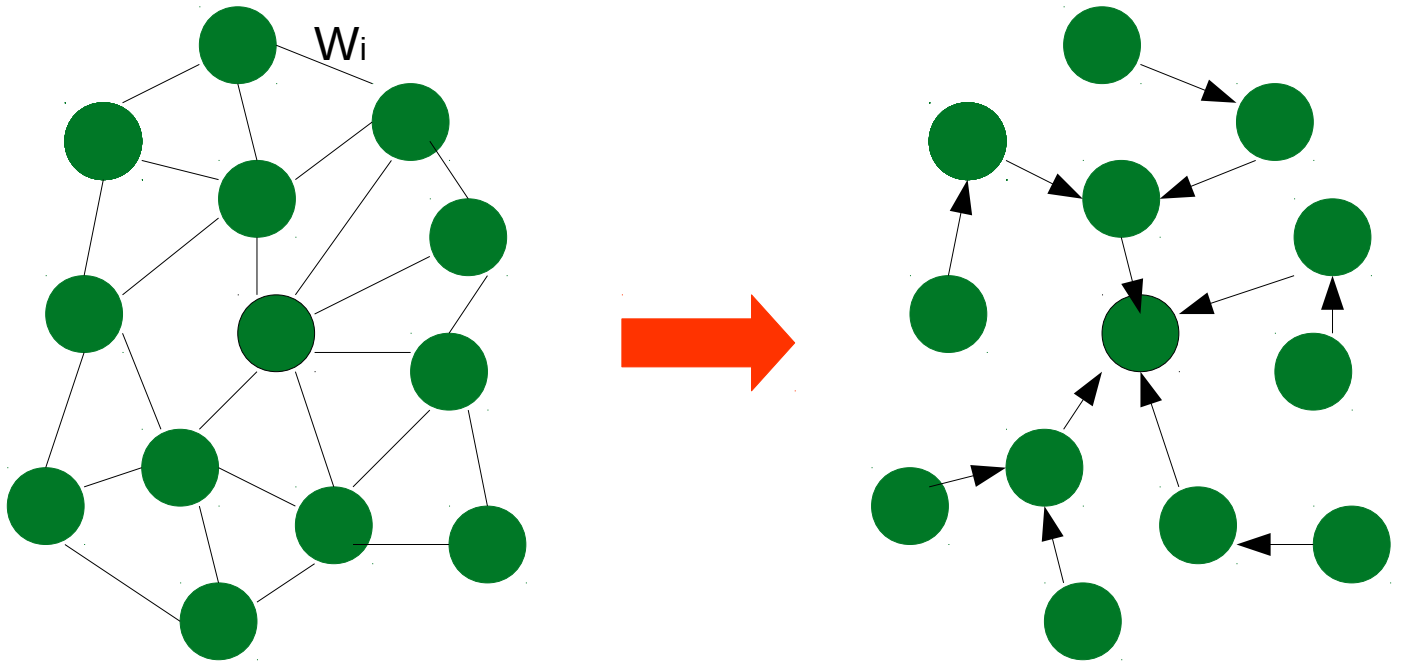
Gallager-Humblet-Spira (GHS) Algorithm in WSN

Sergio Diaz

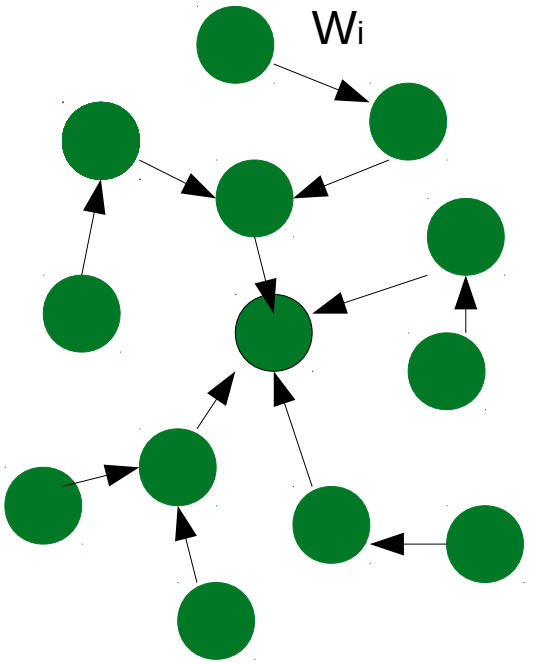
GHS Algorithm - Problem Statement

- Given a graph **G** with **N** nodes and **E** edges, find the Minimum Spanning Tree (MST)
- Minimum Spanning Tree
  - It includes all of the vertices of **G,** and
  - the sum of the weights of its edges is minimal.



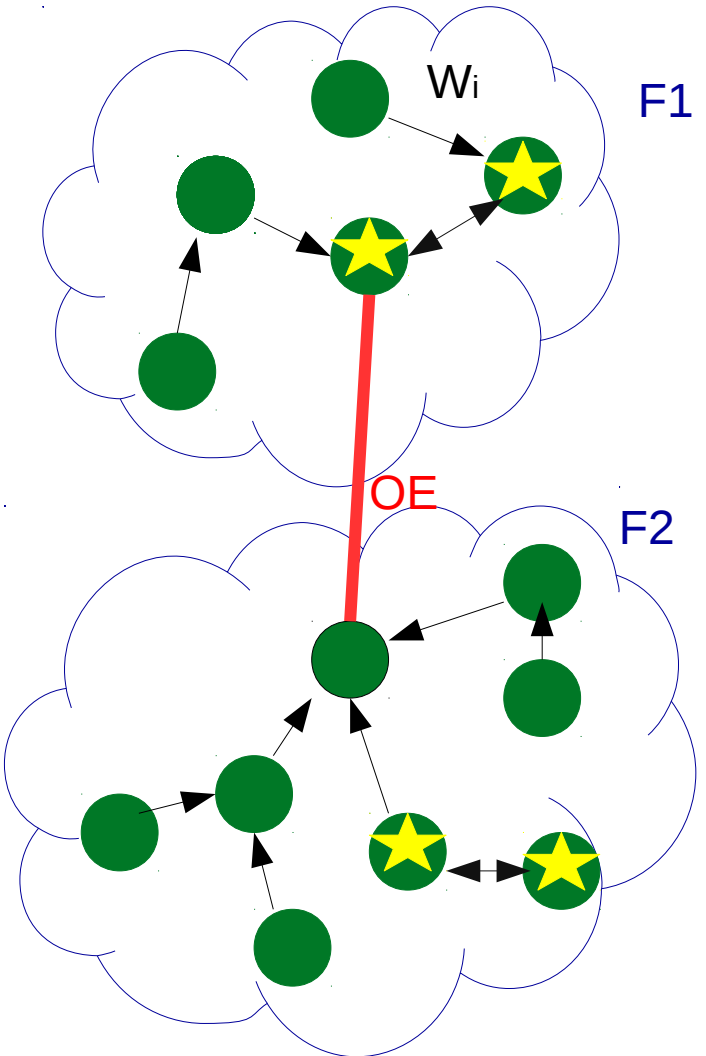
GHS Algorithm - Overview

- **Assumes**: Each node already knows the weight of each edge.
- Distributed algorithm for constructing a MST
- **Functioning**:
  - Sending msg over adjoining links,
  - Waiting for incoming msg, and
  - Processing
- **Characteristics**:
  - Asynchronous
  - Deadlock-free
  - Tolerates unpredictable but finite delay
  - One message contains at most
    - One edge weight + $\log_2 8N$ bits
  - Worst-case message complexity:
    - O(E + N log N): It is optimal



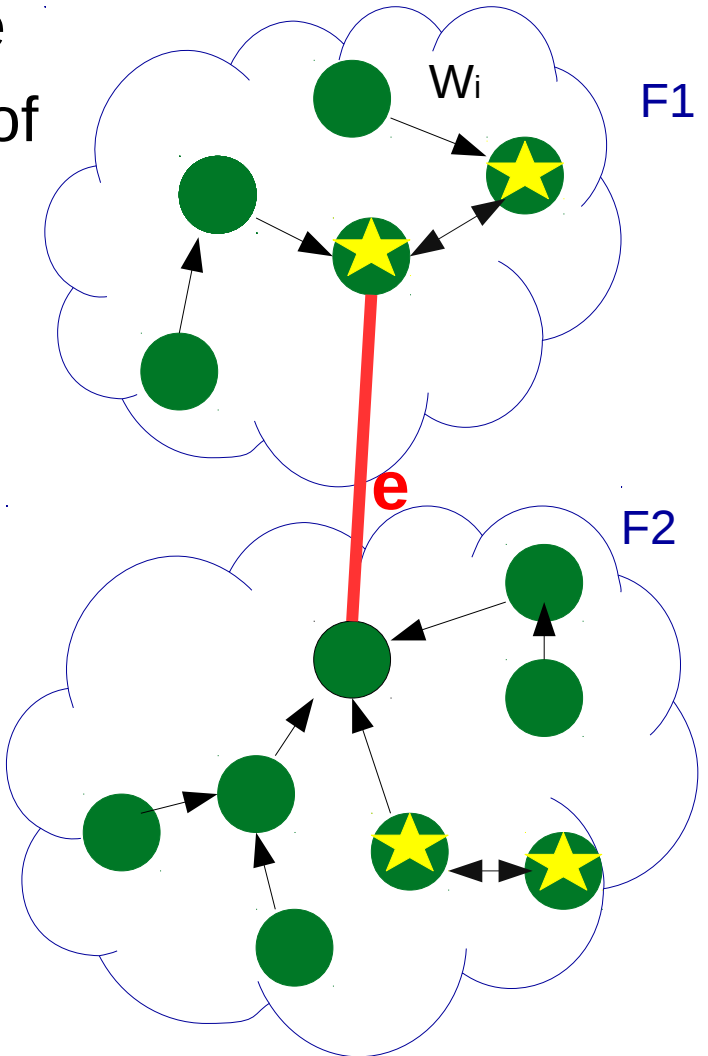
GHS Algorithm – Concepts (1)

- **Fragment (F)**: Any connected subgraph of the MST
  - **Core Nodes:**
    - Central Computing unit of the fragment
    - Receives reports of the lowest-weight edge
    - The join to another fragment is initiated
- **Outgoing Edge:** If exactly 1 of the nodes connected by the edge is in the fragment.
- **Type of messages:**
  - **Connect:** Request a connection to another F
  - **Initiate:** Initiate connection with another F
  - **Test:** Test whether the edge is an outgoing edge
  - **Accept:** Accept the edge as an outgoing edge
  - **Reject:** Reject the edge as an outgoing edge
  - **Report:** To report the lowest-weight of a node
  - **Change Root:** Change the core nodes of the fragment.



GHS Algorithm – Concepts (2)

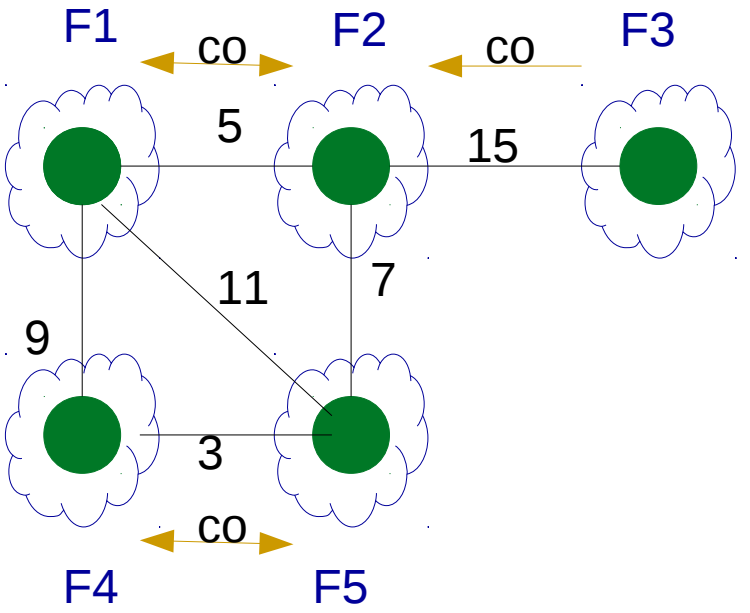
- GHS algorithm is built on the following property
  - **Property:**
    - Given a fragment of an MST, let **e** be a **minimum-weight outgoing edge** of the fragment.
    - Then joining **e** and its adjacent nonfragment node to the fragment yields another fragment of an MST.



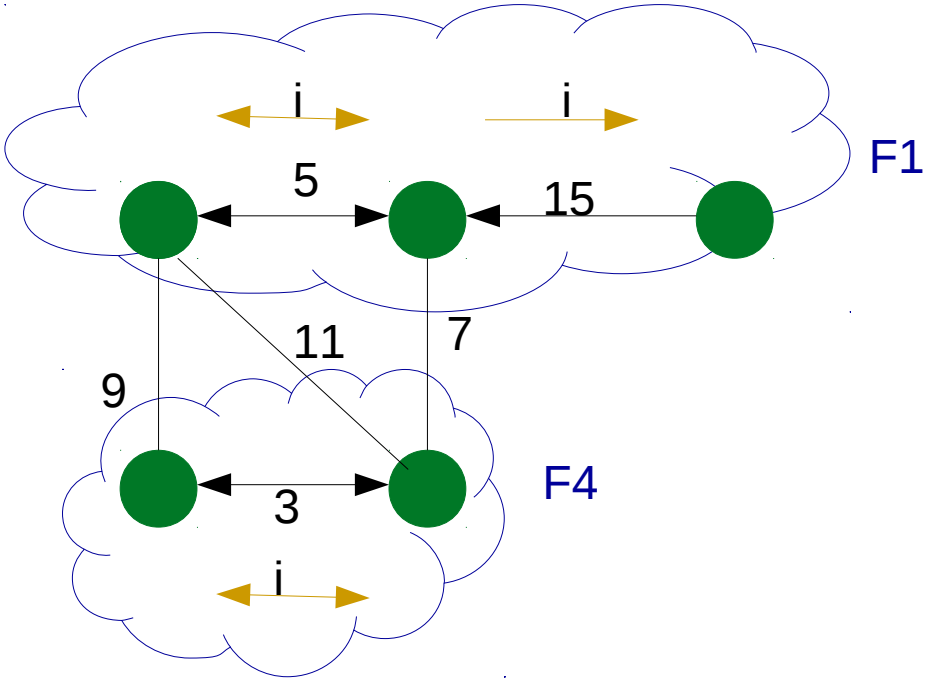
- We can join fragment F1 with F2 as long as e is a **minimum-weight outgoing edge**.
- By joining fragments the MST is found.

GHS Algorithm – Example

- **Initially (Connect):**
  - Each node belongs to a different fragment
  - Each node sends a Connect **(co)** msg via its lowest-weight edge

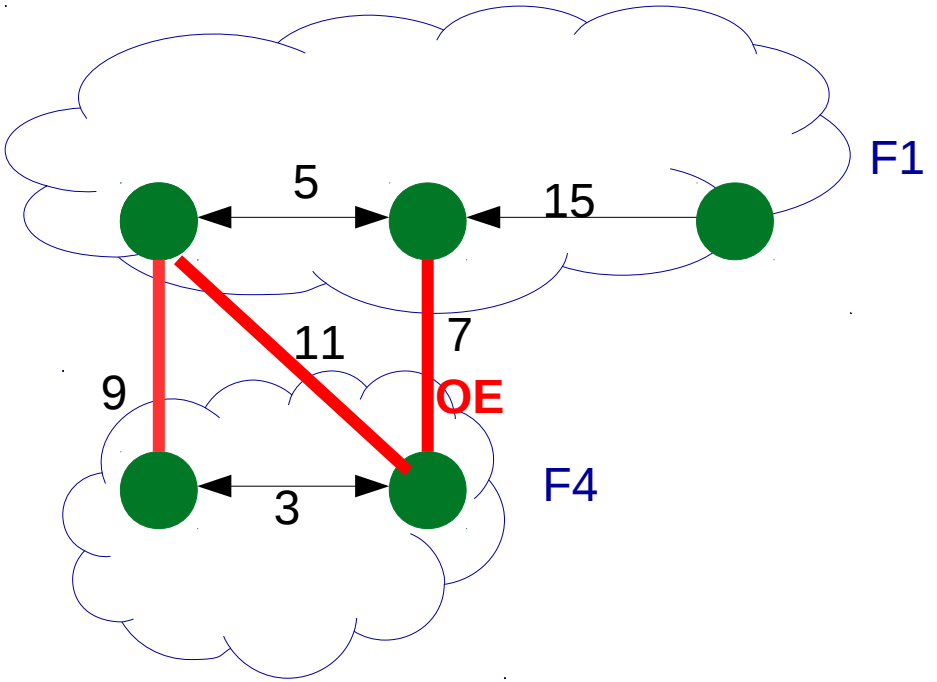


- **Initiate:**
  - Each node responds a Connect msg with an initiate (i) msg.
  - Then, the Fs that received an i msg merge.
  - At the reception of an initiate msg the node selects the sender as its parent

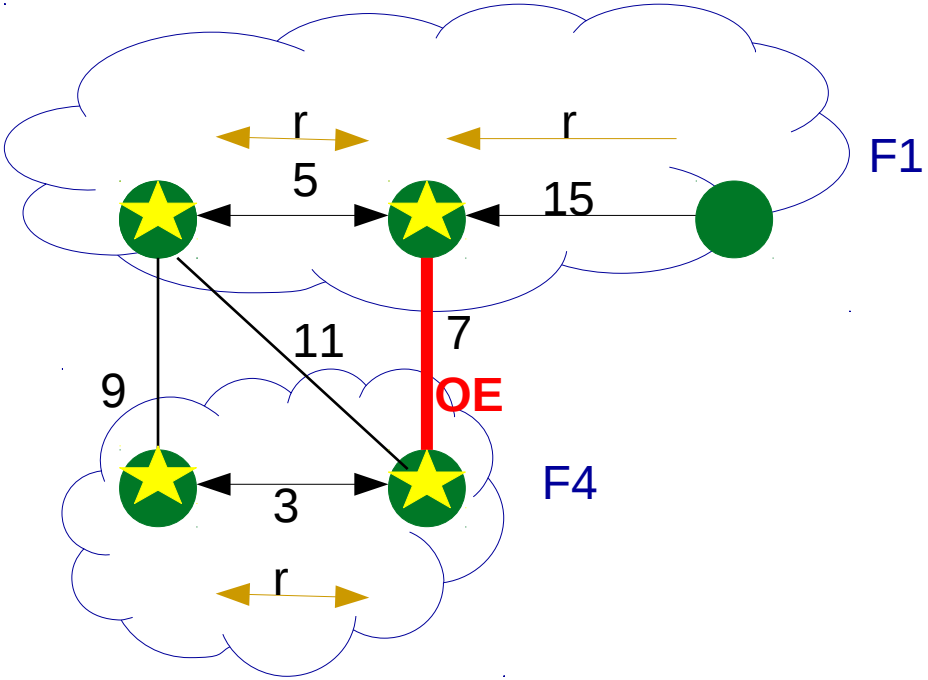


GHS Algorithm – Example

- **Test: Accept or Reject:**
  - Each node sends test (t) msg to find the outgoing edges (**OE**)
  - The edge is **accepted** if it belongs to a different fragment
  - The edge is **rejected** if it belongs to the same fragment

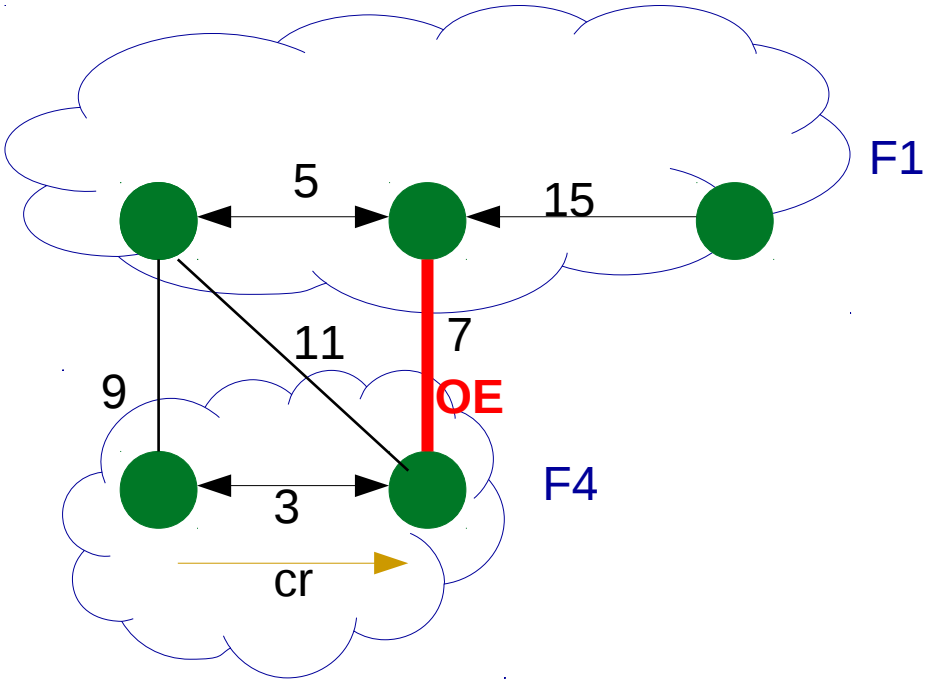


- **Report:**
  - Each node reports its lowest weight outgoing edge.
  - The **core nodes** process the report msg and find the *lowest weight outgoing edge* of the fragment.

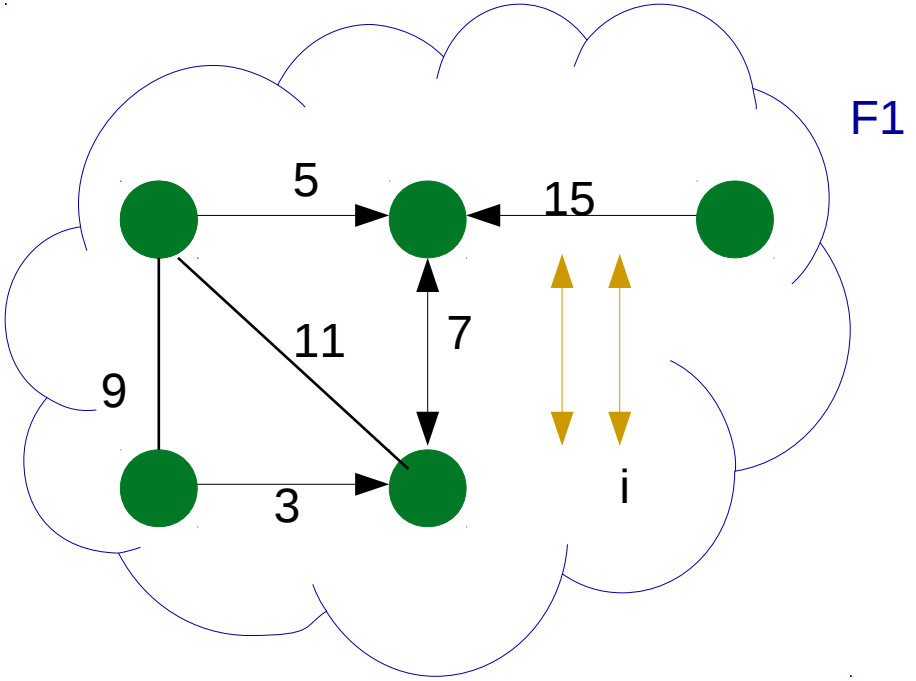


GHS Algorithm – Example

- **Change Root:**
  - A Change root msg is sent to the node that owns the *lowest weight outgoing edge* in each fragment.

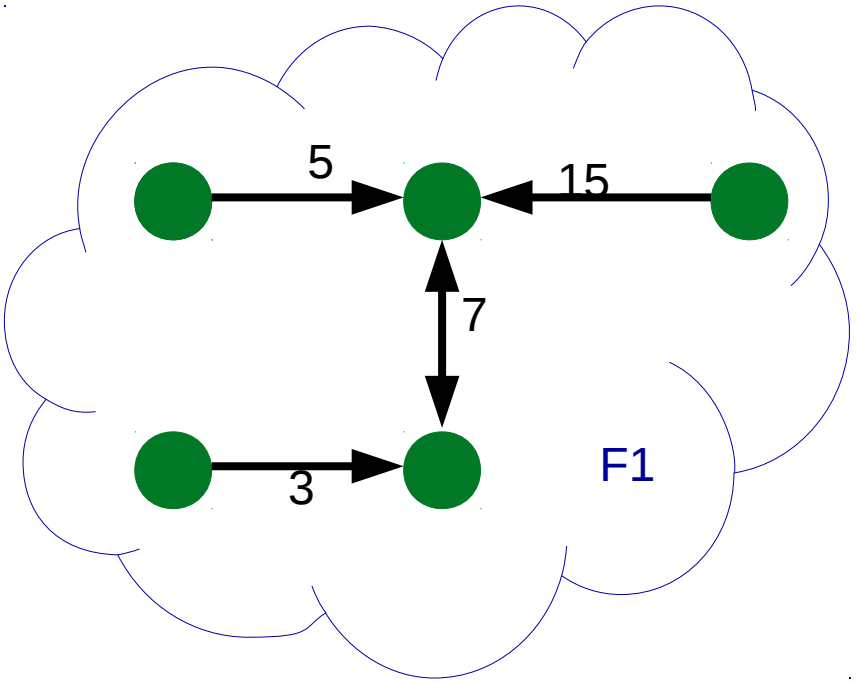


- **Connect:**
  - The fragments send a connect msg, followed by a initiate msg.
  - The fragments joined together
  - The nodes change their parent according to the last initiate msg.



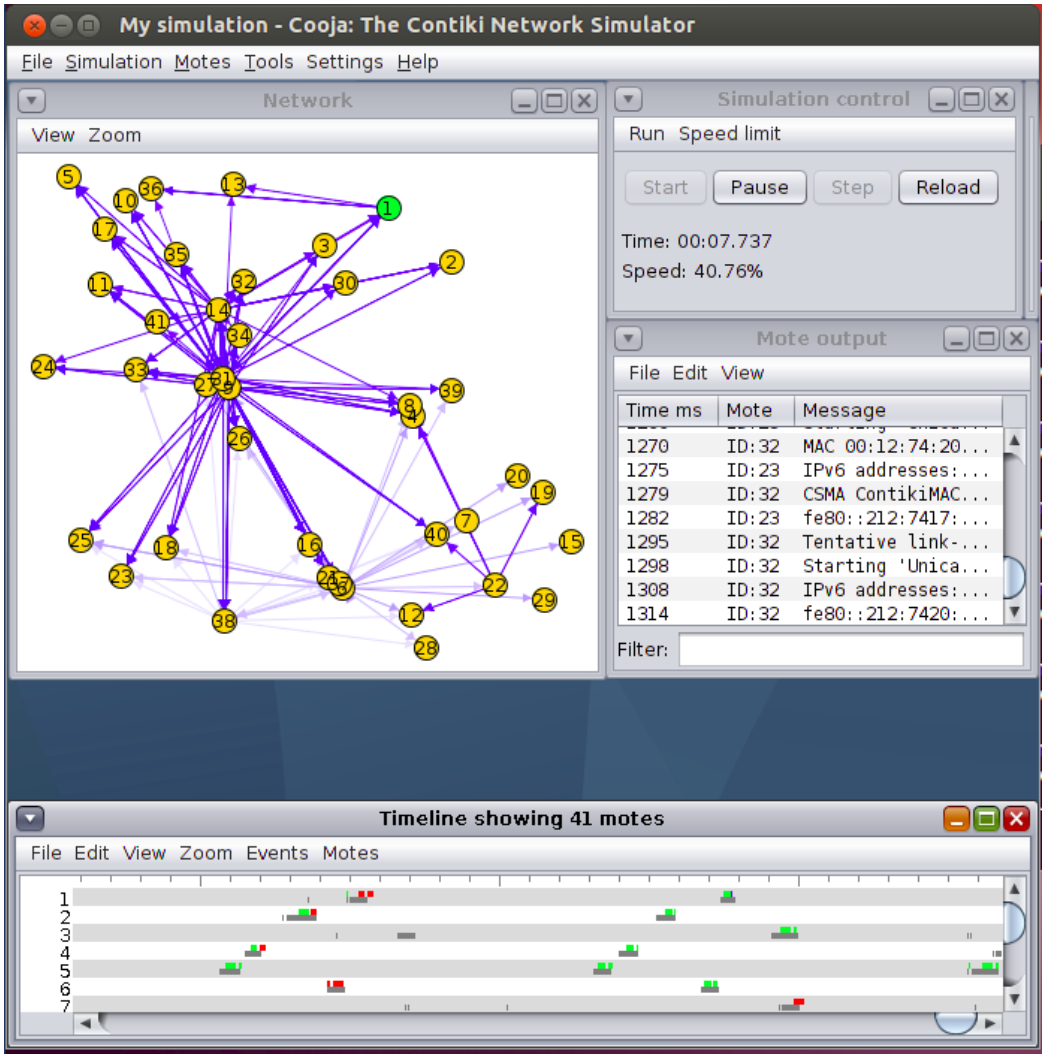
GHS Algorithm – Example

- **Final result:**
  - Only one fragment remains
  - Minimum Spanning Tree
  - Each node has a parent
  - The more costly edges are avoided



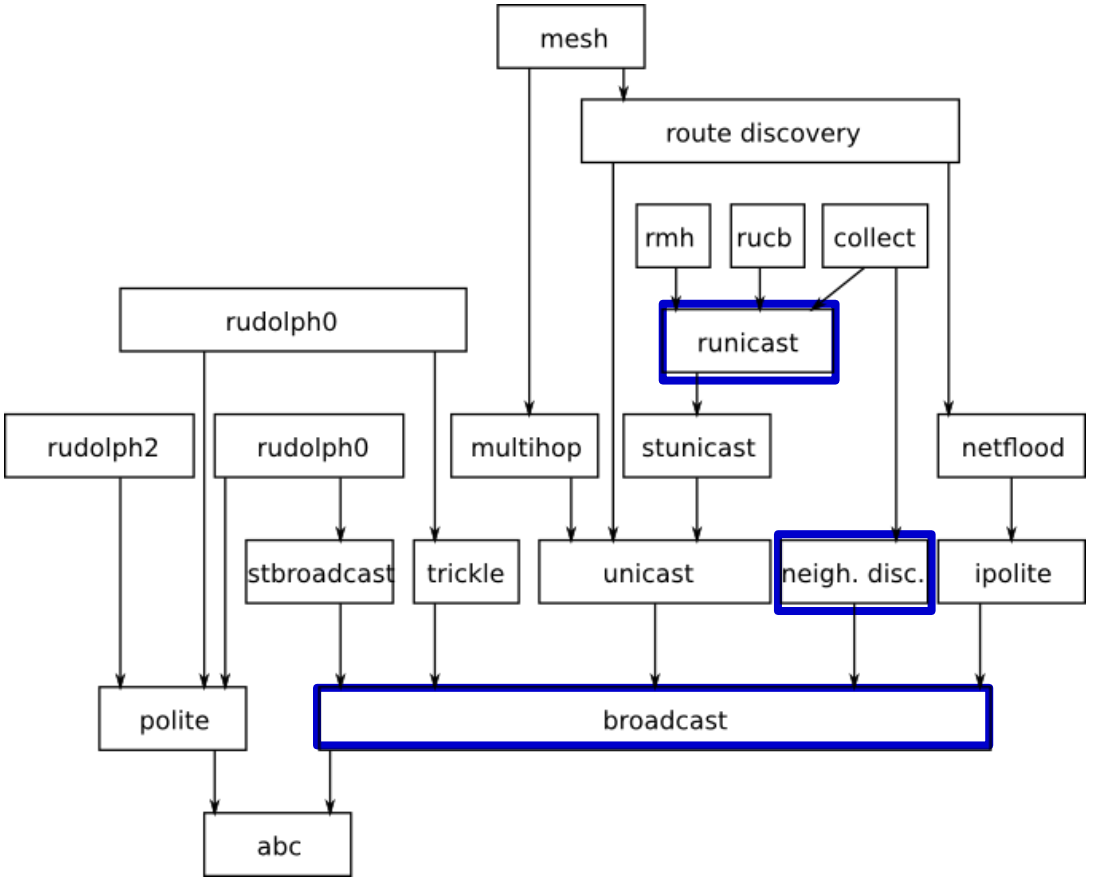
GHS implementation in Contiki OS - Overview

- **Contiki Characteristics**:
  - It is an open source Operating System for IoT.
  - It is event-driven and it is written in C
  - The protocol stack can be chosen between
    - IPv4
    - IPv6
    - **Rime**: We selected rime to implement the GHS algorithm.

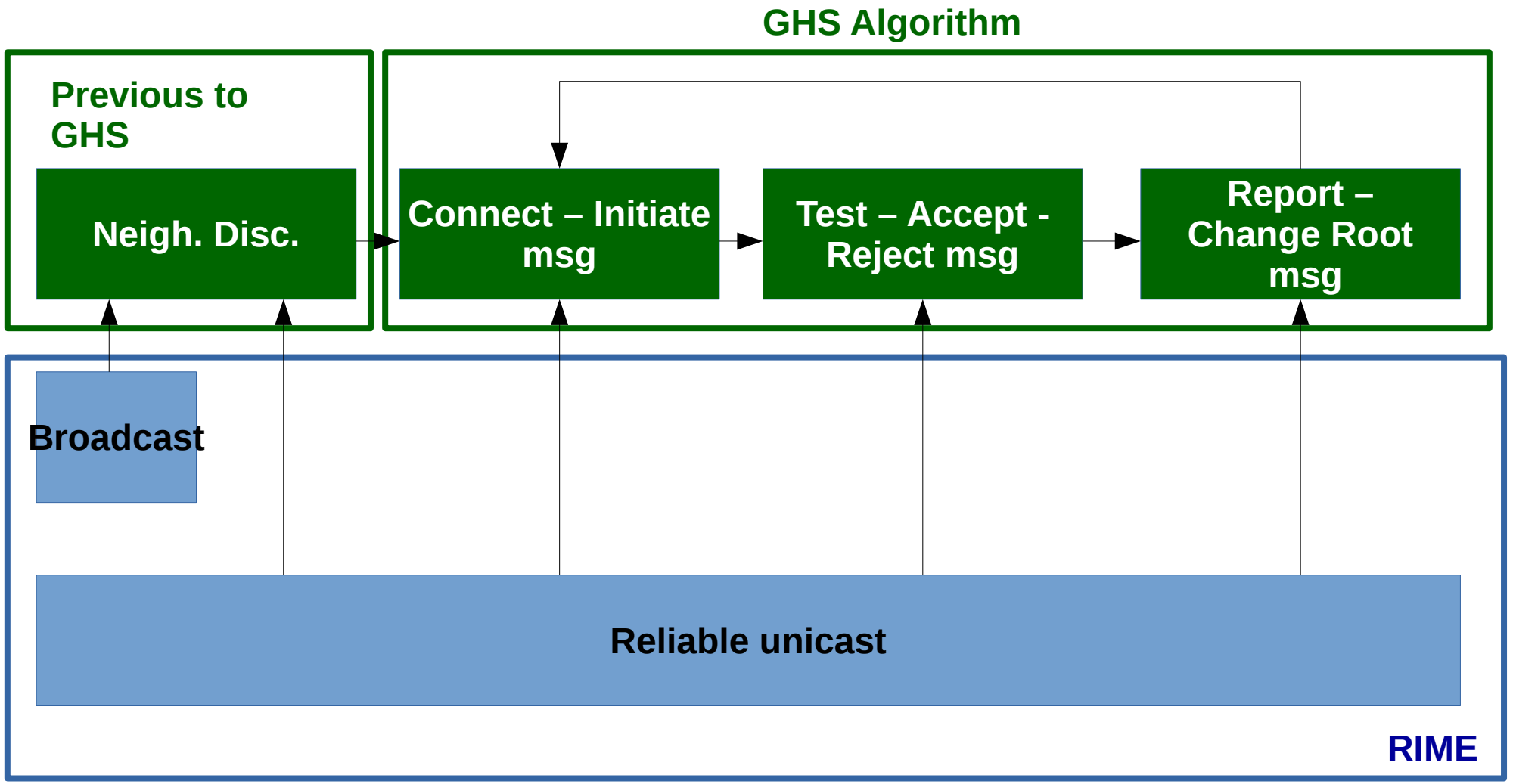


GHS implementation in Contiki OS - Rime

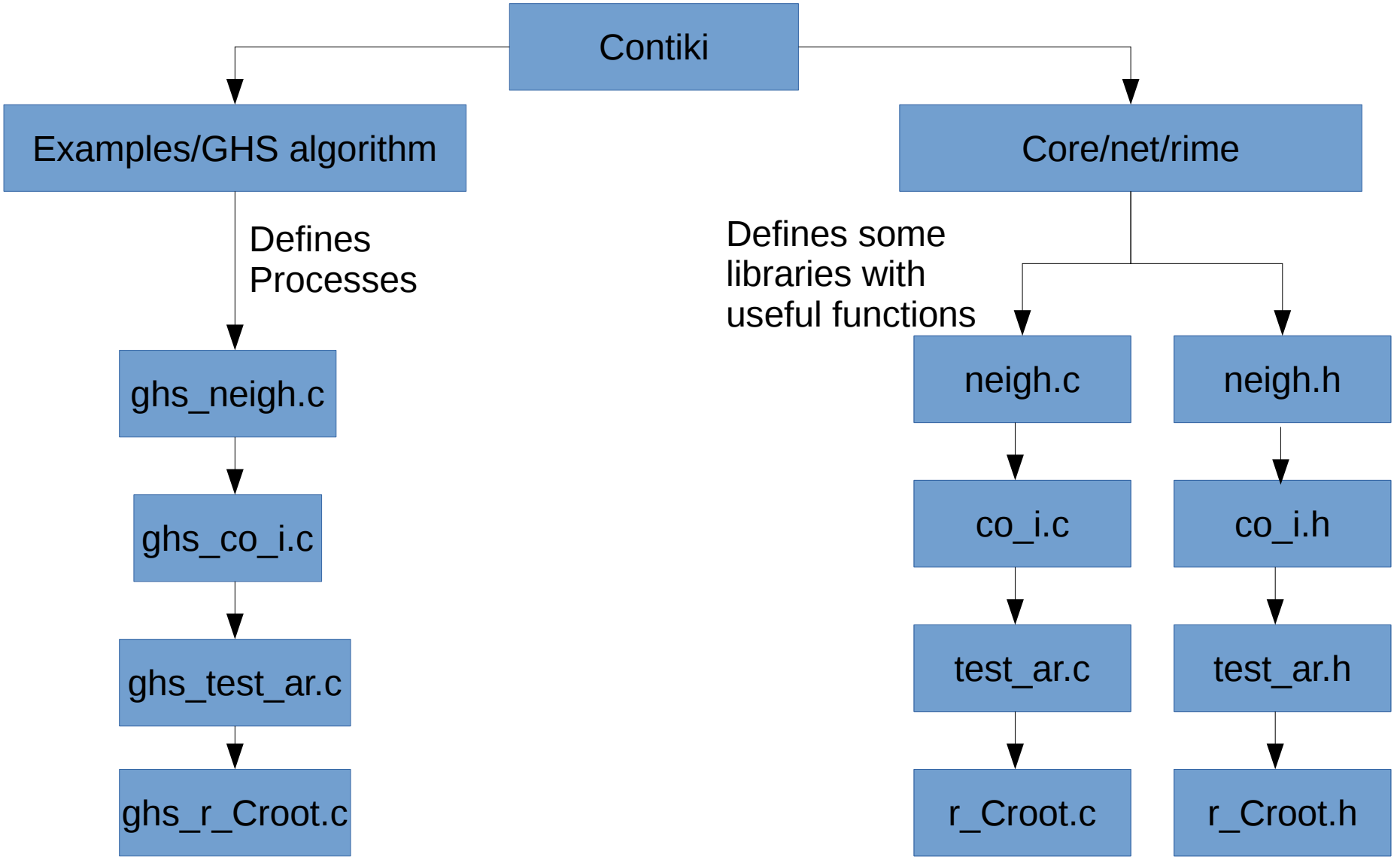
- **Rime**:
  - It provides a hierarchical set of wireless network protocols
  - It provides from a simple anonymous broadcaster to a mesh network routing



GHS implementation in Contiki OS - Rime

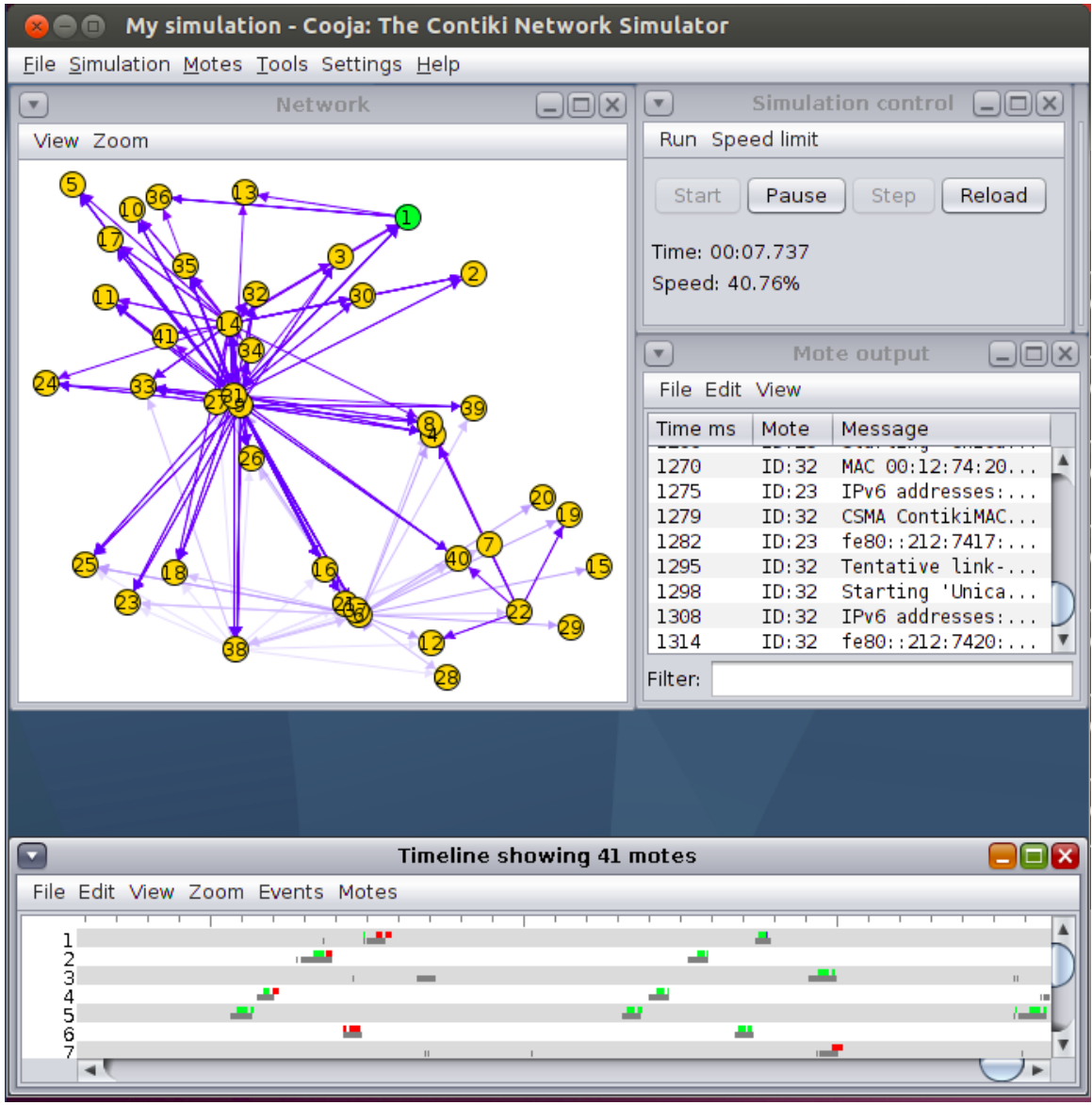


GHS implementation in Contiki OS - Rime



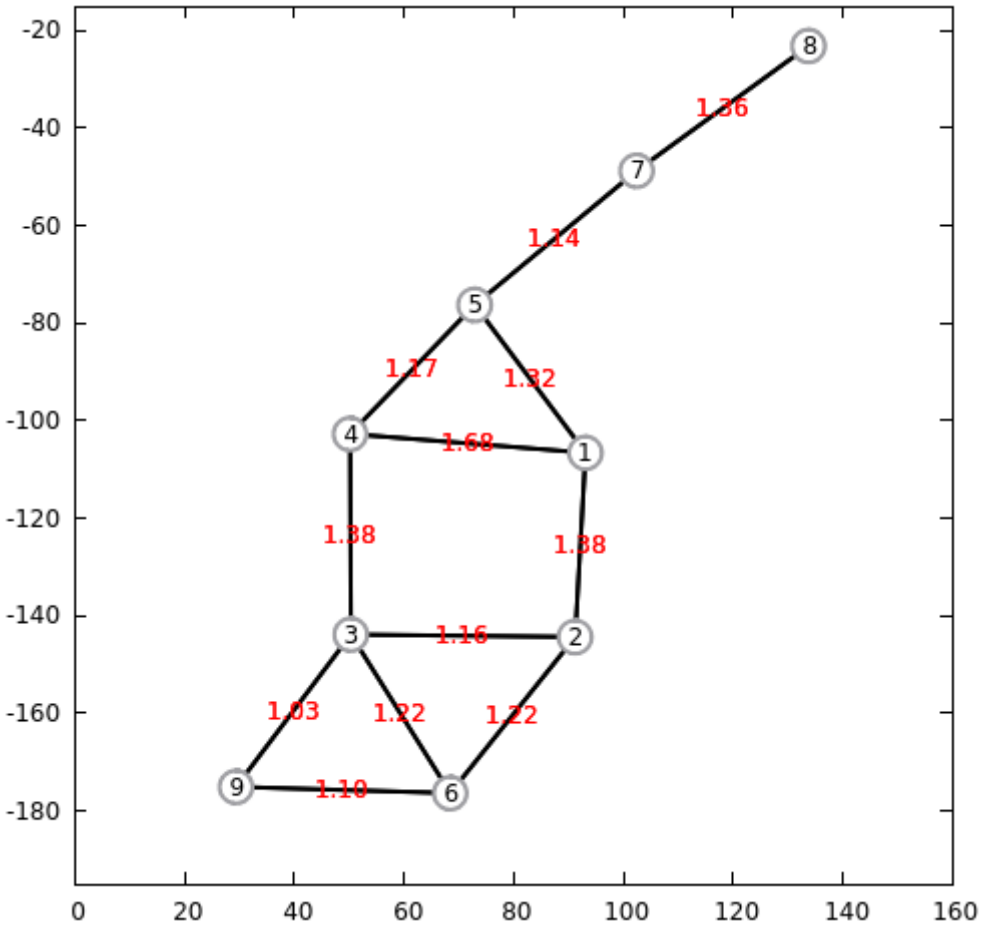
GHS implementation in Contiki OS - Cooja

- **Cooja**
  - It is the Contiki network emulator.
  - A simulated node in Cooja is an actual compiled and executing Contiki system.

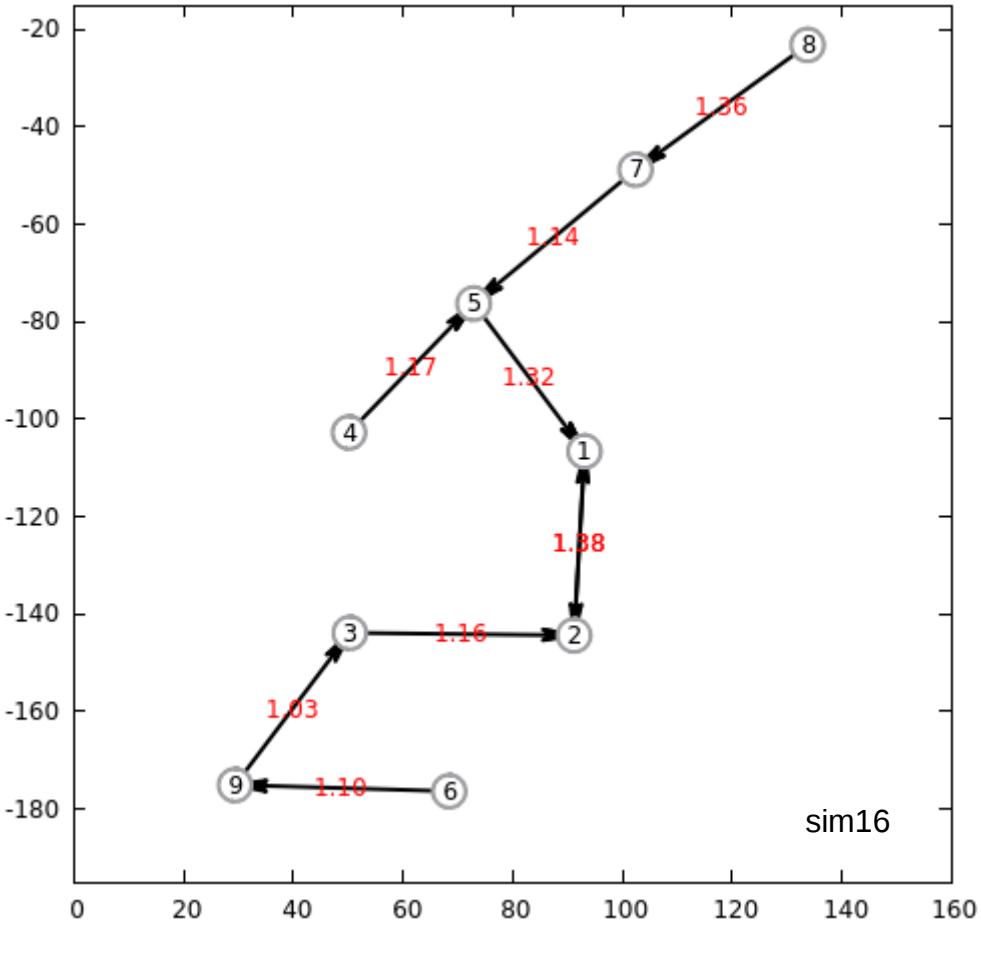


GHS Results

Graph

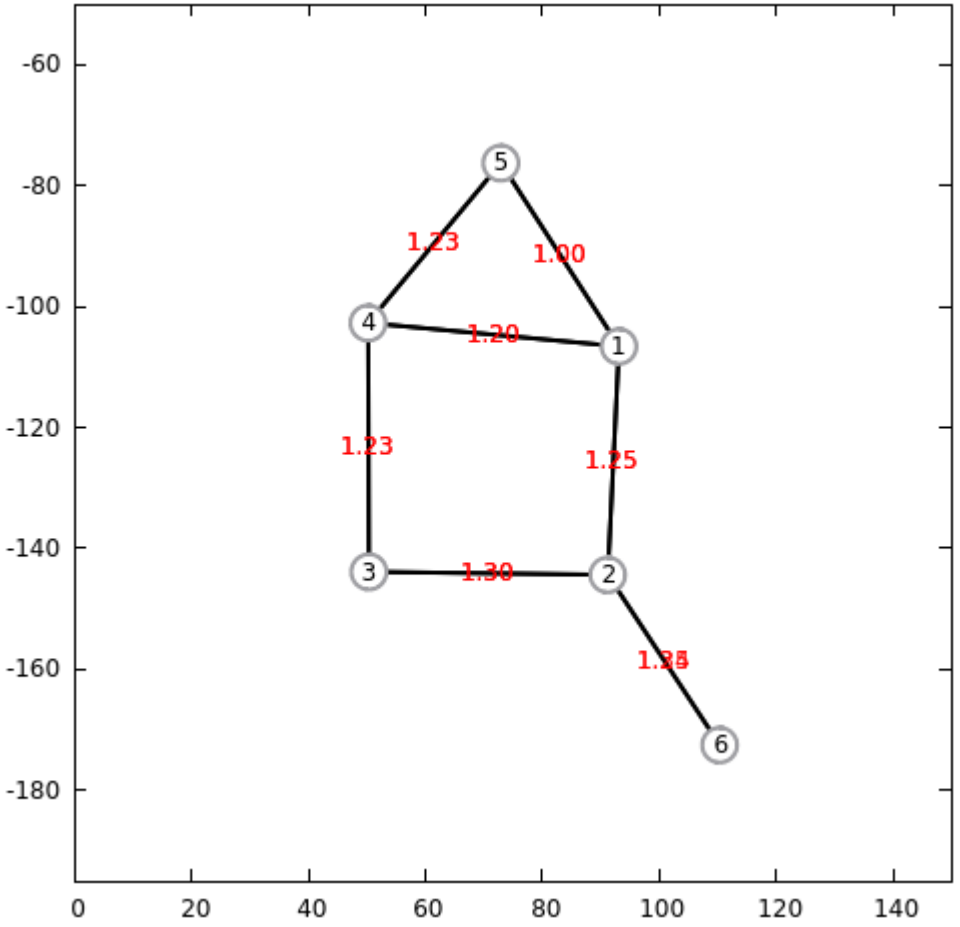


Minimum Spanning Tree

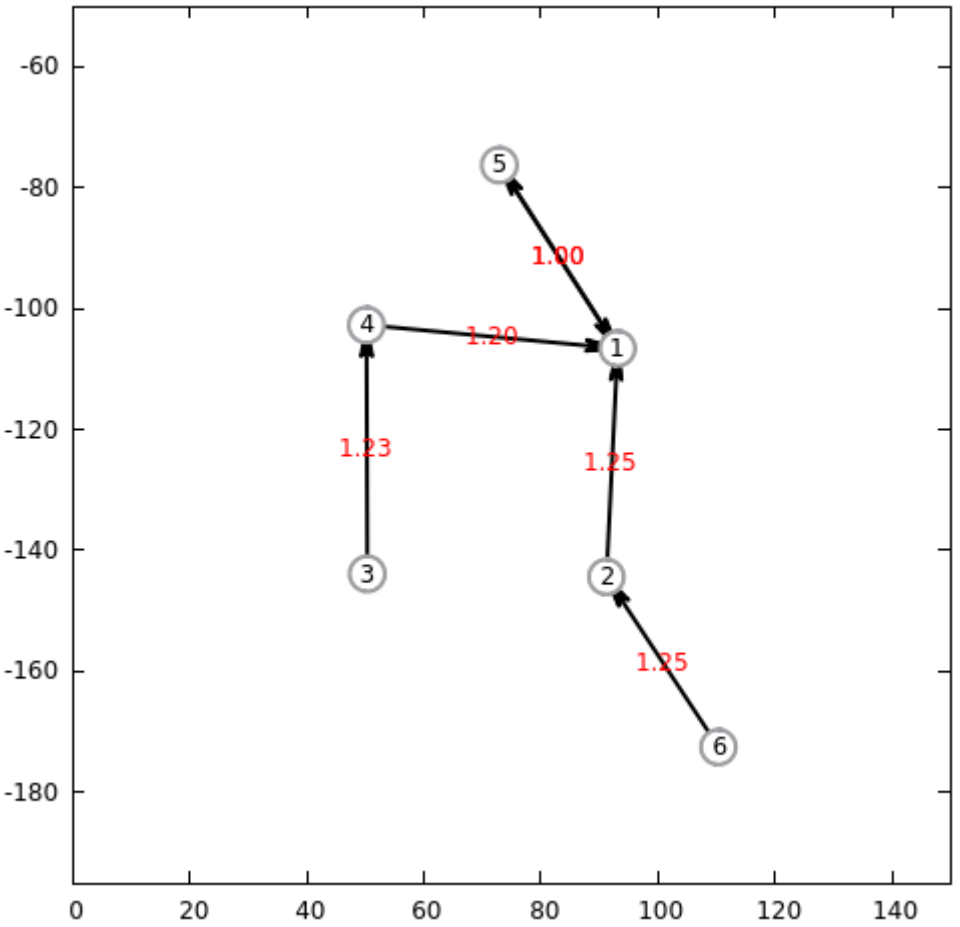


GHS Results

Graph

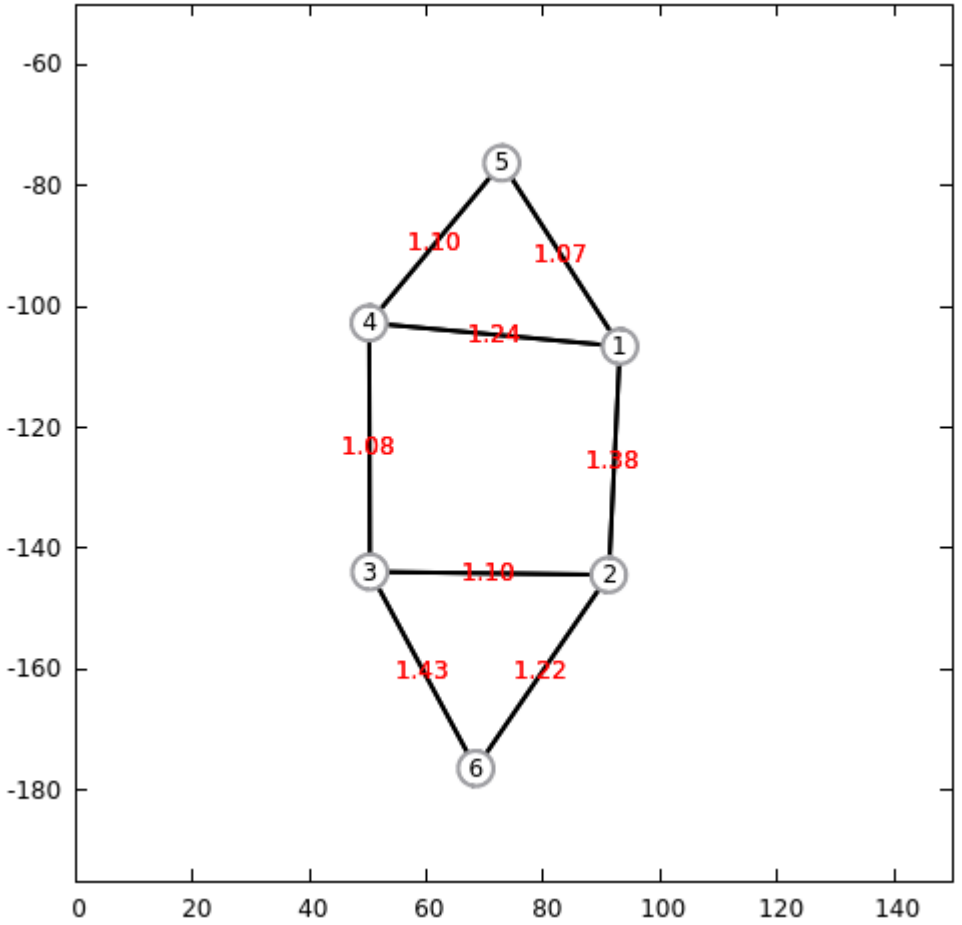


Minimum Spanning Tree

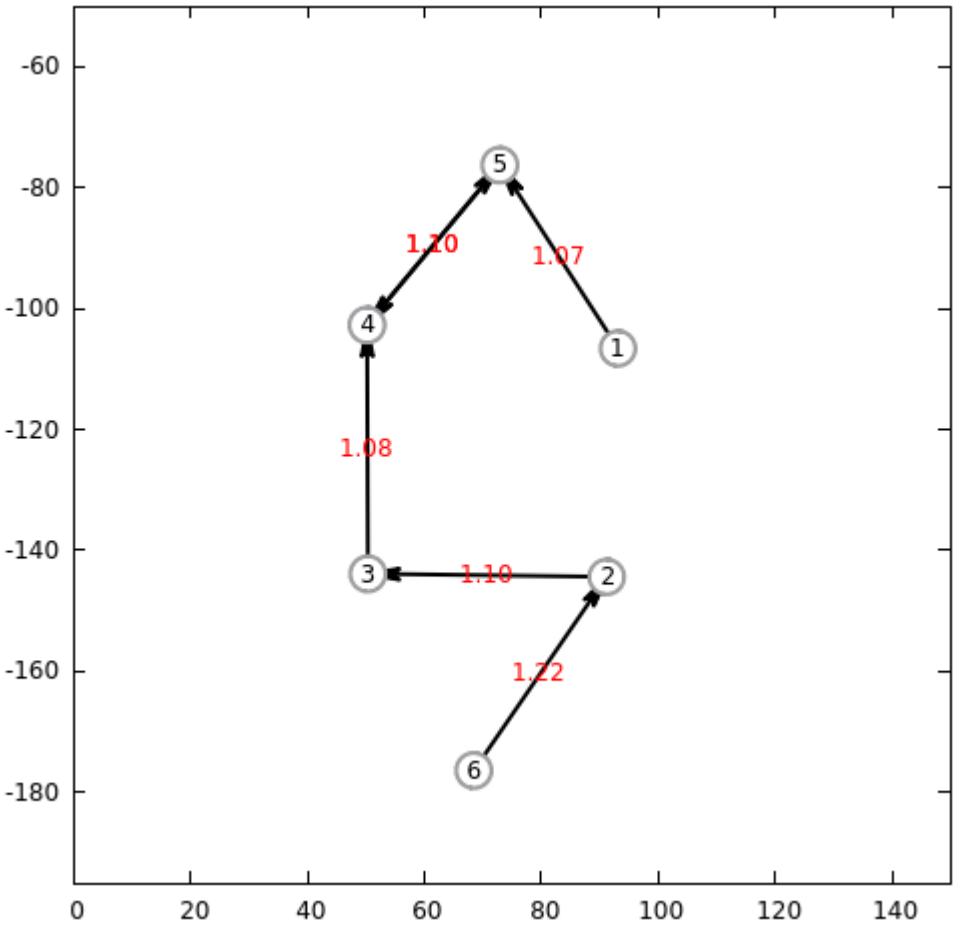


GHS Results

Graph

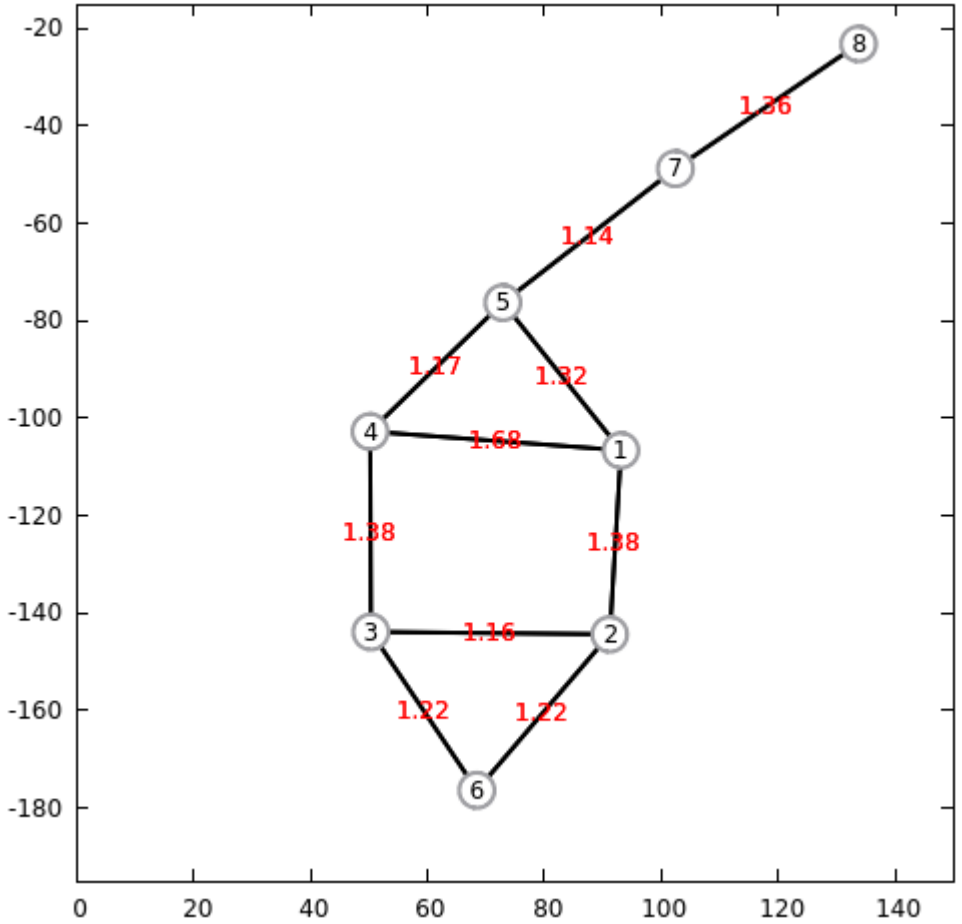


Minimum Spanning Tree

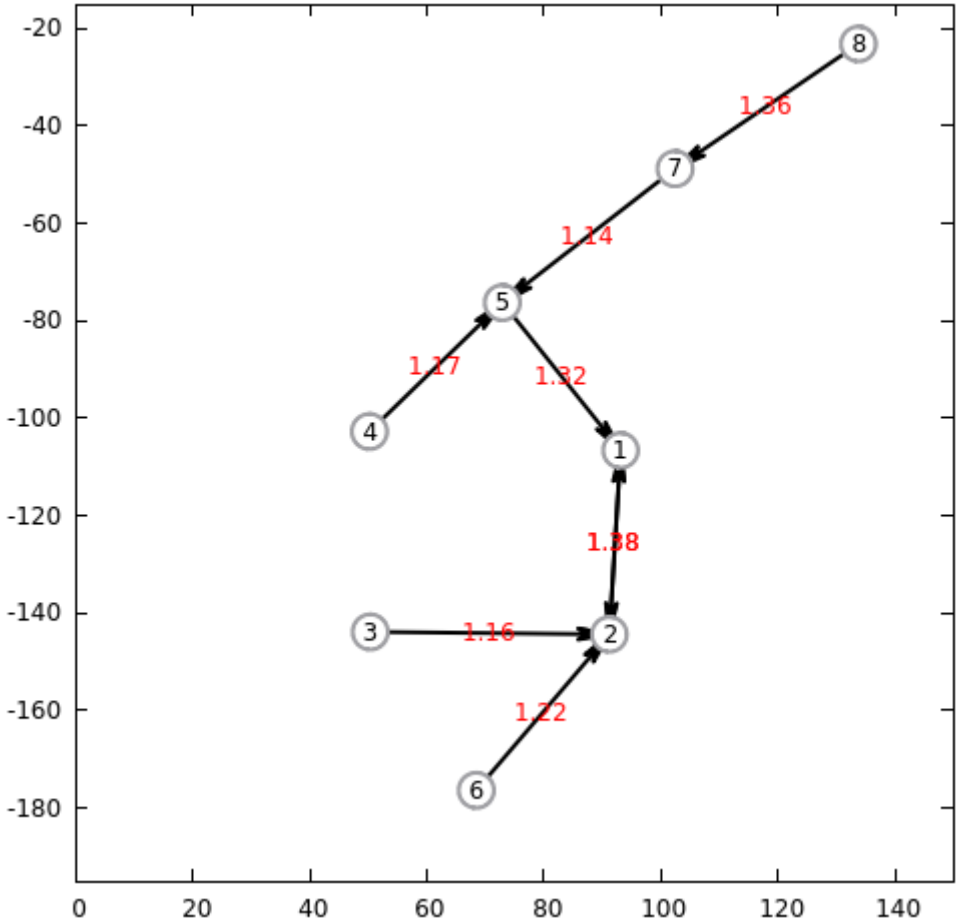


GHS Results

Graph

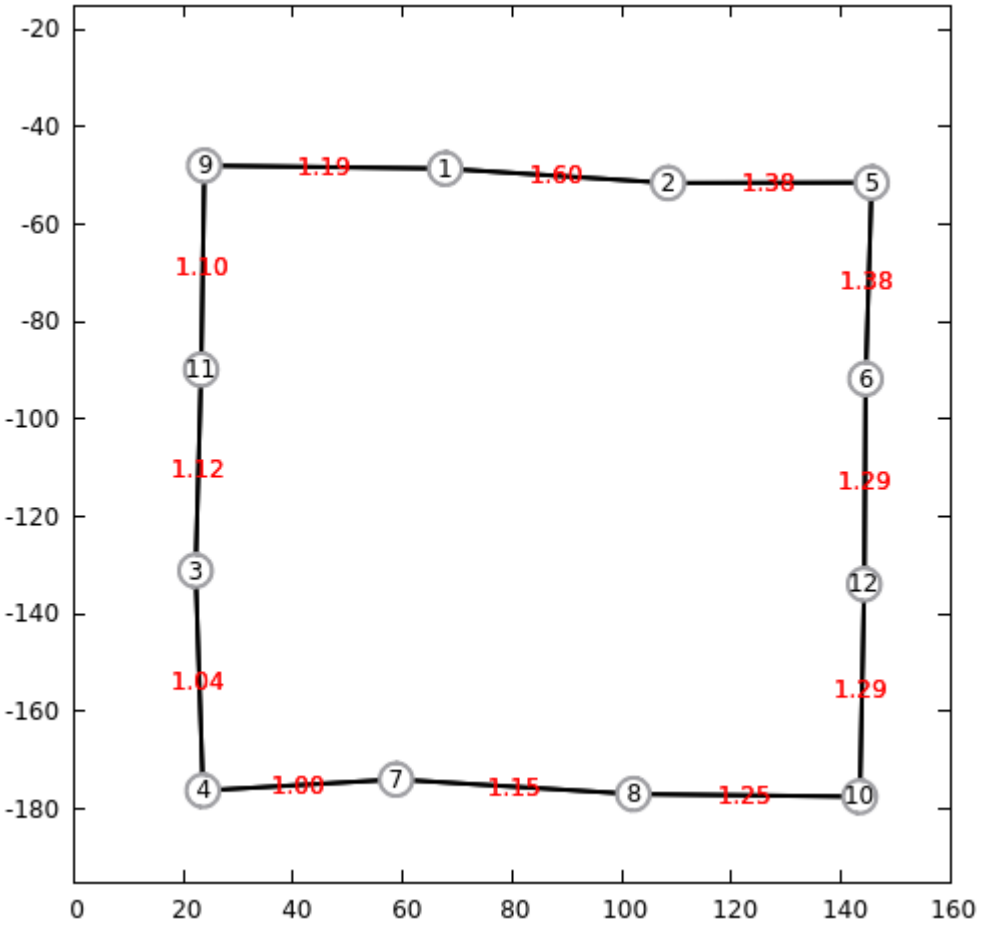


Minimum Spanning Tree

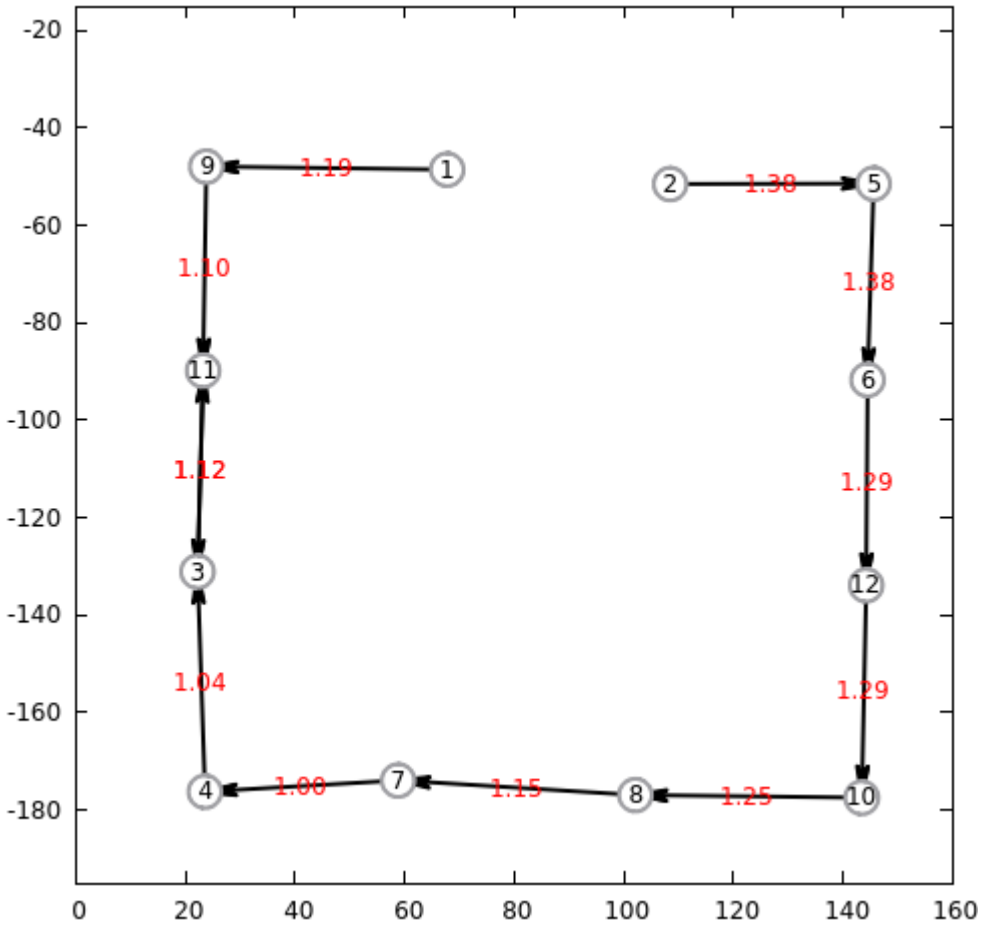


GHS Results

Graph



Minimum Spanning Tree




ANY
QUESTIONS?