CSCI 460 — Operating Systems

Lecture 16

Distributed Minimum Spanning Tree

GHS (Gallager, Humblet and Spira)
1. Distributed Minimum Spanning Tree

• Problem
  – 1. Nodes are the vertices of a weighted graph.
  – 2. Each node knows its neighboring nodes, and the weights of the incident edges.
  – 3. No control node.
  – 4. MST is computed by message passing.

• Restriction of the sequential Prim’s and Kruskal’s Algorithm
  – 1. Both require processing one vertex (edge) at a time, making it difficult to run them in parallel.
  – 2. Both need to know the whole state of the graph, making it difficult in the message-passing model.

• GHS is the first such algorithm
2. Overview

- **Preconditions:**
  - 1. The graph is undirected and connected.
  - 2. Edge weights are all distinct (ties can be broken by the IDs).
  - 3. Initially each node knows the weight of each incident edge.
  - 4. Initially each node is in the quiescent (sleeping) state and it either spontaneously awakens or is awakened by an incoming message.
  - 5. Messages can be transmitted in both directions on an edge and arrive after some limited delay, with no error.
  - 6. Each edge delivers messages in the FIFO order.

- **Properties of MST:**
  - 1. Define a subtree of the MST $T$ as a *fragment*. Given a fragment of an MST $T$, let $e$ be a minimum-weight outgoing edge of the fragment.
  - 2. Then, joining $e$ and its adjacent non-fragment node yields another fragment of $T$.
  - 3. If all the edges of a connected graph $G$ have different weights, then the MST of $G$ is unique.
3. Description of the algorithm

• A non-decreasing level number is assigned to each fragment, initially with a value 0. Each non-zero level fragment has an ID, which is the ID of the core edge in the fragment, which is selected when the fragment is first built. During the running of the algorithm, each node can classify each of its incident edges into 3 categories:

  – 1. Branch edges are those in the MST.
  – 2. Rejected edges are those not in the MST.
  – 3. Basic edges are undecided ones (neither branch nor rejected edges).

• For level-0 fragments, each awakened node will do the following:
  – 1. Choose its minimum-weight incident edge and mark that edge as a branch edge.
  – 2. Send a message via the branch edge to notify the node on the other side.
  – 3. Wait for a message from the other end of the edge.

• The edge chosen by both nodes it connects becomes the core with level 1.

• For a non-zero level fragment, the algorithm can process it in three stages:
1. **Broadcast**: The two nodes adjacent to the core broadcast messages to the remaining nodes in the fragment. These messages are sent via the branch but not the core edge. Each broadcast message contains the ID and the level of the fragment. At the end of this stage, each node receives the new fragment ID and the level.

2. **Convergecast**. At this stage, all the nodes in the fragment cooperate to find the minimum weight outgoing edge of the fragment. The messages sent at this stage are in the opposite direction of the broadcast stage. Initialized by all the leaves (the nodes having only one branch edge), a message is sent through the branch edge. The message contains the minimum weight of the incident outgoing edge it found ($+\infty$ if no such edge exists). *Details will be covered a bit later.* For each non-leaf node, let the number of its branch edges be $n$. Then after receiving $n - 1$ convergecast messages, it will pick the minimum weight from the messages and compare it to the weights of its incident outgoing edges. The smallest weight will be sent toward the branch it received the broadcast from.

3. **Change core**. After the Convergecast stage, the two nodes connected by the core can inform each other of the best edges they received. Then they can identify the minimum outgoing edge from the entire fragment. A message will be sent from the core to the minimum outgoing edge via a path of branch edges. Finally, a message will be sent
out via the chosen outgoing edge to request to combine the two fragments that the edge connects. Depending on the levels of those two fragments, one of the two combined operations are performed to form a new fragment (details will be discussed below).
4. How to find the minimum weight incident outgoing edge?

- As discussed above, every node needs to find its minimum weight outgoing incident edge after the receipt of a broadcast message from the core. If node $x$ receives a broadcast, it picks its minimum weight basic edge and sends a message to the node $x'$ on the other side with its fragment’s ID and level. Then node $x'$ will decide whether the edge is an outgoing edge and send back a message to notify node $x$ of the result. The decision is based on the following:

- 1. **Case 1:** Fragment_ID($x$) = Fragment_ID($x'$). Then, node $x$ and $x'$ belong to the same fragment (so the edge is not outgoing).

- 2. **Case 2:** Fragment_ID($x$) \neq Fragment_ID($x'$) and Level($x$) \leq Level($x'$). Then, node $x$ and $x'$ belong to different fragments (so the edge is outgoing).

- 3. **Case 3:** Fragment_ID($x$) \neq Fragment_ID($x'$) and Level($x$) > Level($x'$). Then, we cannot make a decision. In this case, the algorithm lets node $x'$ postpone the response until its level \geq the level it received from node $x$. 
5. How to combine two fragments?

- Let $F$ and $F'$ be the two fragments that need to be combined. There are two ways:
  - 1. **Merge**: This operation occurs if both $F$ and $F'$ share a common minimum weight outgoing edge, and $\text{Level}(F) = \text{Level}(F')$. The level of the combined fragment will be $\text{Level}(F') + 1$.
  - 2. **Absorb**: This operation occurs if $\text{Level}(F) < \text{Level}(F')$. The combined fragment will have the same level as $F'$.

- When an “Absorb” operation occurs, $F$ must be at the stage of core changing while $F'$ can be at any stage. Let $e$ be the edge that $F$ and $F'$ want to combine with and let $x$ and $x'$ be the two nodes connected by $e$ in $F$ and $F'$ respectively. There are two cases:
  - 1. **Case 1**: Node $x'$ has received a broadcast message, but it has not sent a convergecast message back to the core. Then, fragment $F$ can join the broadcast process of $F'$.
  - 2. **Case 2**: Node $x'$ has already sent a convergecast message back to the core.

- In summary, fragments are combined by either the “Merge” or “Absorb” operation. The “Absorb” operation doesn’t change the maximum level among all fragments. The “Merge” operation may increase the maximum level by 1. In the worst case,
all fragments are combined by “Merge” operations, so the number of fragments decreases by half at each level. Therefore the maximum number of levels in $O(\log V)$, where $V$ is the number of nodes in the graph.

- As the fragments with the lowest level will not be blocked, and the maximum level cannot decrease, it is easy to see that the algorithm will terminate with an MST.