A Distributed Algorithm for Spanning Trees

R. G. GALLAGER, P. A. HUMBLET, and P. M. SPIRA

present by Robert Pate and My Luc

Overview

- Introduction
- Spanning trees and the high level of MST algorithm
- Description of MST distributed algorithm
- Communication cost analysis
- Time analysis

Introduction

- Connected undirected graph with N nodes and E edges (distinct weights)
- Each node knows the weight its edges
- Cost of sending message in both directions is the same

-> asynchronous distributed algorithm to determine the minimum-weight spanning tree (MST)

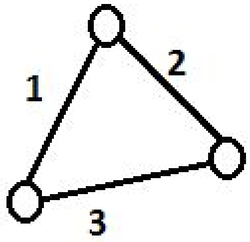
Applications

- Broadcast information in communication networks
- Generate spanning tree from any node when network failure
- Find highest number node in network

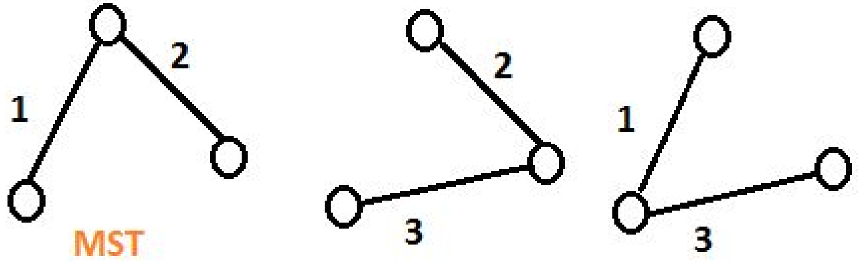
What is minimum spanning tree?

- An undirected graph G = (V, E)
- Spanning tree of G: a subset of the edges of G that connect all vertices without any cycle
- Minimum spanning tree: a spanning tree with the minimum total edge weight

Graph G:

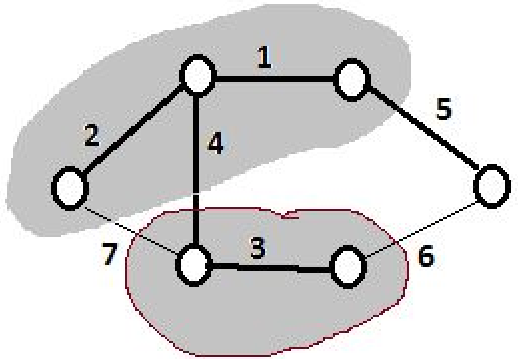


Spanning trees of G:



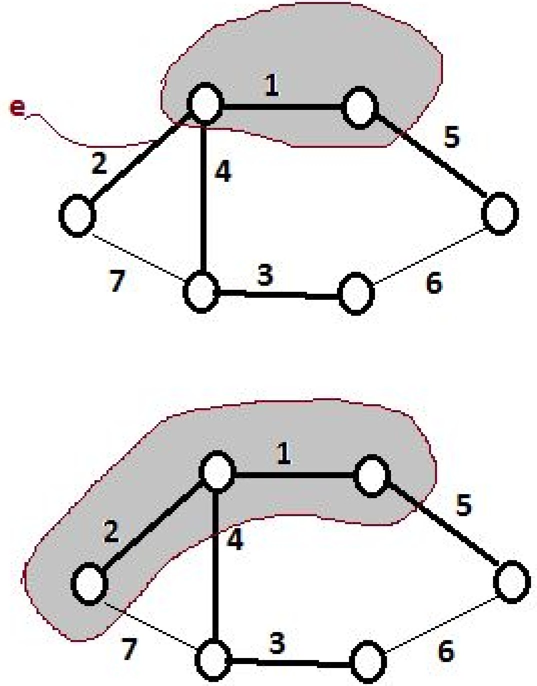
MST fragment

- A fragment of MST: a subtree of the MST
- Outgoing edge of a fragment if one adjacent node is in the fragment and the other is not.



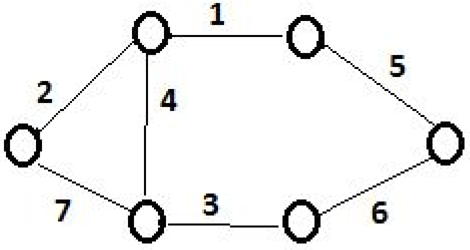
MST Property 1

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



MST property 2

If all the edges of a connected graph have different weights, then the MST is unique.



Distributed MST

- Each fragment finds its minimum-weight outgoing edge
- Then it tries to combine the fragment at the other edge
- How and when to combine depending on the levels of the two fragments

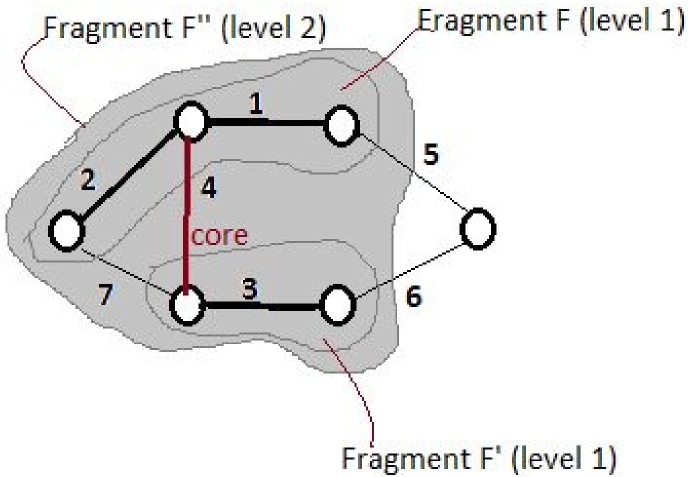
Distributed MST - Level

- Fragment with single node has level 0
- Fragment F (level L) wants to connect to fragment F’ (level L’):

If L < L’, F is absorbed as part of F’ becoming new fragment at level L’

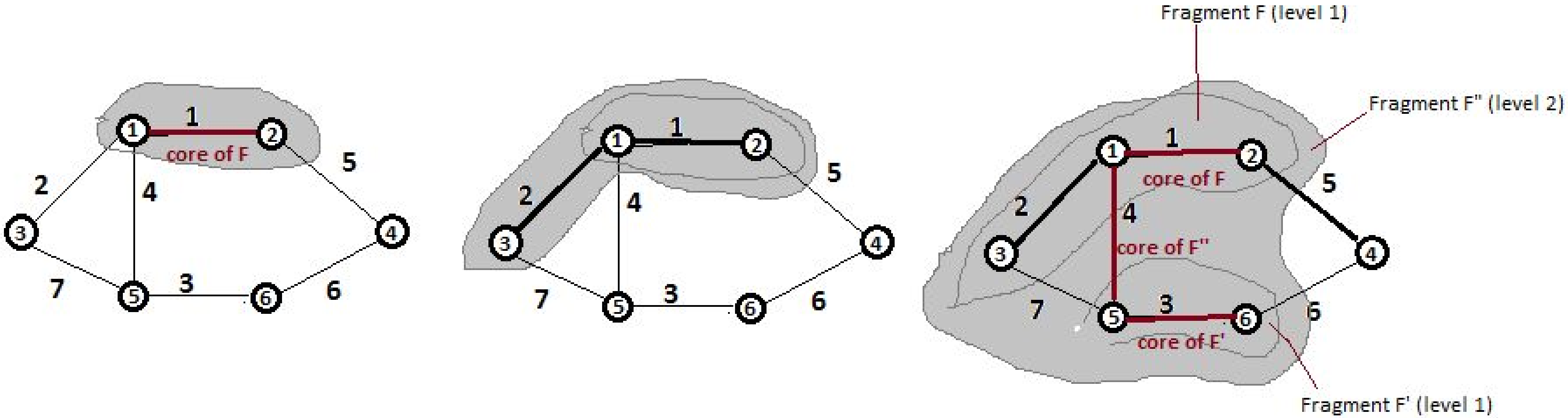
If L = L’, F and F’ combine into new fragment at level L +1. Combining edge is called core of the new fragment.

- Identity of fragment is the weight of its core

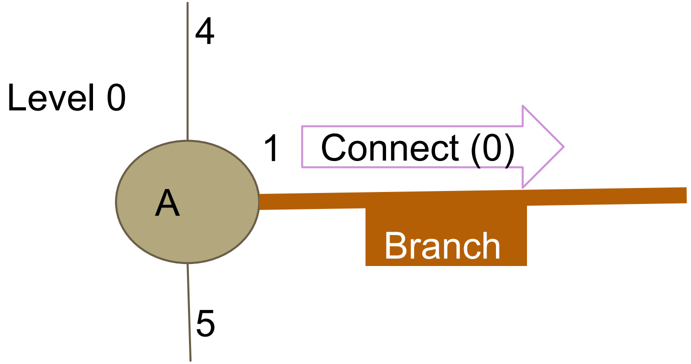


Distributed MST

- Node 1 and node 2 combine on their common minimum weight edge
- Node 3 and its minimum weight edge are then absorbed -> fragment F
- Node 5 and node 6 combine on their common minimum weight edge -> fragment F’
- F and F’ combine on their minimum weight edge to form level 2 fragment F”
- Node 4 can be absorbed to F or F” depending on the timing

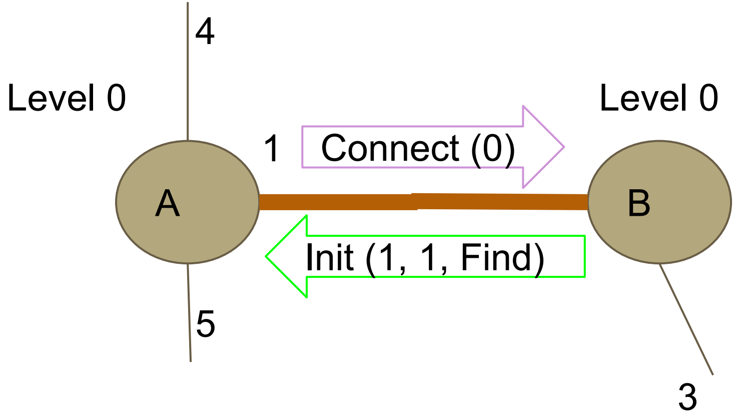


Part 3: Description - Wake a Single Node



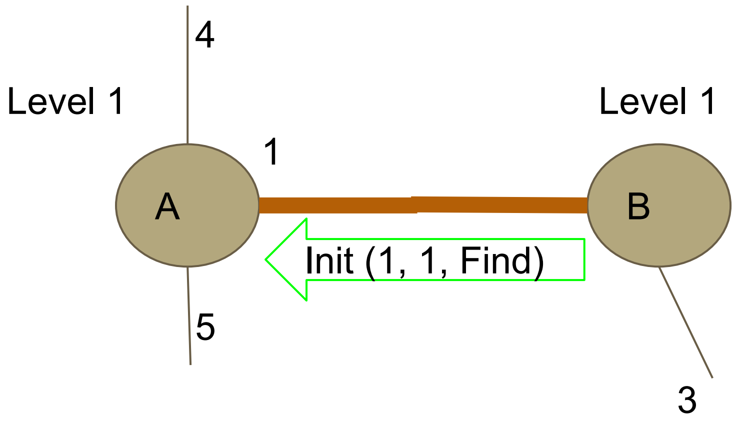
1. Awaken
2. Mark the shortest edge as a Branch
3. Send Connect *(LN)* to the node on it
    a. *LN* is the node level, starts at 0
4. Change state to Found

Part 3: Description - Two Nodes



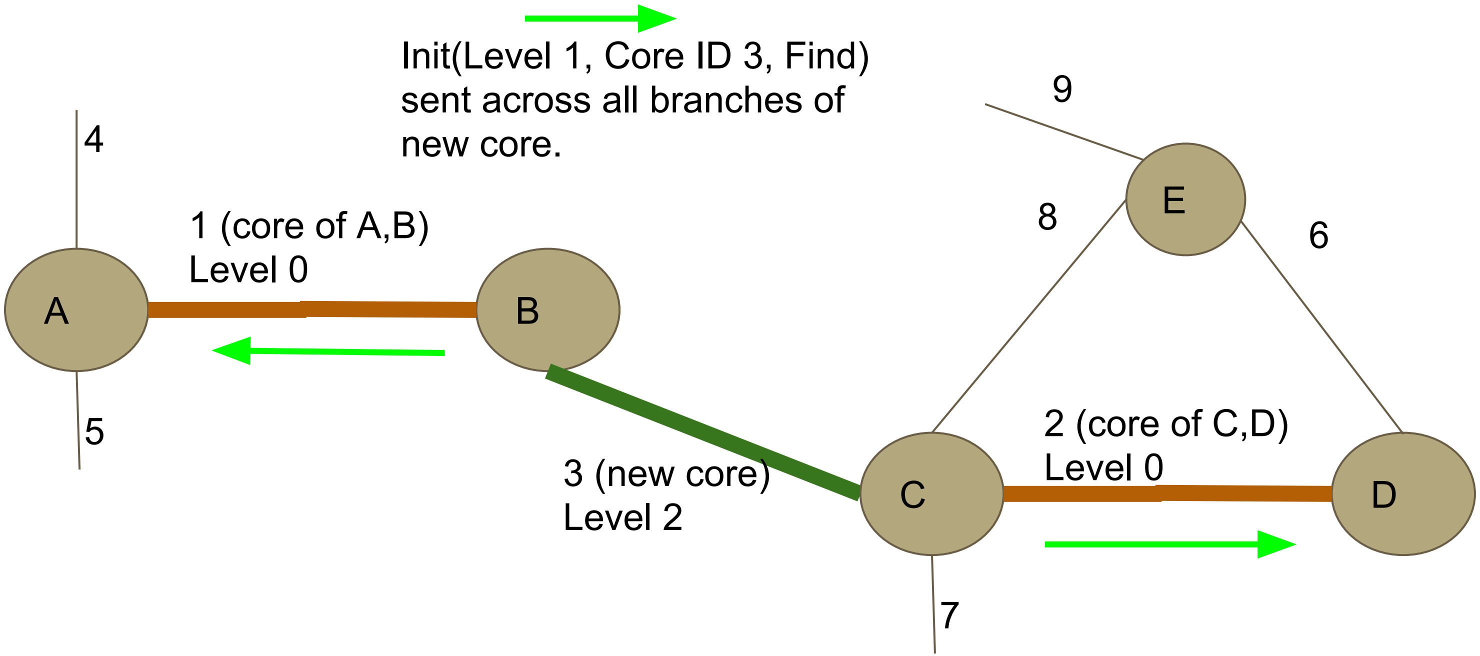
1. Connect(0) wakes Node B
2. Repeat the previous wakeup process
3. Run the “On Receive Connect” Process:
    a. Test if the sent level is less than current
    b. Test if the edge is marked as basic
    c. Else send Init(Level 1, Weight 1, Find)

Part 3: Description - Two Nodes

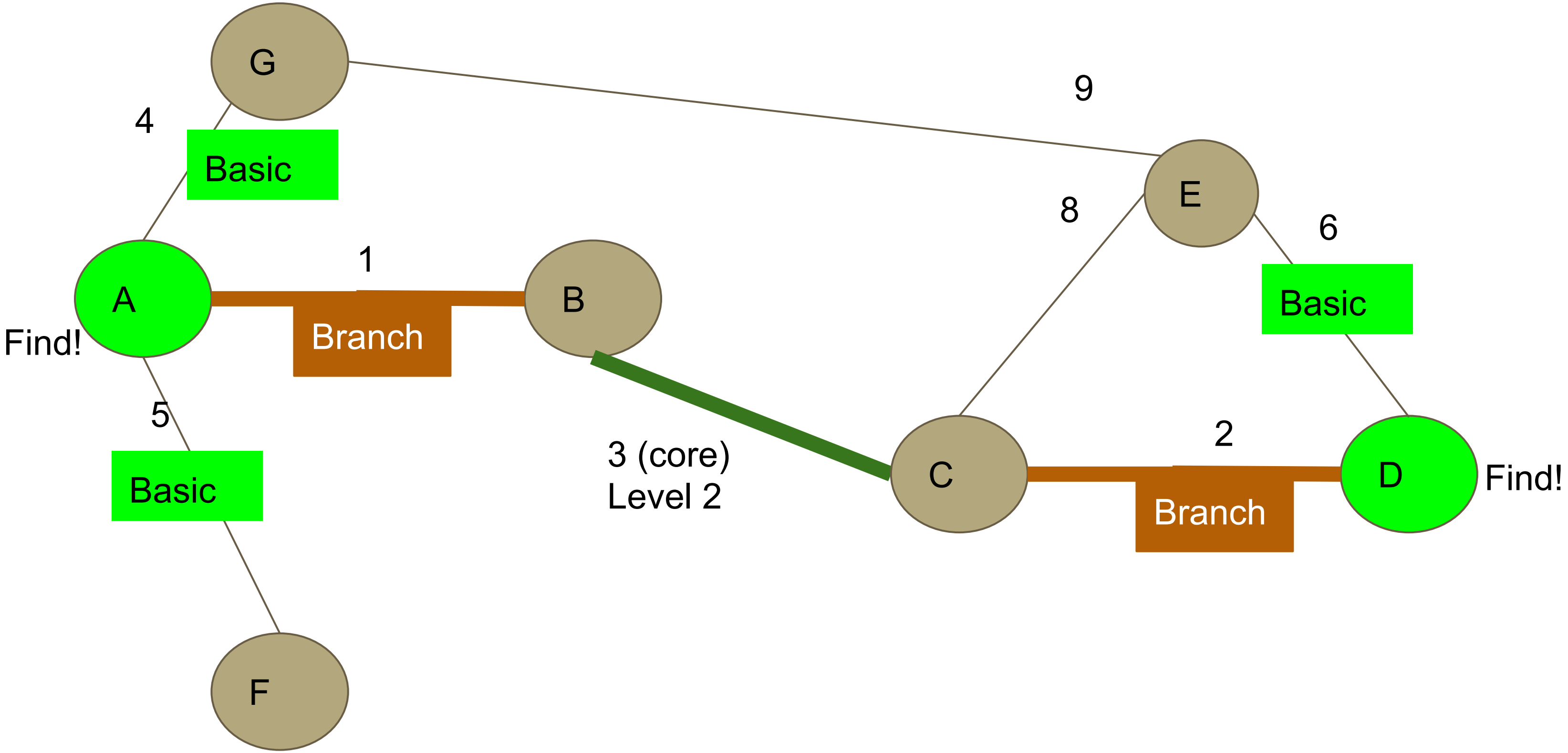


1. Node A receives the Init() from B
2. Accepts Branch 1 as its fragment identity
3. Proceeds to Init() and Test() other edges
    a. More on Test() in a minute!

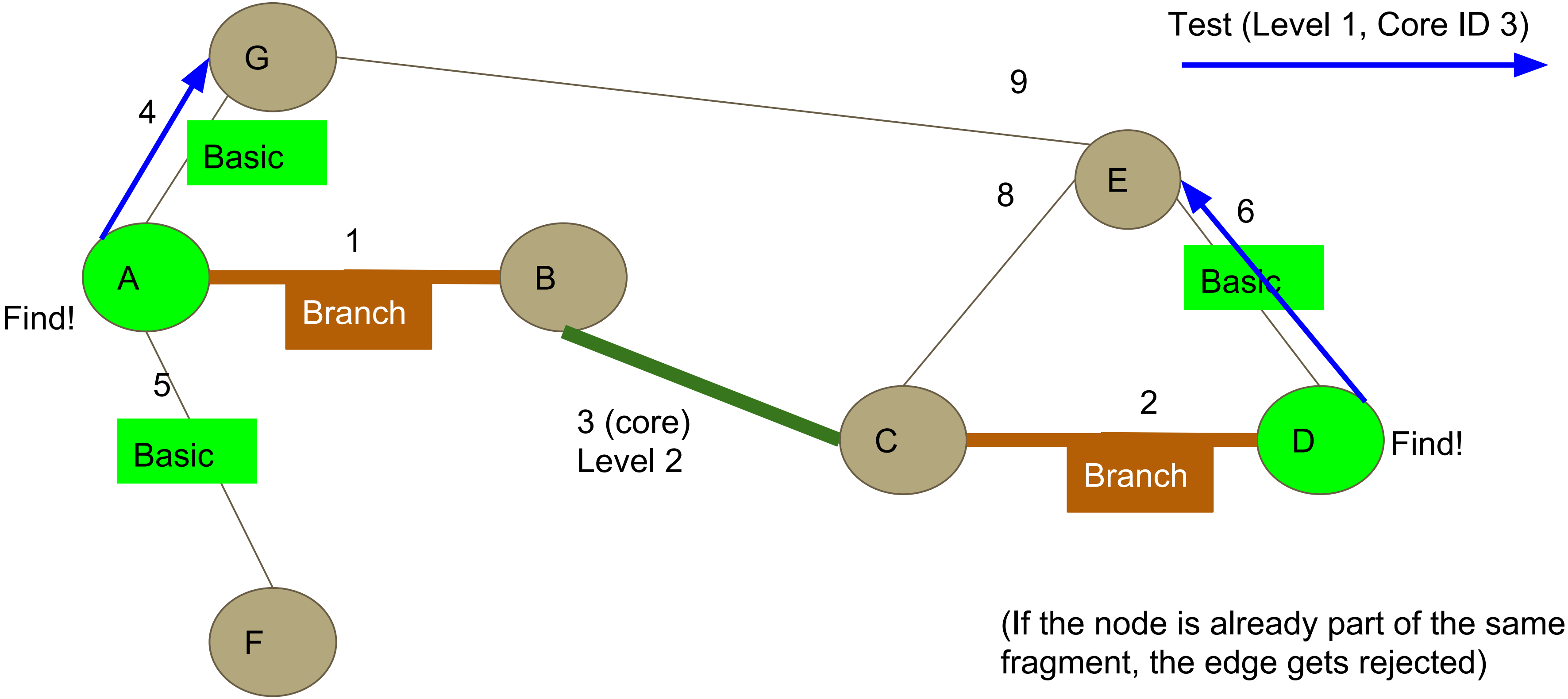
Part 3: Description - Two Fragments and Initiate



Part 3: Description - Find Min-Outgoing Edges

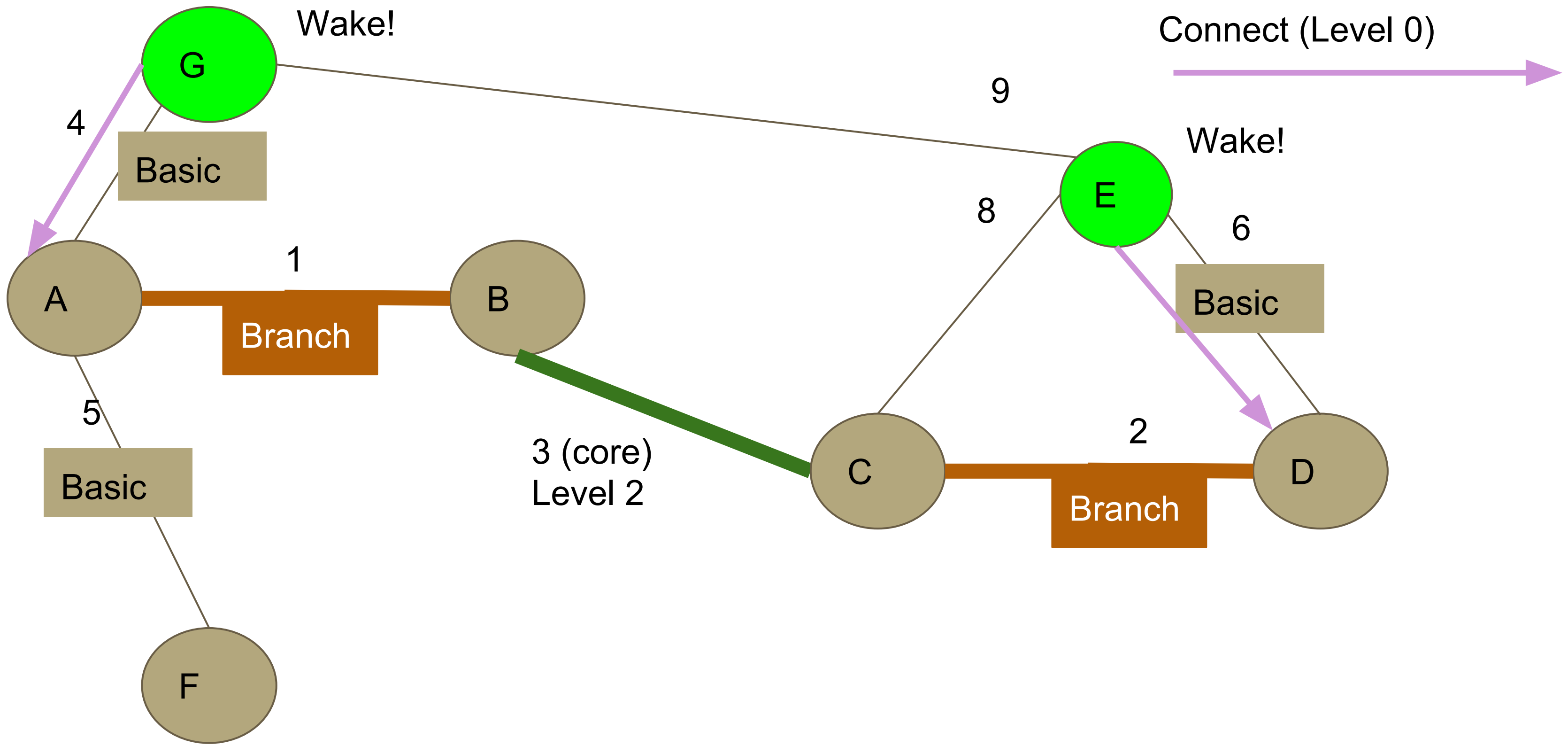


Part 3: Description - Test Basic Branches



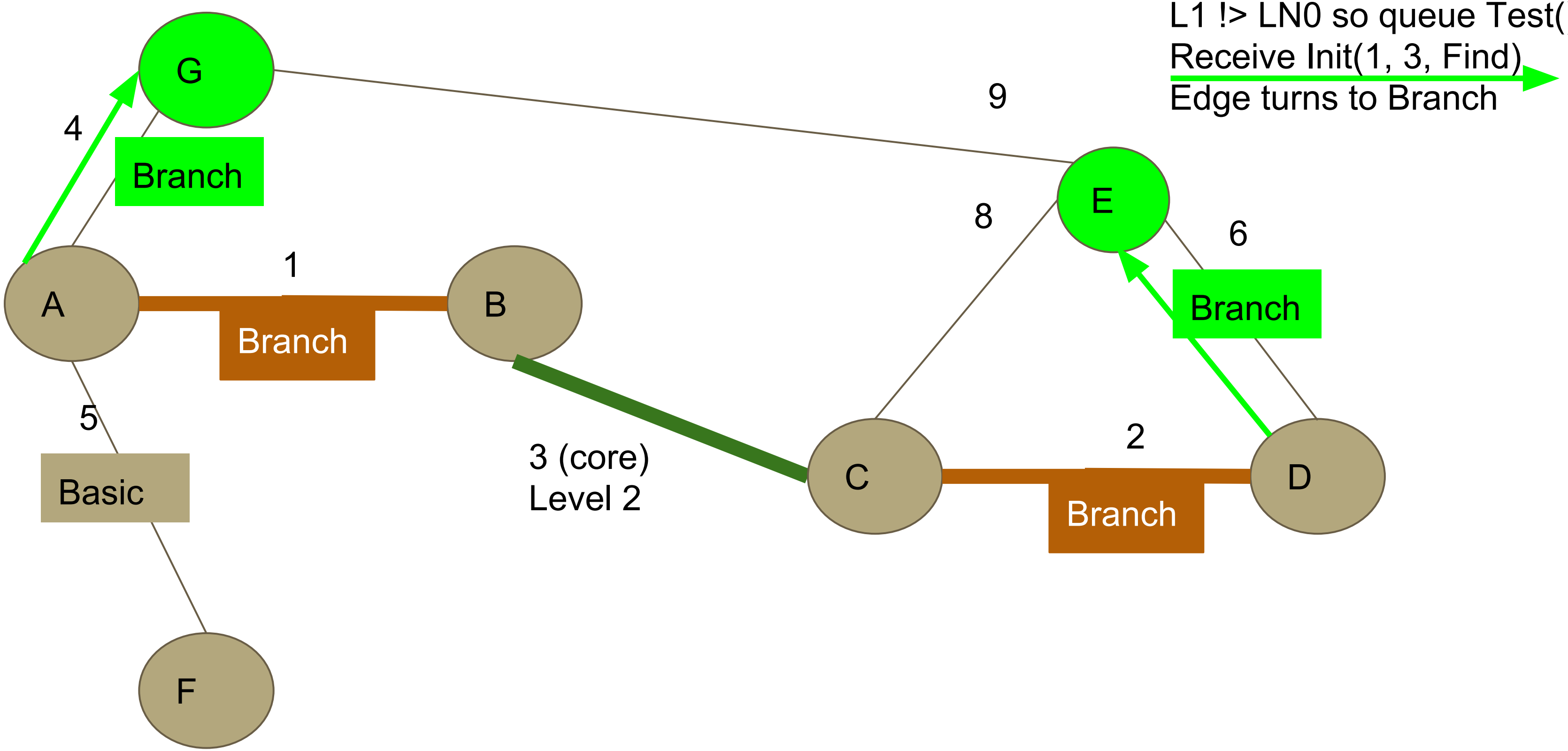
(If the node is already part of the same fragment, the edge gets rejected)

Part 3: Description - Wake and Connect

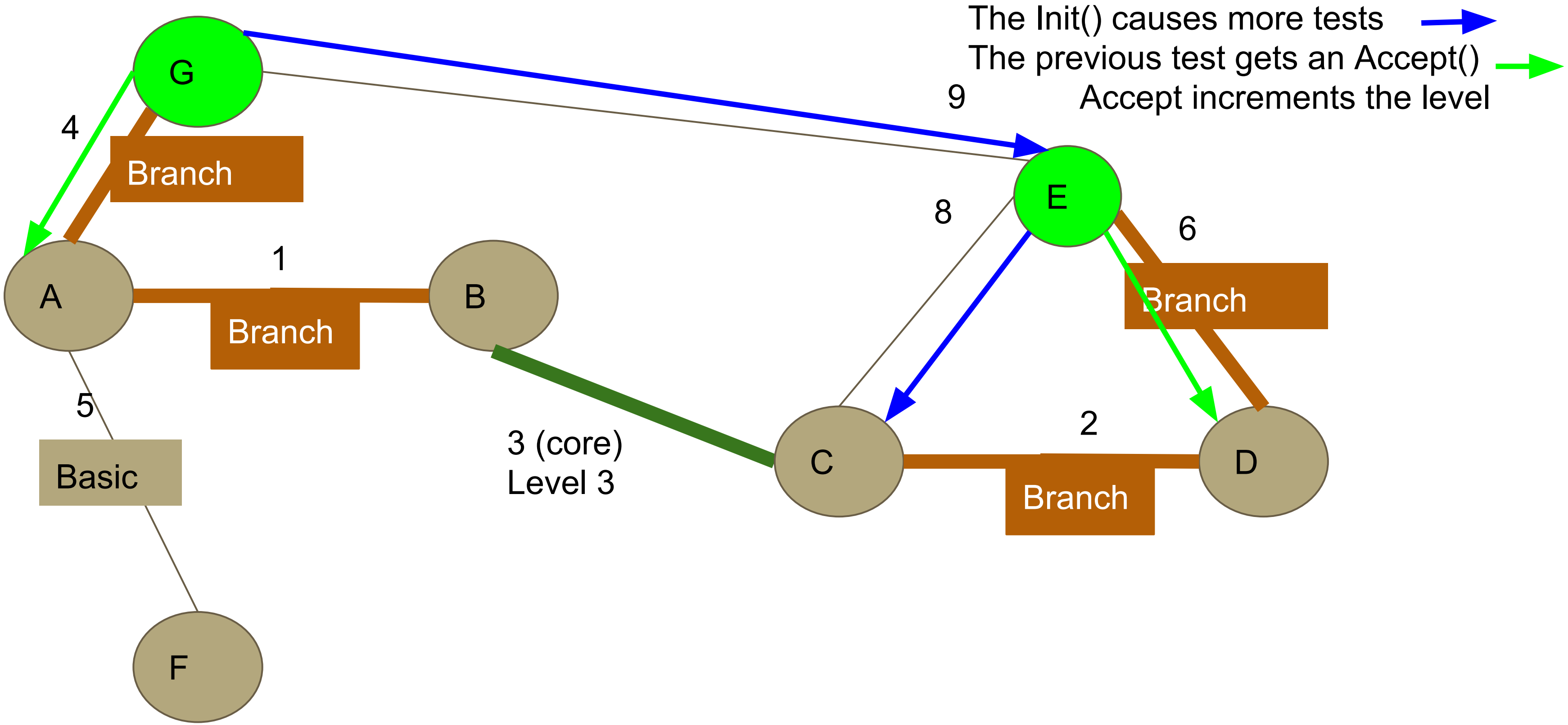


Part 3: Description - Init and Test

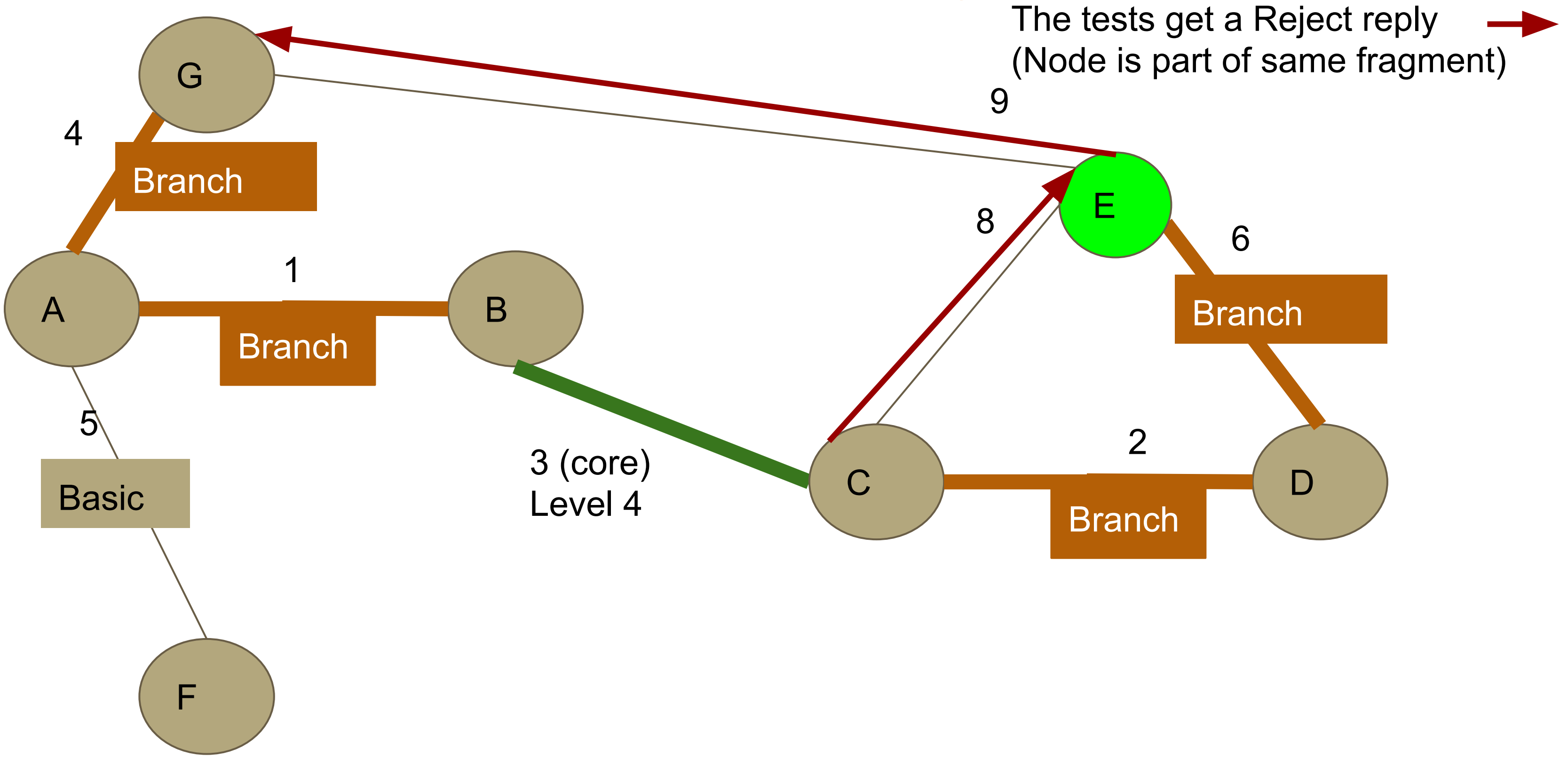
L1 !> LN0 so queue Test();
Receive Init(1, 3, Find)
Edge turns to Branch



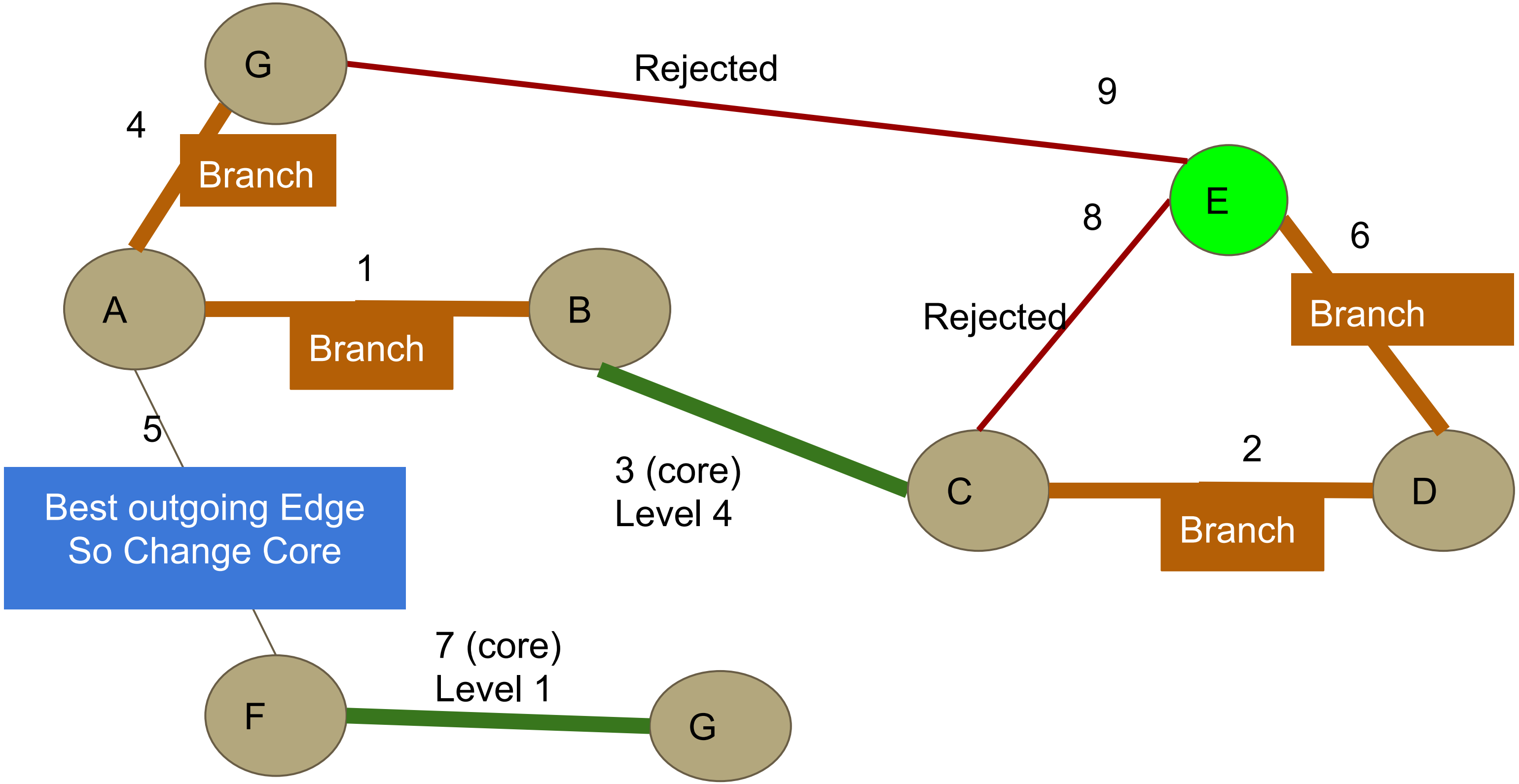
Part 3: Description - Test and Accept



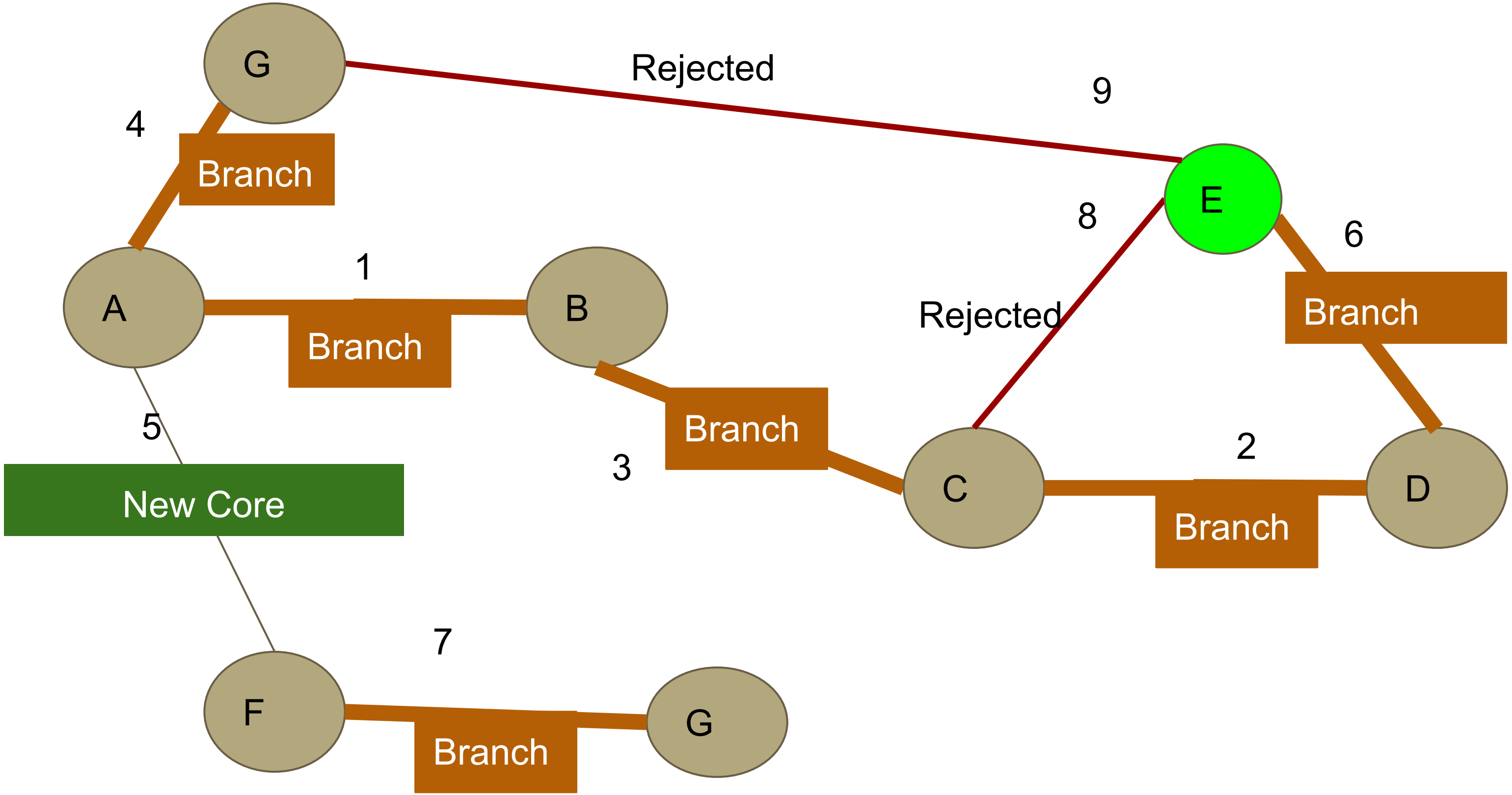
Part 3: Description - Test and Reject



Part 3: Description - Finding Outgoing Edge



Part 3: Description - Finding Outgoing Edge



Part 4: Communication Cost - Weight

Components of most complex message:

- One edge weight
- A level between zero and *log N*
- A bit representing message type

Part 4: Communication Cost - Count

Message Count

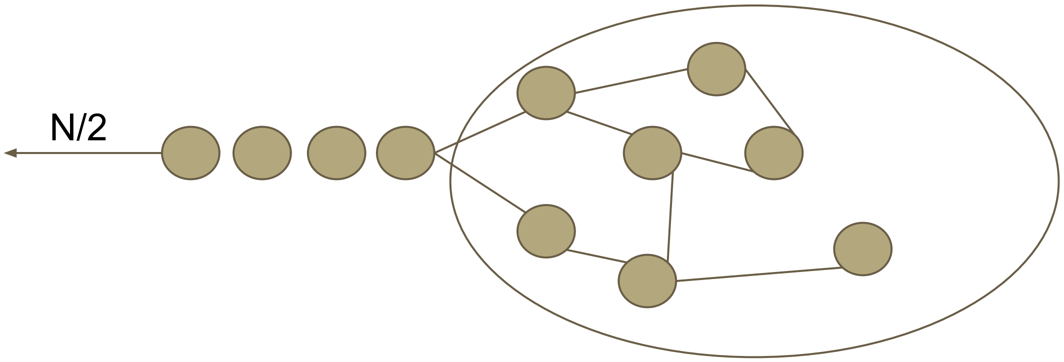
- One rejection per edge and two messages per rejection
  - **2E**
- Five Messages for levels besides first (zero) and last ($\log_2 N$)
  - Init, accept, successful-test, report, change-root/connect
  - **5*N*(-1 + log *N*)**
- First and Last
  - First: Init, Connect
  - Last: Report
  - Both **less than 5, so call it 5 for simplicity**

Maximum:

5*N* log *N* + 2*E*

Part 5: Timing Analysis

- Waking one at a time could lead to at worst $N(N - 1)$ sequential messages
- Waking all is better, $N\text{-}1$ time to wake all and $5N \, Log_2 N$ to complete.
- Worst case is a graph split equally into a handle and head: $O\,(N \log N)$



Sampling of Related Papers 1

"O jistém problému minimálním"
Jarník, V. (1930)

"Shortest connection networks And some generalizations"
Prim, R. C. (1957)

"A note on two problems in connexion with graphs"
Dijkstra, E. W. (1959)

Sampling of Related Papers 2

“Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and related problems”
Awerbuch, Baruch (1987)

“A Highly Asynchronous Minimum Spanning Tree Protocol”
Singh, Gurdip and Bernstein, Arthur J. (1995)

“Distributed Maintenance of a Spanning Tree using Labeled Tree Encoding”
Garg, Vijay K. and Agarwal, Anurag (2005)