#ASLI ENGINEERING

EIG Algorithm for Distributed Consensus

SWIPE

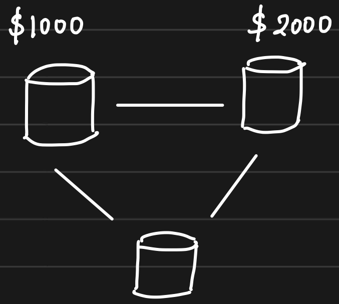
BY

ARPIT BHAYANI

Exponential Information Gathering for distributed consensus

Reaching consensus is extremely important in any distributed network.

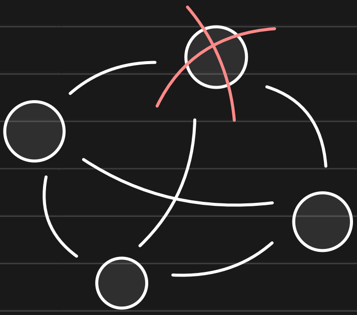
eg: we cannot have two datanodes in a cluster
such that one thinks price = \$1000
while the other thinks price = \$2000



Depending on which node the request hits, the user would see the corresponding value, giving an inconsistent view

Somehow, the nodes need to agree on one value.

Achieving distributed consensus
- is easy when No failures
- is impossible when network unreliable
- and tricky when unreliable process



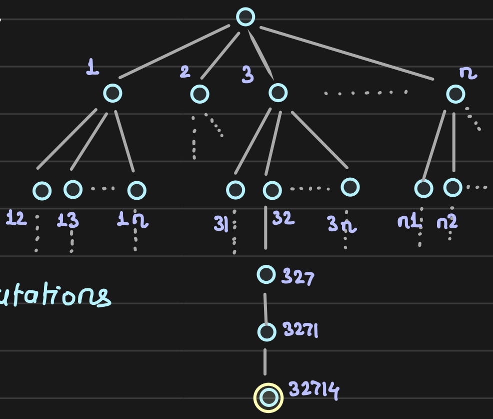
Exponential Information Gathering

Core Idea: Relay the values across rounds, record the communication path, and decide.

ARPIT BHAYANI

EIG Data Structure

EIG data structure is a tree that grows exponentially. The paths from the root of the tree represent the communication path from which the message is received (propagated)



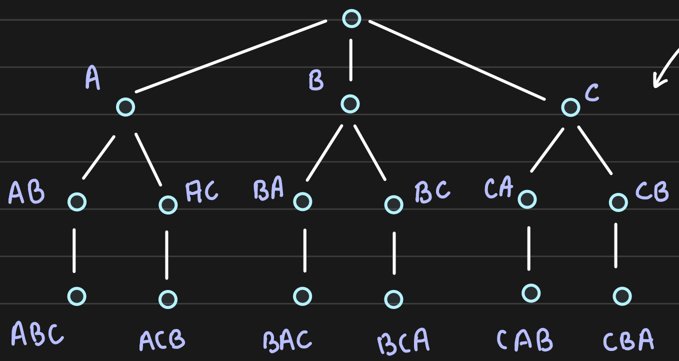
The tree is constructed level by level and is designed to hold all possible permutations of length k. (distinct paths)

At each level k, every node has n-k children to maintain uniqueness of the path

Root node is labelled as " "

if a node received a message labelled [3,2,7,1,4] it holds the message in the tree along that path

Hence the data structure is like a trie



3-level deep EIG constructed over 3 nodes A, B and C

ARPIT BHAYANI

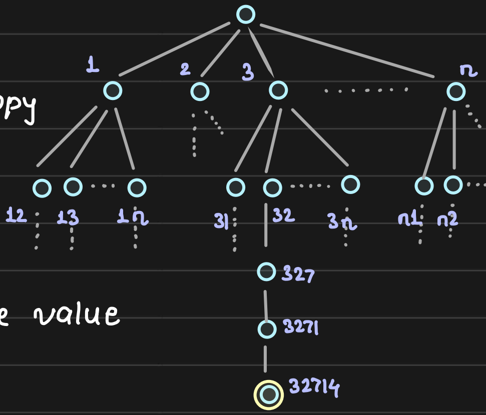
The algorithm

we assume at max f nodes would fail while achieving consensus. The algorithm runs for $f+1$ rounds giving chances for f processes to fail.

In each round, a new level in EIG Tree is built using prev level

Each process maintains its own EIG Tree

The processes upon receiving values from other nodes / processes, update their own copy of the tree.



After $f+1$ rounds, the nodes/procs refer to their local EIG Tree to decide the value

* Node independently takes the decision

Building the tree is a way to gather entire information

The decision rule is totally upto the usecase at hand.

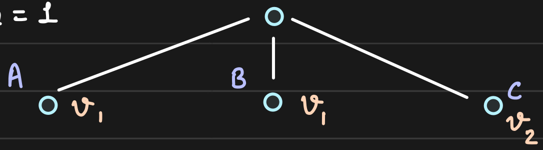
Round 1: Every process i,

1. sends its value to the entire network (including itself)
2. receives value (v) from other node j

   ↳ update the tree $[j] = v$

ARPIT BHAYANI

at the end of round 1, every node in the network will have the same EIG Tree with depth = 1

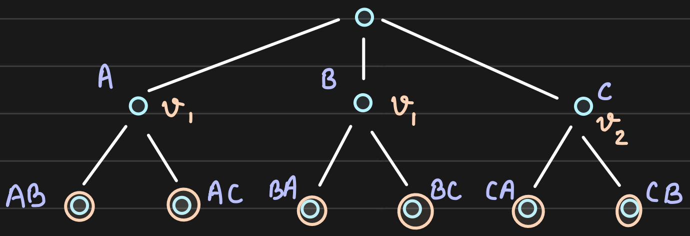


Round k, $2 \leq k \leq f+1$: Every process i,

1. sends all pairs (x, v) from k-1 level in the network where i is not in x

eg: process B would send tree[A] and tree[C] in round 2
process C would send tree[A] and tree[B] in round 2

Node A can thus form the next level of the EIG Tree with path AB, CB, AC, BC.



Node A will send the (x, v) to itself as well, thus receiving values of tree[B] and tree[C] from A

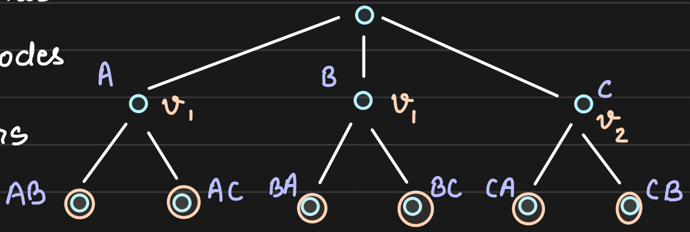
It uses these values to construct path BA and CA thus completing the entire level 1.

This process is repeated at each node and thus every node will have the exact same EIG Tree.

ARPIT BHAYANI

Thus we see how every node has

1. all values from all the nodes
2. all values across all paths



The algorithm stops after $f+1$ rounds

Each node simply goes through its own copy of EIG tree and gathers all values seen so far

if singleton set, choose v

if set has multiple values, choose default v_0

Alternative decision strategy

Depending on the usecase, we may choose any decision strategy

1. pick the smallest one
2. pick the newest one

} so long as we have total ordering of the values

$1000, 9:00:00 am ← Total ordering on timestamp

$2000, 9:00:01 am

$\boxed{\text{\$ 1500, 9:00:02 am}}$ ← nodes deciding on the latest one

ARPIT BHAYANI