Distributed Algorithms : Basics and Some Advances

Sushanta Karmakar

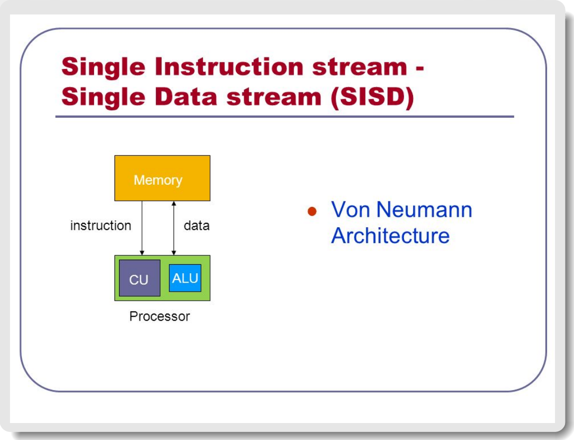
Indian Institute of Technology Guwahati
https://www.iitg.ac.in/sushantak/

October 6, 2018

Centralized Computing

algorithm running on a single machine

- input is centralized
- single processor
- output is centralized

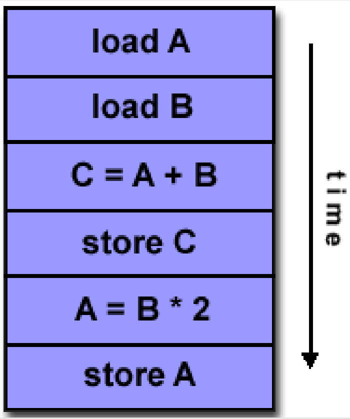


Some examples of centralized computing

algorithm running on a single machine

- sorting
- searching
- matrix multiplication
- Computation happens in a single m/c
- centralized graph problems, (**max flow, matching**)

An example

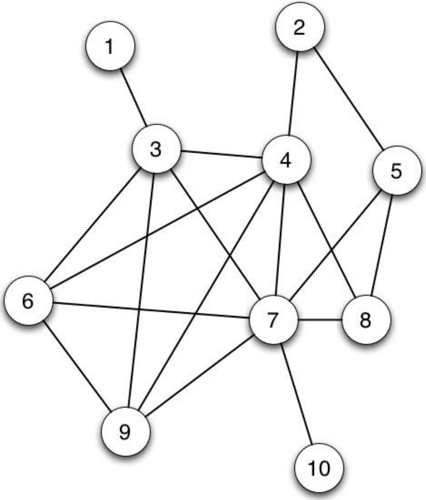


Distributed computing

algorithm running on many machines

- structure formation (DFS, BFS, MST, SPT, CDS, MIS etc)
- leader election
- mutual exclusion
- distributed graph algorithms (coloring etc)
- others (snapshot, deadlock, consensus etc)

An example

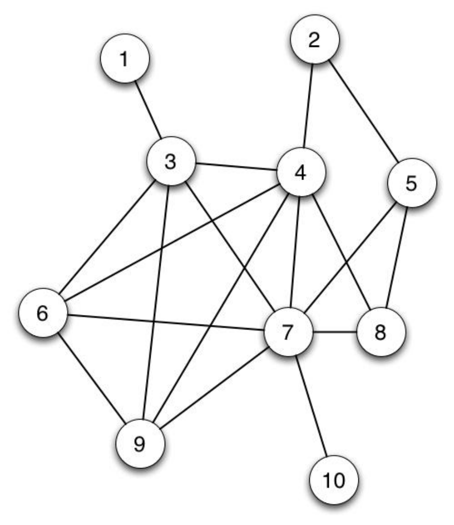


Why Distributed computing

algorithm running on many machines

- distributed environment
- speed up
- memory-efficiency
- resource sharing
- fault-tolerance

An example

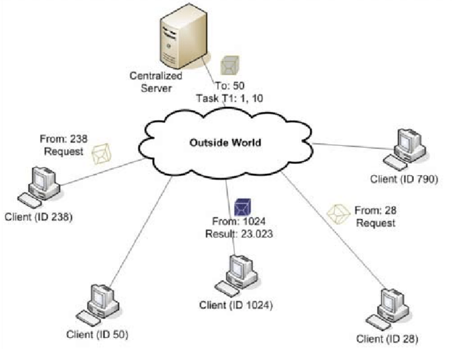


Application of Distributed computing

algorithm running on many machines

- World wide web
- Network file server
- Banking, railway/airline reservation
- P2P systems
- Block-chain, sensor network, cloud computing

An example

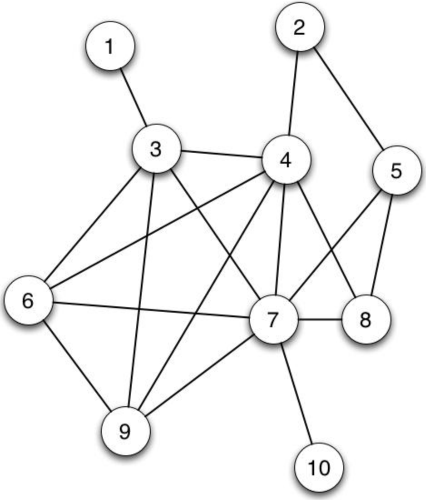


Model of Distributed computing

algorithm running on many machines

- message passing/shared memory
- synchronous/asynchronous
- knowledge of a node
- network topology
- failure model
- strong vs weak model
- change in model changes the problem

An example

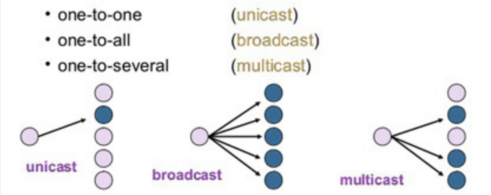


Broadcast : a standard problem

send a message to everybody

- message passing model
- synchronous/asynchronous
- knowledge of a node
- network topology unknown
- no failure

An example



A simple broadcast algorithm

send a message to everybody

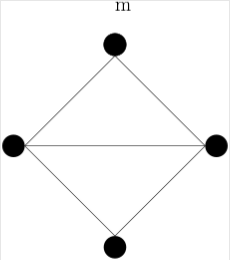
For source r: send M to each $v \in N(r)$

For any other node v : upon receiving M from u
send M to each $w \in N(v) \setminus \{u\}$

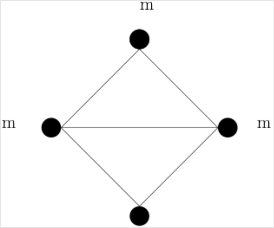
An execution

send a message to everybody

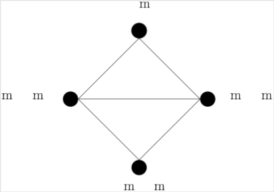
Round 0



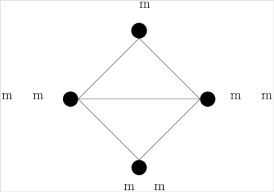
Round 1



Round 2



Final state



Solving the redundancy problem in broadcast

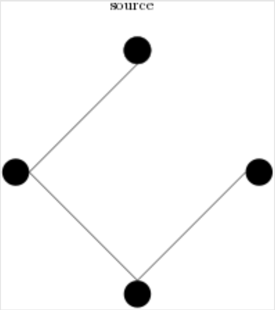
send a message to everybody

use a spanning tree rooted at the source r

A spanning tree T of a graph $G = (V, E)$ is a subgraph (V, E') such that

- for all pair $u, v \in V$, there is a path between u and v
- there is no cycle in T

Spanning tree



Some understanding

regarding broadcast problem

- need a structure for efficient broadcast
- time to complete the broadcast is $O(depth)$
- message complexity $=$ # send of messages $= n - 1$
- broadcast can also be done using other structures (bfs, dfs, cds etc)
- faster broadcast possible (but with redundancy)
  - push-pull approach of randomized broadcast
  - gossip / rumor spreading

Some questions

on broadcast in a graph

What is the best broadcast structure ?

Does the same algorithm work in asynchronous system ?

What happens if channels are not FIFO ?

Does the network topology affect the broadcast ?

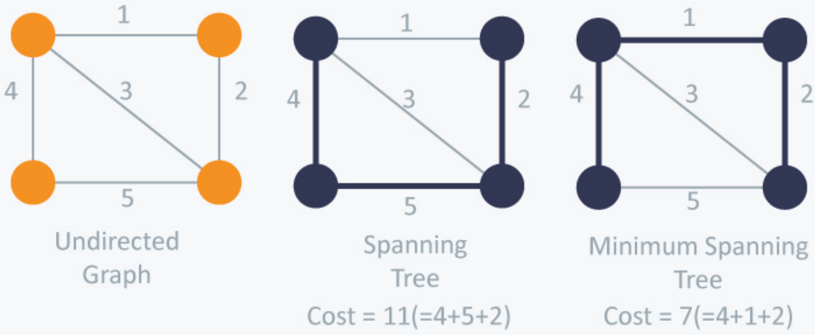
What happens when one/more node(s)/link(s) fail(s) ?

Minimum spanning tree

an important structure

Given a graph $G = (V, E)$, an MST of G is a subgraph $T = (V, E')$ such that T is a spanning tree and the value of $W = \sum_{e \in E'} w_e$ is minimum.

MST



MST construction

distance traveled by agent is minimized

centralized algorithm

Kruskal's algorithm, time complexity is $O(n \log n)$, concept of union-find data structure

distributed algorithm

GHS algorithm (round complexity is $O(n \log n)$), algorithm by B. Awerbuck ($O(n)$), and many more

best known deterministic distributed algorithm

Kutten and Peleg, $O(D + \sqrt{n} \log^* n)$

Other interesting problems

Steiner tree problem

Given a connected undirected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}^+$, and a set of vertices $Z \subseteq V$, known as the set of terminals, the goal of the ST problem is to find a tree $T' = (V', E')$ such that $\sum_{e \in E'} w(e)$ is minimized subject to the conditions that $Z \subseteq V' \subseteq V$ and $E' \subseteq E$.

Prize-collecting steiner tree

Given a connected weighted graph $G = (V, E, p, w)$ where V is the set of vertices, E is the set of edges, $p : V \to \mathbb{R}^+$ is a non-negative prize function and $w : E \to \mathbb{R}^+$ is a non-negative weight function, the goal is to find a tree $T = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$ that minimizes the following function:

$$GW(T) = \sum_{e \in E'} w_e + \sum_{v \notin V'} p_v$$

Some fundamental questions

structural aspect of distributed algorithm

What are the best known centralized algorithms for the ST and PCST problems ?

What are the best known distributed algorithms for the ST and PCST problems ?

Is there any possibility of further improvement ?

What about the problems in the dynamic graph setting ?

ST algorithm

structural aspect of distributed algorithm

centralized

- best approx factor is $1.386 + \epsilon$, running time is $O(n^5 \log n)$, Byrka et al. in STOC 2010
- A 2-approximate algorithm generally takes $O(n|Z|^2)$ time

distributed

- 2 approx with round complexity is $O((S + \sqrt{min\{S|Z|, n\}}) \log n)$, Lenzen and Patt-Shamir, PODC 2014
- $2(1 - \frac{1}{\ell})$ approx with round complexity $O(S + \sqrt{n} \log^* n)$, upcoming in ICDCN 2019

PCST algorithm

structural aspect of distributed algorithm

centralized

- $2(1 - \frac{1}{n-1})$ approx factor, running time is $O(n^2 \log n)$, Goemans and Williamson, SIAM J. of Applied Math. 1995.
- $2(1 - \epsilon)$ approx factor, specifically 1.9672, running time is $O(n^5 \log n)$, Archer et al. , SIAM J. of Computing, 2011

distributed

Nothing much known

Some more questions

structural aspect of distributed algorithm

Distributed algorithm for PCST, fault-tolerant algorithm for ST and PCST under crash, transient, byzantine faults

Is there any possibility of further improvement ?

What about the problems in the dynamic graph setting ?

Thank you

Questions ?