5 TERMINOLOGY AND BASIC ALGORITHMS

TOPOLOGY ABSTRACTION AND OVERLAYS

- Physical topology

  nodes represent all network nodes (including switching elements, non-app executing hosts nodes)

  edges represent all communication links

- Superimposed topology (topology overlay)

  higher level topology superimposed on logical topology usually a regular structure like tree, ring, mesh, or hypercube. It provides a specialized path for efficient information dissemination and/or gathering as part of a distributed algorithm.

- application execution

  execution of distributed application program.

- control algorithm (protocol)

  executed in order to monitor application execution, or perform various auxiliary functions: creating a spanning tree, creating a connected dominating set, achieving consensus among nodes, dist. transaction commit, dist. deadlock detection, global predicate detection, termination detection, ...

- logical topology

  nodes represent all end hosts where application executes.

  edges are logical channels among these nodes

- Centralized algorithm

  most work is performed by one (or few) processors, whereas others play a relatively small role in performing the joint task.

- distributed algorithm

  each processor plays an equal role in sharing the msg, time and space overhead.

- symmetric algorithm

  all processors execute the same logical functions.

- asymmetric algorithm

  different processors execute logically different functions

{structural elegance}

- anonymous algorithm

does not use process identifiers in code.

{allows scalability transparency}

- Uniform algorithm

does not use n (total no of processes) as parameter in code.

- adaptive algorithm

complexity of the algorithm can be expressed in terms of k (no of participating nodes).

- deterministic recieve primitive

specifies the source from which it wants to recieve a message.

- non-deterministic recieve primitive

can recieve a msg from any source.

- deterministic execution

program that contains no non-deterministic recieves.

- non-deterministic execution

program that contains atleast one non-deterministic primitive.

- execution inhibition

protocols that require processors to suspend their normal execution until some action has been performed are called inhibitory or freezing protocols.

- Locally, globally inhibitory
- Send, recieve, internal event inhibition

- Synchronous system

system that has known upper bound in:

- message delay
- clock drift rate
- logical step execution time.

- asynchronous system

system which satisfies none of above 3 conditions.

- on-line algorithm

executes as data is being generated.

- off-line algorithm

requires all data to be available before execution begins.

- wait-free algorithm

can execute in an (n-1) process fault tolerant manner.

(resilient to n-1 process failures)

PROCESS FAILURE MODELS

- fail-stop

Execution stops at some instant. other processes learn of this failure

- crash

Execution stops at some instant. other processes do not learn of this failure

- recieve omission

intermittently fail to recieve some messages.

- send omission

intermittently fail to send some messages.

- general omission

intermittently fail to recieve / send some msgs.

- byzantine, or malicious failure, with authentication

processes may exhibit arbitrary behaviour, but signatures can be used to verify msg integrity.

- byzantine or malicious failure

processes may exhibit arbitrary behaviour.

(no authentication techniques applicable)

COMMUNICATION FAILURE MODELS

- crash

link stops carrying msgs at some instant.

- omission

intermittently fails to carry some messages

- byzantine failure

link can exhibit arbitrary behaviour, including creating spurious messages, and modifying the msgs sent on it.

- timing failures

these can occur in synchronous systems either for processes, or for links.

- complexity measures & metrics
  - space, time & node vs. system wide
  - message complexity

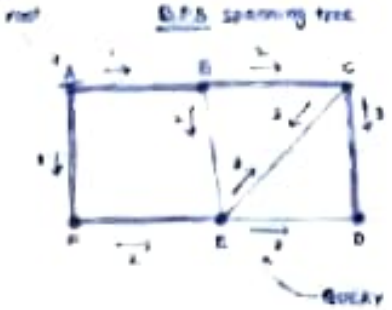
    number, size, time.

SYNCHRONOUS SINGLE INITIATOR SPANNING TREE ALGORITHM USING FLOODING

each node already knows its neighbours. spanning tree is known, when each node knows its parent. this algorithm assumes a designated root node.

root: • starts by sending QUERY to its neighbours.

node: • on receiving QUERY(s) one random node is chosen as parent. QUERY is sent to all remaining neighbours.



6 nodes 8 links

3 rounds 9 QUERYs

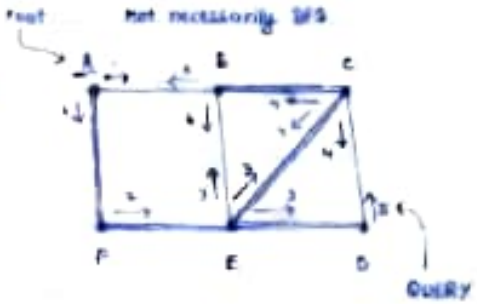
Colge uses "visited", and "depth" variables)
(diameter - longest distance in graph)

ASYNCHRONOUS SINGLE-INITIATOR SPANNING TREE ALGORITHM USING FLOODING

this algorithm assumes a designated root node. algorithm terminates when &response to all QUERYs is received.

root: • starts by sending QUERY to its neighbours.

node: • on receiving first QUERY, it is chosen as parent and ACCEPT reply is sent to it. to remaining QUERY msgs, REJECT reply is sent. QUERY is sent to all remaining neighbours.



ASYNCHRONOUS CONCURRENT-INITIATOR SPANNING TREE ALGORITHM USING FLOODING

~~suppress~~ wake: any node may spontaneously initiate spanning tree algorithm provided it has not already been invoked locally due to receipt of a QUERY message

Suppress the instance initiated by one root and continue the instance initiated by the other root, based on tie-breaking using the processor identifier. this algorithm uses QUERY (newroot), ACCEPT (newroot), and REJECT (newroot) messages.

Root: starts by sending QUERY(i) to its neighbours

node: when QUERY(newroot) received from j:

↳ newroot > myroot : suppress current execution. reset data structures.
join j's subtree with myroot = newroot.

↳ newroot = myroot : already have a parent from same root
hence send REJECT to j

↳ newroot < myroot : j's root has lower priority, so send REJECT to j.
j will eventually recieve QUERY (myroot) from i.

when ACCEPT (newroot) received from j:

↳ newroot = myroot : ACCEPT is in response to QUERY sent by i.
process normally.

↳ newroot < myroot : ACCEPT in response to old QUERY from i (with old root).
ignore message.

↳ newroot > myroot : cannot happen.

when REJECT (newroot) received from j: similar to ACCEPT above.

when children ∪ unrelated = neighbours - parent; send ACCEPT (myroot) to parent
(ACCEPTed, REJECTed)

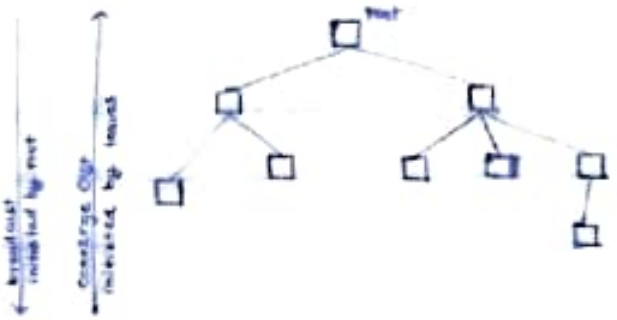
drawback: only root knows when the algorithm has terminated. it must broadcast others to inform this. (along spanning tree)

ASYNCHRONOUS CONCURRENT INITIATOR DEPTH FIRST SEARCH SPANNING TREE ALGORITHM

similar to the algorithm before, though here QUERY is sent to only one neighbour at a time after the reciept of a QUERY, ACCEPT, or REJECT.

BROADCAST AND CONVERGECAST ON A TREE

Spanning tree is useful for distributing info to all nodes via - broadcast and collecting info from all nodes via - convergecast <



SINGLE SOURCE SHORTEST PATH ALGORITHM: SYNCHRONOUS BELLMAN-FORD

each node sends its length$_i$ to all its neighbours.

on recieving length$_j$ the node updates its length if

length$_j$ + weight$_j$ < length.

- length: initially ∞
- weight$_j$: weight between nodes and neighbour

msg complexity: $O(n-1)E$

- E: links

after k rounds, length of nodes k hops away from source is stabilized.

longest path length = n-1

rounds to stabilize = n-1

initially, length = ∞

for source, length = 0

algorithm assumes no cyclic paths having negative weight.

SINGLE SOURCE SHORTEST PATH ALGORITHM: ASYNCHRONOUS BELLMAN-FORD

Very similar to previous algorithm above, a node terminates after sending UPDATE to all neighbours. Unfortunately msg complexity - $\mathcal{O}(e^n)$, time comp - $\Omega(e^n \cdot d)$

MINIMUM-WEIGHT SPANNING TREE (MST) ALGORITHM IN A SYNCHRONOUS SYSTEM

Kruskal's algorithm: begings with a forest of graph components. in each iteration it identifies the min weight edge that connects 2 different components, and uses this edge to merge 2 components. this continues until all components are merged into single component.

Prim's algorithm & dijkstra's algorithm: a single node component is selected. in each iteration, a min weight edge incident on the component is identified, and the component expands to include that edge and the node at the other end of that edge. after n-1 iterations, all the nodes are included.

distributed algorithm by gallagher, humblet, and spira (GHS) uses kruskal's strategy.
MWOE - min weight outgoing edge
this algorithm knows the value of n. it assumes that weight of each edge in the network is unique, which is necessary to guarantee unique MST. (if weights are not unique, the ids of the nodes on which they are incident can be used as tie-breakers by defining a well-formed order)

- initially each component has one node which is the leader.
- leader broadcasts SEARCH_MWOE (leader) on tree edges to request edges to find MWOE.
- edges send EXAMINE (leader) to non-tree edges after recieving SEARCH_MWOE.
- details of potential MWOEs are convergecast to leader with REPLY_MWOE (localid, remoteid).
- leader broadcasts ADD_MWOE (localid, remoteid) to add MWOE and identify new leader.
- new leader is max (localid, remoteid) which touches the MWOE.
- new leader broadcasts NEW_LEADER (leader) after merging components.

- a cycle of length 2 is possible, but since they have a common MWOE, there would be a single spanning tree, and a unique leader in merged comp.
- a cycle of length 3 or more is not possible as that would mean 2 different minimum weight edges to one tree, which is not possible.

→ Iteration time complexity = $O(\log n)$ total time complexity = $O(n \log n)$

n msgs to tree edges, E msgs to find MWOE. → msg complexity = $O((n+e)\cdot \log n)$

MAXIMAL INDEPENDENT SET (MIS)

proposed by Luby

each iteration:

- each process selects a random number, random$_i$.
- if its random$_i$ is least among neighbours it gets to be in MIS.
- each P_i informs its selection status through SELECTED (yes/) msg.
- if a process finds atleast one of its neighbours is SELECTED(yes) it eliminates itself from MIS.
- each P_i informs its elimination status through ELIMINATED(yes/) msg.
- each P_i removes from its neighbours, who send ELIMINATED (yes) msg.
- proceed to next iteration.

no of iterations = $O(\log n)$