COMP90020: Distributed Algorithms

15. Information Gathering Algorithms

Learning Important Facts about Your Distributed System

Miquel Ramirez

THE UNIVERSITY OF MELBOURNE

Semester 1, 2019

Agenda

A Pattern Emerges

Information Gathering Problems in Distributed Systems

A process p_i needs to gather information from all other processes p'.

Protocols follow this basic structure:

1. p_i sends message to processes p',
2. processes p' then reply with requested information,
3. a special, internal decide event triggers for p_i.

Question!

In which of the following problems have we seen this "subproblem"?

(A): Server Synchronization (B): Leader Election

(C): Atomic Commit Protocols (D): Failure Detection

→ (A): The Network Time Protocol (NTP) is arranged to involve pairs of processes, information gathering is about 1-to-N data exchanges.

Wave Algorithms for Information Gathering

Wave algorithms formalize the notion of *information gathering* processes triggering special decide events.

Properties of Wave Algorithms

Executions of wave algorithms, called waves, satisfy the following properties:

1. *Termination*: it is finite,
2. *Decision*: contains one or more *decide* events
3. *Dependence*: for each *decide* event a and process p, $b' \prec a$ for some event b' at every other process p'.

What does Dependence Mean?

Idea Behind Wave Algorithm

Executions of wave algorithms give rise to decisions where all processes *have a say*.

As a consequence, decide events e can only be triggered at the initiator process p_i if a message has been received from every other process.

Question!

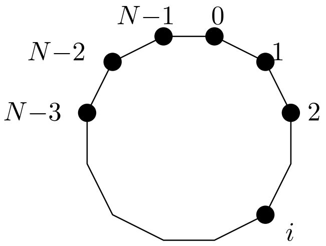
Will a wave algorithm complete if a process p refuses to take part in the execution?

(A): Yes (B): No

$\rightarrow$ (No): The property of dependence requires decide events to be causally related to previous events. There exists at least one process p' not triggering an event b' justifying the decision event a at p.

Wave algorithms - Example

In a *ring–based algorithm*, p_0, the initiator, sends a token θ to p_1, which is passed on by all other processes to their neighbours.



The initiator decides after the token has returned.

Question!

For the decide event e at p_0, which events b_i are causally before e?

(A): b_{N-1} (B): $b_{N-1}, ..., b_i, ..., b_1$

$\rightarrow$ (B): $b \prec a$ are transitive. $b_{N-1} \prec e$ implies $b_{N-2} \prec e$, since $b_{N-2} \prec b_{N-1}$.

Traversal Algorithms

A traversal algorithm is a *centralized* wave algorithm; i.e., there is one initiator, which sends around a token.

- In each execution, the token first visits all processes.
- Eventually, the token returns to the initiator, triggering a *decide* event.

Traversal algorithms build a spanning tree:

- the initiator is the root; and
- each noninitiator has as parent the neighbor from which it received the token first.

Tarry's Algorithm (from 1895)

Processes p and p' in DS connected via channels $p \rightarrow p'$, $p' \rightarrow p$.

- R1 A process p never forwards the token θ through the same channel twice.
- R2 A process only forwards θ to its parent when there is no other option.

Gaston Tarry (1843-1913)

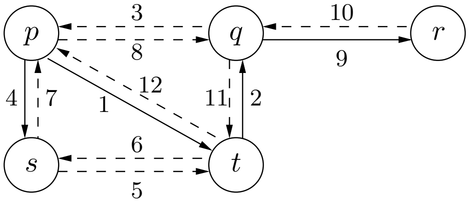
θ travels through each channel both ways, and finally ends up at the initiator.

Efficiency

- Message complexity: $2E$ messages
- Time complexity: $\leq 2E$ time units

Tarry's algorithm - Example

p is the initiator.



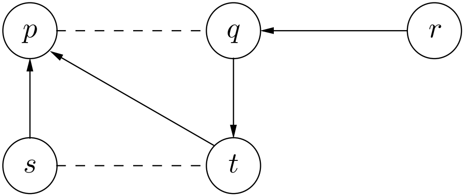
The network is undirected and unweighted.

Arrows and numbers mark one possible path of the token.

Solid arrows establish a parent-child relation e.g. p *is parent of* t or t *is child of* p.

Tarry's Algorithm & Spanning Trees

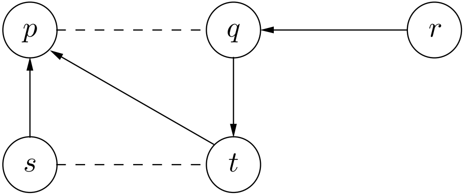
We will use the "child-of" convention by reversing the direction of messages that establish a parent-child relation between processes.



Tree edges, which are part of the spanning tree, are solid.

Frond edges, which aren't part of the spanning tree, are dashed.

Question



Question!

Could this spanning tree have been produced by a depth-first search (DFS) starting at p?

(A): Yes (B): No

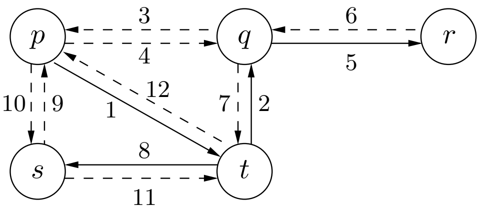
→ (No): In a spanning tree resulting from DFS s and t would never have p as a parent.

Depth-first Search (DFS) for Distributed Systems

Depth-first search (DFS) can be obtained adding to Tarry's algorithm:

R3 When a process p receives the token θ, it immediately sends it back through the same channel if this is allowed by R1 & R2.

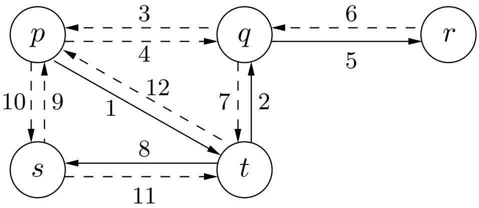
Example:



Property

In the spanning tree of a depth-first search, all frond edges connect an ancestor with one of its descendants in the spanning tree.

Latency in Information Gathering Algorithms



Question!

What is the consequence of sending the token θ back and forth between edges pq and ps for the decide event at p?

(A): A delay of 4 time steps

(B): Nothing to worry about, it will eventually happen

Question!

Can the extra delay be avoided without modifying connectivity?

(A): No

(B): Yes, but messages require more bits.

$\rightarrow$ (Yes): We can add the IDs of processes that have already seen θ

DFS with Neighbour Knowledge

Additions to DFS

To prevent transmission of the token through a frond edge, visited processes id's V are included in the token θ.

R4: θ is not forwarded by p to a process q if $q \in V$, except when p is a child of q.

Efficiency

- Message Complexity $2N - 2$ messages
  - Each tree edge carries 2 tokens.
- Time Complexity: $\leq 2N - 2$ time units
- Bit Complexity: Up to kN bits per message
  - k bits are needed to represent one process identifier.

Cidon's Algorithm

Motivation

Implement V via messaging in an economic manner.

Additions to DFS

1. A process p holding the token θ for the first time, forwards it and notifies neighbours with an *info* message.
2. p records in a local variable fw_p to which process is forwarded the token last.
3. If p receives θ from a process $q \neq fw_p$
   - p marks edge pq as frond, and dismisses θ.
4. If q receives notification from process $q \neq fw_p$
   - q marks pq as frond edge and continues forwarding the token.

Cidon's Algorithm - Efficiency

Message complexity: $\leq 4E$ messages

$\rightarrow$ Each channel carries at most 2 info messages and 2 tokens.

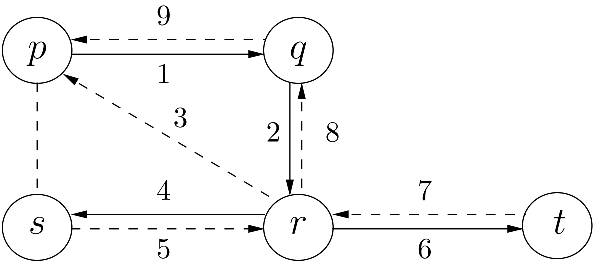
Time complexity: $\leq 2N - 2$ time units

$\rightarrow$ Each tree edge carries 2 tokens.

Property

At least once per time unit, a token is forwarded through a tree edge.

Cidon's Algorithm - Example



Notes:

- θ is forwarded by r through the frond edge pr before p *info* reaches r
- r continues to forward the token to s,
- p's *info* reaches s before θ does, so s does not send θ to p.

Decentralized Waves – Tree algorithm

The tree algorithm is a decentralized wave algorithm for undirected, acyclic networks.

The local algorithm at a process p:

- R1 p waits until it received messages from all neighbors except one, which becomes its *parent*.

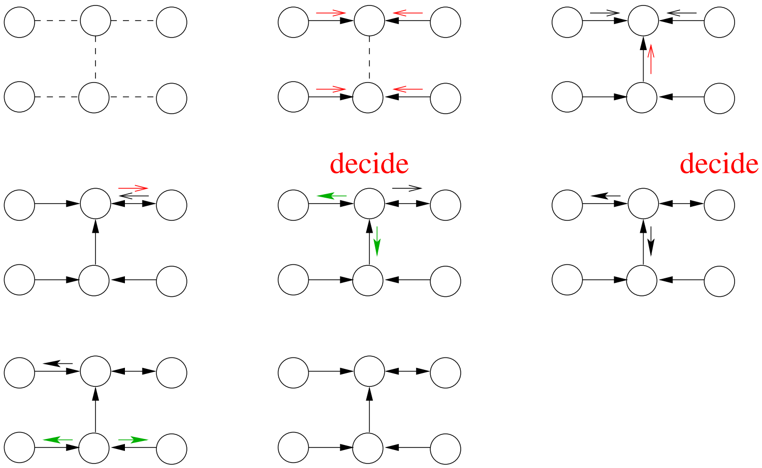
  Then it sends a message to its parent.
- R2 If p receives a message from its parent, it decides.

  It sends the decision to all neighbors except its parent.
- R3 If p receives a decision from its parent, it passes it on to all other neighbors.

Property

Always *two* (neighboring) processes decide.

Tree Algorithm - Example



Question

Question!

Does the Tree Algorithm terminate if applied to a network containing a cycle?

(A): Yes

(B): No

→ (No): The Tree algorithm is not correct for networks with cycles. Consider the a ring with three processes. Since each process has two neighbours, it will wait for a message from one of its neighbours. Hence, all three processes will be waiting for an input, and no event ever happens.

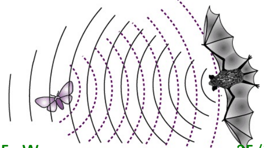
Echo Algorithm

The echo algorithm is a centralized wave algorithm for undirected networks.

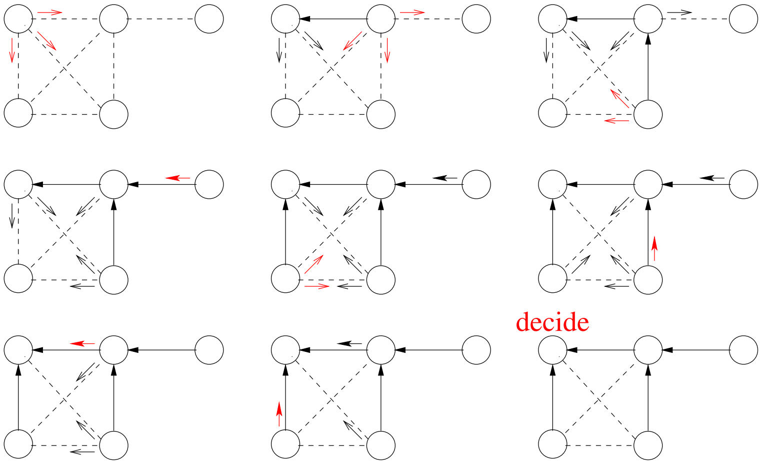
- R1 The initiator sends a message to all neighbors.
- R2 When a noninitiator receives a message for the first time, it makes the sender its *parent*.

  Then it sends a message to all neighbors except its parent.
- R3 When a noninitiator has received a message from all neighbors, it sends a message to its parent.
- R4 When the initiator has received a message from all neighbors, it decides.

Message complexity: $2E$ messages



Echo Algorithm - Example



Summary

Name	Centralized?	Time	Message	Notes
Tarry's Algorithm	Yes	$2E$	$2E$	
Depth First Search	Yes	$2N-2$	$2N-2$	Bit complexity kN
Cidon's Algorithm	Yes	$2N-2$	$4E$	
Tree Algorithm	No	$D/2$	$2E$	Only acyclic
Echo Algorithm	Yes	$2N-2$	$2E$	

Notes:

- N is number of processes, E number of channels
- Time and Message complexity is always worst case complexity
- Best algorithm depends on network topology

Spanning Trees in the Sky

Alternative forms of the Tree and Echo algorithm key to Distributed Control and Sensor Networks design and stabilization.

→ Olfati-Saber and Murray *Consensus Problems in Networks of Agents With Switching Topology and Time-Delays*, 2004

Further Reading

Wan Fokkink's *Distributed Algorithms: An Intuitive Approach*

- Chapter 4, Wave Algorithms

G. Tarry (1895) *Le problème des labyrinthes*, Nouvelles Annales de Mathématiques, 14, pp. 187-190

T.-Y. Cheung (1983) *Graph traversal techniques and the maximum flow problem in distributed computation*, IEEE Transactions on Software Engineering, 9, pp. 504–512

I. Cidon (1988) *Yet another distributed depth-first search algorithm*, Information Processing Letters, 26, pp. 301–305

E. J. H. Chang (1982) *Echo algorithms: Depth parallel operations on general graphs*, IEEE Transactions on Software Engineering, 8, pp. 391–401

A. Segall (1983) *Distributed network protocols*, IEEE Transactions on Information Theory, 29, pp. 23–34