Distributed Algorithms: Wave Traversal
Awerbuch's and Cidon's Depth-first Search Algorithms

Burak Şen buraksenb@gmail.com

CENG532: Distributed Computing Systems
Department of Computer Engineering
Middle East Technical University
Ankara Turkey

May 15, 2024

Outline of the Presentation

Agenda

Wave Traversal in Distributed Systems

Let's say that you have a network of interconnected processes forming a distributed system. Each process holds or manages a portion of critical data or resources. Then you need the processes to traverse the entire distributed system in a coordinated manner to collect information, perform computations, or execute tasks. Without a well-defined traversal strategy, certain processes may remain unvisited, leading to incomplete data collection or task execution.

Wave Traversal Problem

How can these processes traverse the distributed system efficiently and return to the initiator process? The traversal must visit each process exactly once, ensuring comprehensive coverage of the system. However, in a distributed environment, challenges such as message passing, synchronization, and fault tolerance need to be addressed to achieve an optimal traversal strategy.

Agenda

Awerbuch's DFS: A new distributed depth-first-search algorithm

Awerbuch's algorithm is a depth-first search (DFS) wave traversal algorithm that addresses the Wave Traversal problem in distributed systems. It offers the following key features:

- Depth-First Search: Traverses the distributed system in a depth-first manner, ensuring comprehensive coverage.
- Scalable Messages: Message complexity scales proportionally with the network size, ensuring efficient traversal in large-scale systems.

Awerbuch's DFS: A new distributed depth-first-search algorithm

Guaranteed Termination: Ensures the traversal concludes with all processes visited exactly once.

Guaranteed Correctness: Ensures the traversal is correct and complete, visiting all processes in the distributed system.

Agenda

Cidon's DFS Algorithm: Yet another distributed depth-first-search algorithm

Cidon's algorithm is another depth-first search (DFS) wave traversal algorithm that introduces several improvements over Awerbuch's algorithm. It offers the following key features:

- Depth-First Search: Traverses the distributed system in a depth-first manner, ensuring comprehensive coverage.
- Improved Message Complexity: Reduces the message complexity compared to Awerbuch's algorithm, optimizing communication overhead.
- Improved Time Complexity: Reduces the time complexity compared to Awerbuch's algorithm, optimizing traversal time by not waiting for acknowledgment messages.

Agenda

Wave Traversal in Distributed Systems

Why is it important?

Without wave traversal, distributed systems may not be able to efficiently collect data, perform computations, or execute tasks in a coordinated manner. Thus, wave traversal algorithms are essential for distributed systems to achieve optimal performance and resource utilization. Furthermore, efficient wave traversal algorithms are crucial to ensure scalability and robustness in large-scale distributed systems, where the number of processes and the complexity of interactions can be significant.

Agenda

Model, Definitions

Wave Traversal

A traversal algorithm is any algorithm that meets the following criteria:

- In each computation there is only one initiator which starts the traversal with one message.
- A process, upon receiving a message, can send messages to its neighbors or decides.
- Algorithm terminates in the initiator process when each process has sent at least one message.

Background

Tarry's Algorithm (1895) is a random graph traversal algorithm that explores the entire graph by visiting each node exactly once.
It is the oldest known algorithm for graph traversal.

Background

- Baruch Awerbuch (1985) DFS algorithm is a depth-first search algorithm that ensures comprehensive coverage of the distributed system.
- Israel Cidon (1988) builds on Awerbuch's algorithm, introducing optimizations to reduce message complexity and traversal time.

Agenda

Wave Traversal

Awerbuch's DFS Algorithm

The implementation of Awerbuch's DFS algorithm is based on the following key components:

- Initialization: The initiator process sends a start message to its neighboring processes to begin traversal.
- Traversal Logic: Each process selects an unvisited neighbor to visit next and sends it a visit message.
- Acknowledgment: Upon receiving a return message from a child process, a process sends an acknowledgment message back to the parent process.
- Termination: The initiator process waits for acknowledgment messages from all neighboring processes to conclude the traversal.

Wave Traversal

Awerbuch's DFS Algorithm

It recursively visits each unvisited neighbor starting from the initiator process, marking processes as visited to prevent redundant visits. This traversal continues until all processes have been visited exactly once.

var $used_p[q]$: **bool** **init** false for each $q \in Neigh_p$;
(*Indicates wether p has forwarded the token to q*)
$father_p$: process **init** udef ;

For the initiator only, execute once:

begin $father_p := p$; choose $q \in Neigh_p$;
forall $r \in Neigh_p$ **do** send ⟨**vis**⟩ to r ;
forall $r \in Neigh_p$ **do** receive ⟨**ack**⟩ from r ;
$used_p[q] := true$; send ⟨**tok**⟩ to q
end

For each process after receiving $\langle \mathbf{tok} \rangle$ from q_0:

begin if $father_p = udef$ **then**
  begin $father_p := q_0$;
    forall $r \in Neigh_p \setminus \{father_p\}$ **do** send $\langle \mathbf{vis} \rangle$ to r ;
    forall $r \in Neigh_p \setminus \{father_p\}$ **do** receive $\langle \mathbf{ack} \rangle$ from r
  end;
 if p is the initiator **and** $\forall q \in Neigh_p : used_p[q]$
  then decide
 else if $\exists q \in Neigh_p : ($[illegible]$ \wedge \neg used_p[q])$
  then begin if $father$[illegible]$\neg used_p[q_0]$
     then [illegible]
     else choose $q \in Neigh_p \setminus \{father_p\}$ with $\neg used_p[q]$
    $used_p[q] := true$; send $\langle \mathbf{tok} \rangle$ to q
    end
  else begin $used_p[father_p] := true$; send $\langle \mathbf{tok} \rangle$ to $father_p$ **end**
end

For each process after receiving $\langle \mathbf{vis} \rangle$ from q_0:

begin $used_p[q_0] := true$; send $\langle \mathbf{ack} \rangle$ to q_0 **end**

Wave Traversal

Cidon's DFS Algorithm

The implementation of Cidon's DFS algorithm is based on the following key components:

- Initialization: The initiator process sends a start message to its neighboring processes to begin traversal.
- Traversal Logic: Each process selects an unvisited neighbor to visit next and sends it a visit message.
- Acknowledgment: Cidon's algorithm eliminates the acknowledgment messages, reducing the communication overhead.

Wave Traversal

Cidon's DFS Algorithm

Cidon's DFS algorithm shares similarities with Awerbuch's approach but streamlines the traversal process by eliminating the wait for acknowledgment messages. Upon receiving the start message, each process selects an unvisited neighbor and proceeds with traversal without waiting for acknowledgments. This design simplifies the algorithm and reduces traversal overhead, improving efficiency in distributed systems. While termination conditions may still involve waiting for traversal completion, the absence of acknowledgment messages enhances scalability and performance, making Cidon's algorithm an attractive choice for DFS traversal in distributed environments.

var $used_p[q]$: **bool** **init** false for each $q \in Neigh_p$;
(* Indicates, wether p has sent the token to q *)
$father_p$: process **init** udef ;
$last_p$: process **init** udef ;

For the initiator only, execute once:
begin $father_p := p$; choose $q \in Neigh_p$;
forall $r \in Neigh_p$ **do** send **vis** to r ;
$used_p[q] := true$; $last_p := q$; send **tok** to q
end

For each process after receiving **vis** from q_0:
begin $used_p[q_0] := true$;
if $q_0 = last_p$ **then** (* Interpret as **tok** *)
forward **tok** message as upon receipt of **tok** message
end

For each process after receiving **tok** from q_0:

begin if $last_p \neq udef$ **and** $last_p \neq q_0$ **then** $used_p[q_0] := true$
    (* This is a frond edge, interpret as **vis** *)
    else(* Act as in Awebach's algorithm *)
        begin if $father_p = udef$ **then**
            begin $father_p := q_0$;
                forall $r \in Neigh_p \setminus \{father_p\}$ **do** send **vis** to r
            end ;
        if p — initiator **and** $\forall q \in Neigh_p : used_p[q]$
            then decide
        else if $\exists q \in Neigh_p : (q \neq father_p \wedge \neg used_p[q])$
            then begin if $father_p \neq q_0 \wedge \neg used_p[q_0]$
                then $q := q_0$
                else choose $q \in Neigh_p \setminus \{father_p\}$
                    with $\neg used_p[q]$;
                $used_p[q] := true$; $last_p := q$; send **tok** to q
            end
        else begin $used_p[father_p] := true$; send **tok** to $father_p$
            end

Agenda

Conclusions

Wave Traversal Algorithms in Distributed Systems

- Through testing is not done yet but the expected results are linear increase in message complexity and time complexity with the number of nodes.
- Moreover, Cidon's Algorithm is expected to have better performance in terms of message complexity and time complexity compared to Awerbuch's Algorithm. Because it reduces the number of messages and does not wait for acknowledgment messages.
- Future work includes testing the algorithms in various network topologies and configurations to evaluate their performance and scalability.

References

[1] B. Awerbuch, "A new distributed depth-first-search algorithm," *Information Processing Letters Volume 20*, 1985.

[2] I. Cidon, "Yet another distributed depth-first-search algorithm," *Information Processing Letters Volume 26*, 1988.

Questions

THANK YOU

Distributed Algorithms: Wave Traversal
Awerbuch's and Cidon's Depth-first Search Algorithms

presented by Burak Şen buraksenb@gmail.com

May 15, 2024