Assignment sheet 3

Problem 3.1 *Wave Algorithms*

Recall the definition of wave algorithms (slide 11, lecture10). Argue that there are issues in case of message duplication.
Propose a modification of the Echo Algorithm which can operate properly even if message duplication occurs.

As we know, in wave algorithms events are causal, and each process keeps track of its own state. When message duplication happens, it may cause a process to incorrectly update its state causing it to continue/be in an inconsistent state. Therefore, in the presence of message duplication, the *dependence* requirement of a wave algorithm may be disrespected.

Explanation:

The dependence requirement states that in each computation, each decide event is causally preceded by an event in each process. As processes may execute algorithms, and keep track of send and receive events, message duplication will cause events to be processed out of order (eg. their vector clocks, or array of receive events may be inconsistent). Furthermore this leads to violation of the causal relationship of the events to be disrespected which may cause the dependence requirement to be disrespected.

Example:

If a node receives a duplicate message after it has already processed the message it can cause the node to update its local state multiple times, leading to an inconsistent view of the system state.

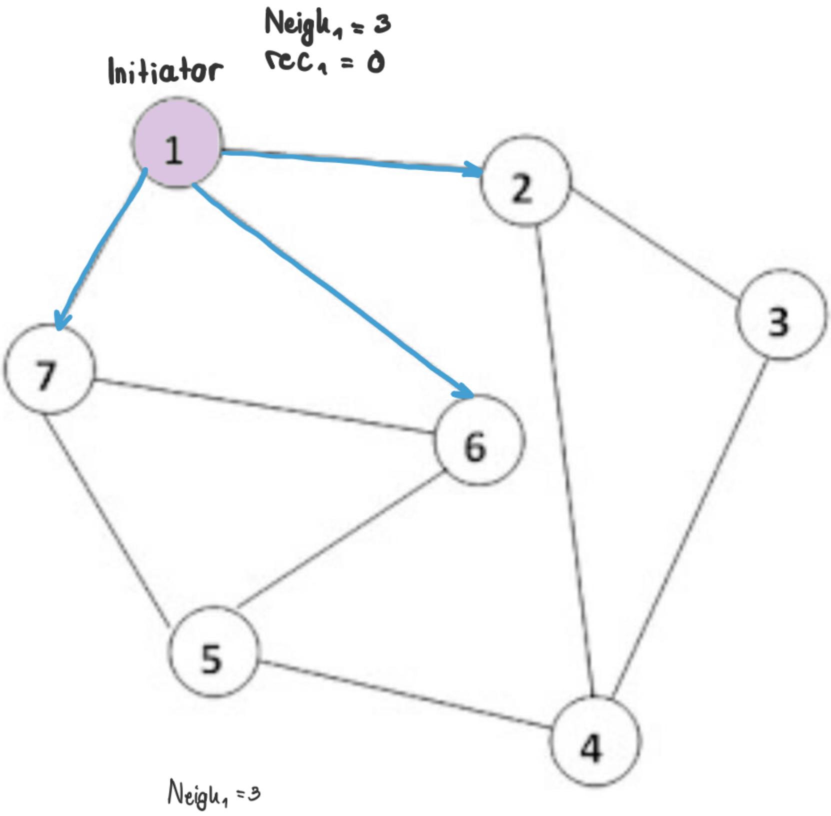
Idea to modify the echo algorithm:

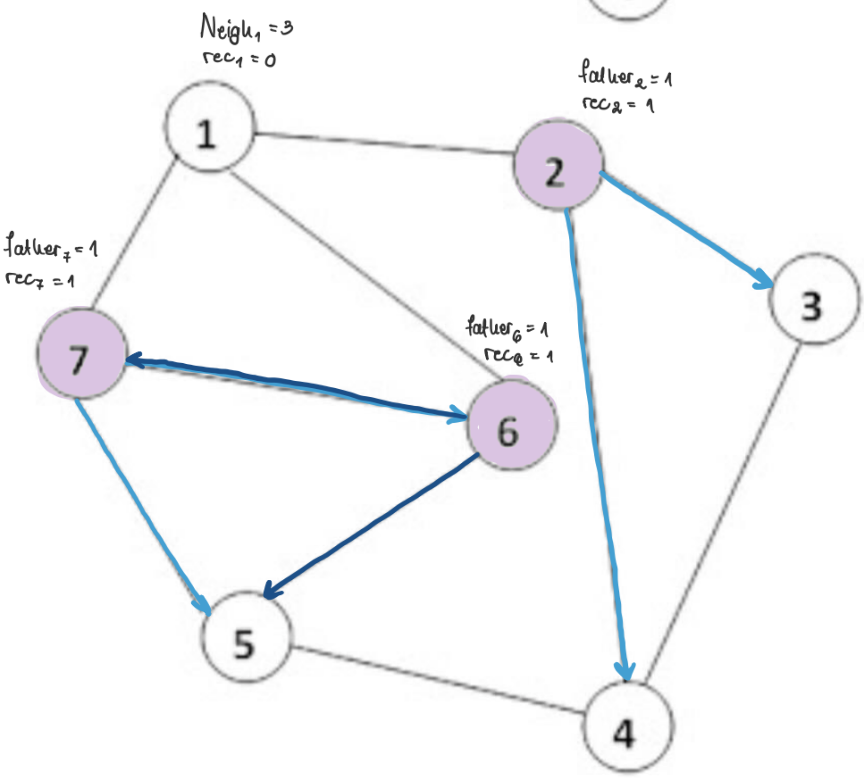
To modify the Echo Algorithm and avoid inconsistencies caused by message duplication, a unique identifier can be added to each propagated message. This identifier is generated by the source node and attached to the message, allowing nodes to keep track of the messages they have already processed. Example:

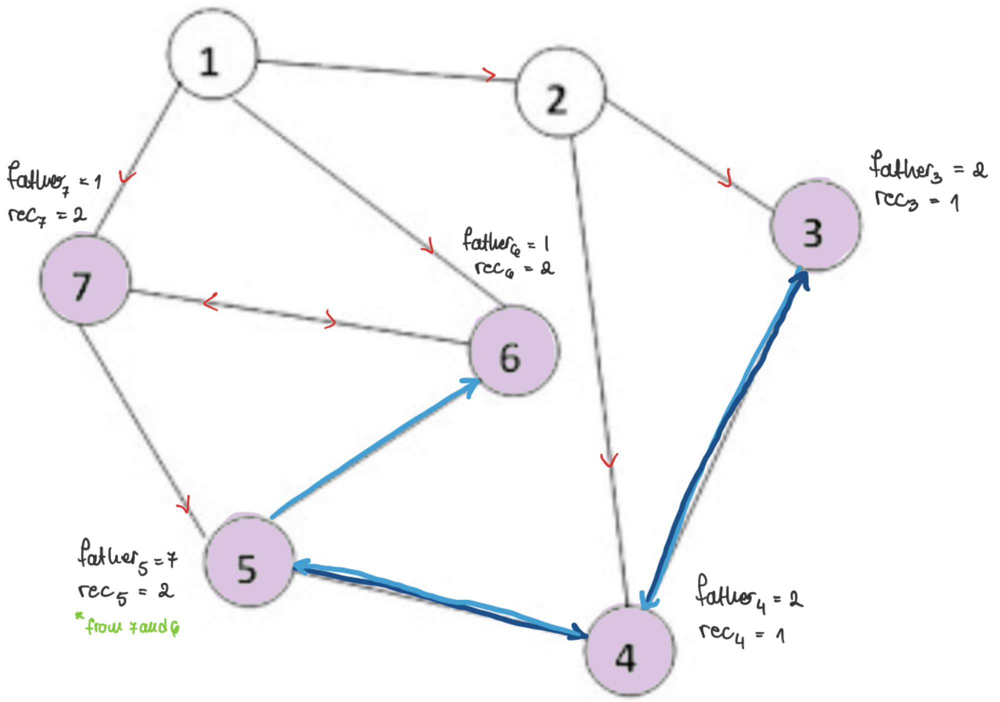
1. Each message M sent by the source process is assigned a unique identifier ID(M).
2. When a process receives a message M with identifier ID(M), it checks if it has already processed a message with that identifier.
3. If the process has not already processed a message with the same identifier, it does the steps provided by the echo algorithm initially.

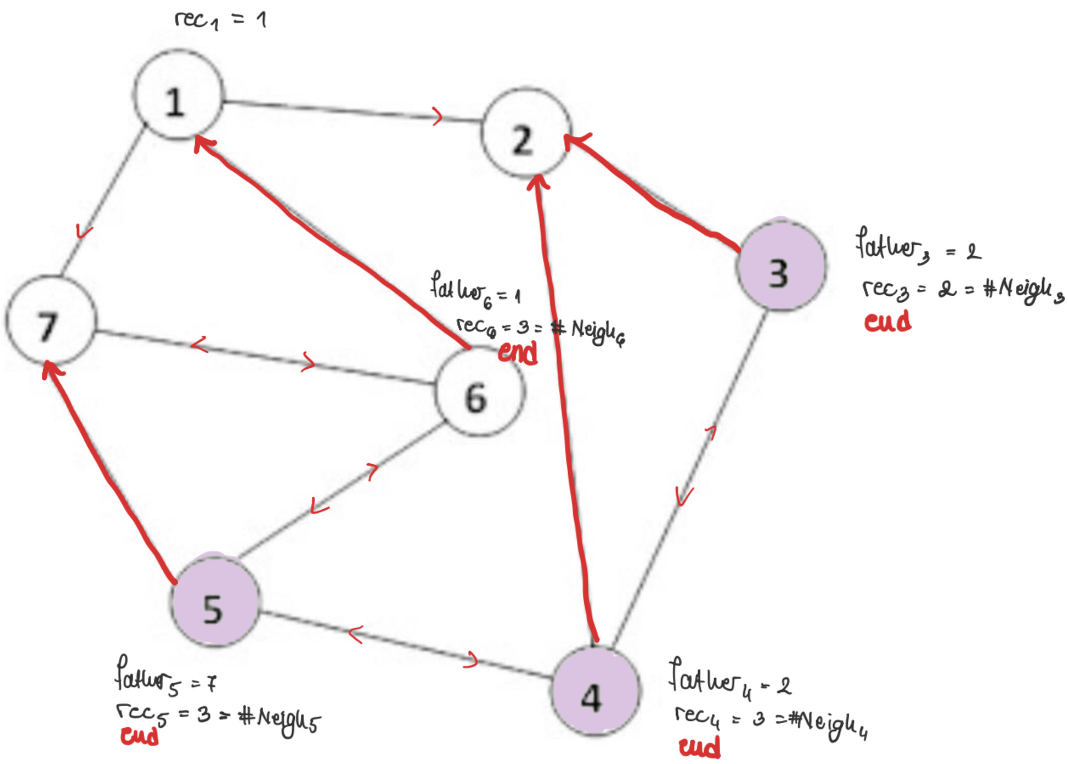
4. If the node has already processed a message with the same identifier, it discards the message and does not forward it.

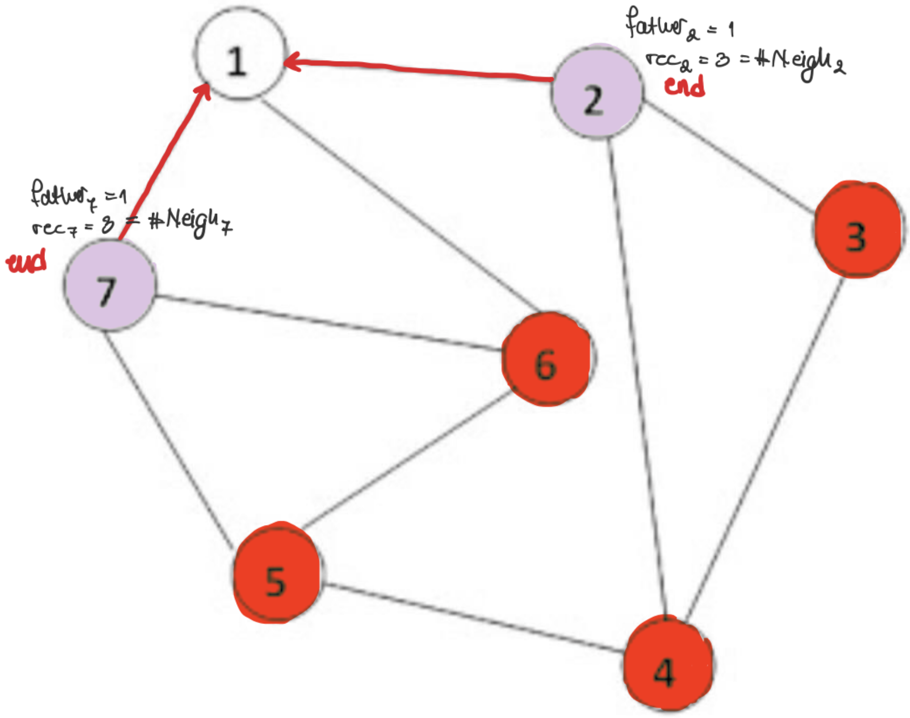
Problem 3.2 *Echo Algorithm Execution*

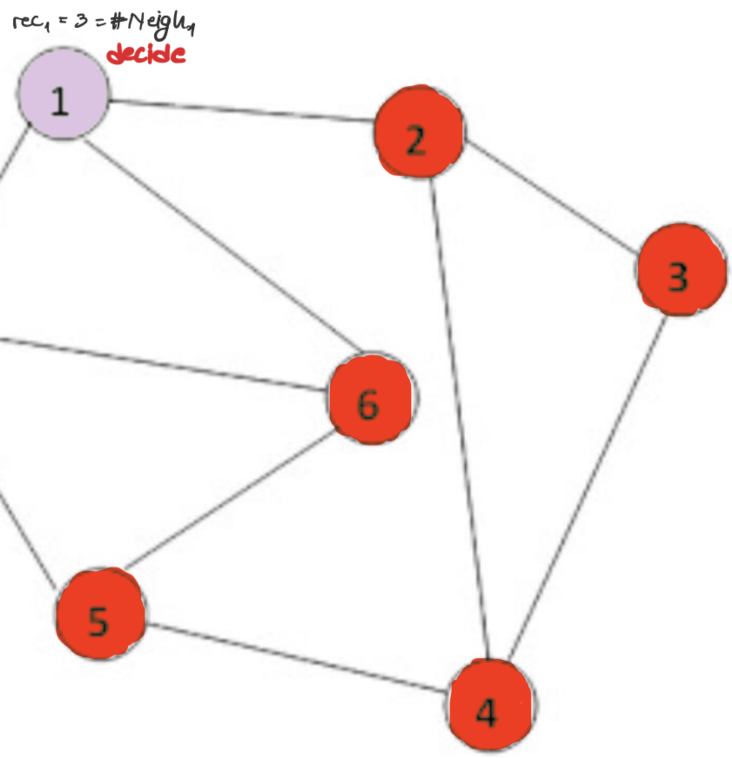





1
2
3
4
5
6
7
father$_7$ = 1
rec$_7$ = 2
father$_6$ = 1
rec$_6$ = 2
father$_3$ = 2
rec$_3$ = 1
father$_5$ = 7
rec$_5$ = 2
*from 7 and 6
father$_4$ = 2
rec$_4$ = 1


rec$_1$ = 1
1
2
3
4
5
6
7
father$_6$ = 1
rec$_6$ = 3 = #Neigh$_6$
end
father$_3$ = 2
rec$_3$ = 2 = #Neigh$_3$
end
father$_5$ = 7
rec$_5$ = 3 = #Neigh$_5$
end
father$_4$ = 2
rec$_4$ = 3 = #Neigh$_4$
end


father$_2$ = 1
rec$_2$ = 3 = #Neigh$_2$
end
father$_7$ = 1
rec$_7$ = 3 = #Neigh$_7$
end
1
2
3
4
5
6
7


rec$_1$ = 3 = #Neigh$_1$
decide
1
2
3
4
5
6
7

Problem 3.4 ***Hypercubes***

a)

The number of ways to traverse the edges of an N-dimensional hypercube can be approximated by N2^(N-1). To see why, consider that each vertex of the hypercube has N edges connected to it, and each edge is shared by exactly two vertices. Therefore, there are N*2^(N-1) edges in the hypercube. To traverse all of these edges, we can start at any vertex and then follow one of its N edges to the next vertex, and continue in this way until we have traversed all N*2^(N-1) edges. However, there are N possible choices for which edge to follow at each vertex, except for the starting vertex, which has only N-1 choices. Therefore, the total number of ways to traverse the edges of the hypercube is approximately N2^(N-1).

b)

To make the choice of path in hypercube traversal deterministic, we can use one of the following methods:

Use a fixed traversal order: A fixed traversal order ensures that adjacent vertices, edges, or faces are visited in a consistent and repeatable order. One example of a fixed traversal order is the binary Gray code, which is a sequence of binary numbers where adjacent numbers differ by only one bit. When applied to a hypercube, a Gray code traversal ensures that adjacent elements are visited in a consistent and repeatable order.

Define a specific starting point and direction: By always starting from the same corner and traversing in the same direction, you can ensure that the traversal is consistent and deterministic. For example, you could start at one corner of the hypercube and traverse along a specific axis or direction.

In summary, to make the choice of path in hypercube traversal deterministic, we can use a fixed traversal order such as binary Gray code or define a specific starting point and direction for the traversal.

c)

https://www.btw2017.informatik.uni-stuttgart.de/slidesandpapers/F8-11-13/paper_web.pdf