Bracha-Toueg Deadlock Detection Algorithm

- A process that suspects it is deadlocked, initiates a (Lai-Yang) snapshot to compute the wait-for graph.
- Each node u takes a local snapshot of:
    - requests it sent or received that weren’t yet granted or dismissed;
    - and grant and dismiss messages in edges.
- Then it computes:
    - Out_u : the nodes it sent a request to (not granted)
    - In_u : the nodes it received a request from (not dismissed)
    - $Requests_u$: the number of grants u requires to become unblocked.

Bracha-Toueg Deadlock Detection Algorithm

Initially $\mathit{notified}_u = \mathit{false}$ and $\mathit{free}_u = \mathit{false}$ at all nodes u.

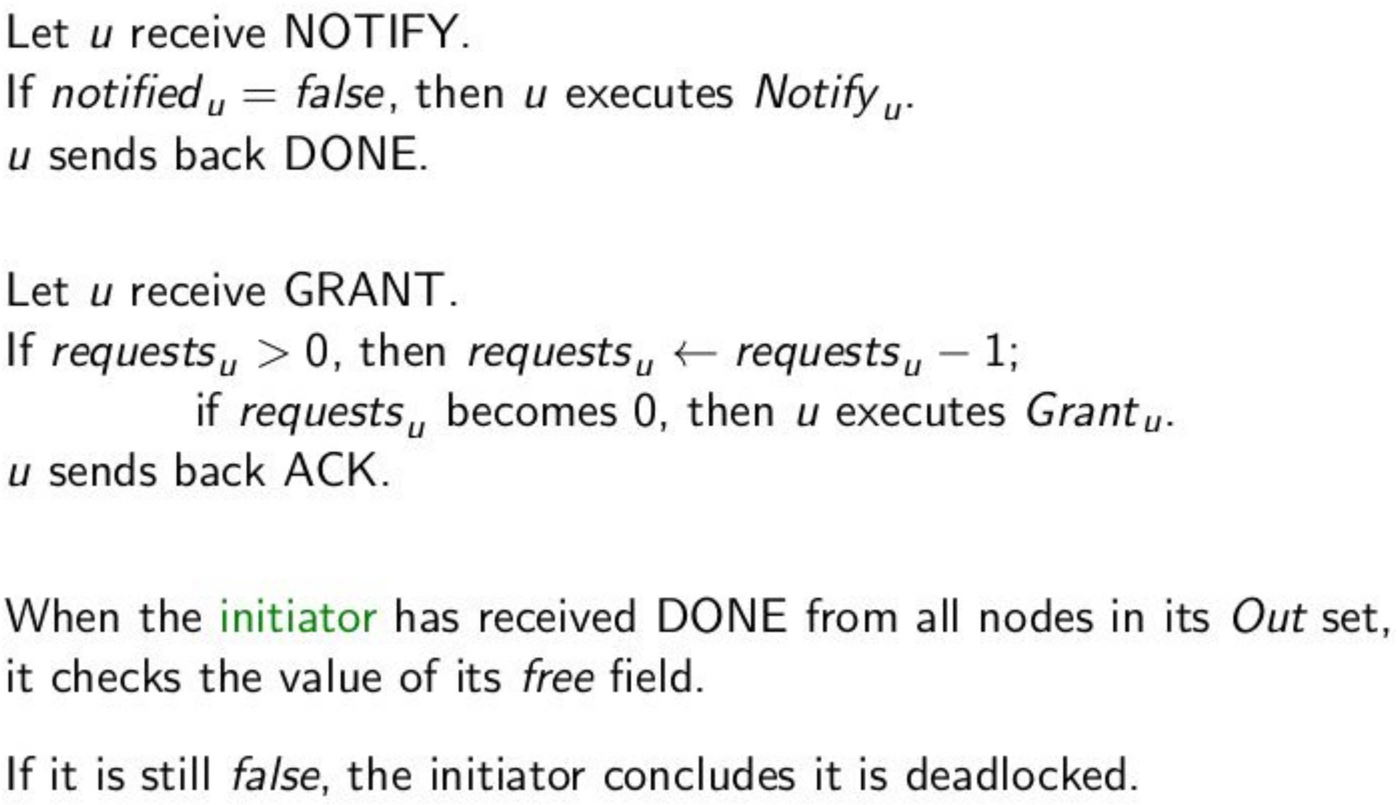
The initiator starts a deadlock detection run by executing *Notify*.

```
$\mathit{Notify}_u$:   $\mathit{notified}_u \leftarrow \mathit{true}$
             for all $w \in \mathit{Out}_u$ send NOTIFY to $w$
             if $\mathit{requests}_u = 0$ then $\mathit{Grant}_u$
             for all $w \in \mathit{Out}_u$ await DONE from $w$

$\mathit{Grant}_u$:    $\mathit{free}_u \leftarrow \mathit{true}$
             for all $w \in \mathit{In}_u$ send GRANT to $w$
             for all $w \in \mathit{In}_u$ await ACK from $w$
```

While a node is awaiting DONE or ACK messages,
it can also process incoming NOTIFY and GRANT messages.

Bracha-Toueg Deadlock Detection Algorithm

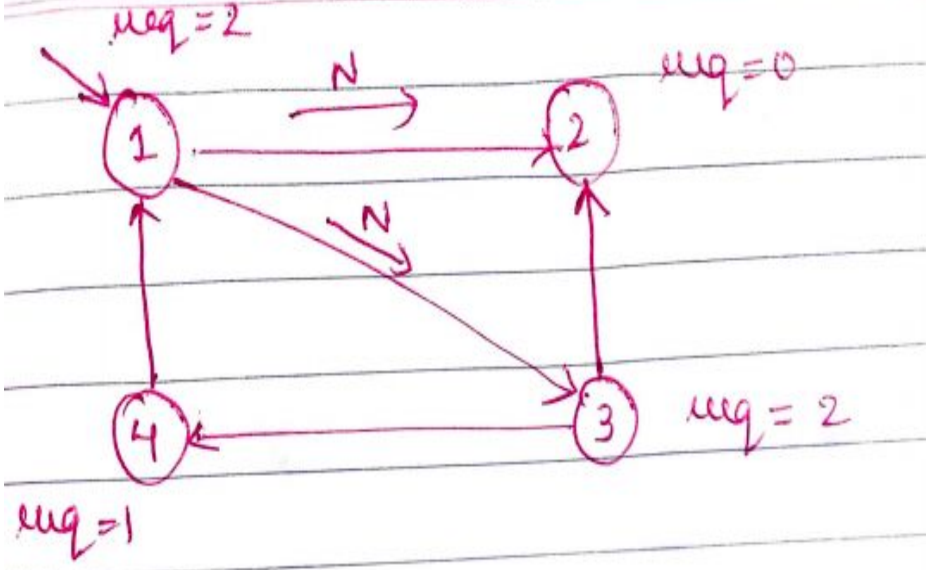


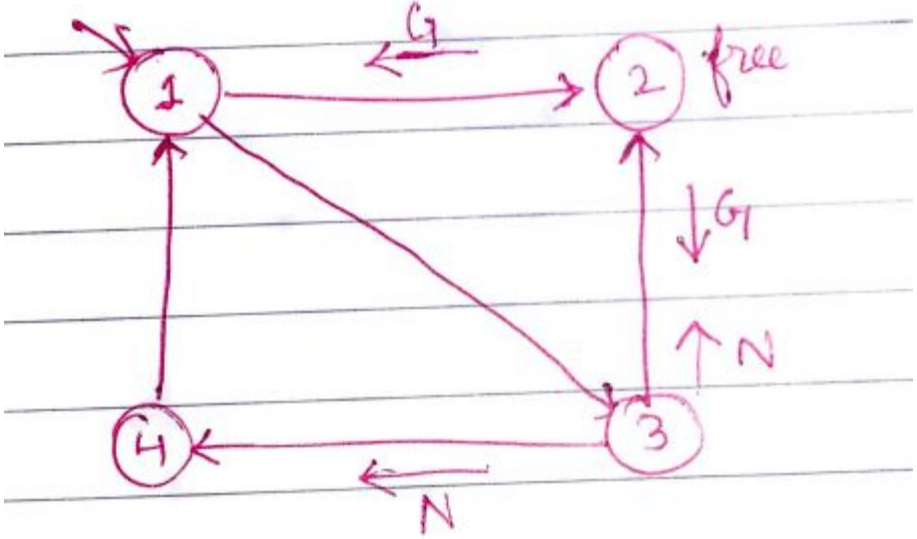
Let u receive NOTIFY.
If $notified_u = false$, then u executes $Notify_u$.
u sends back DONE.

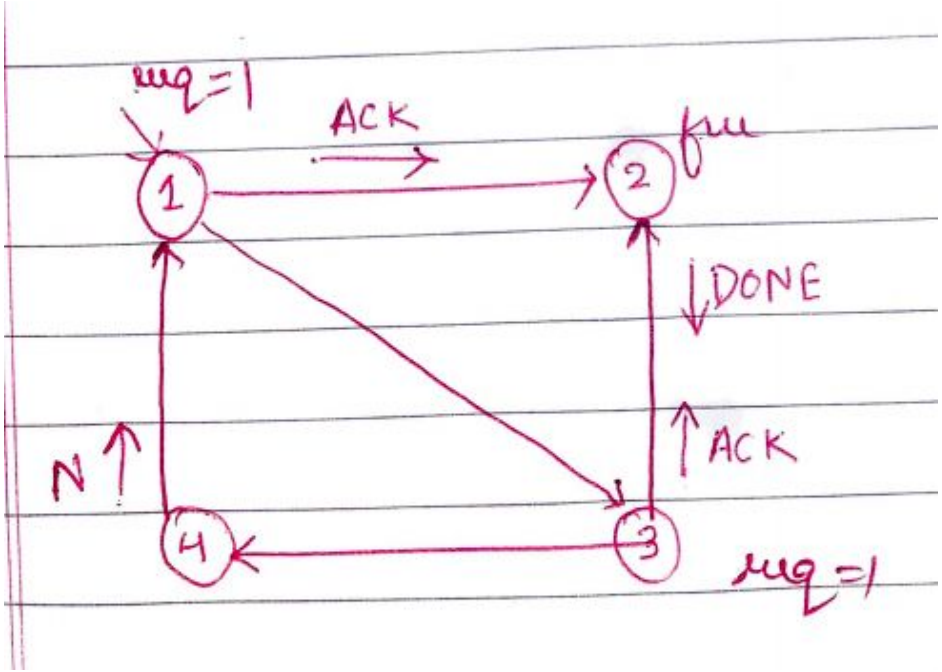
Let u receive GRANT.
If $requests_u > 0$, then $requests_u \leftarrow requests_u - 1$;
        if $requests_u$ becomes 0, then u executes $Grant_u$.
u sends back ACK.

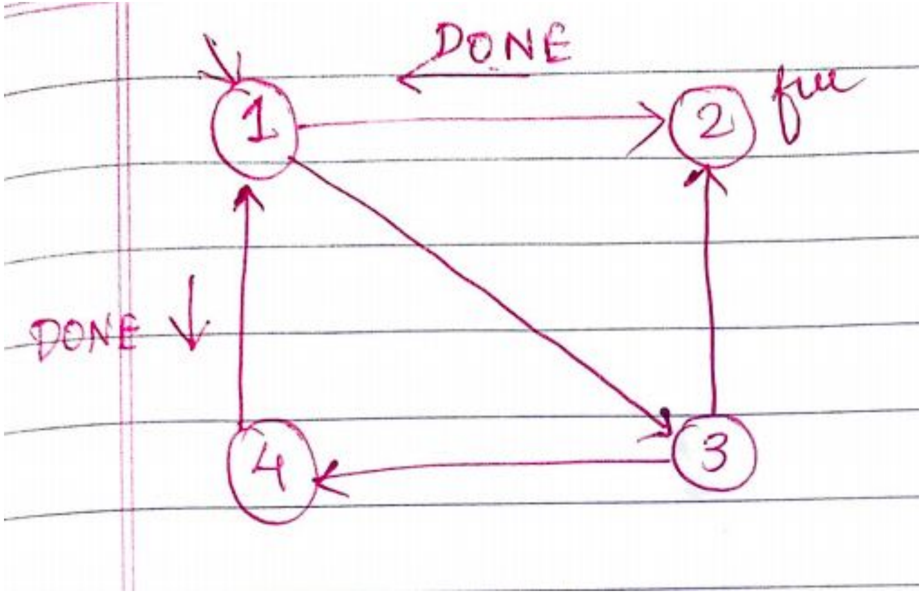
When the initiator has received DONE from all nodes in its *Out* set,
it checks the value of its *free* field.

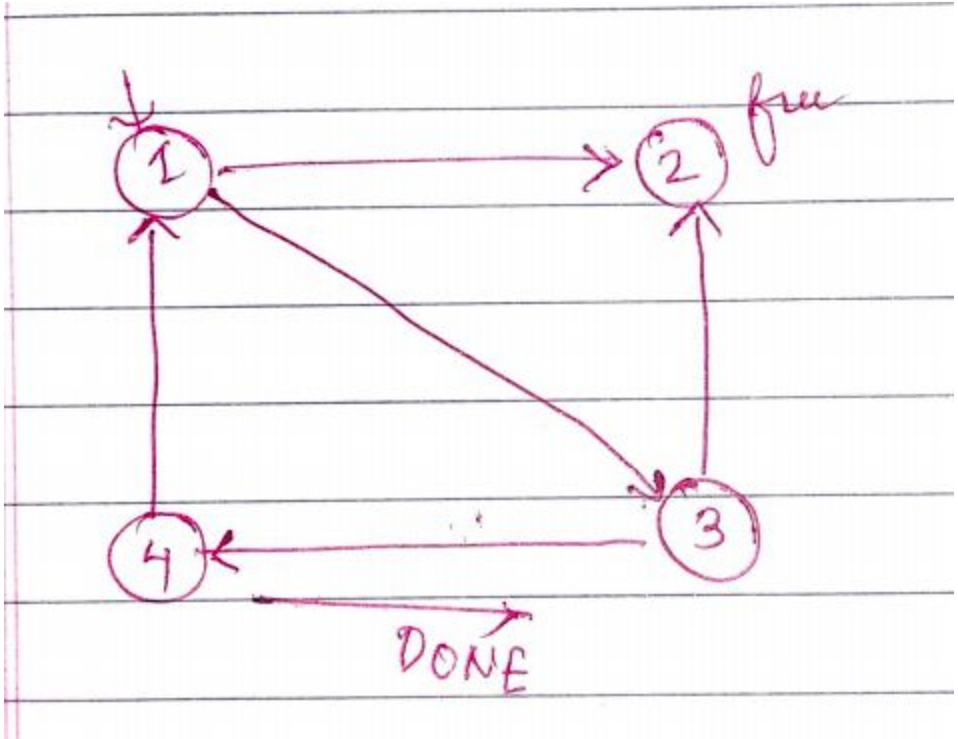
If it is still *false*, the initiator concludes it is deadlocked.

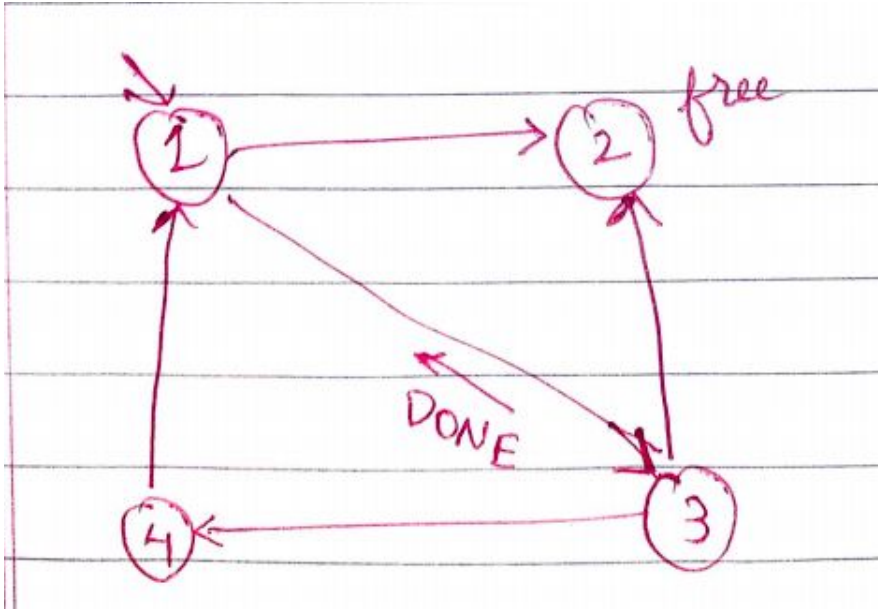

ueq=2
N
ueq=0
1
2
N
4
3
ueq = 2
ueq=1


G
1
2
free
G
N
4
3
N

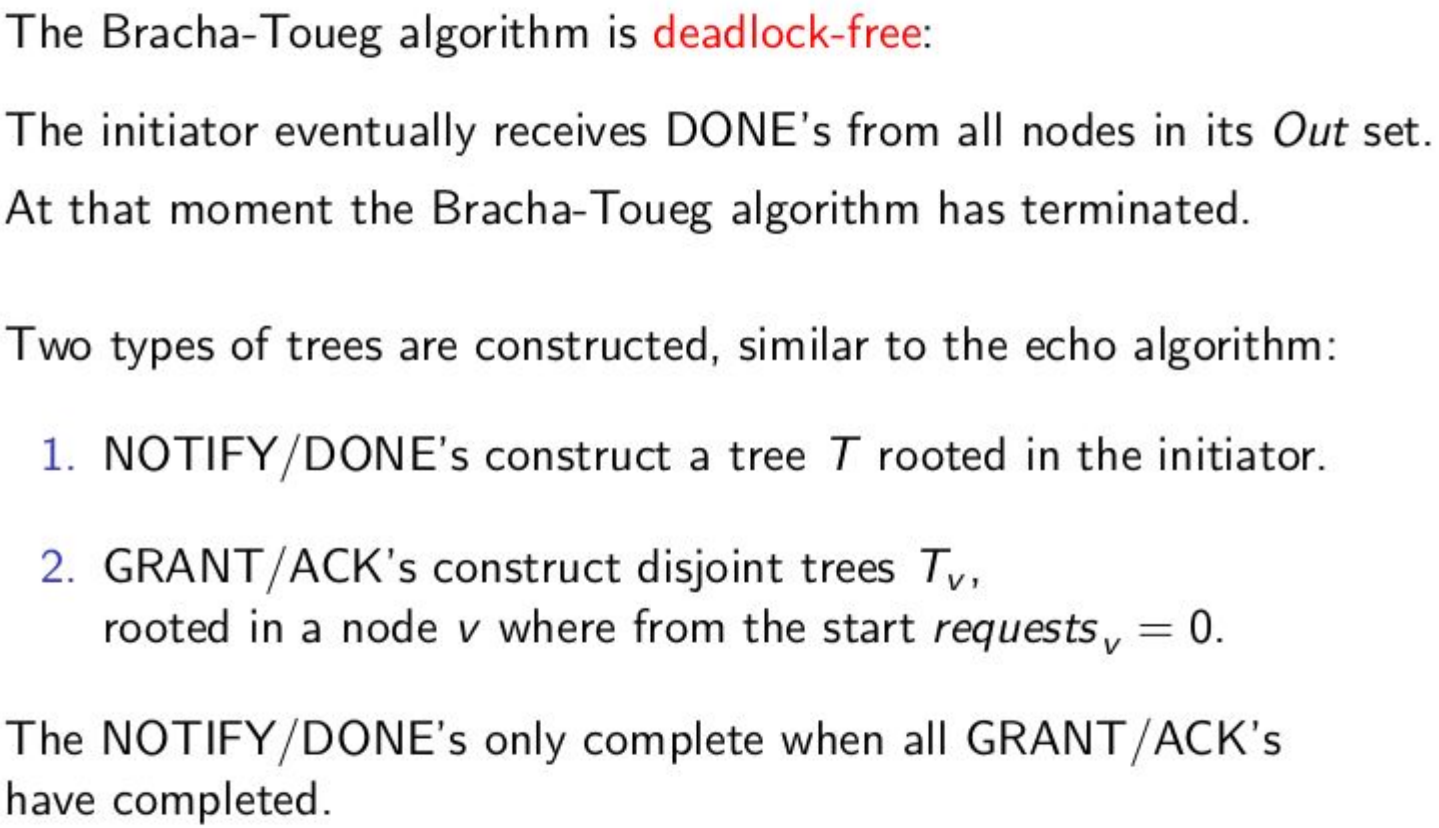

req=1
ACK
1
2
free
DONE
ACK
N
4
3
req=1


DONE
1
2
free
DONE
4
3


1
2
free
3
4
DONE


1
2
free
DONE
3
4

Bracha-Toueg Deadlock Detection Algorithm



The Bracha-Toueg algorithm is deadlock-free:

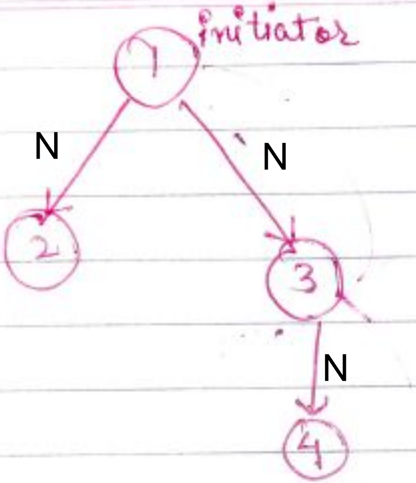
The initiator eventually receives DONE's from all nodes in its *Out* set.

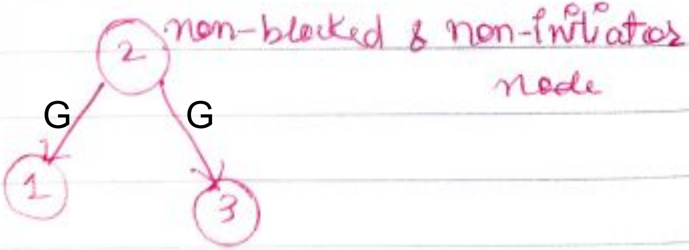
At that moment the Bracha-Toueg algorithm has terminated.

Two types of trees are constructed, similar to the echo algorithm:

1. NOTIFY/DONE's construct a tree T rooted in the initiator.
2. GRANT/ACK's construct disjoint trees T_v,
   rooted in a node v where from the start $requests_v = 0$.

The NOTIFY/DONE's only complete when all GRANT/ACK's have completed.


initiator
1
N
N
2
3
N
4


2
non-blocked & non-initiator node
G
G
1
3

Time Complexity Analysis

- Message Complexity : 4 messages per node, <notify>,<grant>,<done>,<ack>.
- Time complexity: 2d hops, where d = diameter of the WFG.
  - One diameter to send <notify>/<grant> to the farthest node in the graph.
  - Another diameter to receive <done>/<ack> from the same.