Distributed Systems
COMP 212

Lecture 3

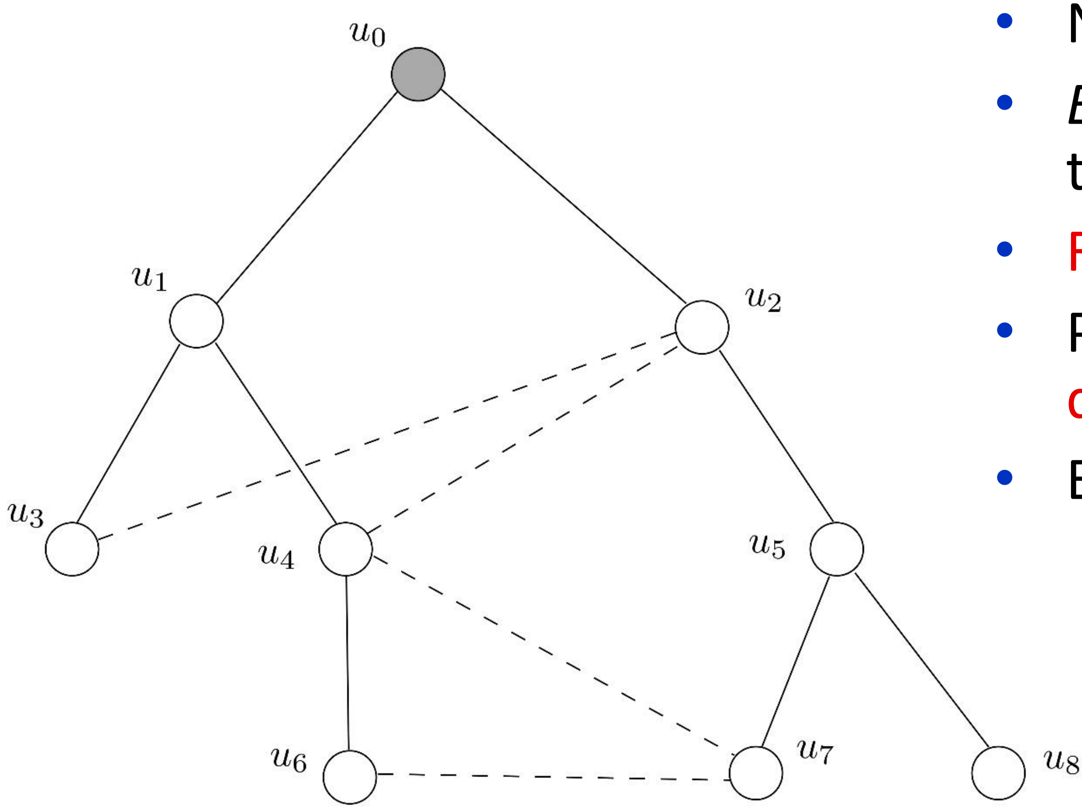
Othon Michail

UNIVERSITY OF LIVERPOOL

Flooding/Broadcast

Broadcast given Spanning Tree

- We start from the case in which a spanning tree of the network is given



- Network $G = (V, E)$
- $E' \subseteq E$ specifies a spanning tree $T = (V, E')$
- Root: u_0 (leader)
- Processors know T in a distributed way
- Each u_i knows:
  - a $parent_i$
  - a set $children_i$

Broadcast given Spanning Tree

Problem:

- u_0 has some information it wishes to send to all processors
  - e.g., a message ⟨M⟩
  - additionally all nodes must have terminated in the end

Solution: Informal Description

- Root u_0 sends ⟨M⟩ to all channels leading to its children and terminates
- When a u_i receives ⟨M⟩ through the channel from its parent
  - it sends ⟨M⟩ to all channels leading to its children and
  - terminates

An Alternative Round

A round:

1. all nodes read incoming messages
2. all nodes update their state
3. all nodes generate new messages and put them in transit
4. all messages are transmitted over the channels and the next round begins

1-3: Local computation by processors

4: Transmission of messages handled by the network (this step could even come first)

Equivalent to the previous type of round

- Use the one that is more convenient to you

Solution: Pseudocode

Algorithm Spanning tree broadcast

State of processor u_i :

- *parent*$_i$: holds a processor index or nil; u_i's parent
- *children*$_i$: holds a set of processor indices (possibly empty); u_i's children
- Boolean *terminated*$_i$: indicates whether u_i has terminated (1) or not (0)

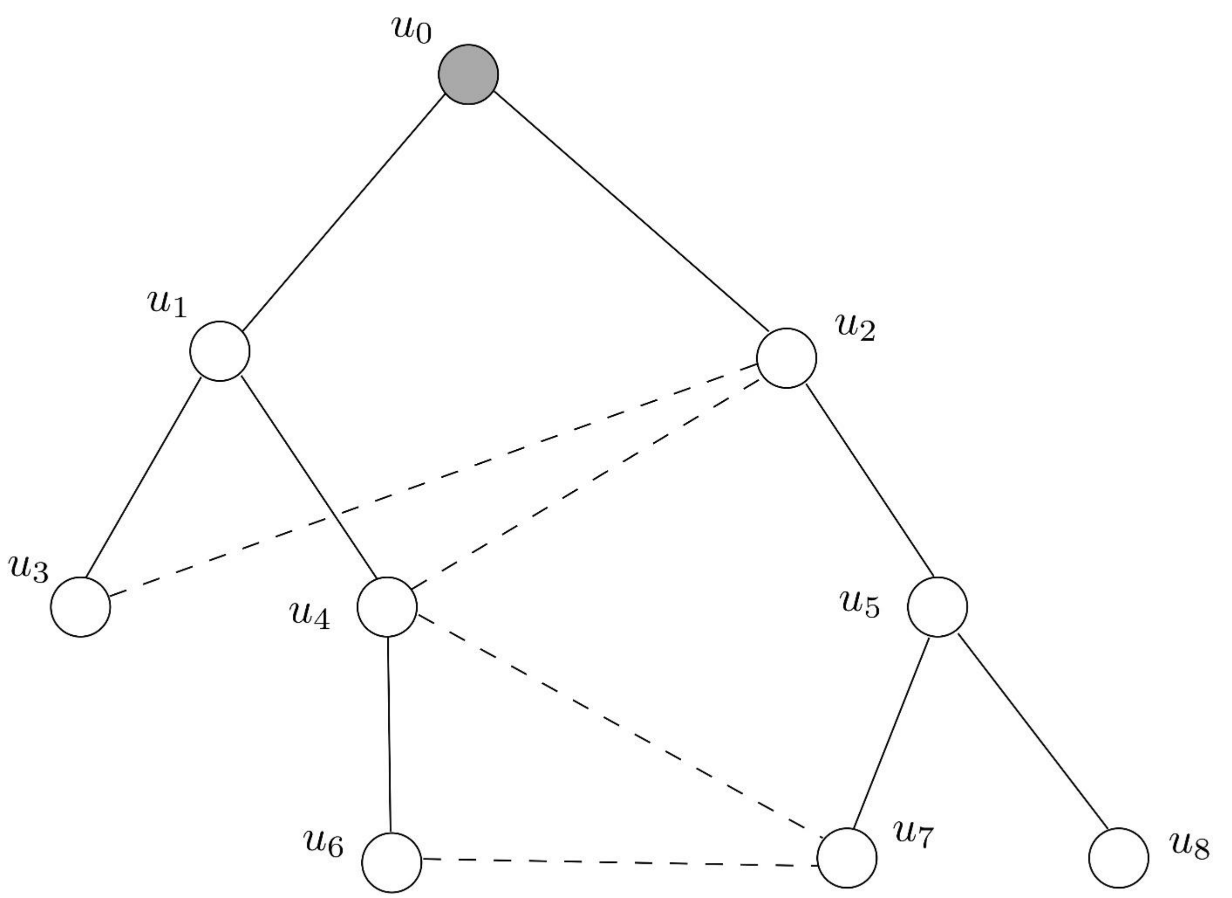
Solution: Pseudocode

Algorithm Spanning tree broadcast
Initially u_0 knows ⟨M⟩

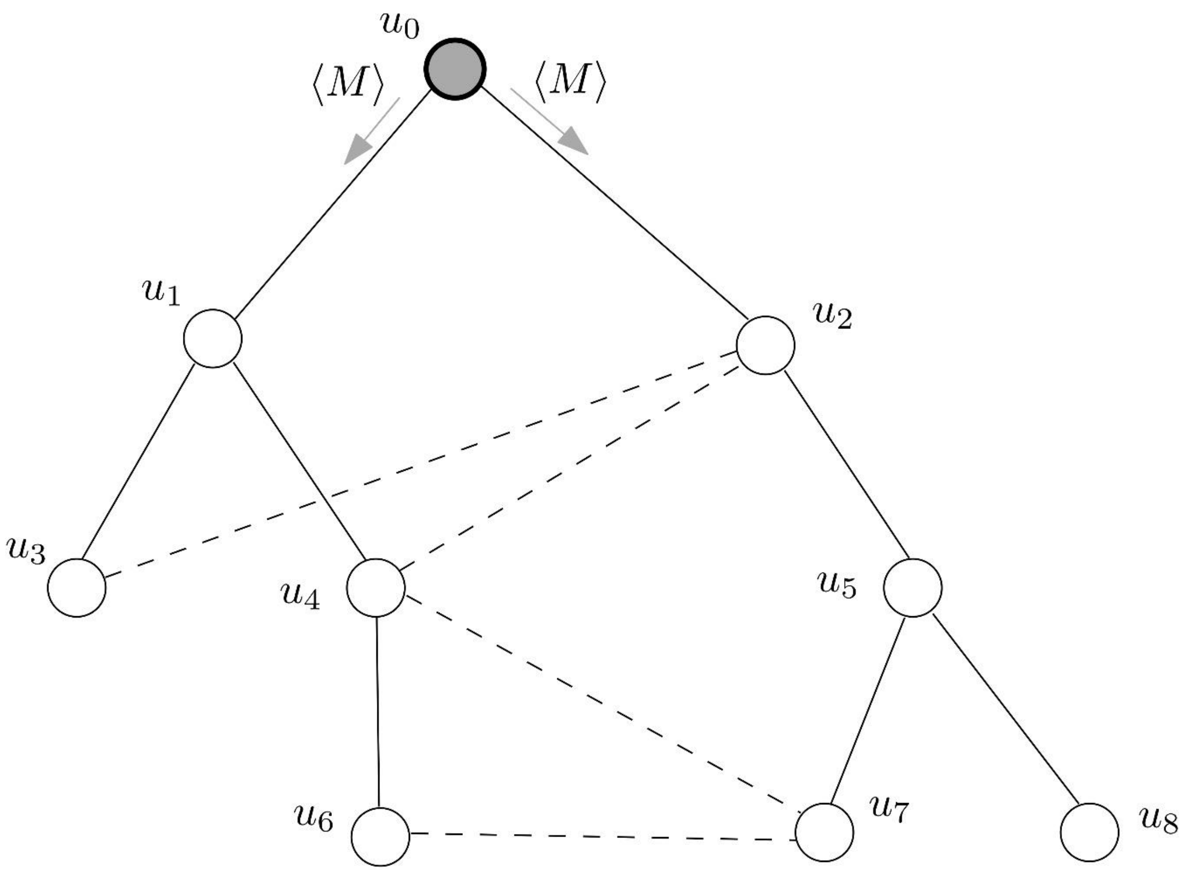
Code for leader (u_0) :
  send ⟨M⟩ to all children
  terminate

Code for non-leader:
  upon receiving ⟨M⟩ from parent:
    send ⟨M⟩ to all children
    terminate

Example Execution

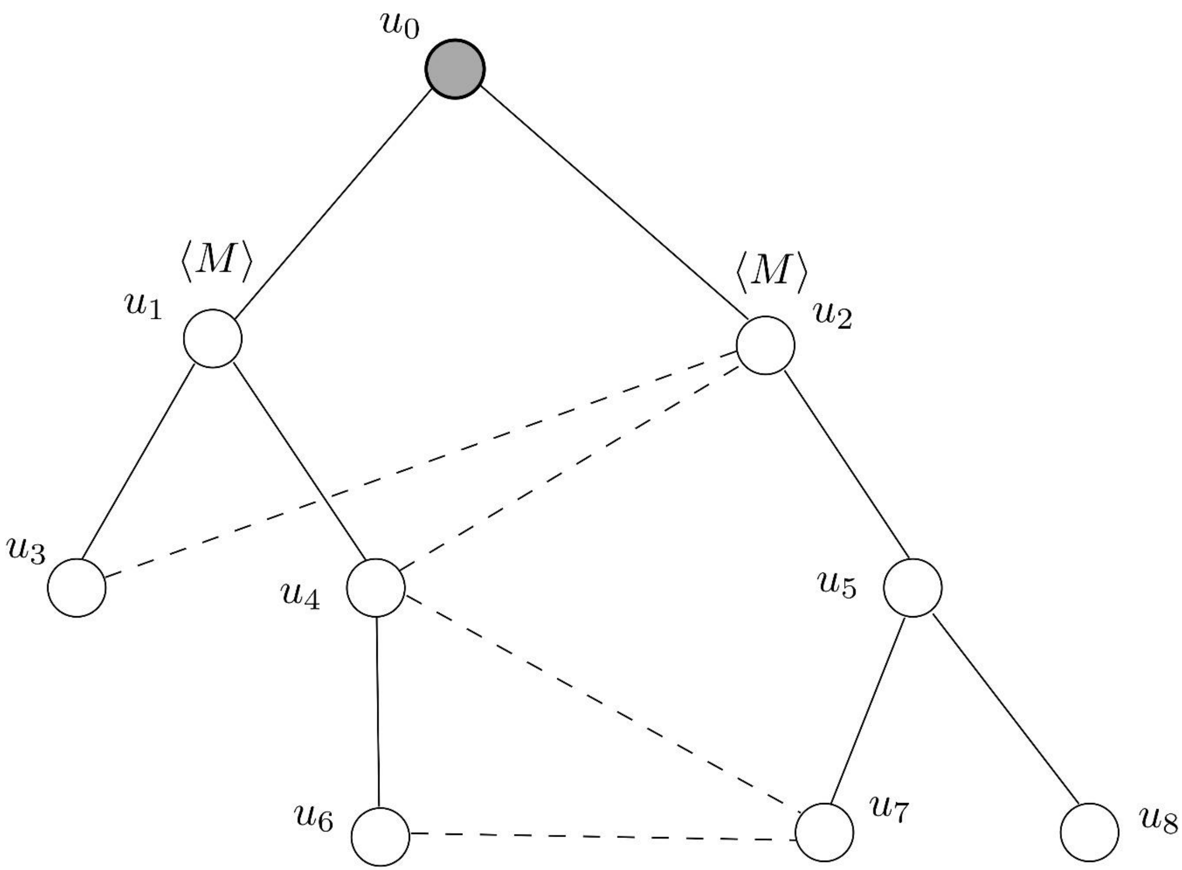


Example Execution



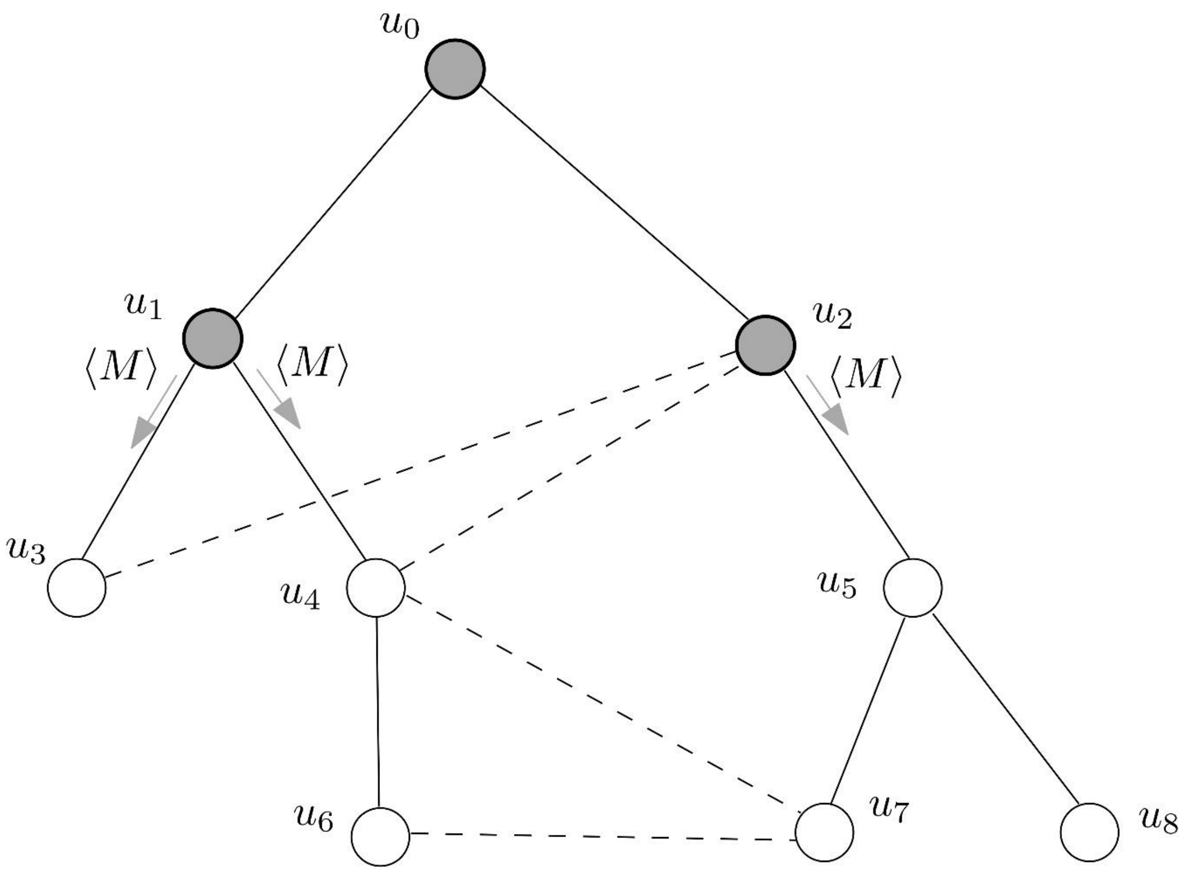
$round = 1$

Example Execution



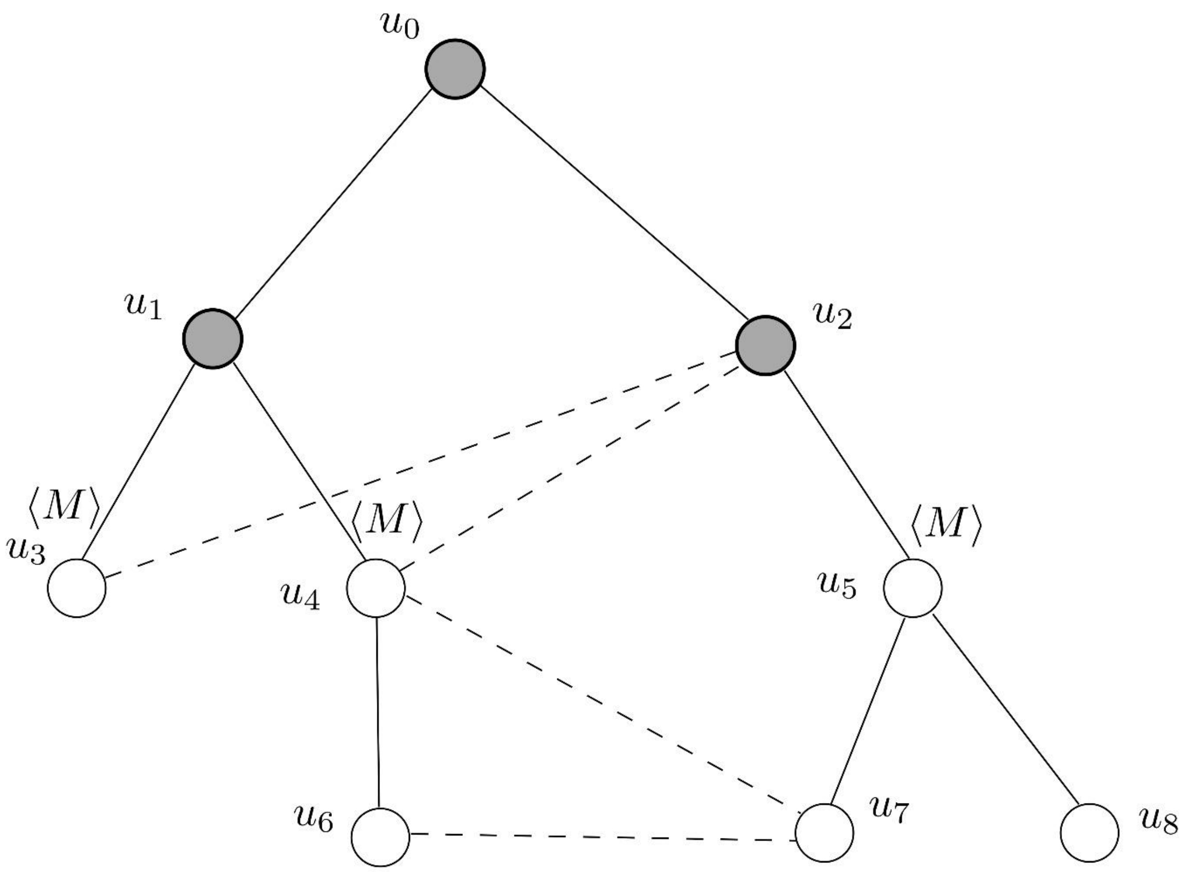
$round = 1$

Example Execution



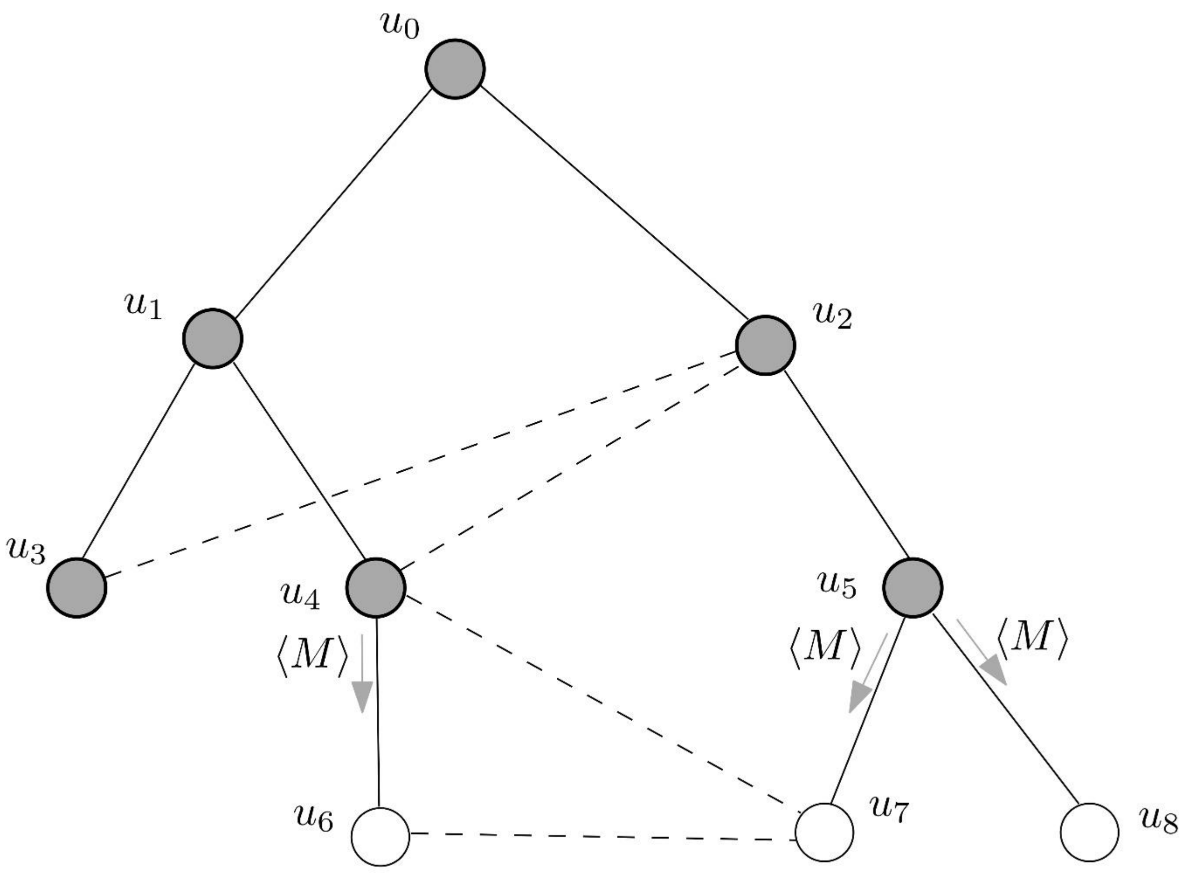
$round = 2$

Example Execution



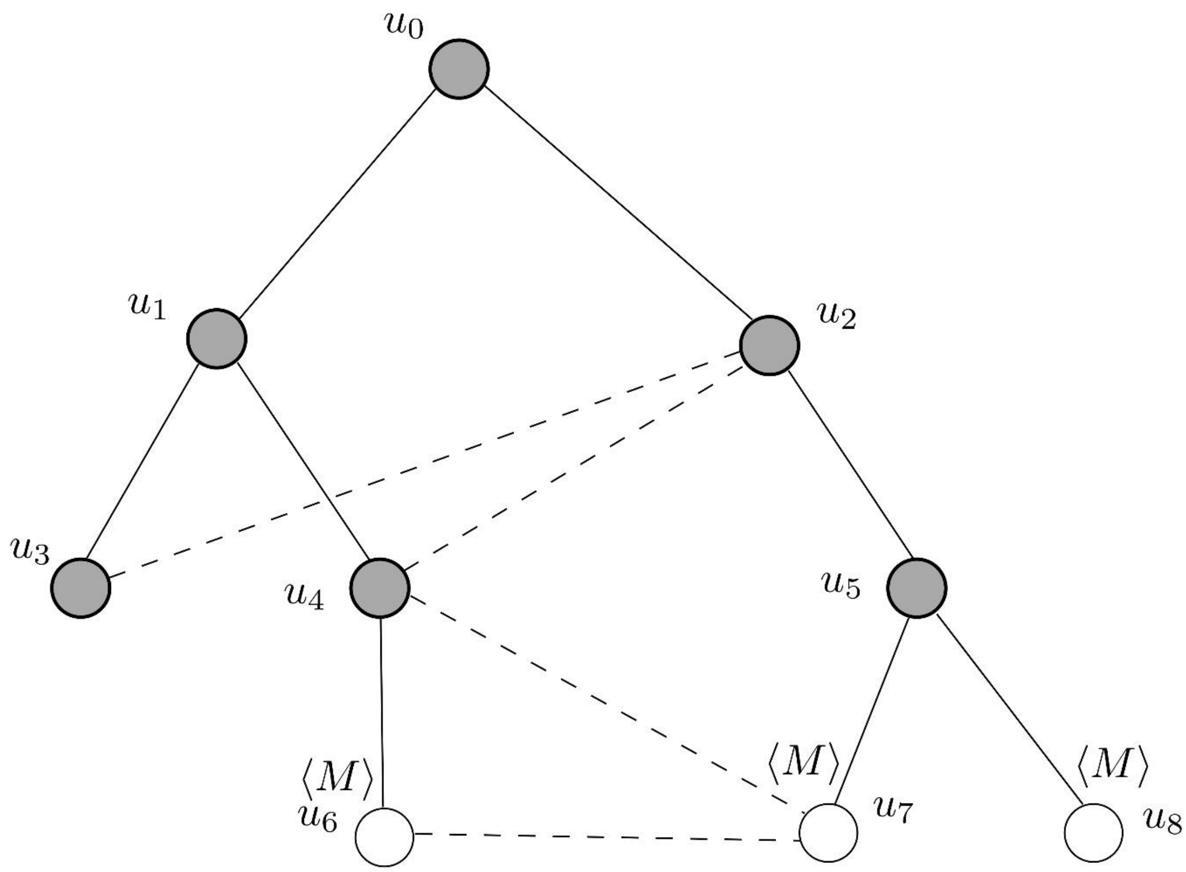
$round = 2$

Example Execution



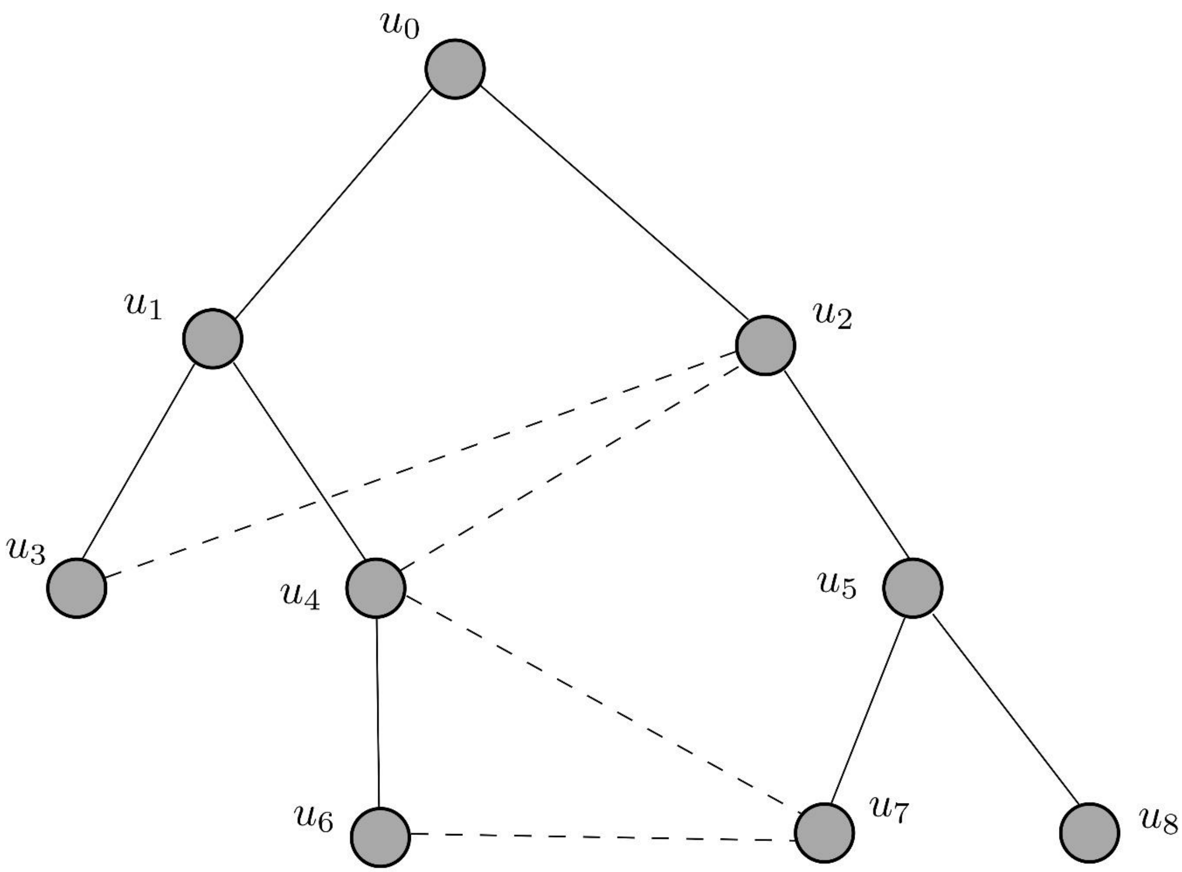
$round = 3$

Example Execution



$round = 3$

Example Execution



$round = 4$

Correctness and Performance

When we devise an algorithm we typically should

1. Convince that it is correct
2. Analyse its performance

- Correctness:
  - Usually a proof that the algorithm does as expected
- Performance:
  - Time Complexity (e.g., #rounds required)
  - Space Complexity (e.g., memory used by processors)
  - Communication Complexity (e.g., total #messages transmitted, size of messages)