Distributed Systems
COMP 212

Lecture 4

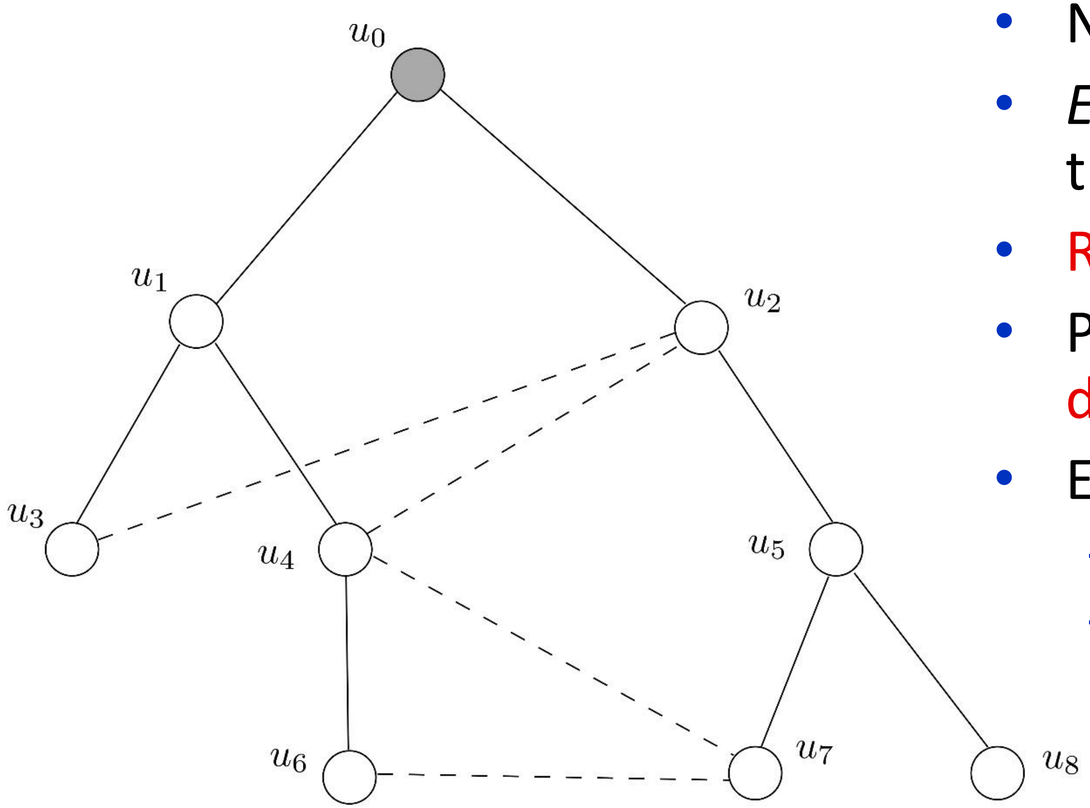
Othon Michail

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: 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

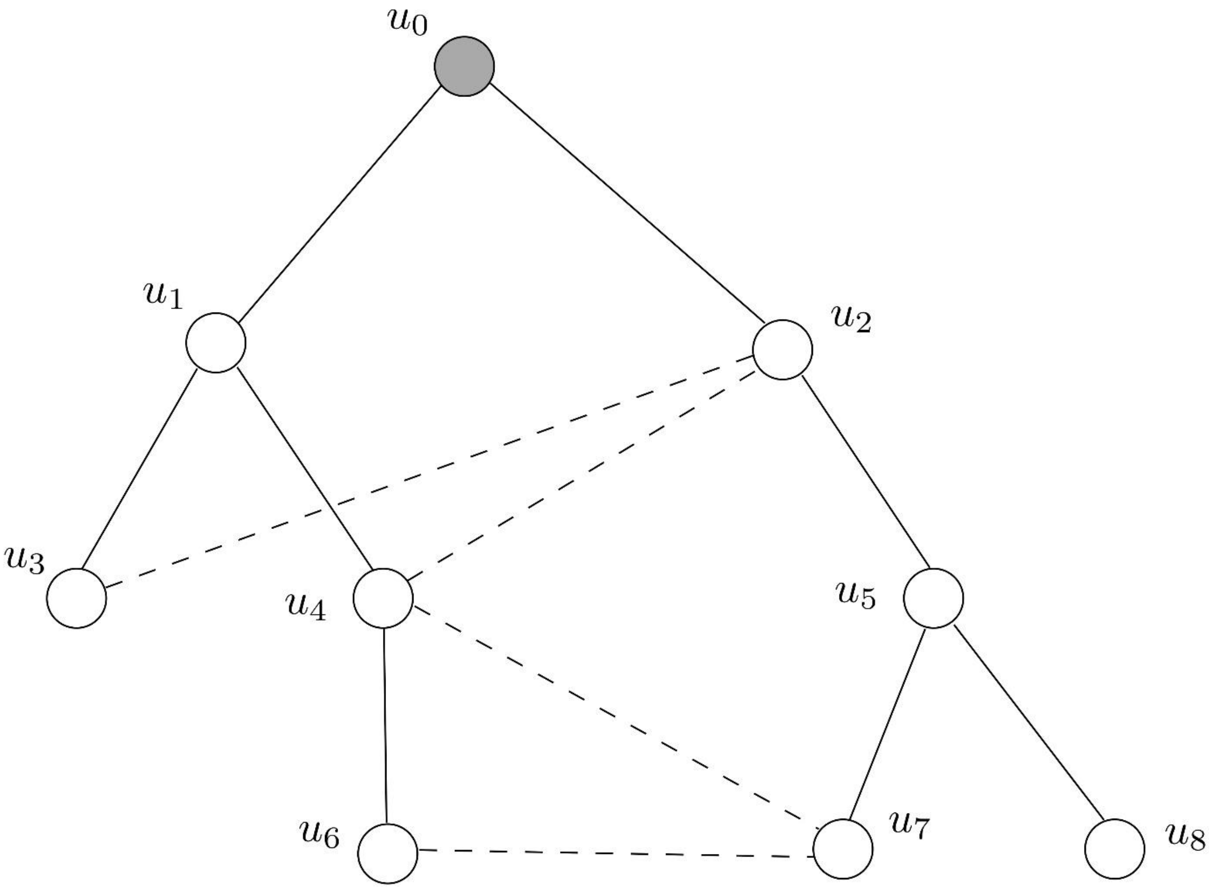
Code for non-leader:

  upon receiving ⟨M⟩ from parent:

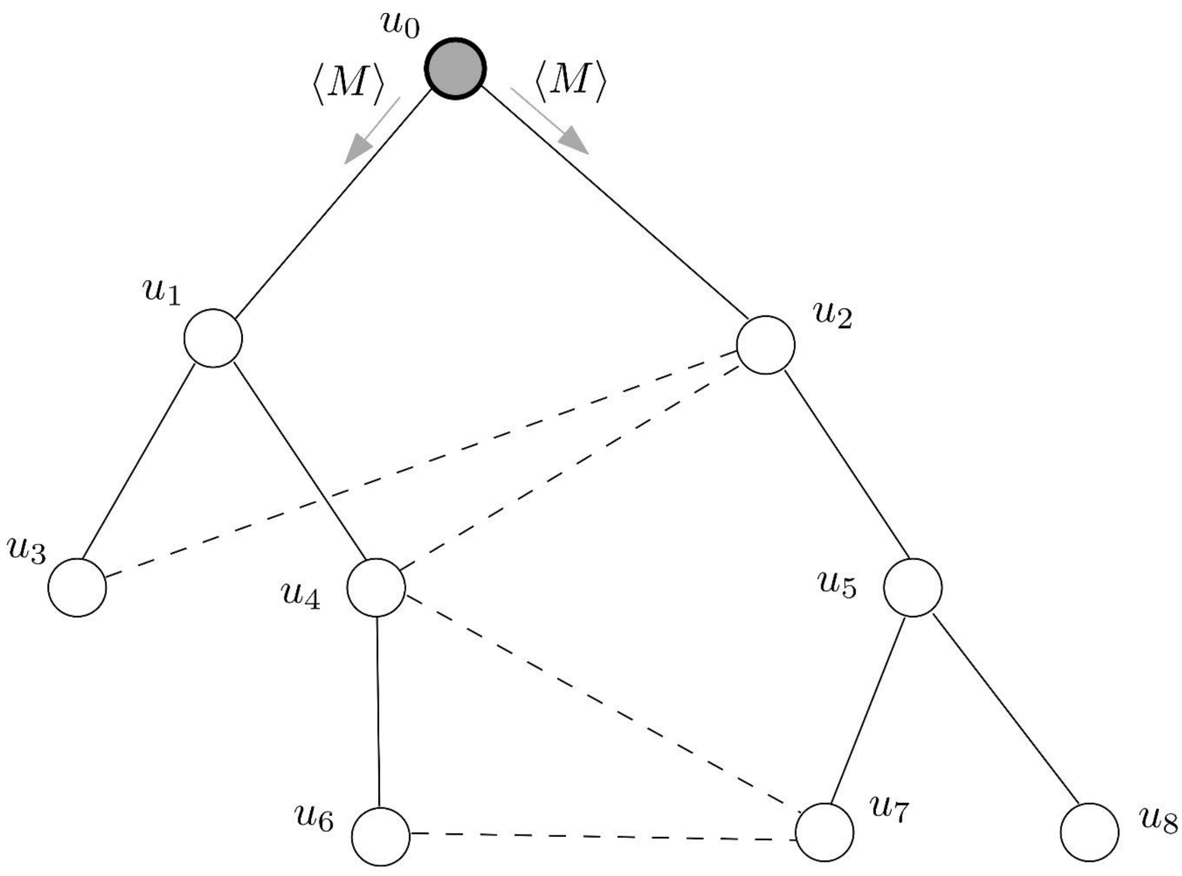
    send ⟨M⟩ to all children

    terminate

Example Execution

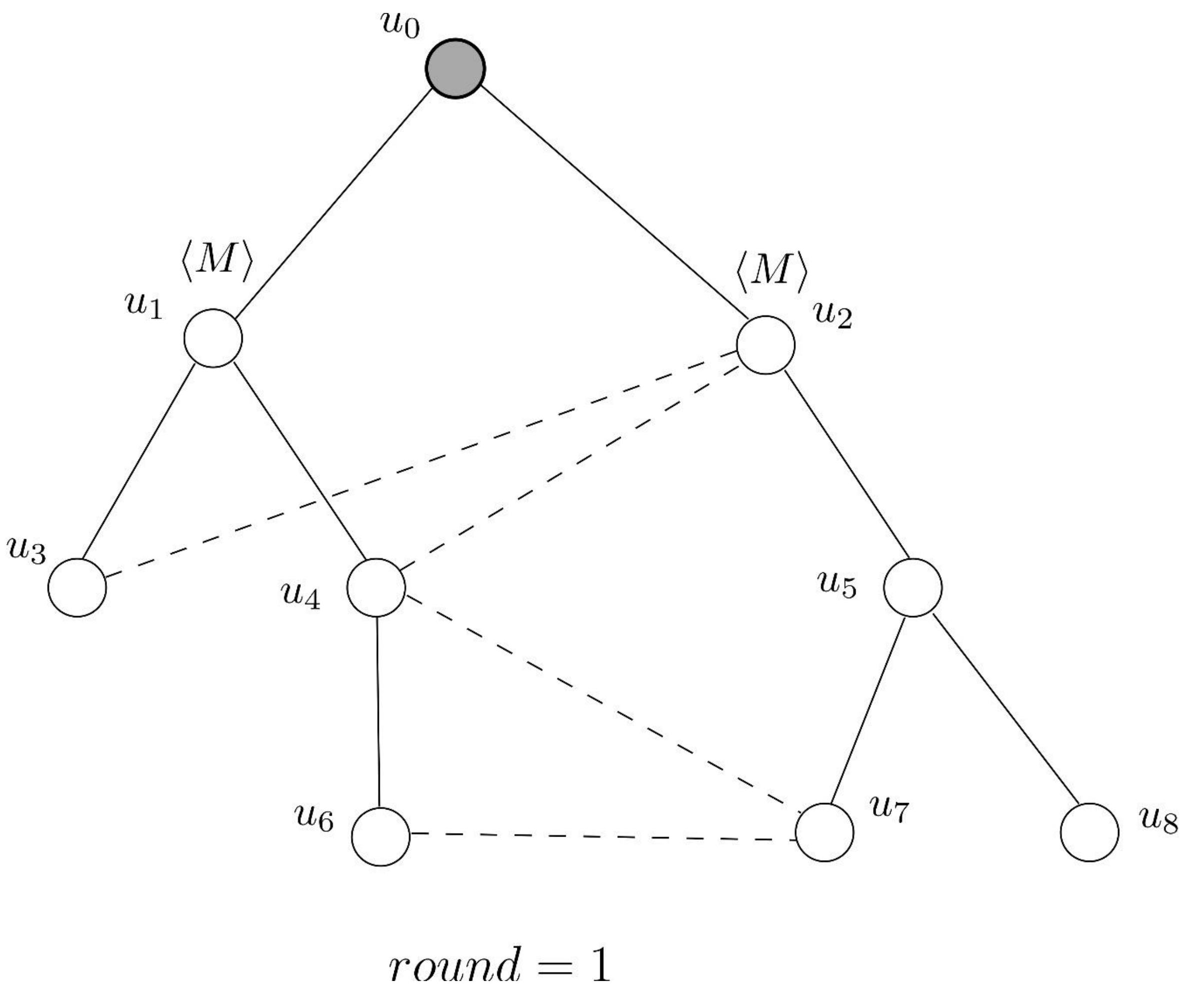


Example Execution

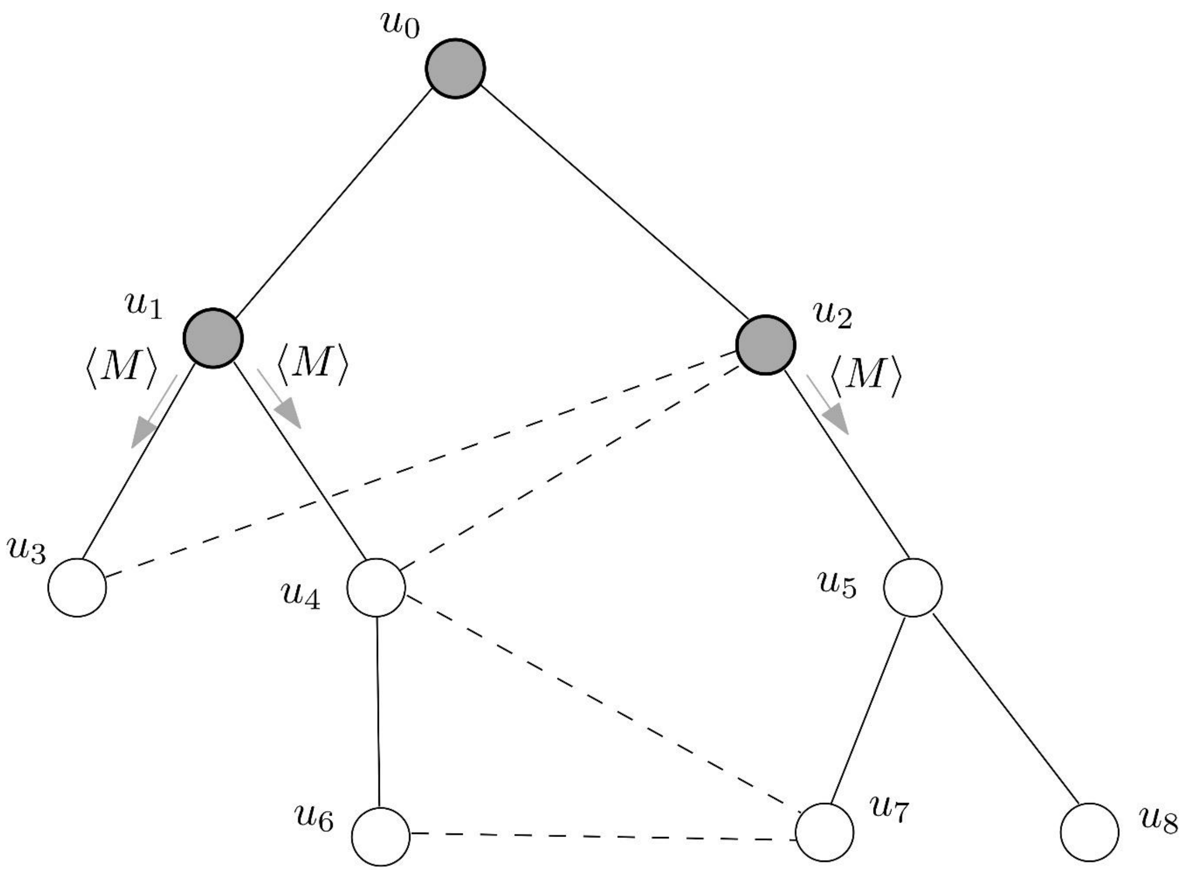


$round = 1$

Example Execution

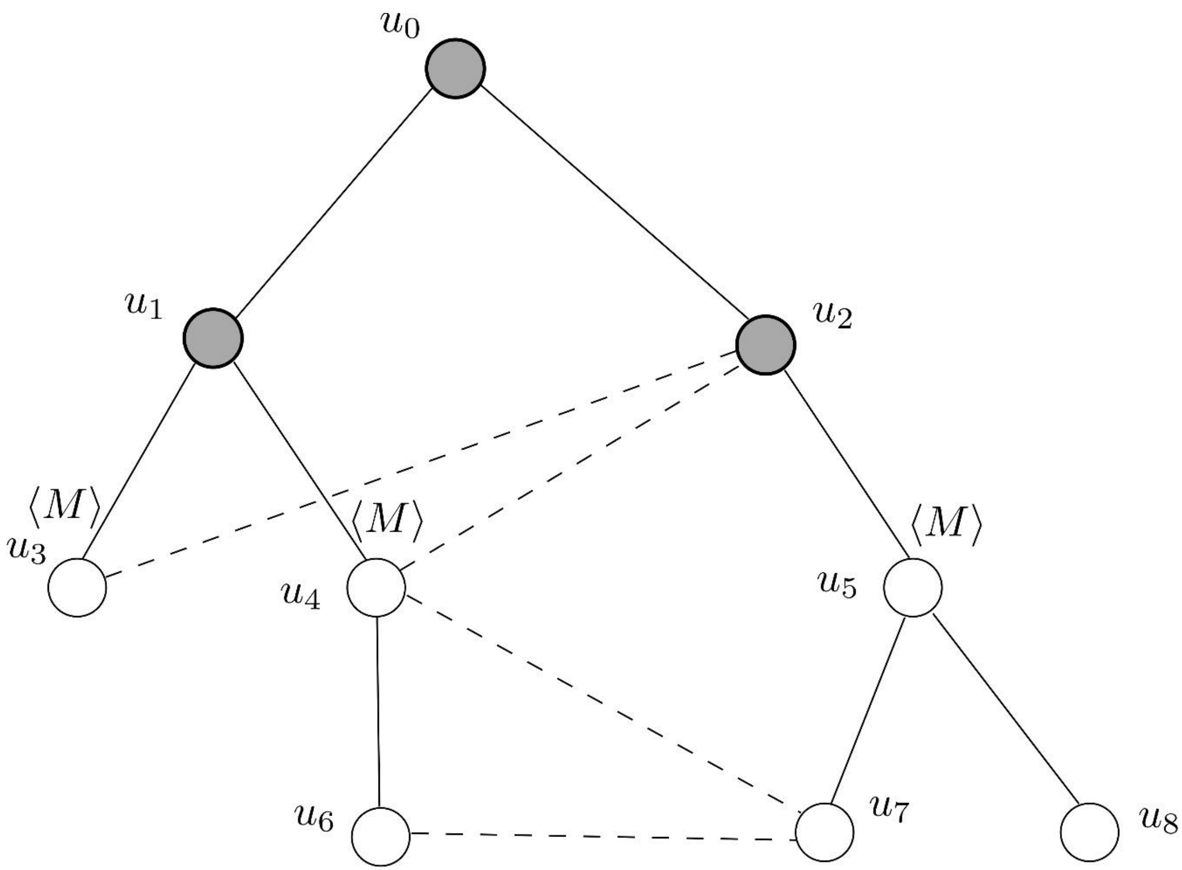


Example Execution



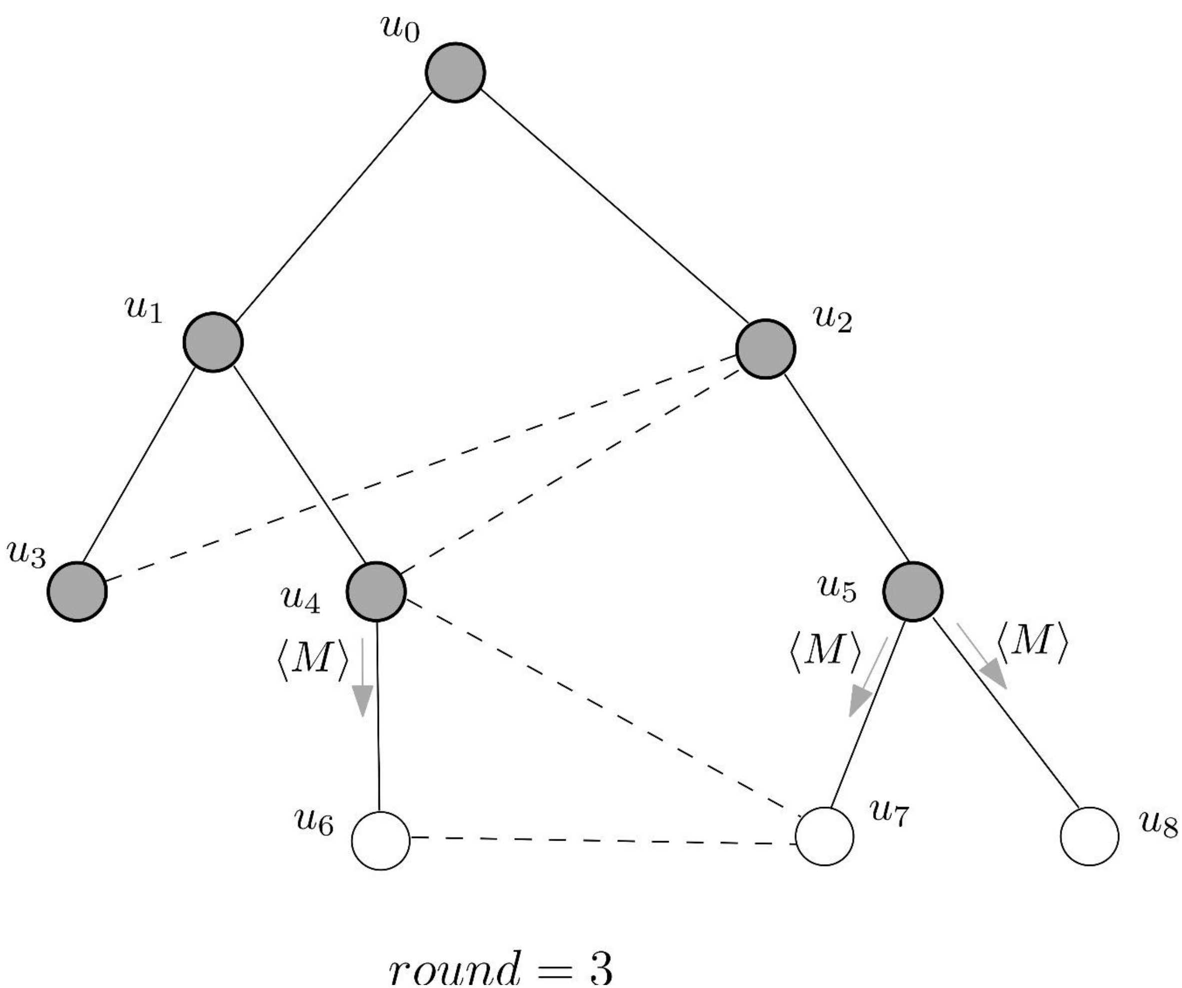
$round = 2$

Example Execution

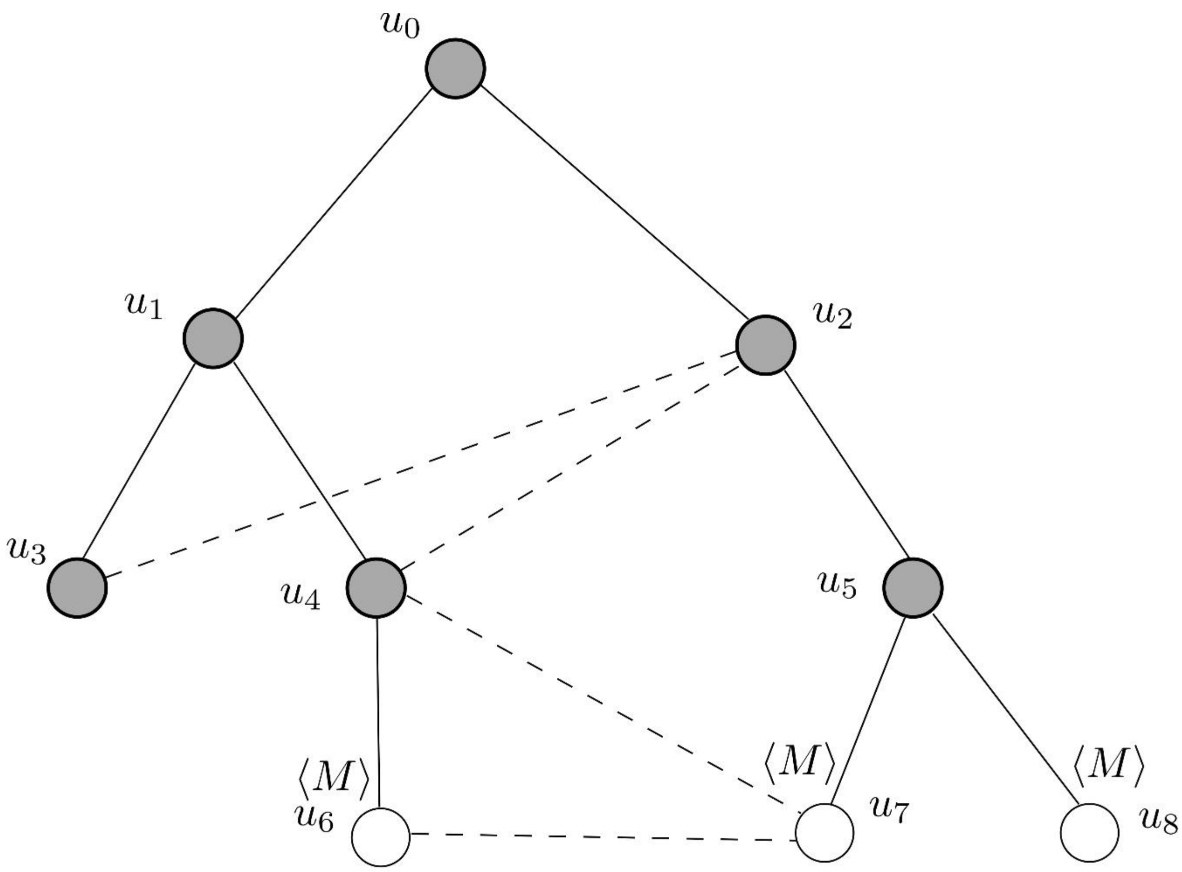


$round = 2$

Example Execution

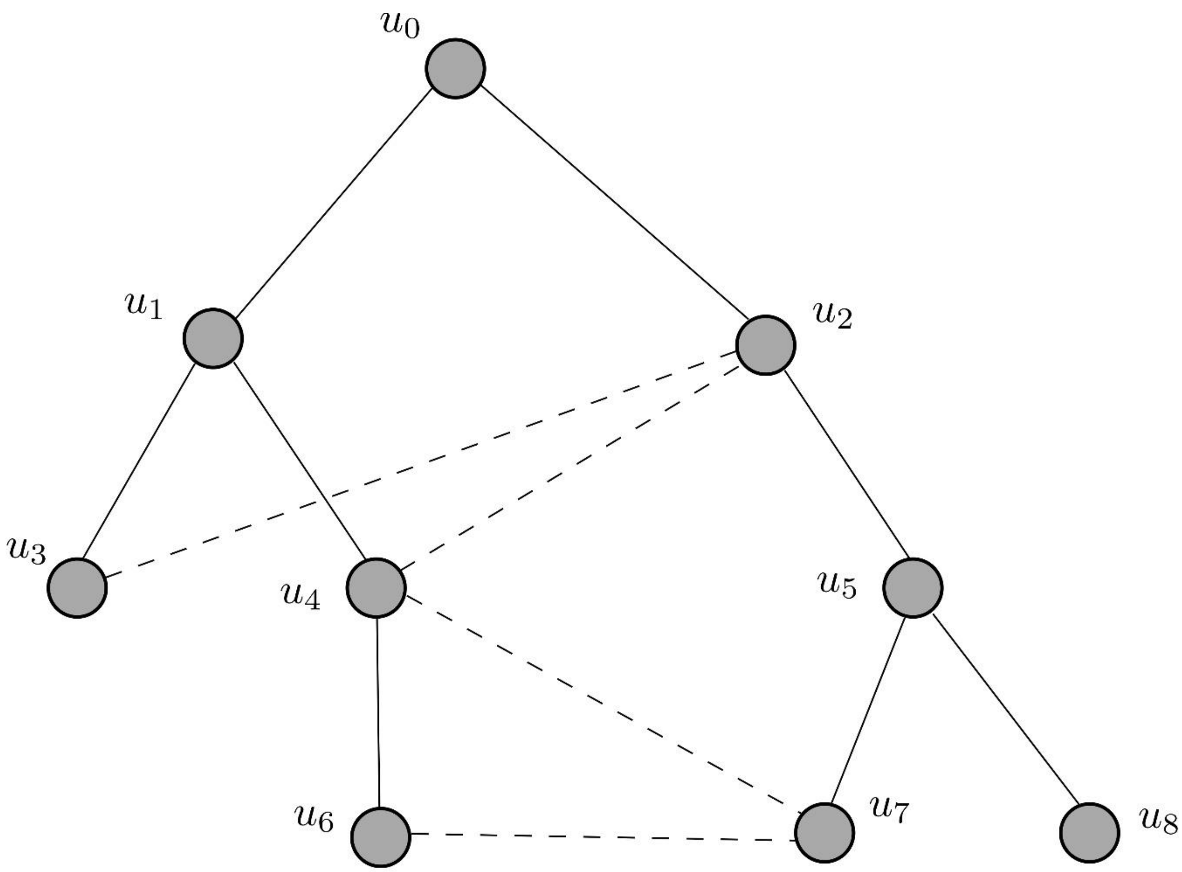


Example Execution



$round = 3$

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); also called message complexity

Spanning Tree Broadcast Correctness

Fairly easy for this algorithm:

- Show that every node will receive ⟨M⟩ and terminate
  - Whenever a u_i receives ⟨M⟩ it terminates by the end of that round
  - Suffices to show that every u_i will receive ⟨M⟩

Proof. Take any $u_i \neq u_0$. As the tree T is spanning, there is a single tree-path from u_0 *to* u_i . By the way the algorithms works, ⟨M⟩ will be forwarded hop-by-hop on the path until it reaches u_i . And this holds for all u_i in the network. □

S. T. Broadcast Time Complexity

- Equal to the #rounds until all nodes have received ⟨M⟩

Lemma. For every u_i whose distance from u_0 in the spanning tree is r, it holds that u_i receives ⟨M⟩ in round r.

Proof. By induction on r.

- For $r = 1$: Holds because in round 1, u_0 transmits ⟨M⟩ to all its children who receive it in round 1
- Assume that it holds for any $r - 1 \geq 1$
  - Means that all processors at distance $r - 1$ receive ⟨M⟩ in round $r - 1$
- Then it must hold also for r
  - The parent u_i of any u_j at distance r is at distance $r – 1$
  - By previous assumption, the parent received ⟨M⟩ in round $r – 1$, therefore transmits it to all its children including u_j in round r, and u_j receives it in round r □

S. T. Broadcast Time Complexity

- This means that for a tree T of depth d the algorithm requires d rounds
- But in general we want our algorithm to run on all possible networks and all their possible spanning trees
  – And not all have the same depth…
- What is the worst-case time complexity of our algorithm?

S. T. Broadcast Time Complexity

- This means that for a tree *T* of depth *d* the algorithm requires *d* rounds
- But in general we want our algorithm to run on all possible networks and all their possible spanning trees
  - And not all have the same depth…
- What is the worst-case time complexity of our algorithm?
- It is the worst-case depth of a spanning tree on *n* nodes

S. T. Broadcast Time Complexity

- This means that for a tree *T* of depth *d* the algorithm requires *d* rounds
- But in general we want our algorithm to run on all possible networks and all their possible spanning trees
  - And not all have the same depth…
- What is the worst-case time complexity of our algorithm?
- It is the worst-case depth of a spanning tree on n nodes $= n - 1$

S. T. Broadcast Communication Complexity

1. Size of largest message transmitted?

S. T. Broadcast Communication Complexity

1. Size of largest message transmitted?
   – Size of ⟨M⟩, typically in bits

- For example, if ⟨M⟩ is a single processor identifier (or *id*) this would typically be $O(\log n)$ bits

S. T. Broadcast Communication Complexity

2. Total #messages transmitted?

- A single observation suffices
- Any ideas?

S. T. Broadcast Communication Complexity

2. Total #messages transmitted?

- A single observation suffices

Observation. *For every edge of the spanning tree, from a parent u_i to a child u_j, exactly one message will be ever transmitted through it.*

- The single transmission of ⟨M⟩ from u_i *to* u_j
- As u_i then terminates it will not happen again

- What is the total #messages then?

S. T. Broadcast Communication Complexity

2. Total #messages transmitted?

- A single observation suffices

Observation. *For every edge of the spanning tree, from a parent u_i to a child u_j, exactly one message will be ever transmitted through it.*

- The single transmission of ⟨M⟩ from u_i *to* u_j
- As u_i then terminates it will not happen again

- What is the total #messages then?
  - #edges of a spanning tree on *n* nodes
  - Always *n* - 1

Spanning Tree Broadcast Summing-up

Theorem. *The Spanning Tree Broadcast algorithm solves the broadcast problem in any connected synchronous network G when a rooted spanning tree T of G is known in advance. The time complexity of the algorithm (in rounds) is equal to the depth d of T. For the communication complexity, the algorithm transmits a total of n - 1 messages and the maximum size of a message is equal to the binary representation of the information to be broadcast.*

Broadcast without a 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
- No spanning tree of the network G is given in advance
- The algorithm should also output a constructed spanning tree of G

Solution: Informal description

- All nodes awake initially
- If awake and have just received ⟨M⟩ from some neighbours,
  - Choose one of those neighbours as your *parent* and let him know
  - forward ⟨M⟩ to the rest of the neighbours
  - Wait for 1 round to collect children (if any) and then sleep
- If neighbours inform you that you are their parent,
  - add those processors to your *children* list
  - sleep
- If you are asleep, do nothing

Solution: Pseudocode

Algorithm Broadcast & Spanning tree construction
Code for processor u_i, $i \in \{0, 1, \ldots, n - 1\}$:

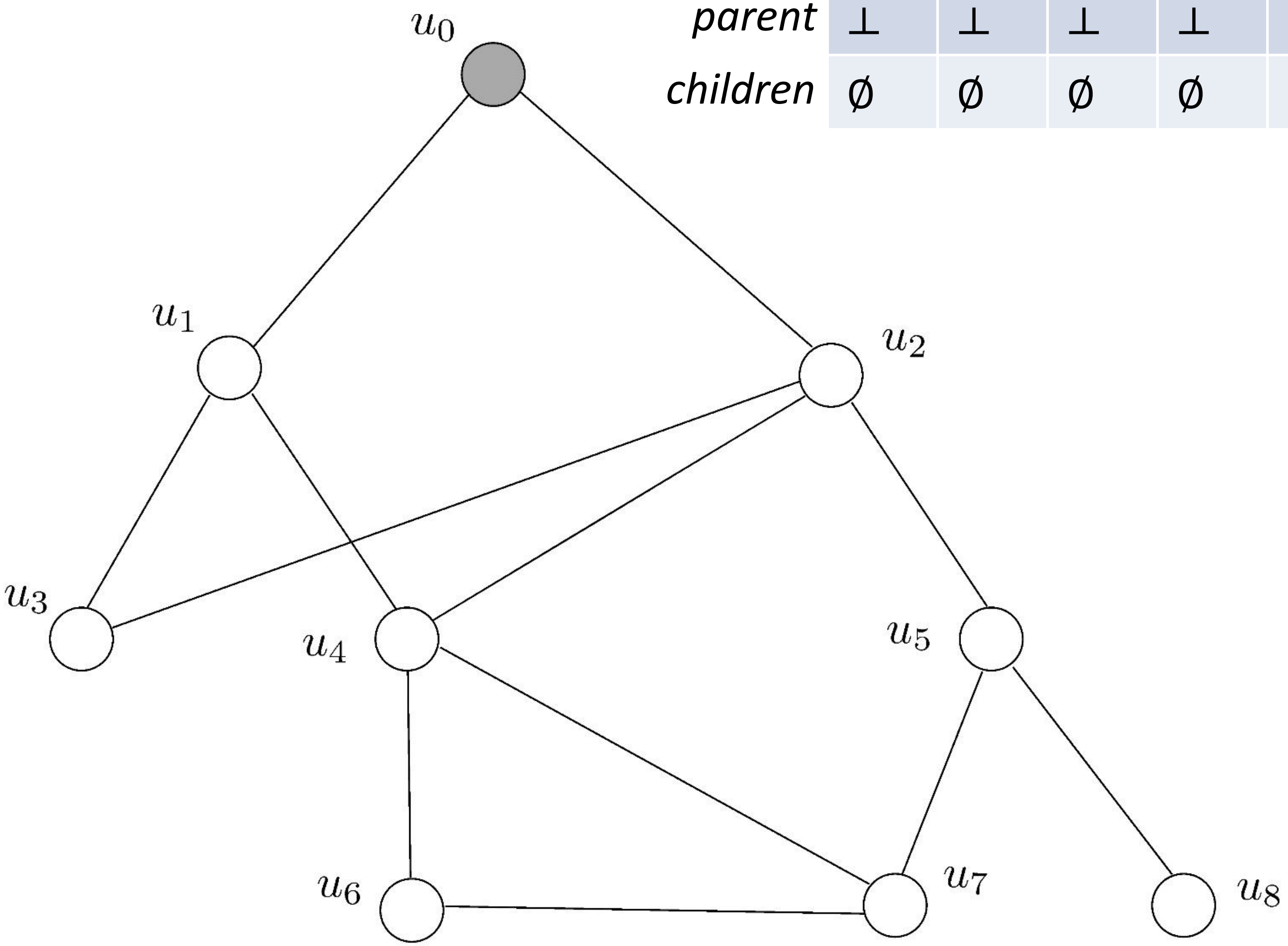
Initially *parent* = $\perp$ and *children* = $\emptyset$

if $u_i = u_0$ and *parent* = $\perp$ then // root has not yet sent $\langle$M$\rangle$
  send $\langle$M$\rangle$ to all neighbours
  parent := u_i

upon receiving $\langle$M$\rangle$ from neighbours N:
  if *parent* = $\perp$ then // u_i has not received $\langle$M$\rangle$ before
    parent := $u_j \in N$ // select one arbitrarily as parent
    send $\langle$"parent"$\rangle$ to u_j
    send $\langle$M$\rangle$ to all neighbours except those in N
    wait for one round to collect children if any and then terminate

upon receiving $\langle$"parent"$\rangle$ from neighbours N:
  add all $u_j \in N$ to *children*
  terminate

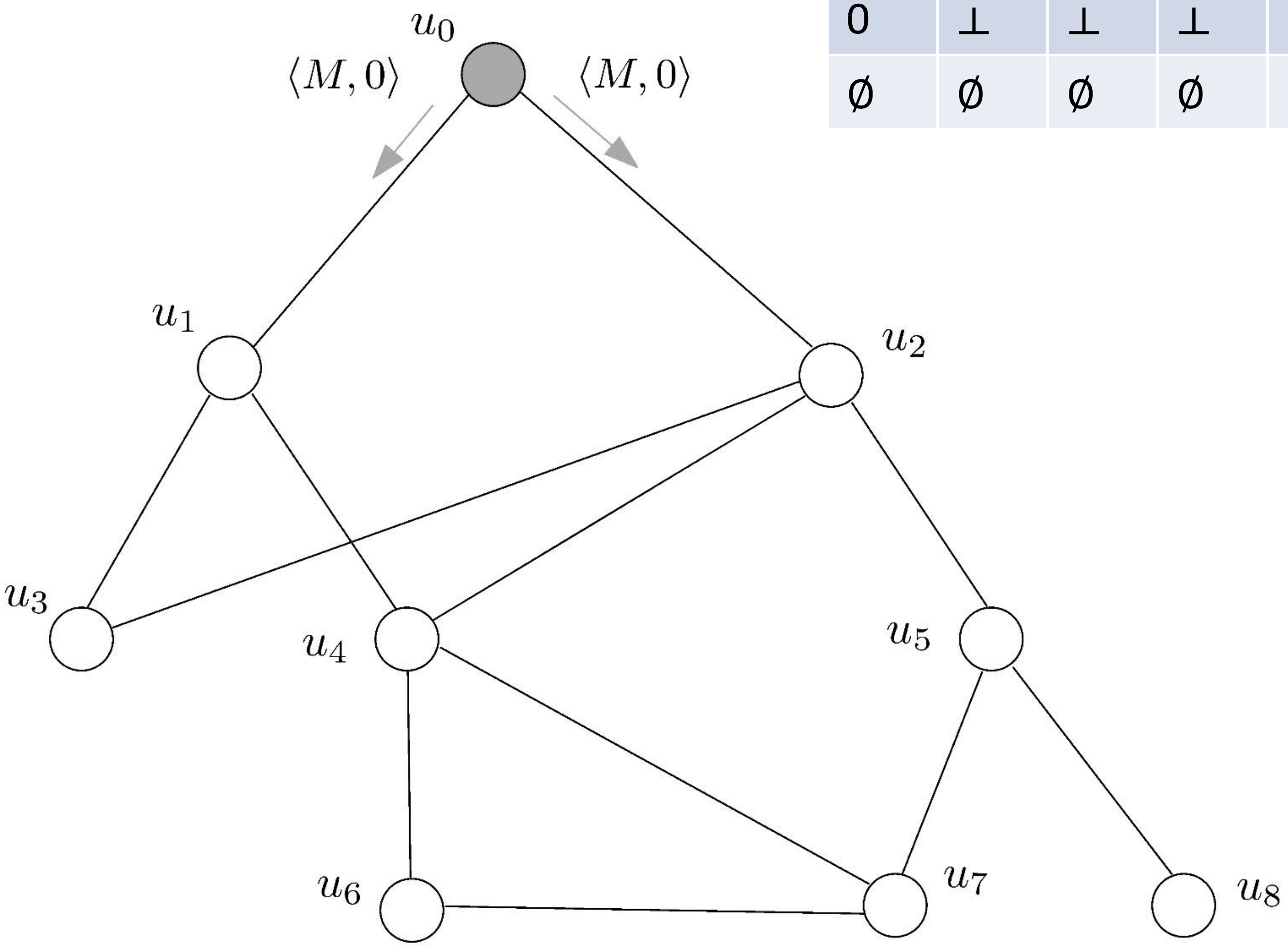
Example Execution



	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
children	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Example Execution

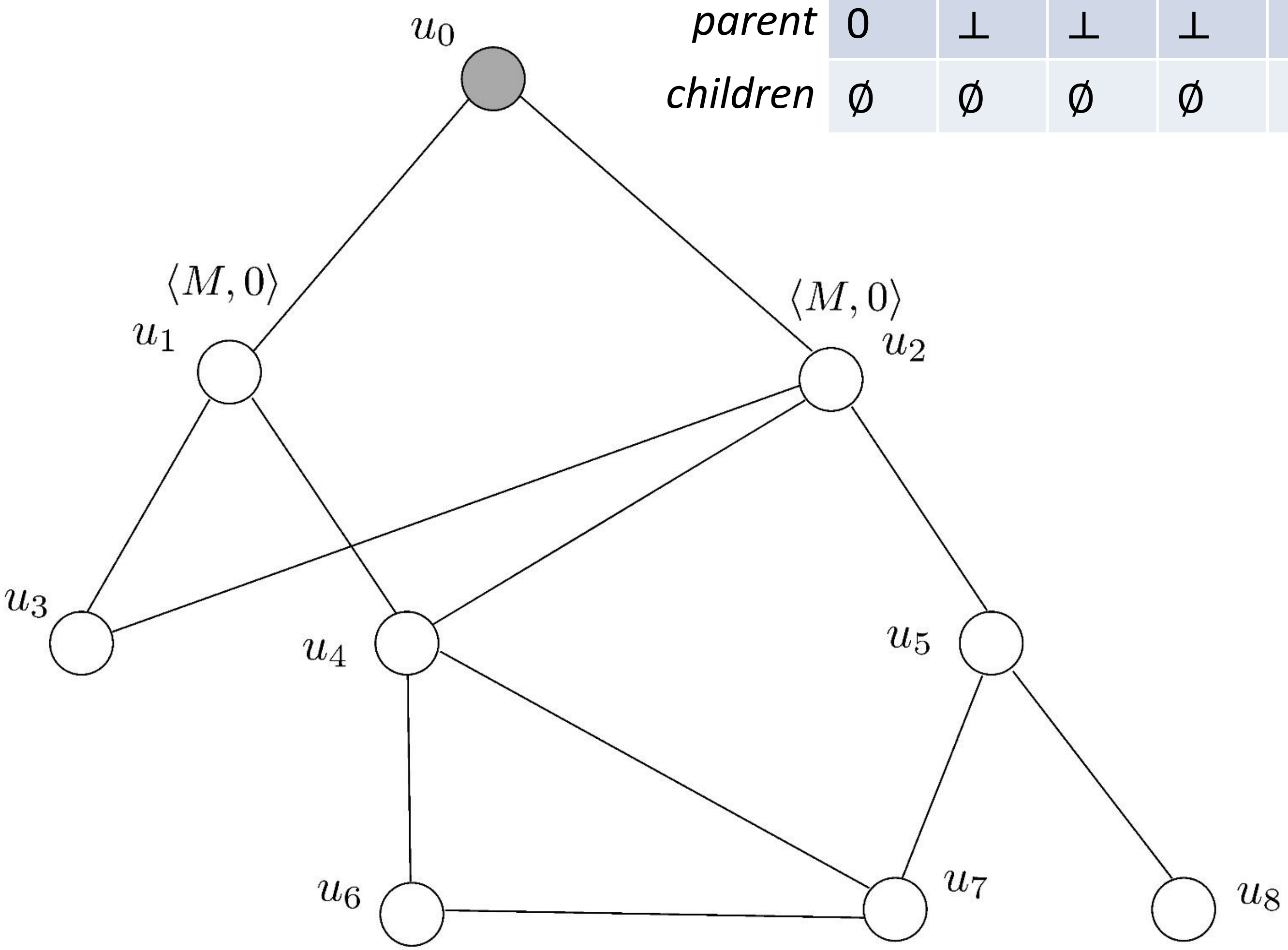
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
0	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø



$round = 1$

Example Execution

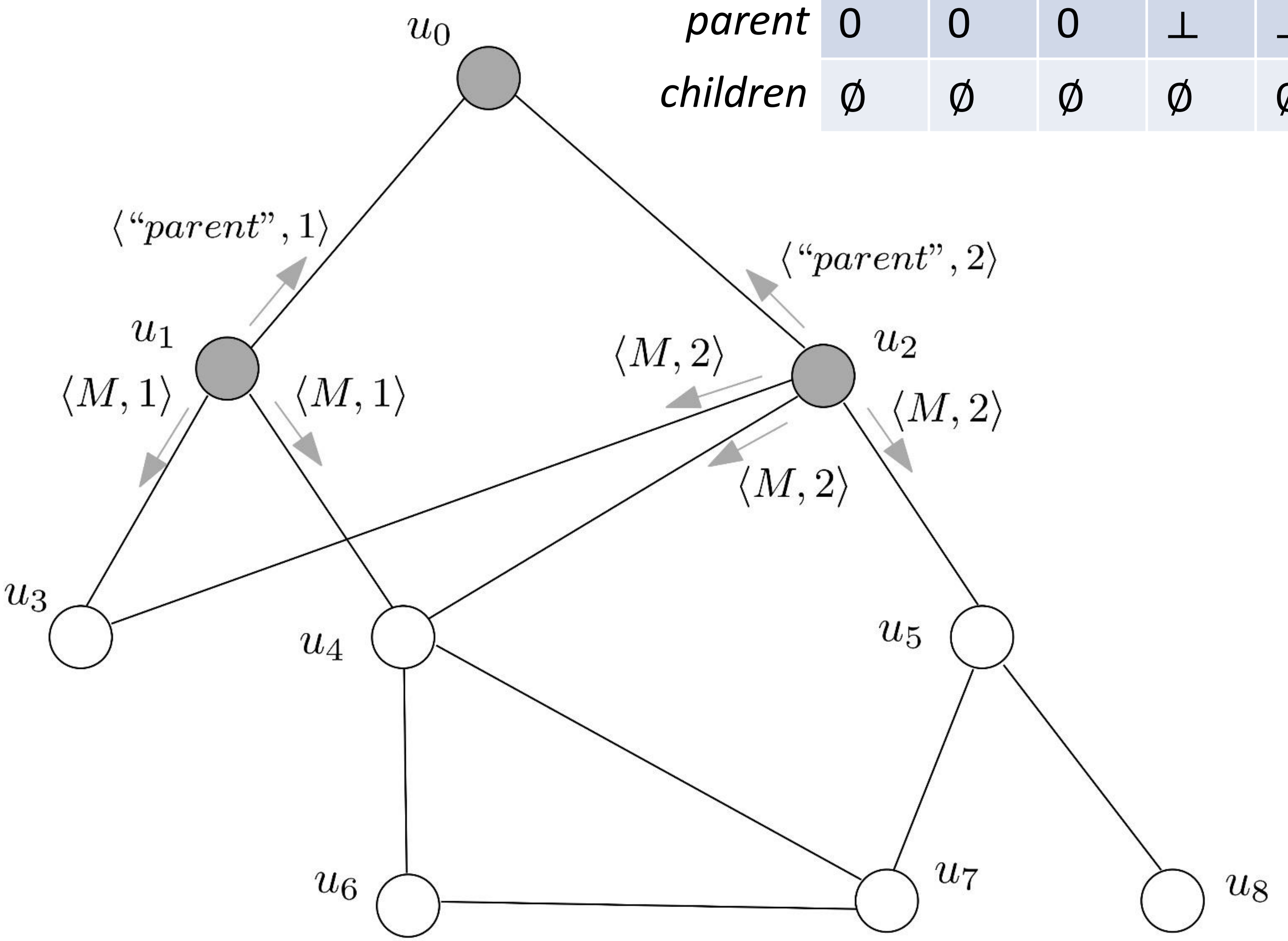
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
children	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$



$round = 1$

Example Execution

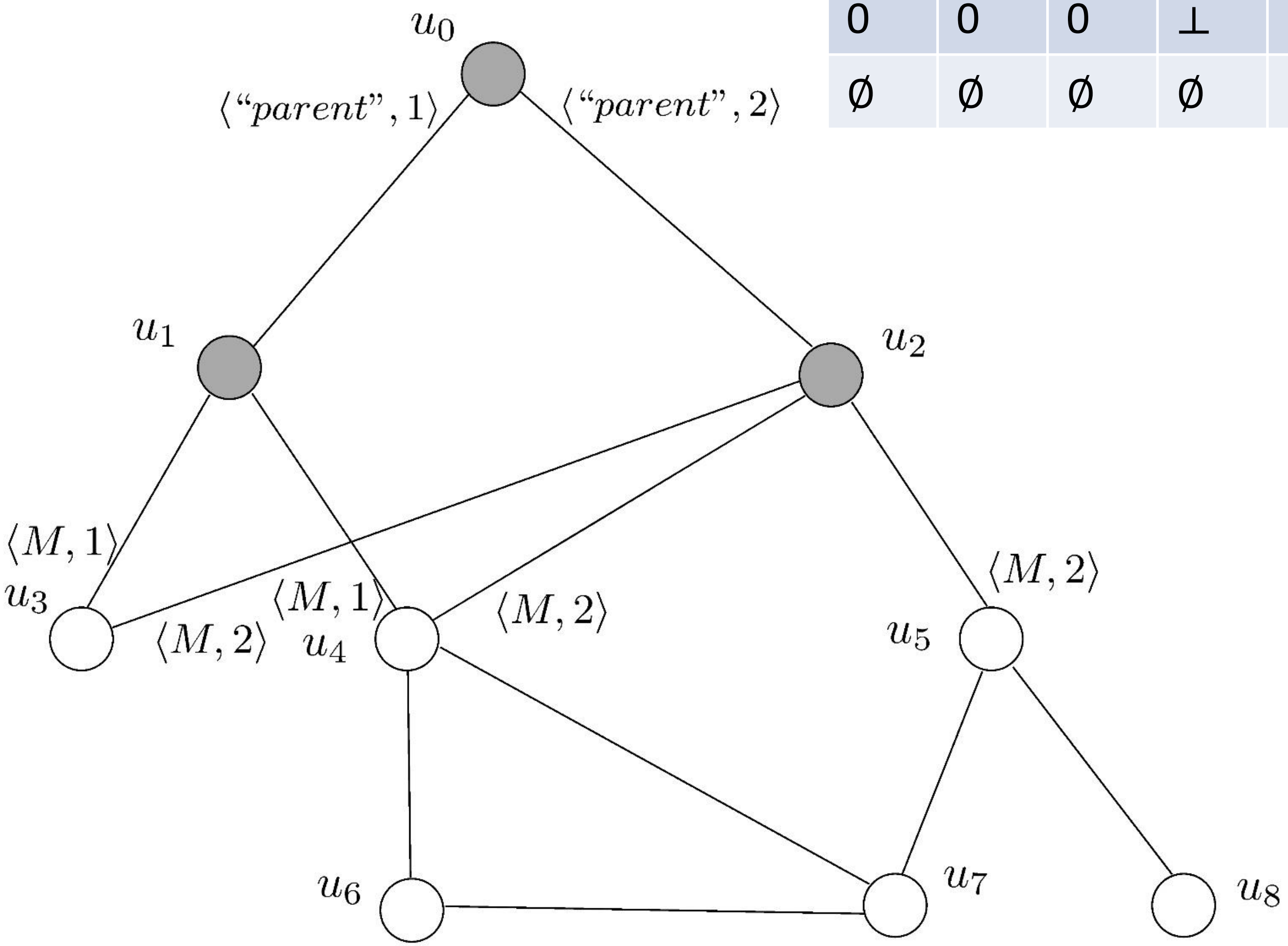
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	⊥	⊥	⊥	⊥	⊥	⊥
children	∅	∅	∅	∅	∅	∅	∅	∅	∅



$round = 2$

Example Execution

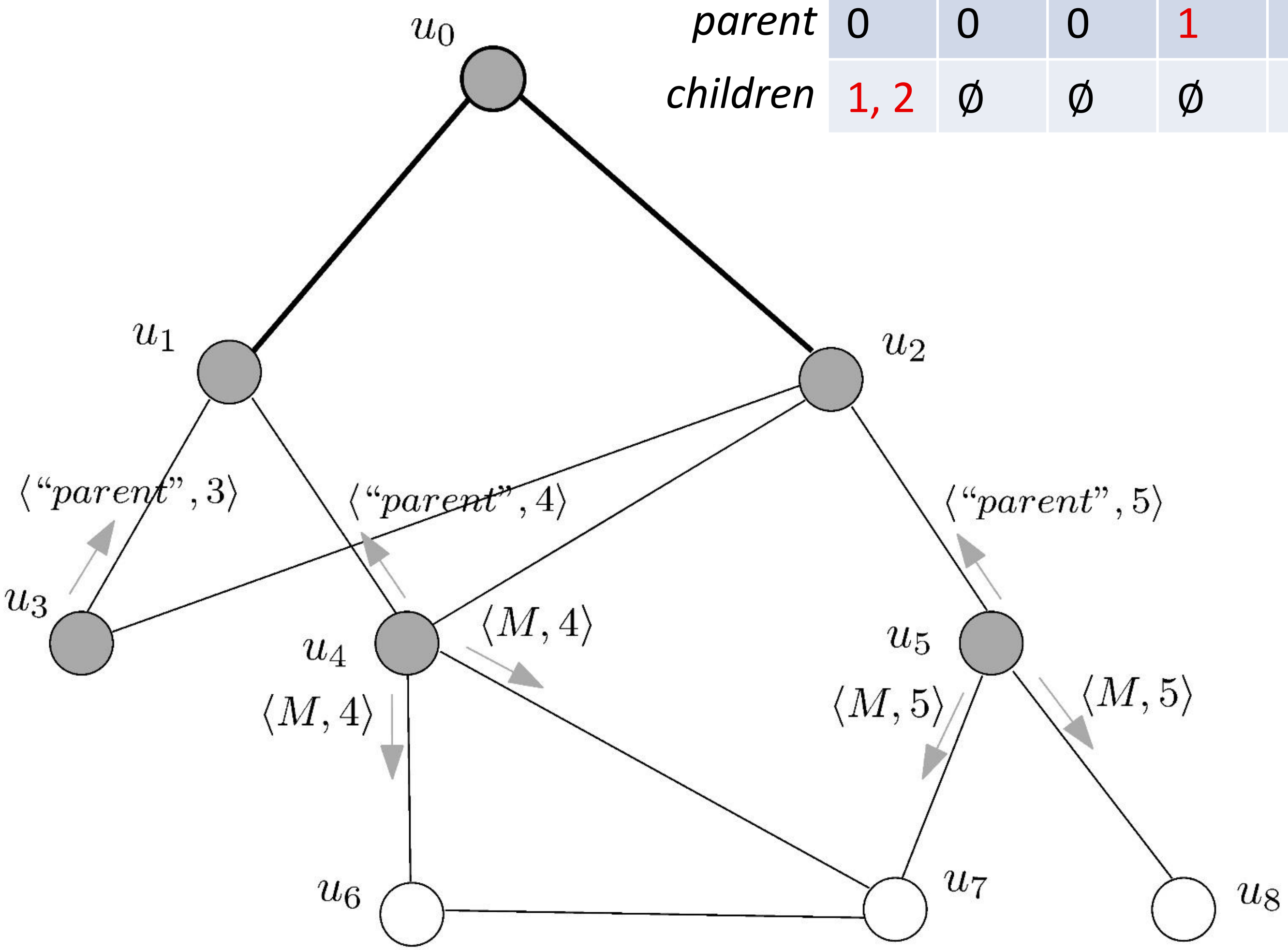
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
0	0	0	⊥	⊥	⊥	⊥	⊥	⊥
∅	∅	∅	∅	∅	∅	∅	∅	∅



$round = 2$

Example Execution

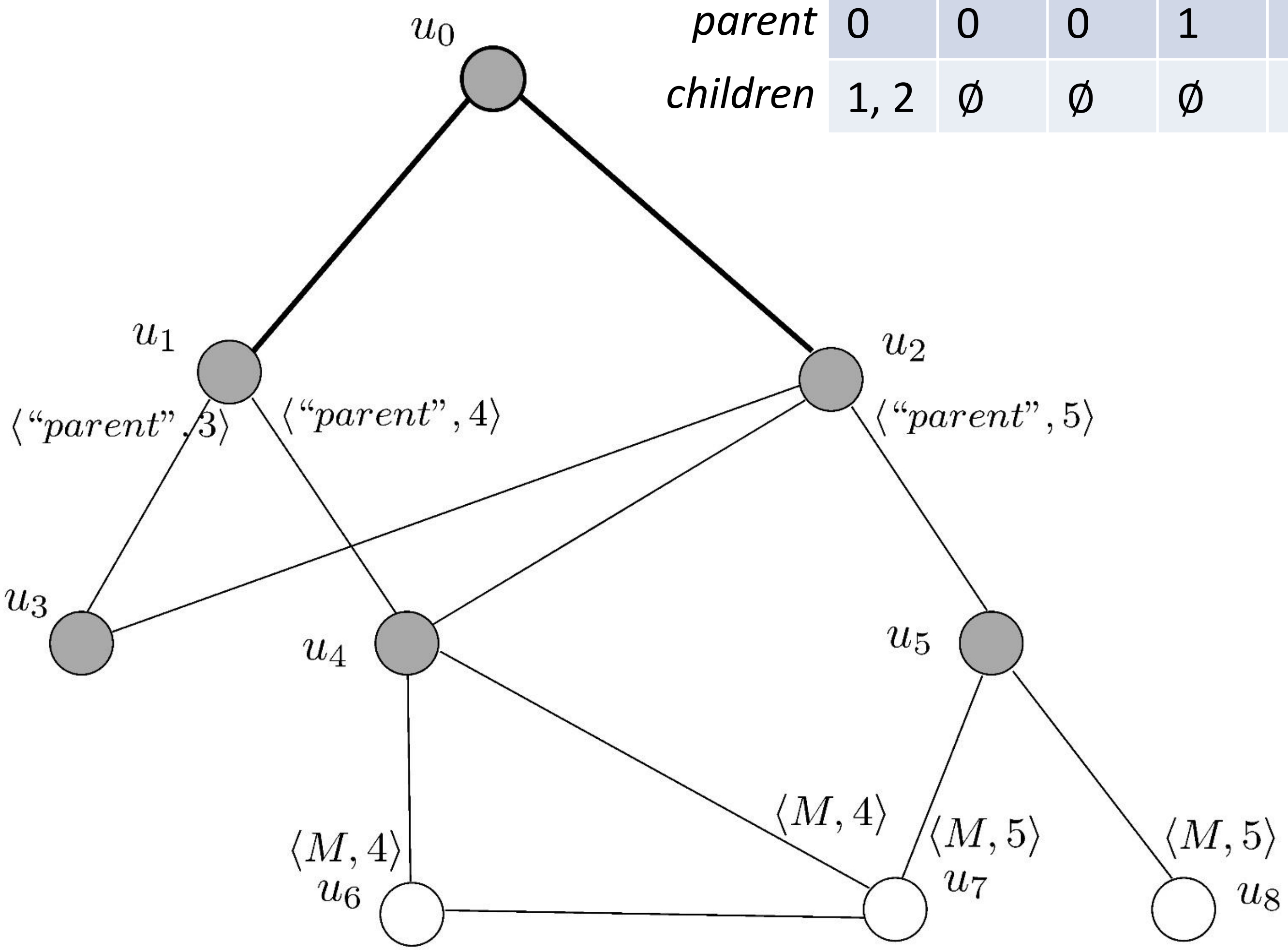
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	1	1	2	$\perp$	$\perp$	$\perp$
children	1, 2	∅	∅	∅	∅	∅	∅	∅	∅



$round = 3$

Example Execution

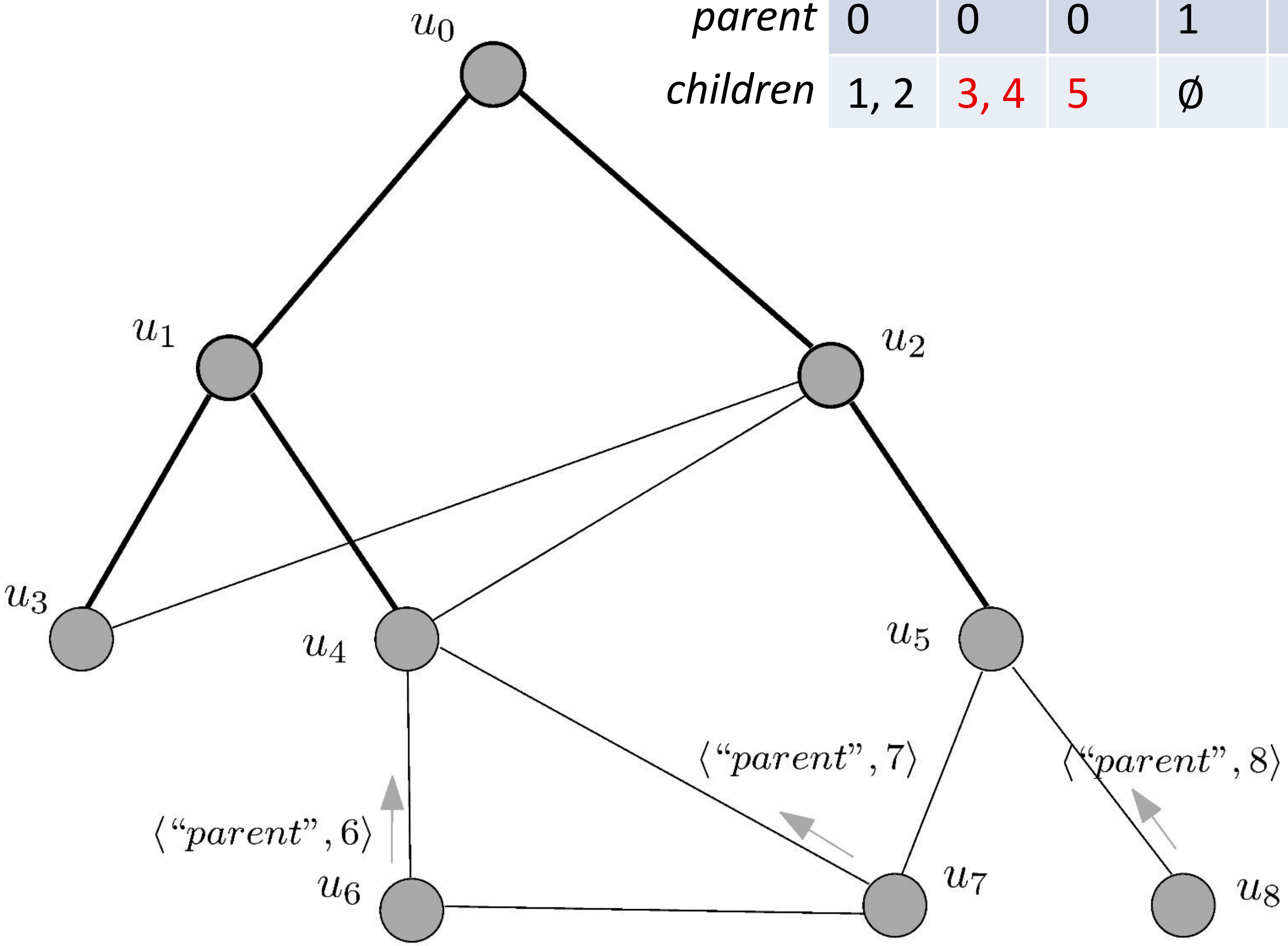
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	1	1	2	⊥	⊥	⊥
children	1, 2	∅	∅	∅	∅	∅	∅	∅	∅



$round = 3$

Example Execution

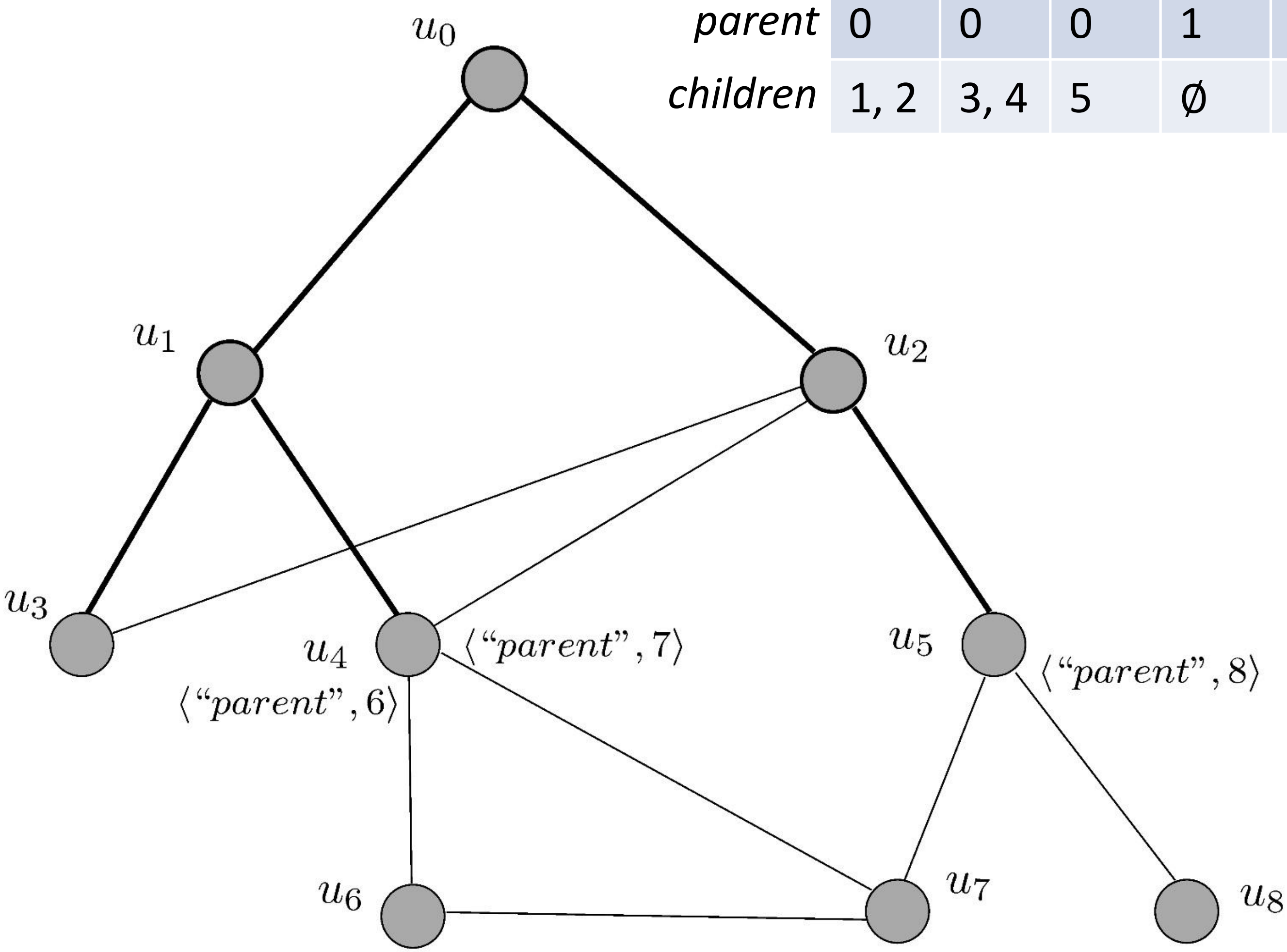
	u_0	u_1	u_2	u_3	u_4	u_5	U_6	u_7	u_8
parent	0	0	0	1	1	2	4	4	5
children	1, 2	3, 4	5	Ø	Ø	Ø	Ø	Ø	Ø



$round = 4$

Example Execution

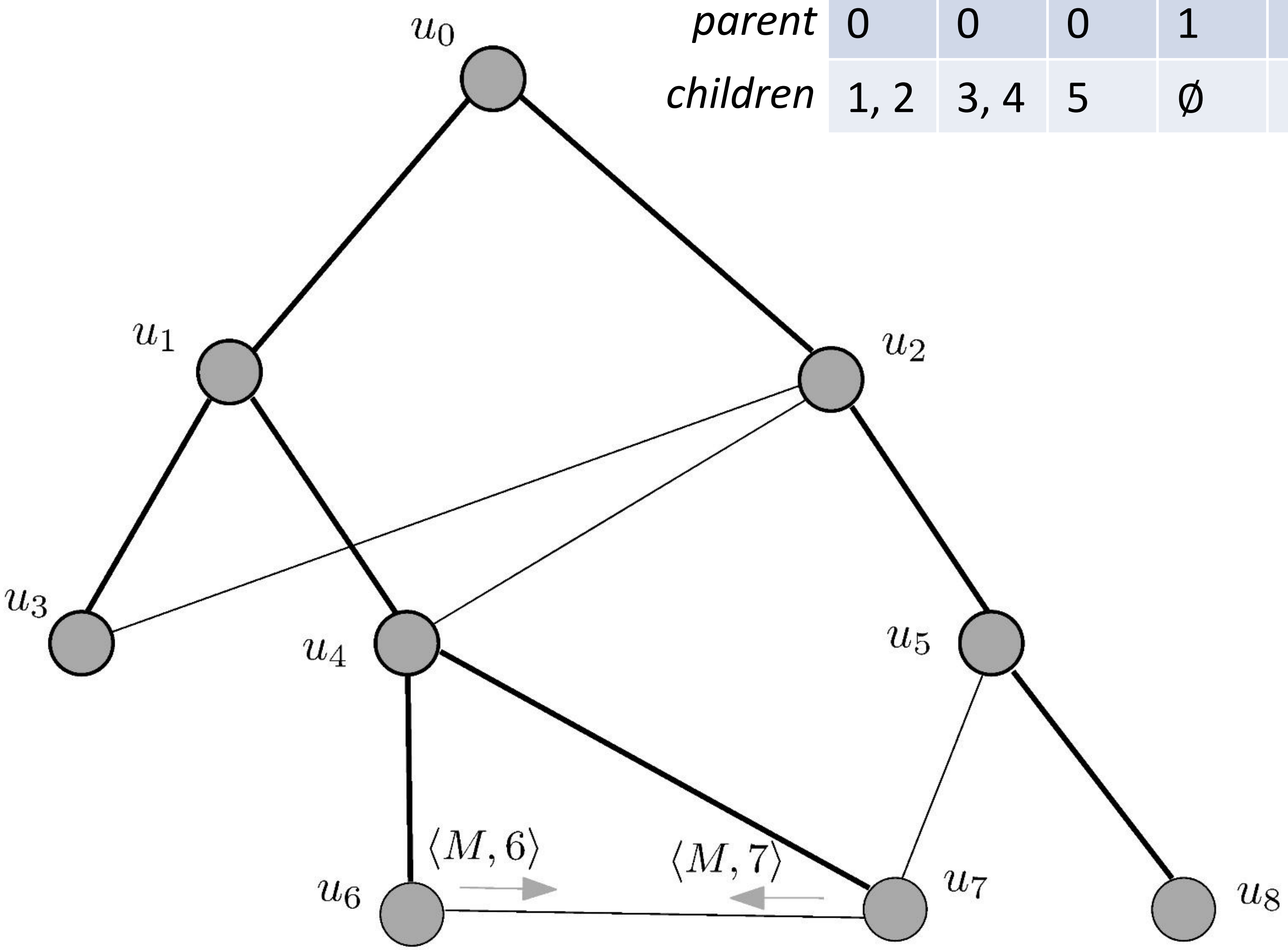
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	1	1	2	4	4	5
children	1, 2	3, 4	5	Ø	Ø	Ø	Ø	Ø	Ø



$round = 4$

Example Execution

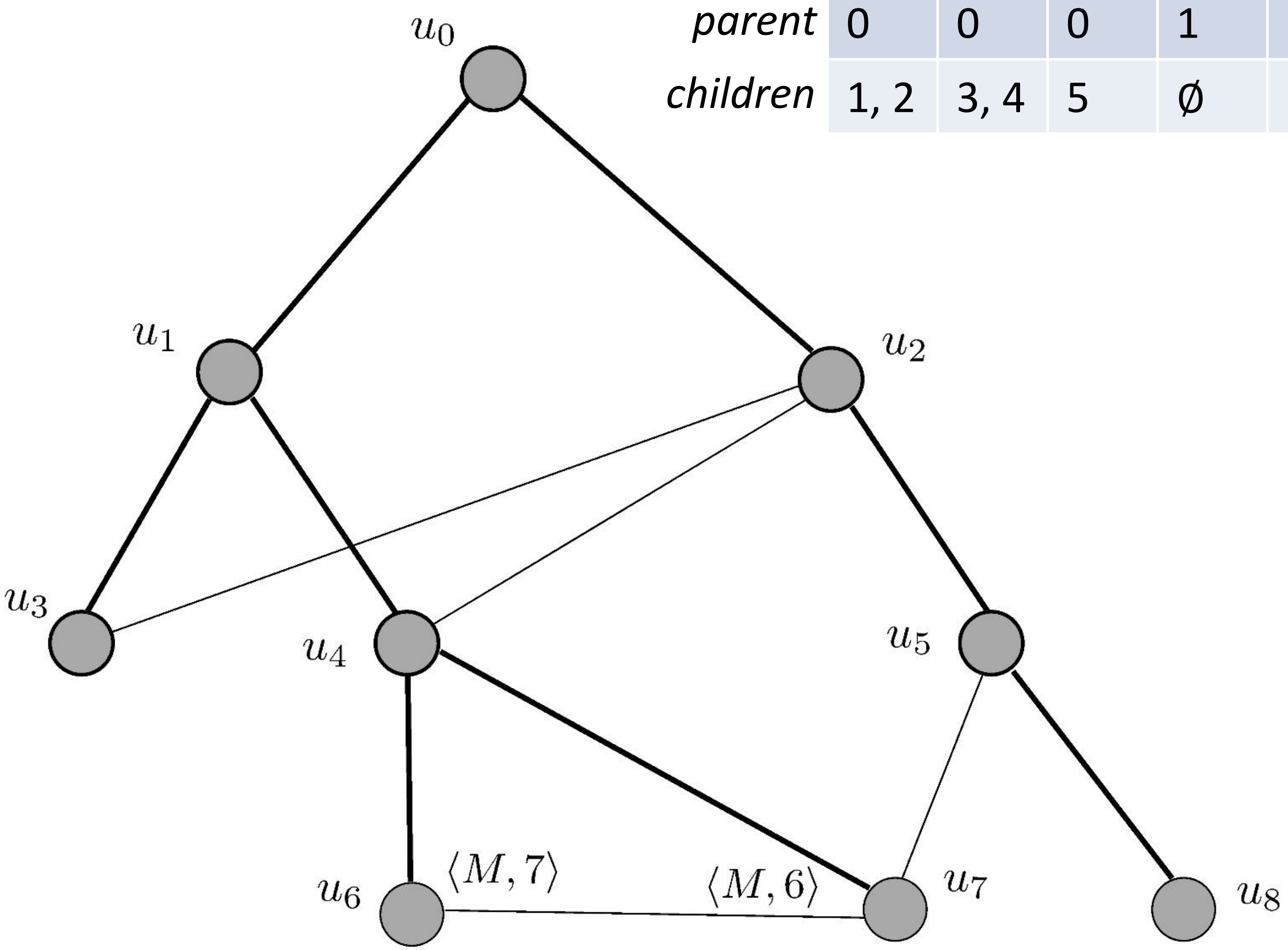
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	1	1	2	4	4	5
children	1, 2	3, 4	5	∅	6, 7	8	∅	∅	∅



$round = 5$

Example Execution

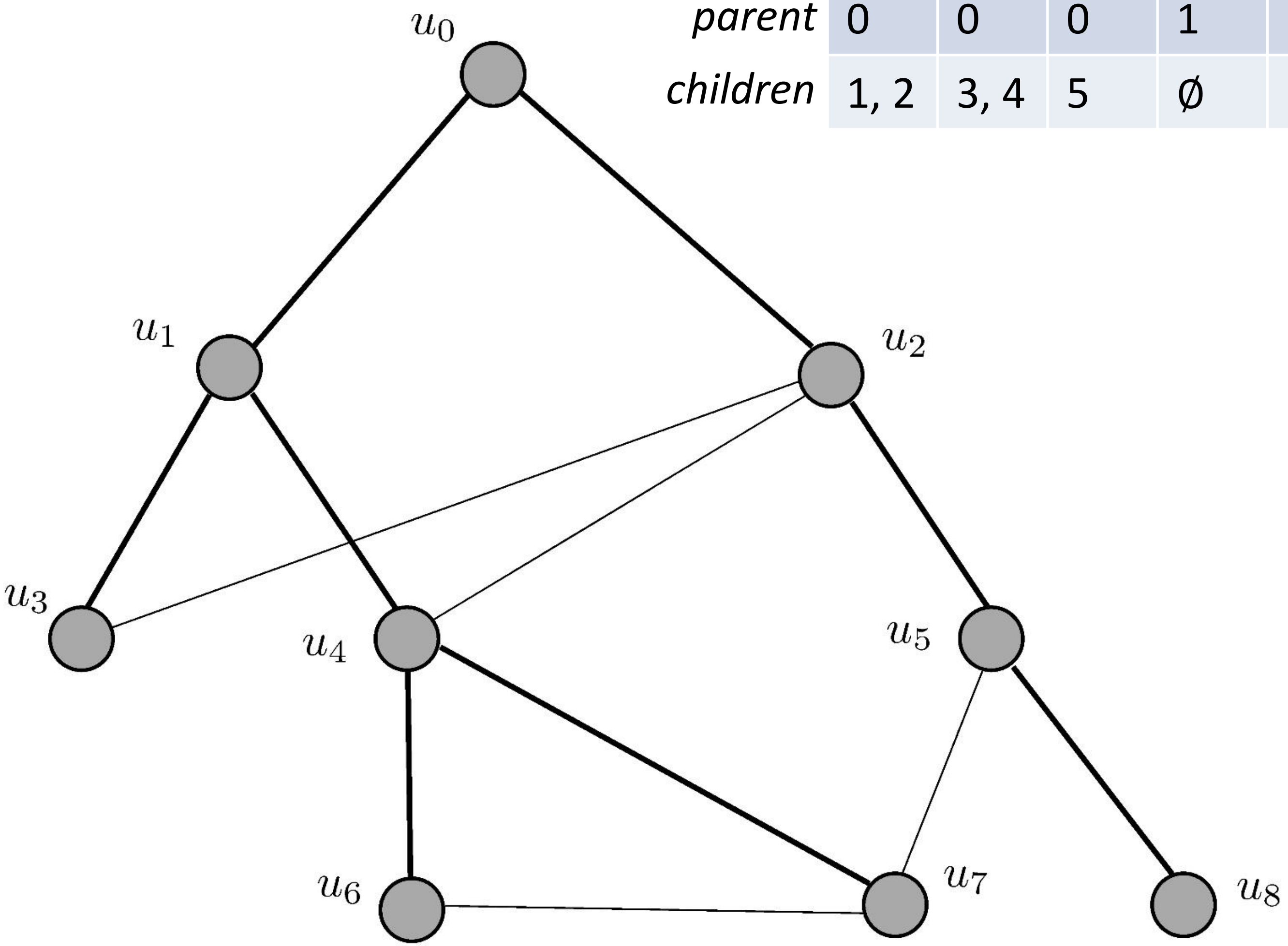
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	1	1	2	4	4	5
children	1, 2	3, 4	5	∅	6, 7	8	∅	∅	∅



$round = 5$

Example Execution

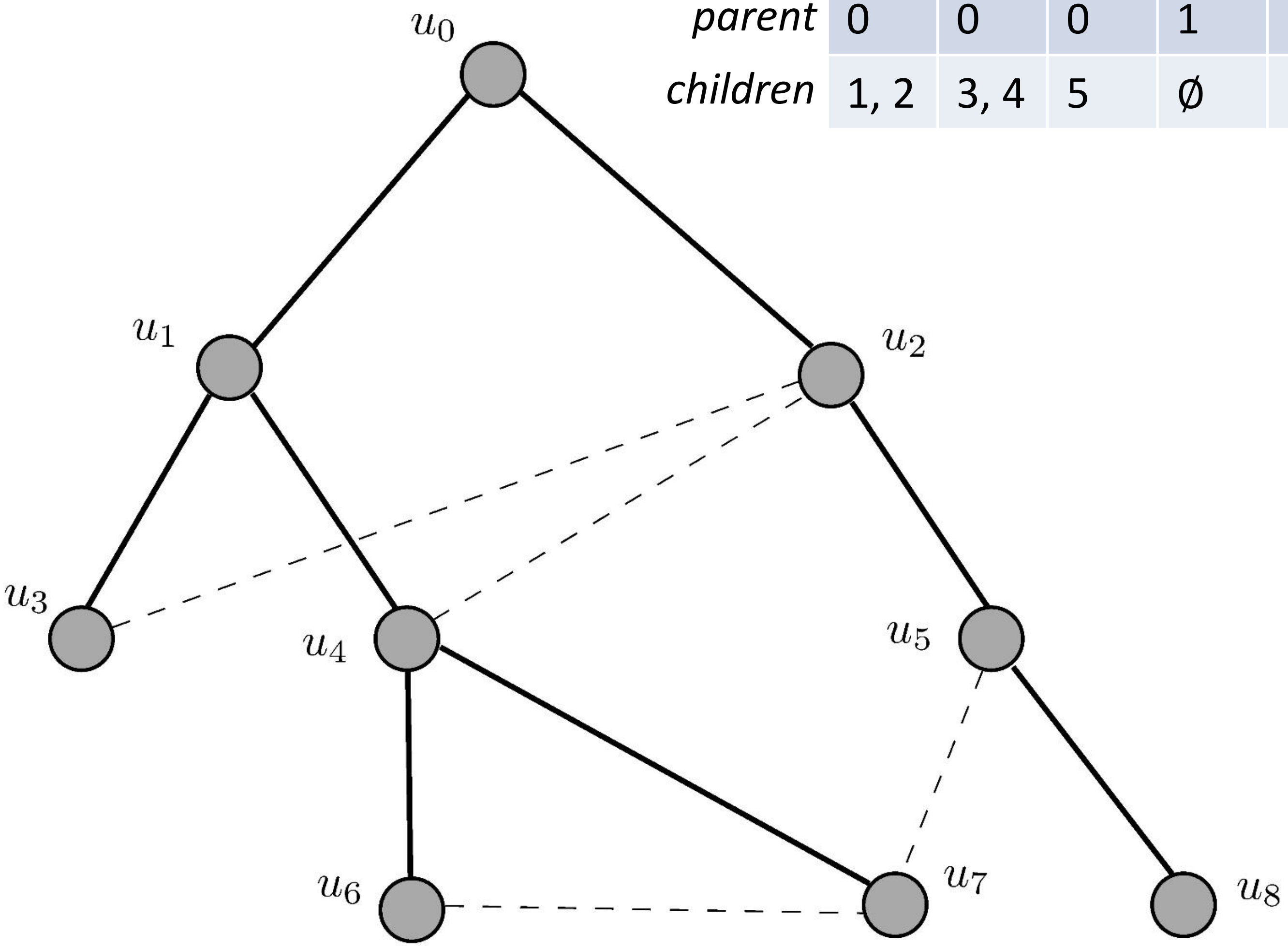
	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	1	1	2	4	4	5
children	1, 2	3, 4	5	∅	6, 7	8	∅	∅	∅



$round = 6$

Example Execution

	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
parent	0	0	0	1	1	2	4	4	5
children	1, 2	3, 4	5	∅	6, 7	8	∅	∅	∅



Correctness and Complexity

- Correctness:
  - correctness of broadcast
  - correctness of spanning tree construction
    - can also be shown that the constructed tree is always a Breadth-first search (BFS) tree
- Time complexity:
  - $O(D)$: where D is the maximum distance of a u_i form u_0 in G
- Communication complexity:
  - size of messages: sends message ⟨M⟩ and an id
  - $O(m)$ messages: where m denotes the #edges of G

Can you prove these at home?