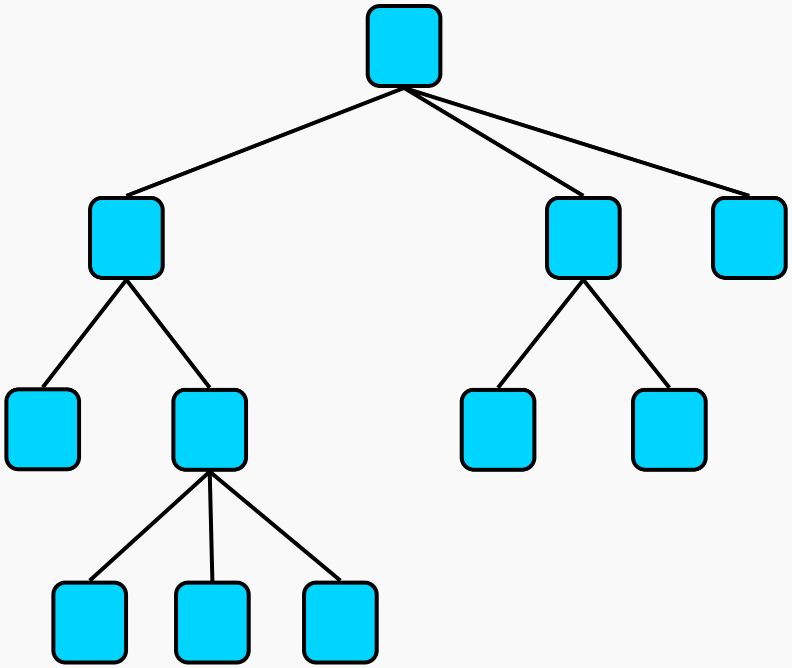
# Computations in Trees

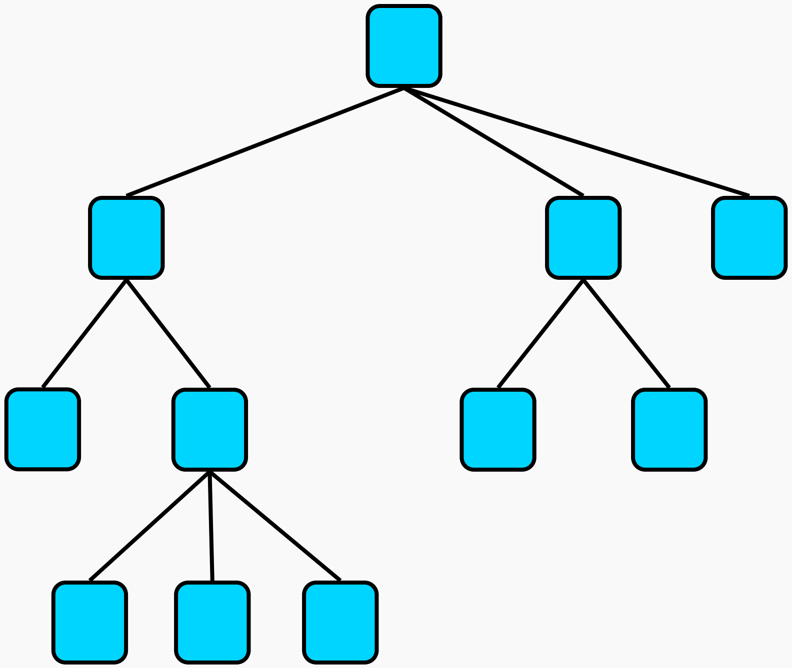
Saturation
Minimum Finding
Eccentricity
Center
Ranking



Paola Flocchini

# Trees

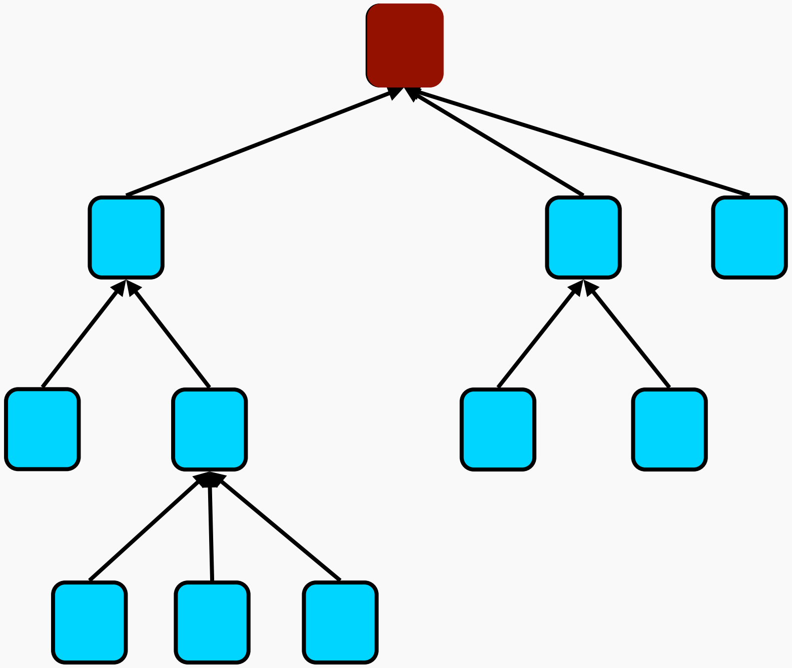
- Acyclic graph
- n entities
- n - 1 links



# Rooted Trees

- Acyclic graph
- n entities
- n - 1 links

Rooted

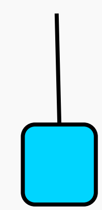


# Saturation Technique

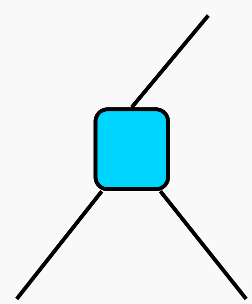
- Bidirectional links
- Ordered messages
- Full Reliability
- Knowledge of the topology

Note:

Each entity knows whether
it is a leaf:



or an internal node:



# SATURATION: A Basic Technique

S = {available, awake, processing }

At the beginning, all entities are available

Arbitrary entities can start the computation (multiple initiators)

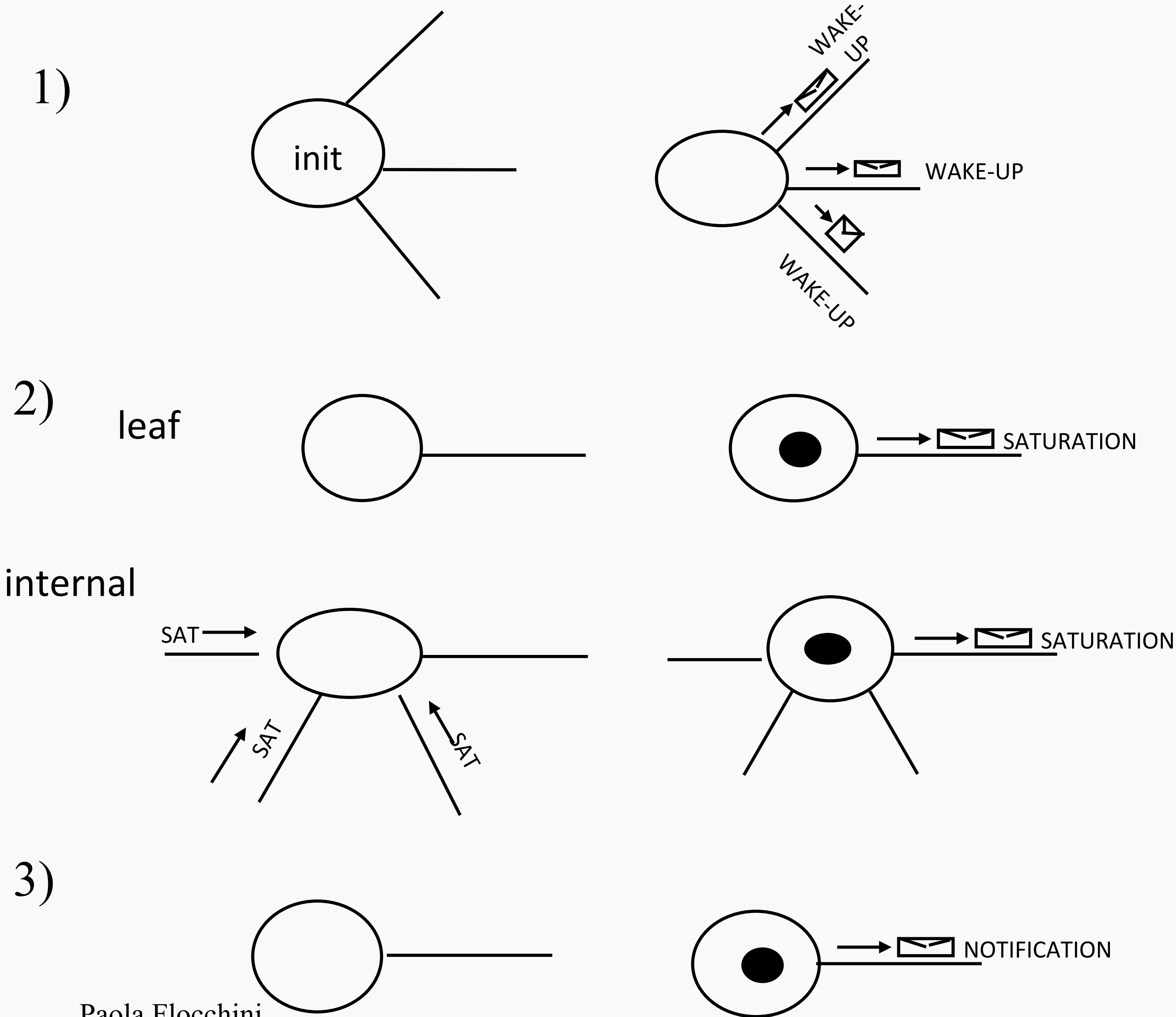
# SATURATION: A General Technique

- **Activation phase:**
  started by the initiators: all nodes are activated

wake-up

- **Saturation Phase:**
  started by the leaves: a unique pair of neighbours is identified (saturated nodes)

- **Resolution Phase:**
  started by the saturated nodes




Paola Flocchini

Saturated node

S = {AVAILABLE, ACTIVE, PROCESSING, SATURATED}
Sinit = AVAILABLE

**AVAILABLE** **I haven't been activated yet**

*Spontaneously*

```
send(Activate) to N(x);
Neighbours:= N(x)
if |Neighbours|=1 then
        M:=("Saturation");                /* special case if
        parent ⇐ Neighbours;              I am a leaf */
        send(M) to parent;
        become PROCESSING;
else
        become ACTIVE;
```

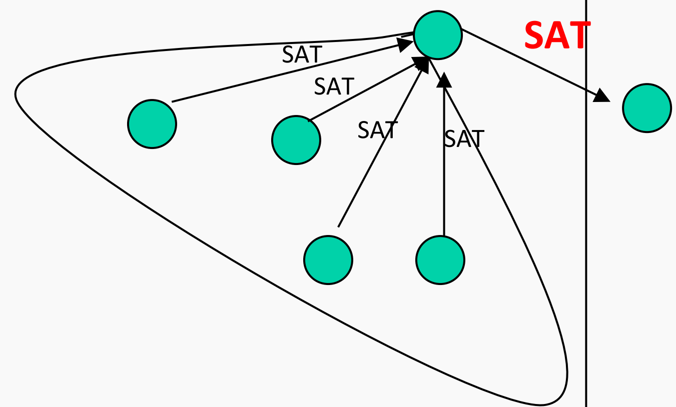
```
Receiving(Activate)
        send(Activate) to N(x)– {sender};
        Neighbours:=  N(x);
        if |Neighbours|=1   then
                M:=("Saturation");
                parent ⇐ Neighbours;
                send(M)   to  parent;
                become PROCESSING;
        else
                become ACTIVE;
```

I am awake but I haven't started sending Saturation messages yet

**ACTIVE**

*Receiving(M)*

Neighbours:= Neighbours - {sender};
if |Neighbours|=1 then
M:=("Saturation");
parent ⇐ Neighbours;
**send**(M) **to parent;**
**become** PROCESSING;

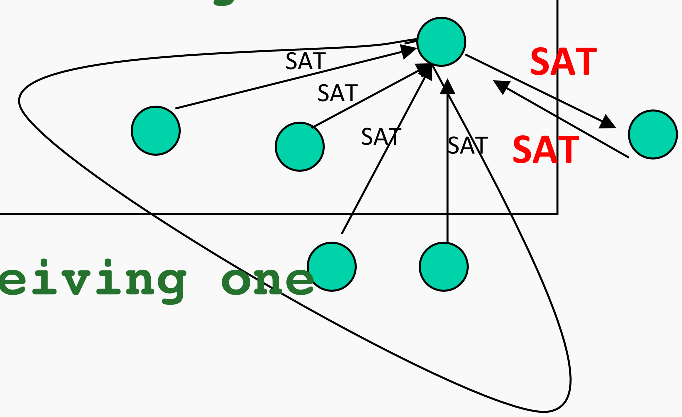



**PROCESSING**

I have already sent my saturation message

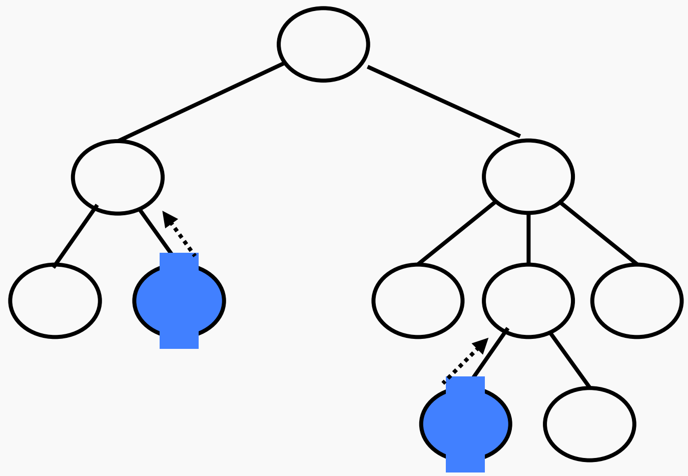
*receiving(M)*

**become** SATURATED;

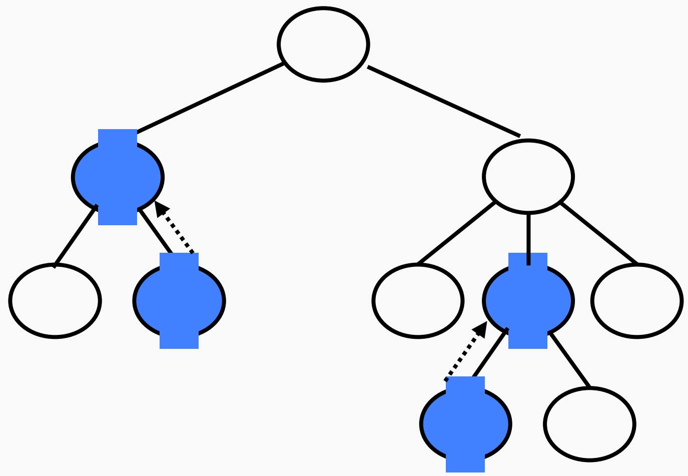



And now I am receiving one

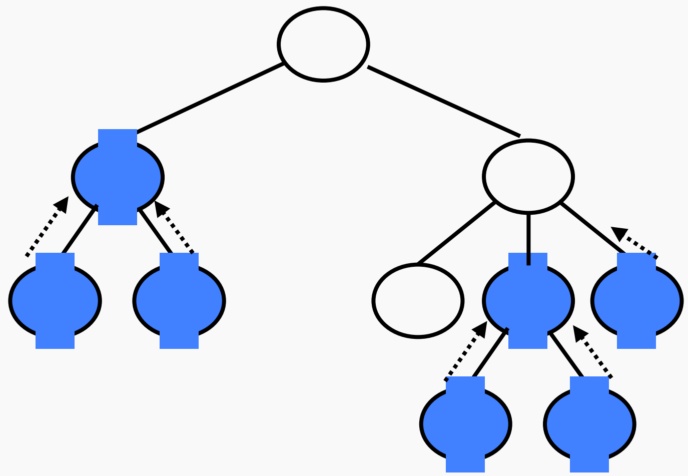
# Example of Saturation Phase (started by some leaves)



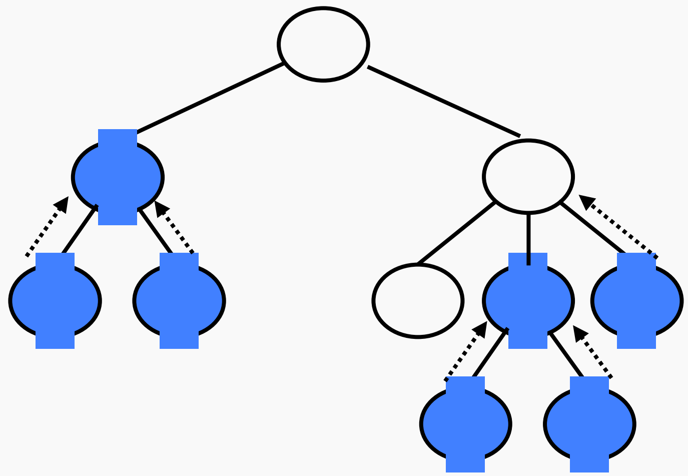
## Example of Saturation



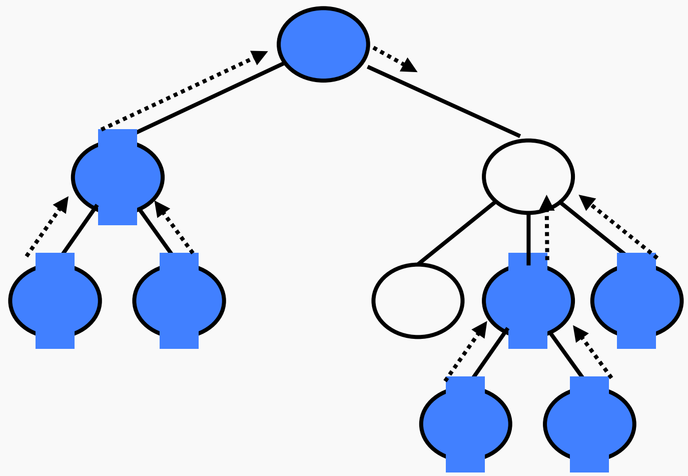
## Example of Saturation



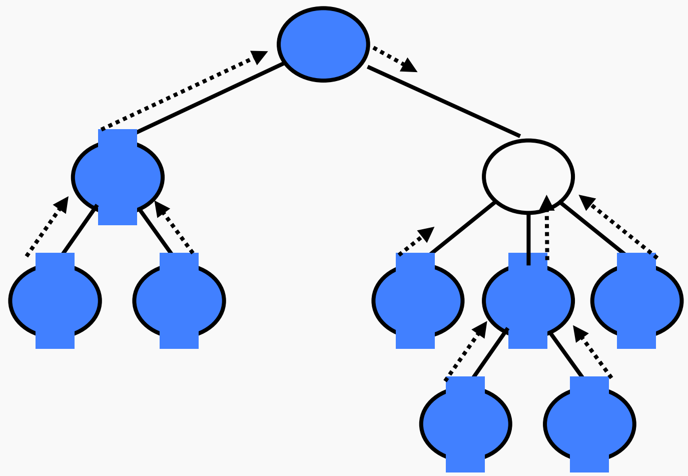
## Example of Saturation



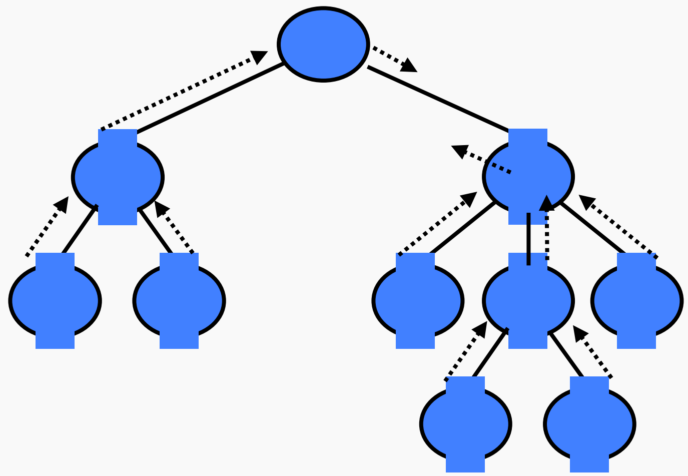
# Example of Saturation



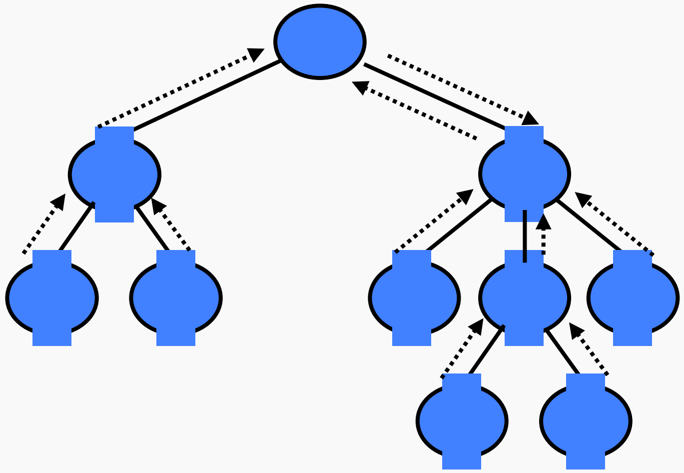
## Example of Saturation



# Example of Saturation

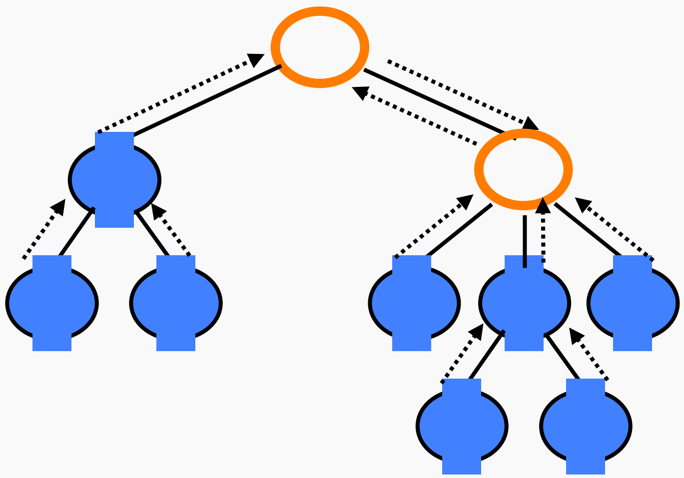


## Example of Saturation

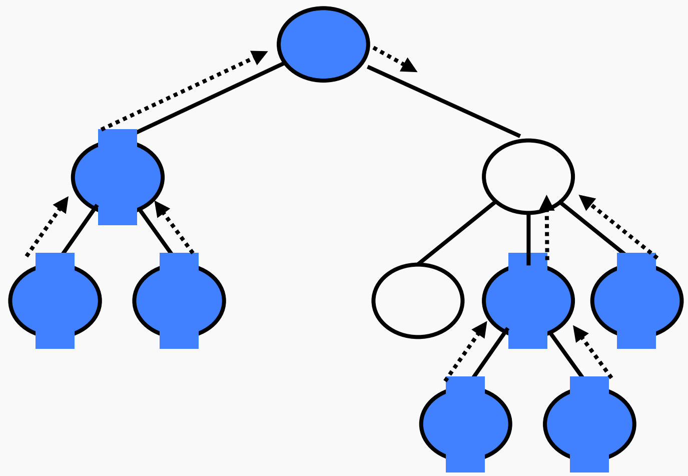


Paola Flocchini

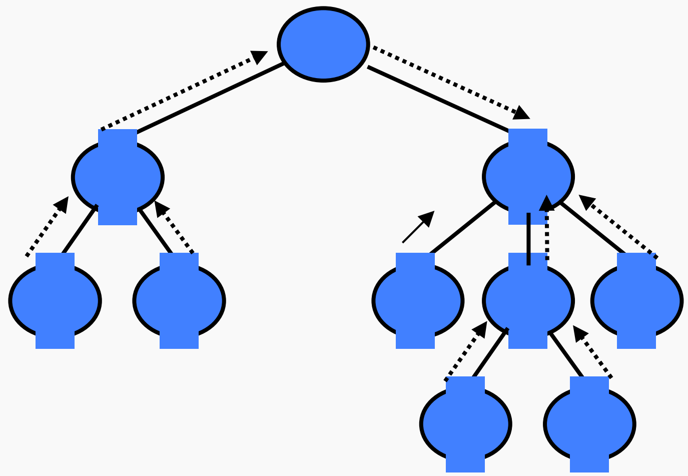
## Example of Saturation



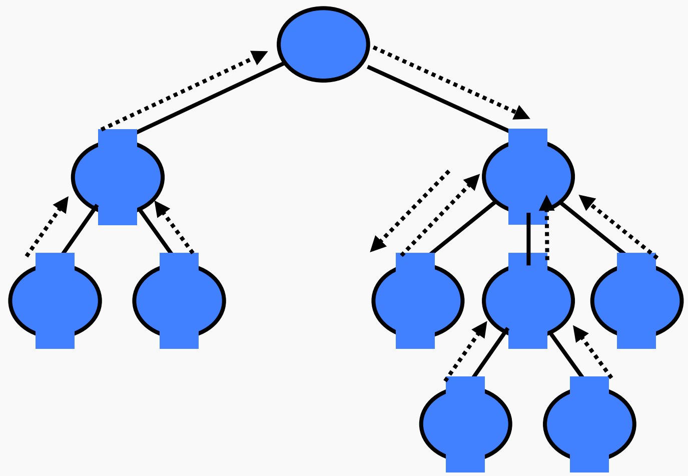
# Example of Saturation - different ending



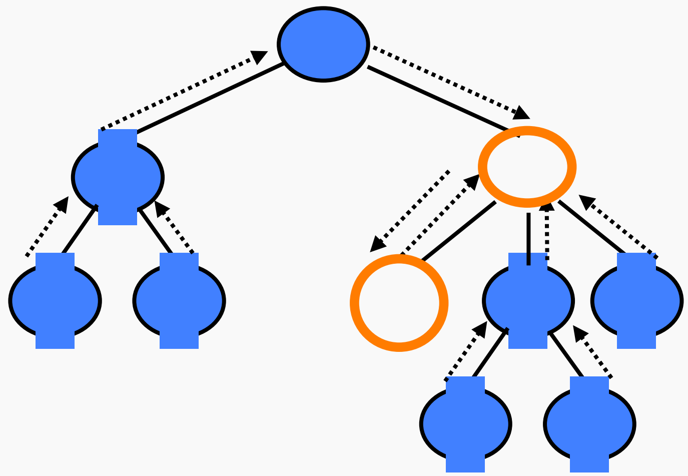
# Example of Saturation - different ending



# Example of Saturation - different ending

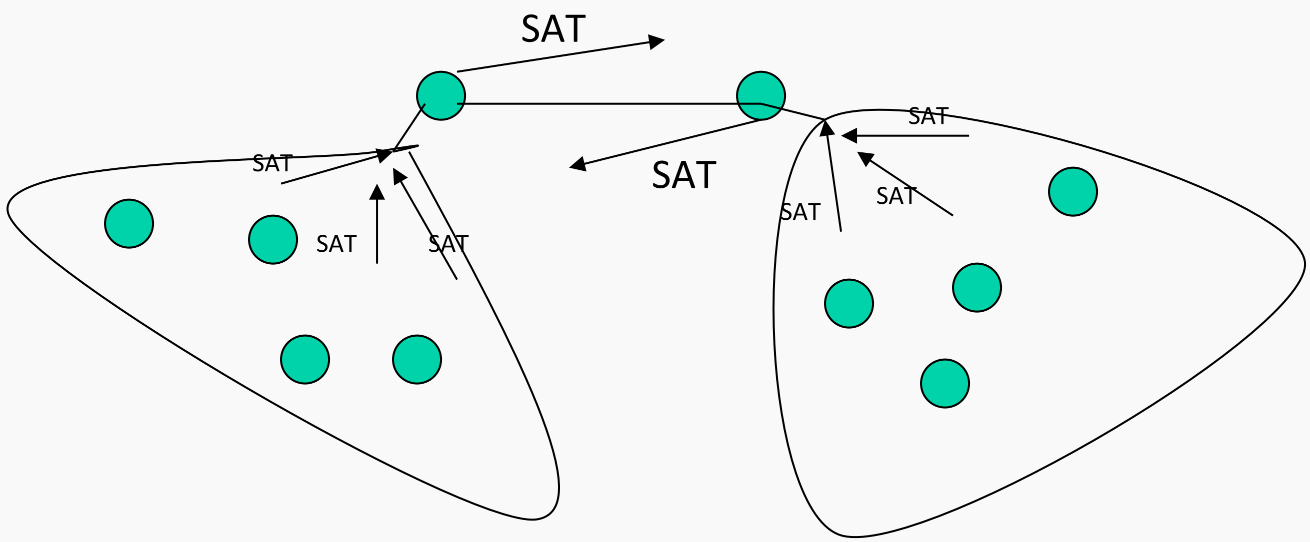


# Example of Saturation - different ending

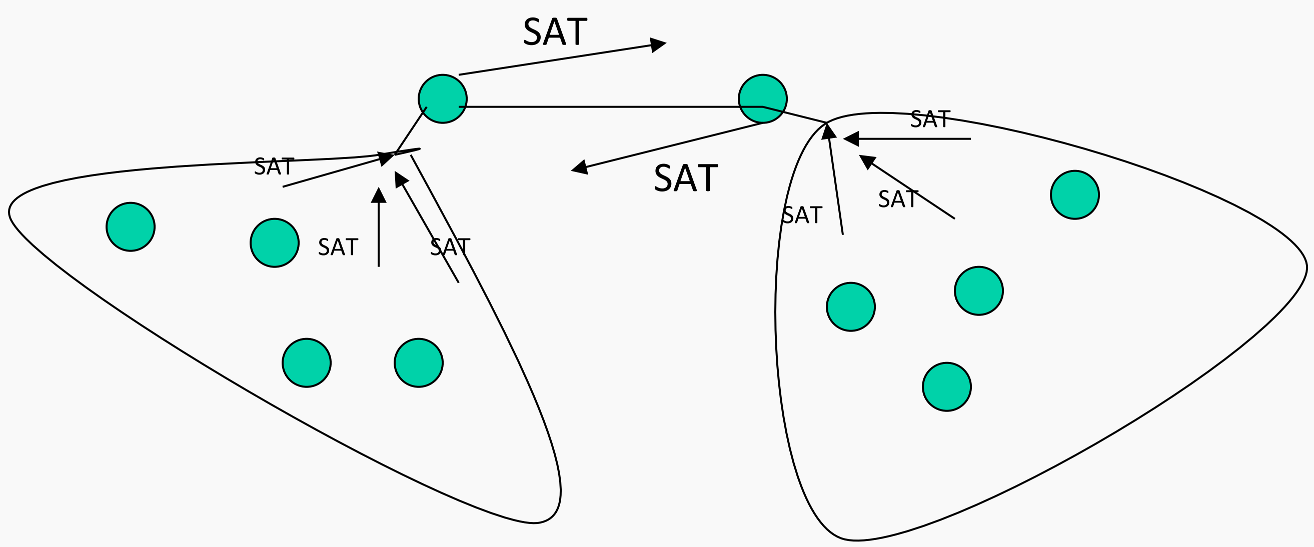


Property:

Exactly two processing nodes become saturated, and they are neighbours.



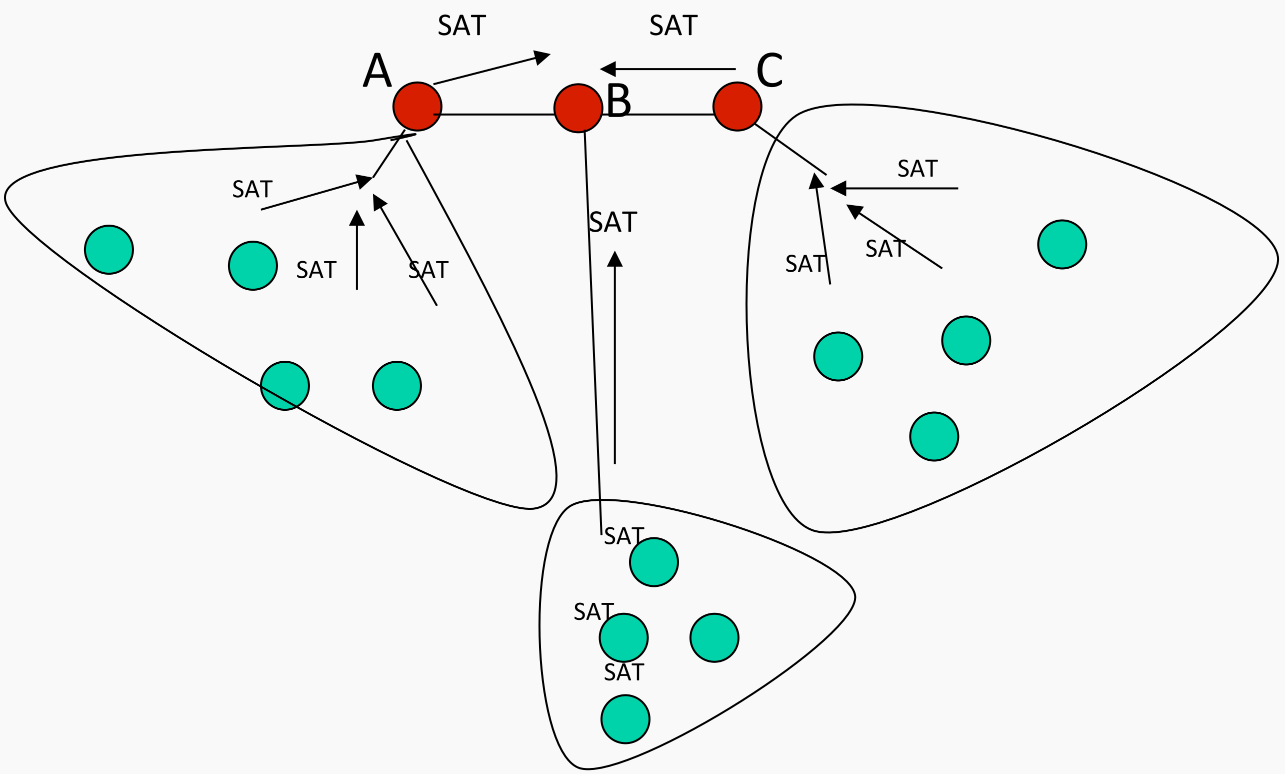
A node becomes PROCESSING only after sending saturation to its parent. Each node sends **only ONE Saturation** message.

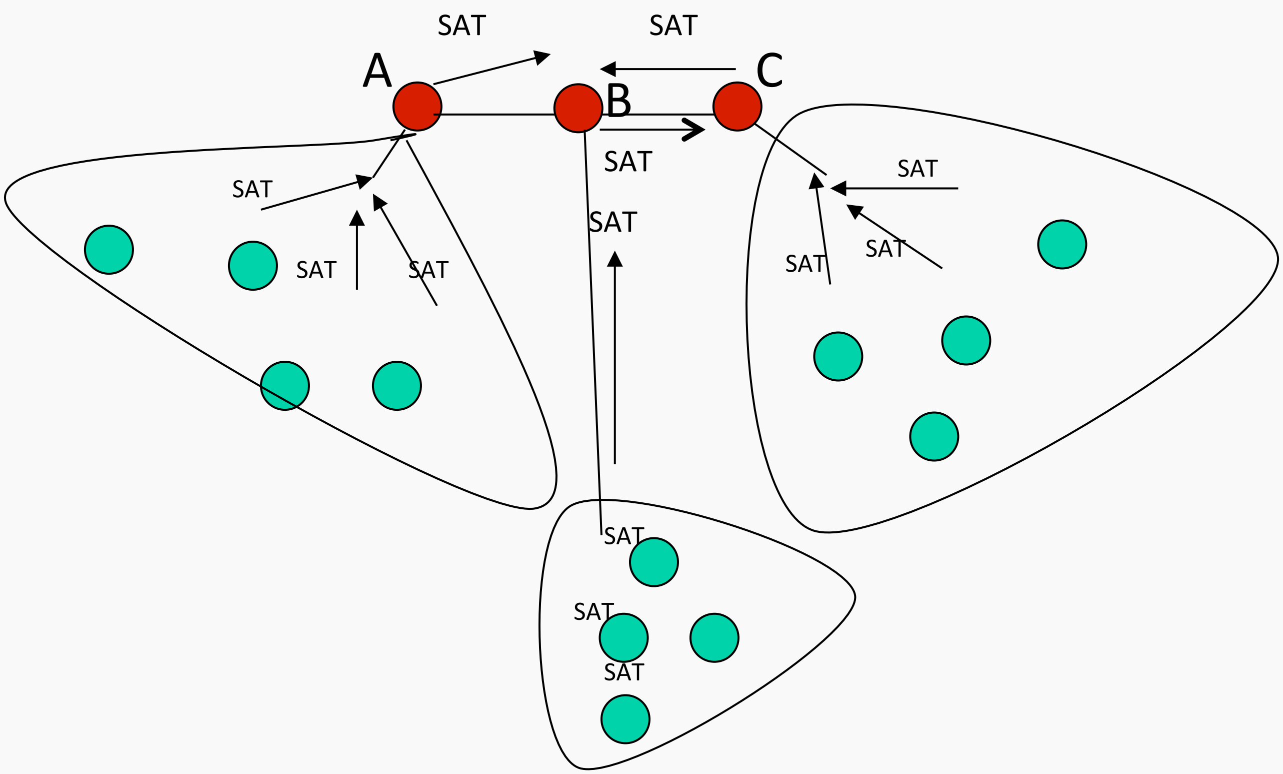



A node become SATURATED only after receiving a message in the state PROCESSING from its parent

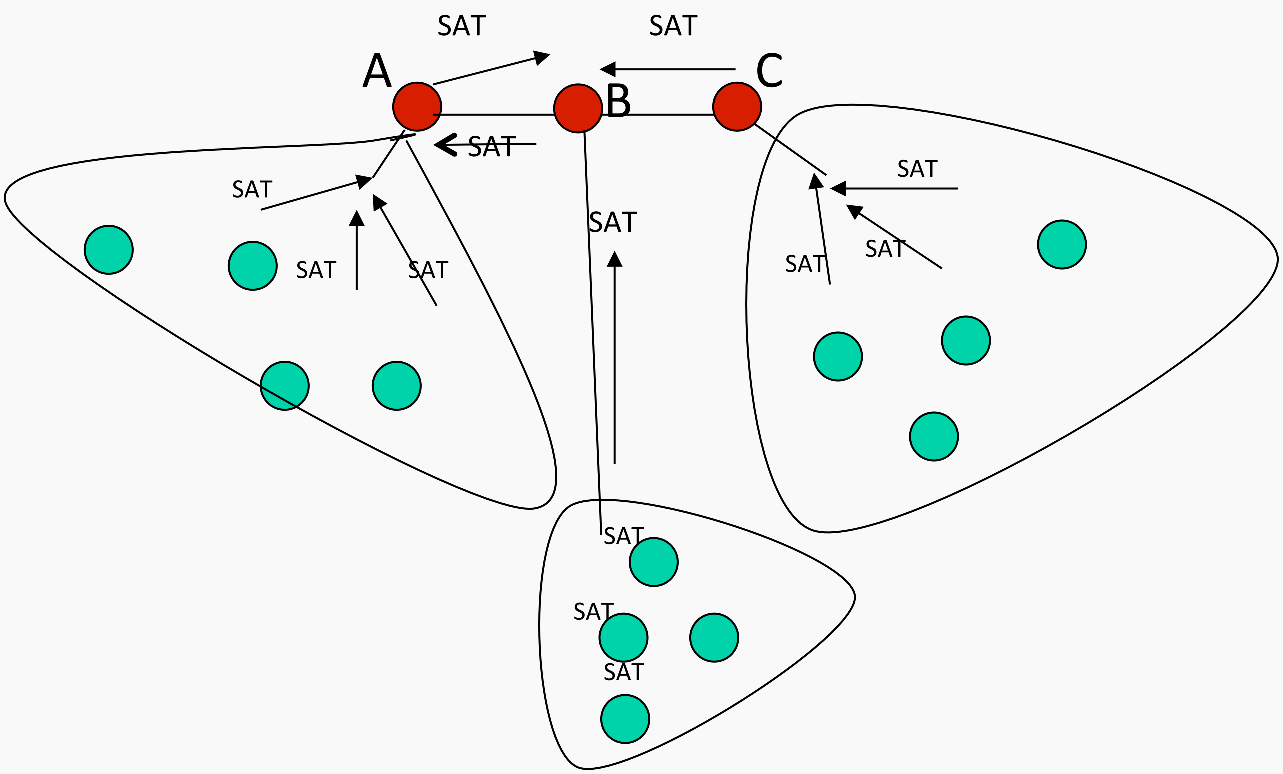
TWO neighbouring entities become saturated

By contradiction: say that three neighbouring entities become saturated:






Paola Flocchini




Paola Flocchini

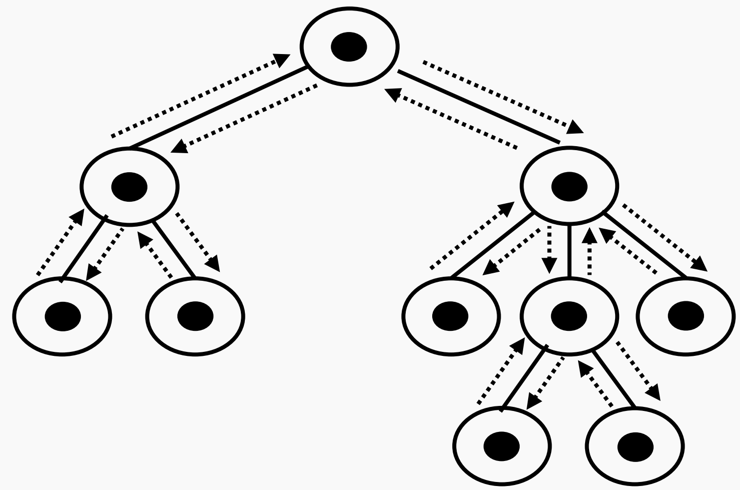
Which entities become saturated depends on the unpredictable delays

Observations and Examples

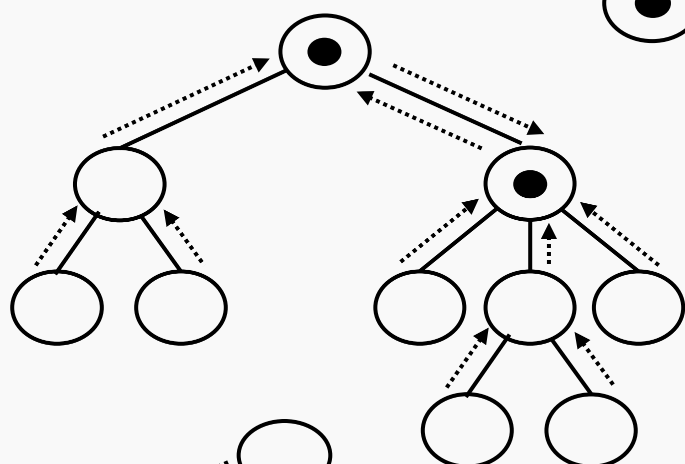
# Message Complexity

**Activation**: Worst case - n initiators

**2(n-1)**

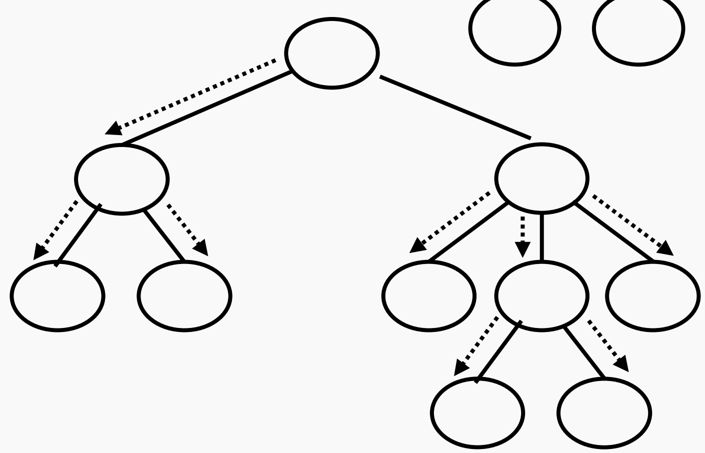


**Saturation**:



**n**

**Notification**:



**n - 2**

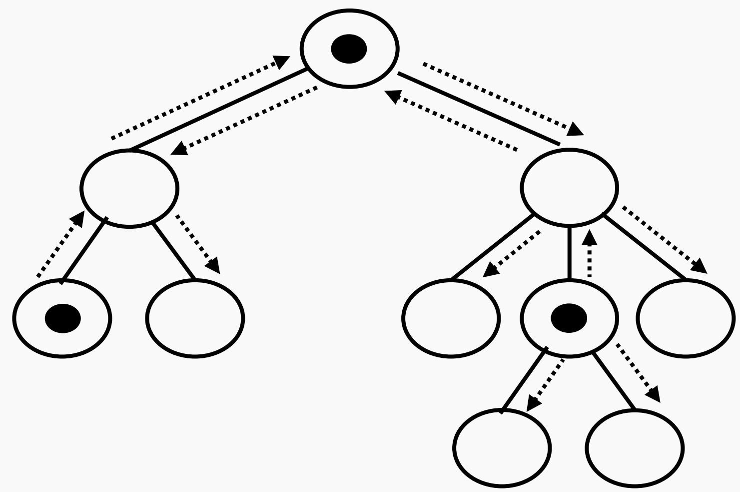
Tot: 2n -2+n+n-2=4n-4

# Message Complexity

**Activation**: In general - k* initiators

**n +k* -2**

(wake-up in the tree)



Let S be the set of initiators

$$\sum_{x \in S} |N(x)| + \sum_{x \notin S} (|N(x)|-1)$$

$$= \sum_{x} |N(x)| - \sum_{x \notin S} 1 \qquad = 2(n-1) - (n-k^*)$$

$$= 2n-2-n+ k^*$$

= **n +k* -2**

# Message Complexity

**Activation**: In general - k* initiators

**n +k* -2**

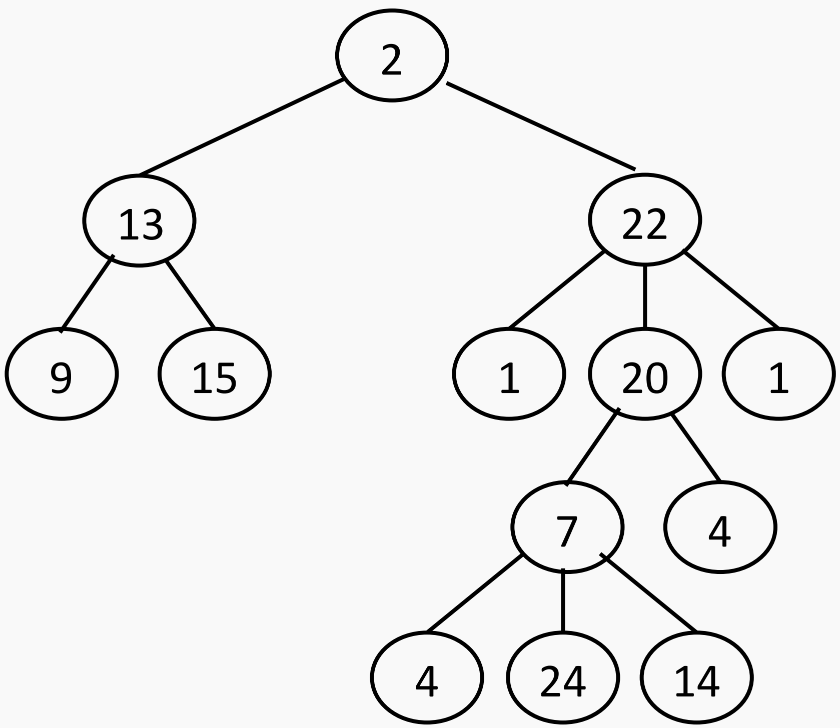
**Saturation: n** does not depend on number of initiators

**Notification**: **n -2**

**TOT**: **3n+k*-4**

## Put information in the saturation message

# Minimum Finding




Entity x has in input value(x)

At the end each entity should know whether it is the minimum or not.

States S {AVAILABLE, ACTIVE, PROCESSING, SATURATED} Sinit = AVAILABLE

**AVAILABLE**

*Spontaneously*

```
send(Activate) to N(x);
min := v(x);
Neighbours:= N(x)
if |Neighbours|=1 then
    M:=("Saturation", min);
    parent ⇐ Neighbours;
    send(M) to parent;
    become PROCESSING;
else become ACTIVE;
```

```
Receiving(Activate )
        send(Activate) to N(x) – {sender};
        min:=v(x);
        Neighbours:=  N(x);
        if |Neighbours|=1   then
            M:=("Saturation", min);
            parent ⇐ Neighbours;
            send(M)   to  parent;
            become PROCESSING;
        else  become ACTIVE;
```

**ACTIVE**

*Receiving(M)*

```
        min:= MIN{min, M}
        Neighbours:= Neighbours - {sender}};
        if |Neighbours|=1 then
                M:=("Saturation", min);
                parent ⇐ Neighbours;
                send(M) to parent;
                become PROCESSING;
```

**PROCESSING**

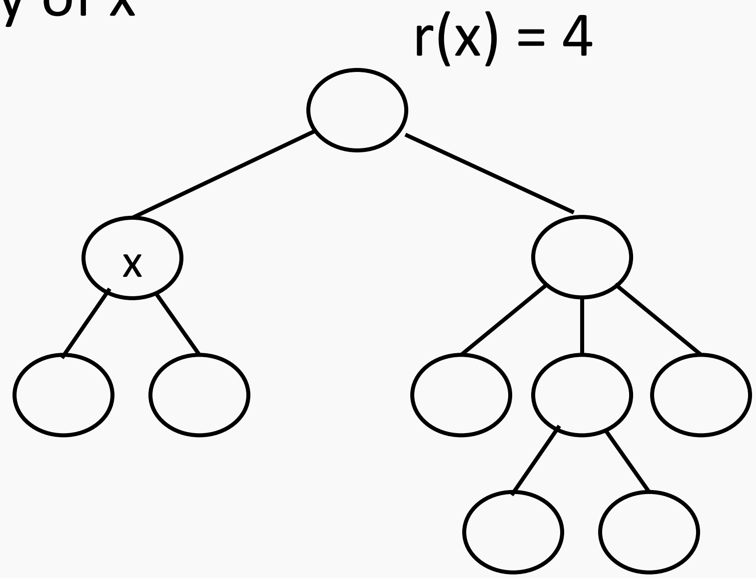
```
receiving(M)
        min:= MIN{min, M}
        Notification:= ("Resolution", min)
        send (Notification)  to N(x) -parent
        if v(x)=min then
                become MINIMUM
        else
                become LARGE

receiving(Notification)
        send(Notification)  to N(x) -parent
        if v(x)=Received_Value  then
                become MINIMUM;
        else
                become LARGE;
```

# Finding Eccentricities

$d(x,y)$ = distance between x and y

$\max_{y}\{d(x,y)\} = r(x)$ eccentricity of x




How to find the eccentricity of all nodes

## Idea 1:

1) Every node broadcasts a request,
2) the leaves send up a message to collect the distances.

Complexity: $O(n^2)$

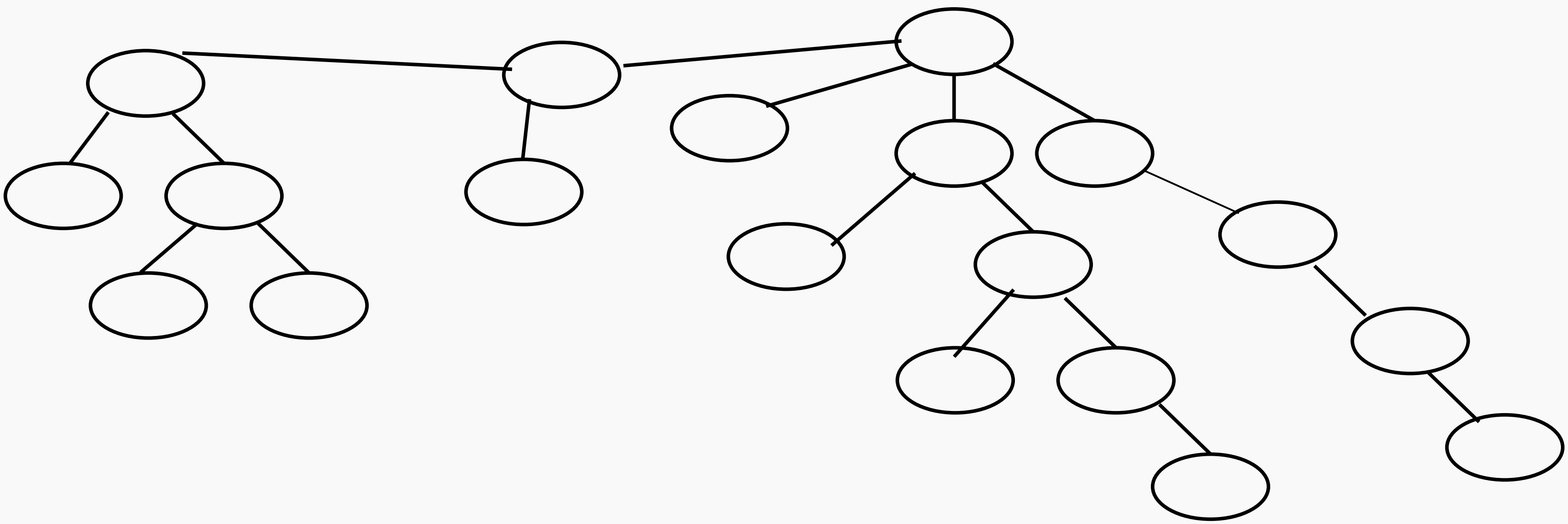
## Other Idea:

Based on the saturation technique:

1) Find the eccentricity of the two saturated nodes
2) Propagate the needed info so that the other nodes can find their eccentricity (in the notification phase)

Complexity = saturation

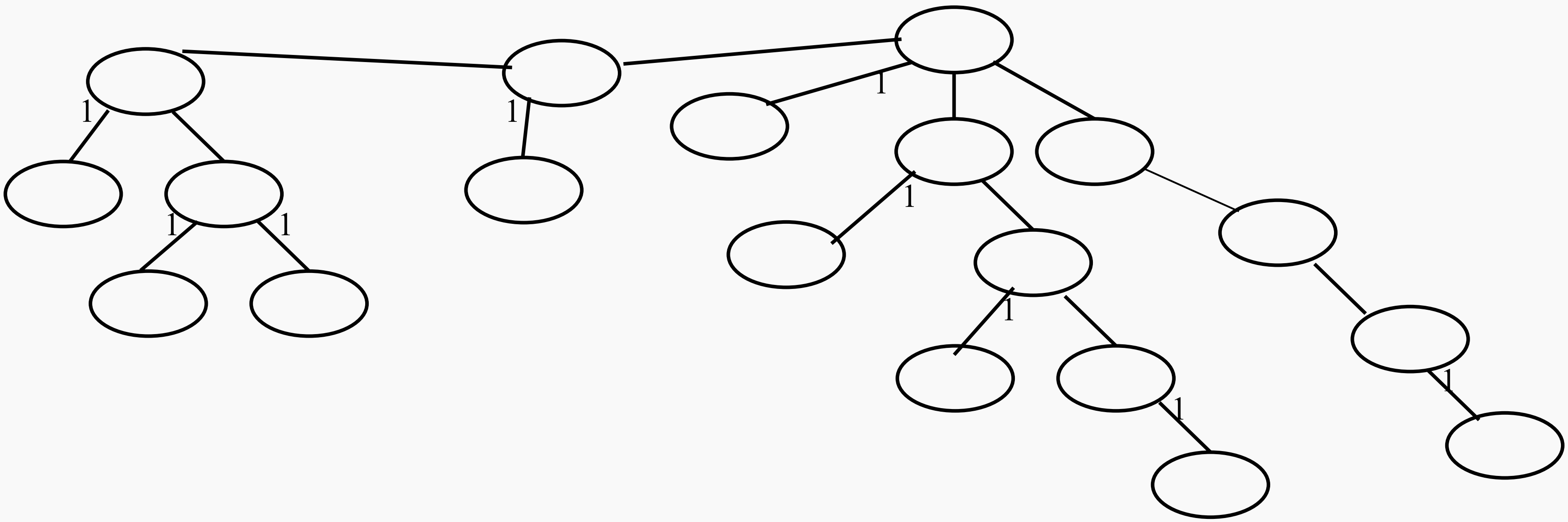
# Observations and Examples



States S {AVAILABLE, ACTIVE, PROCESSING,

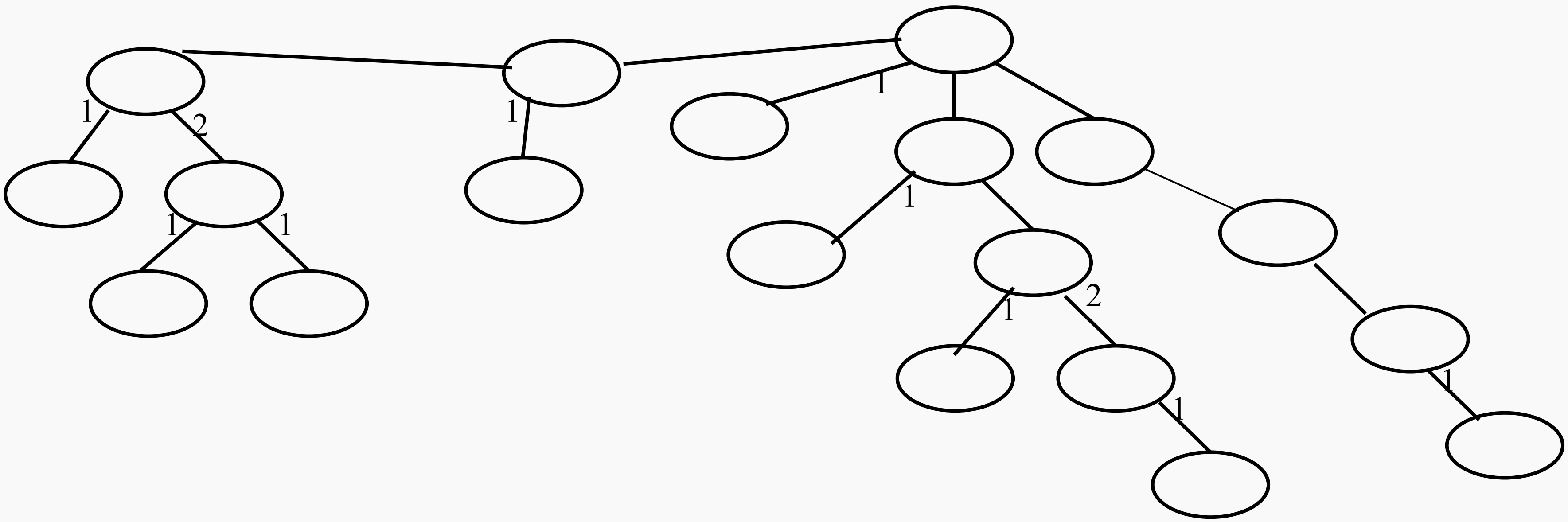
SATURATED}      Sinit = AVAILABLE

# Find the eccentricity



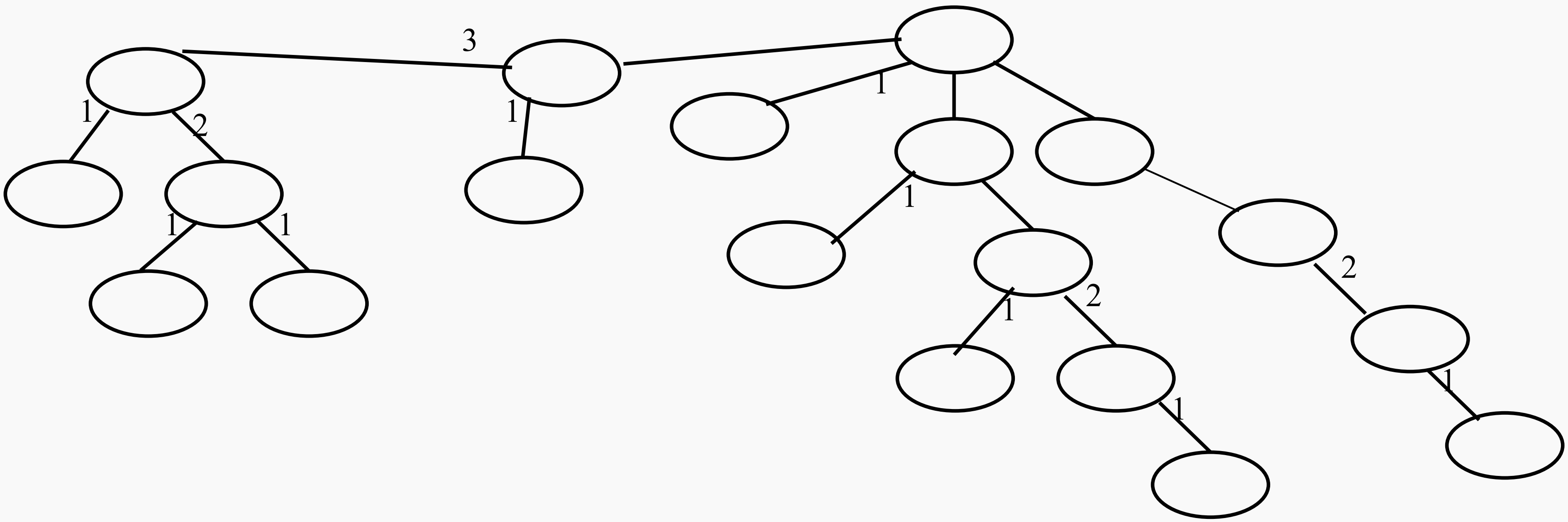

Paola Flocchini

# Find the eccentricity



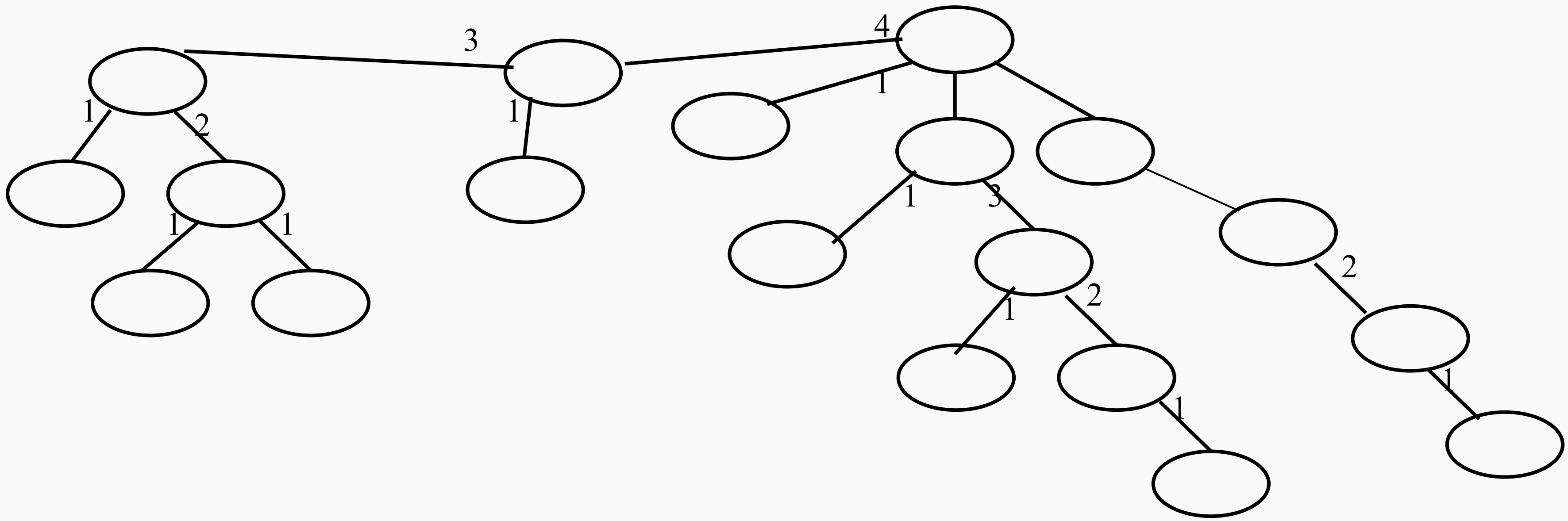

Paola Flocchini

# Find the eccentricity



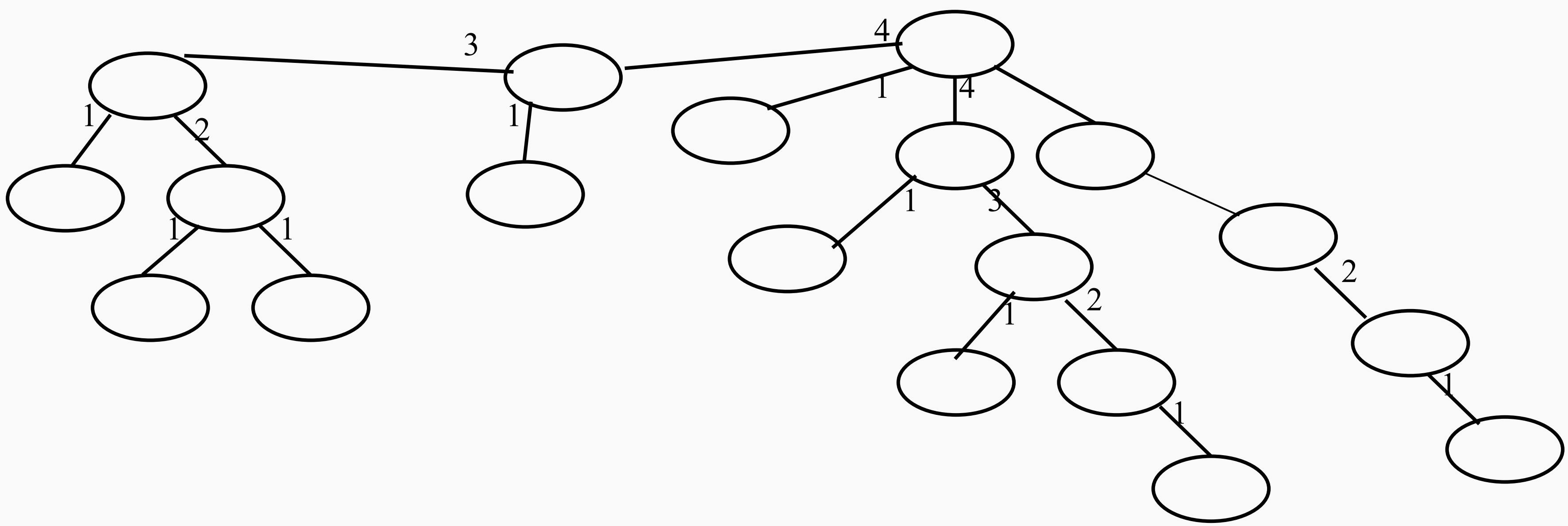

Paola Flocchini

# Find the eccentricity



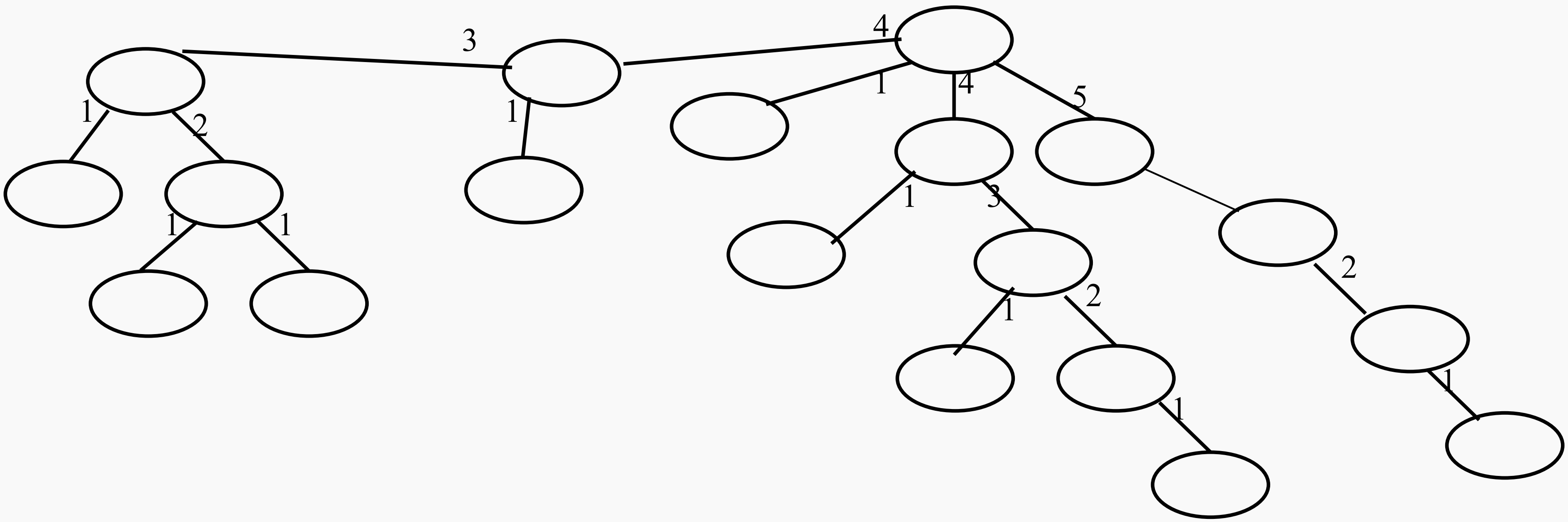

Paola Flocchini

# Find the eccentricity



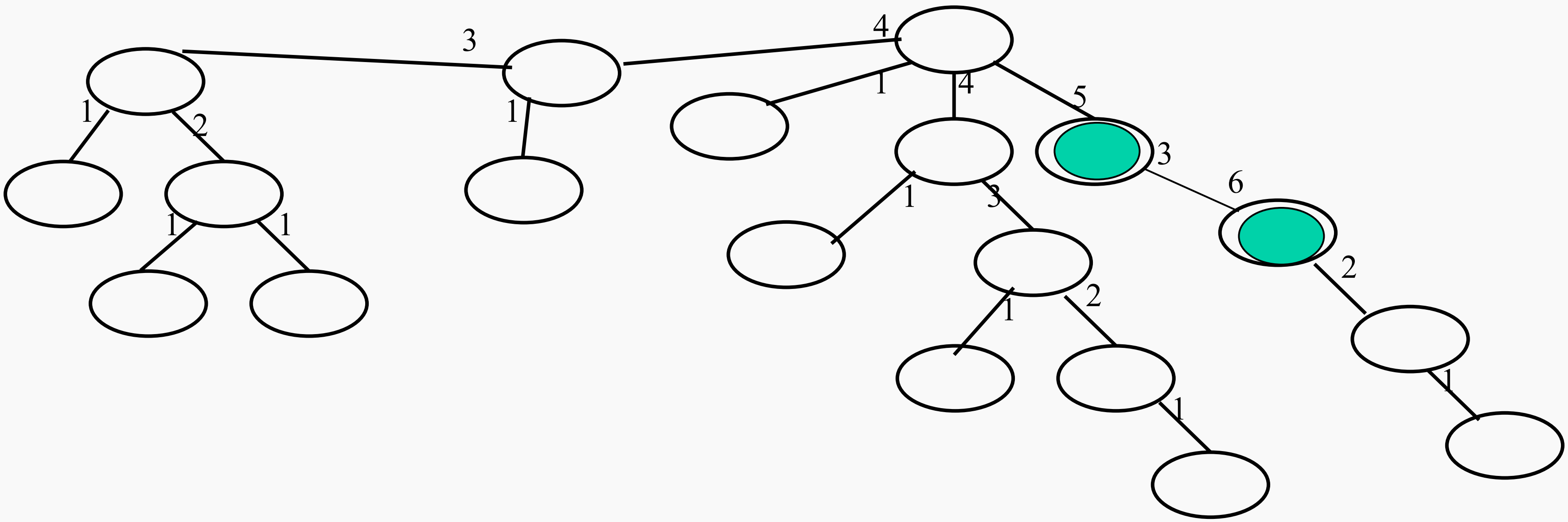

Paola Flocchini

# Find the eccentricity

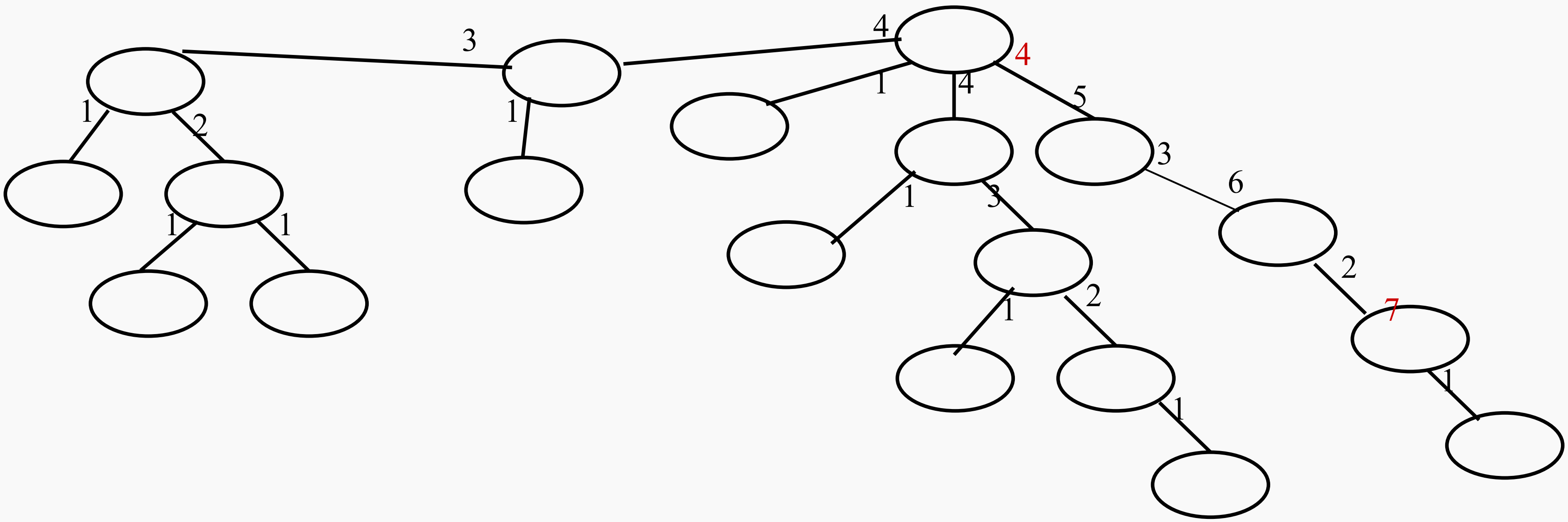



Paola Flocchini

# Find the eccentricity

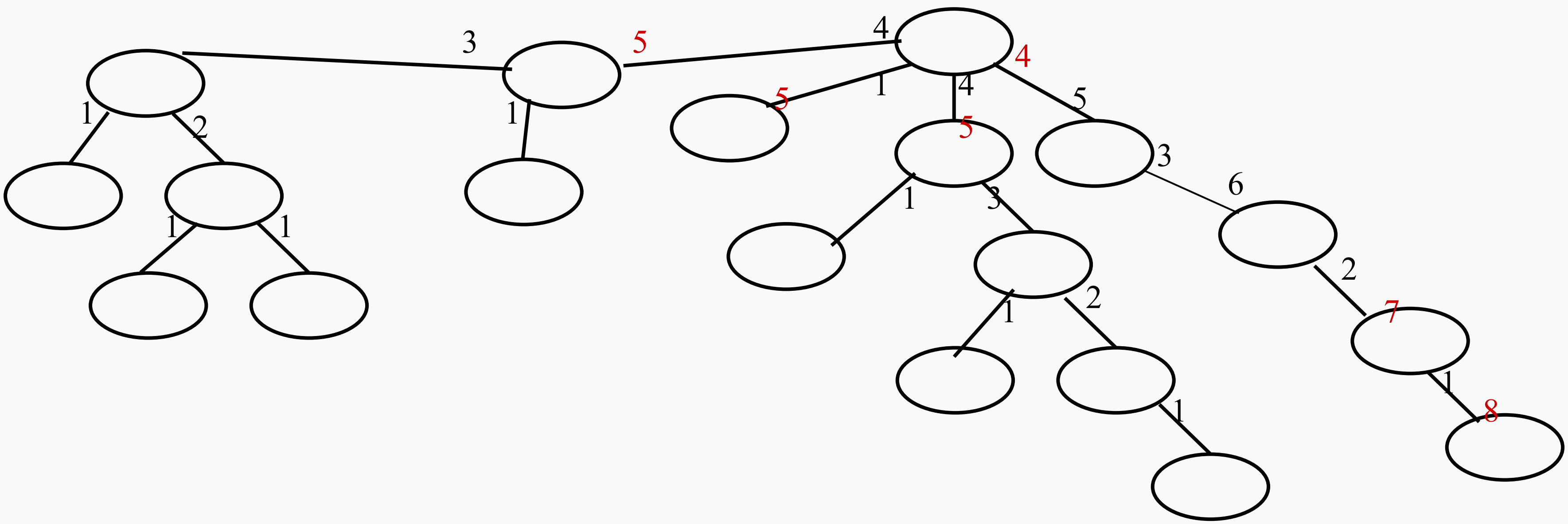


# Find the eccentricity



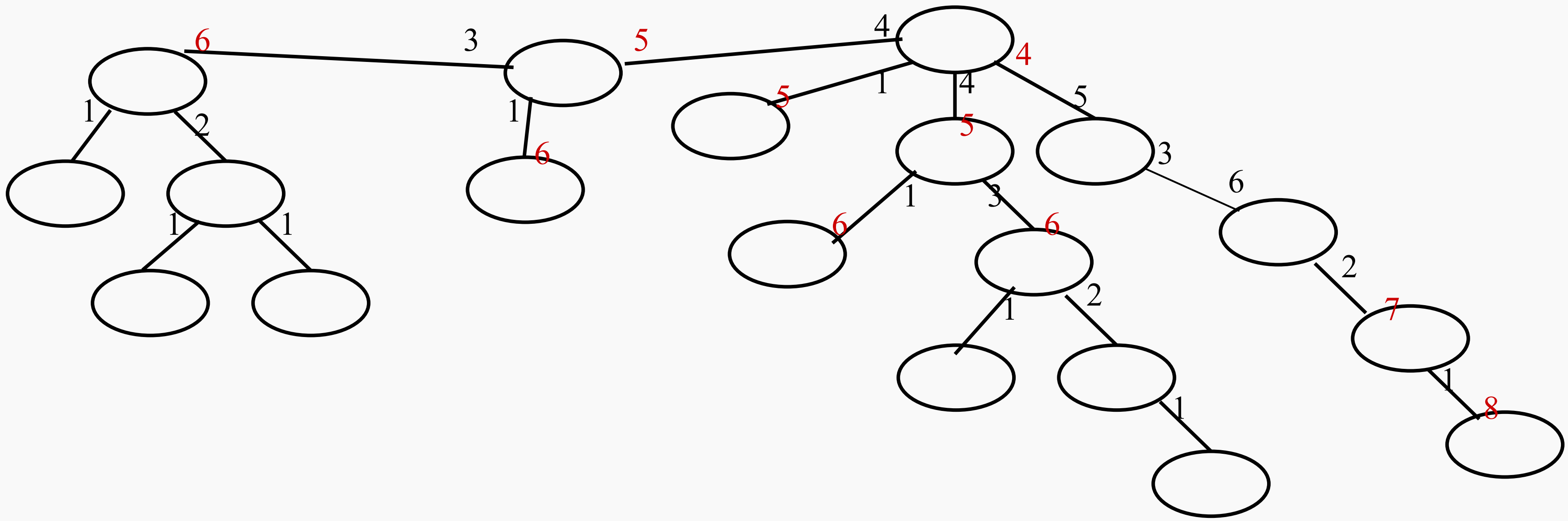

Paola Flocchini

# Find the eccentricity



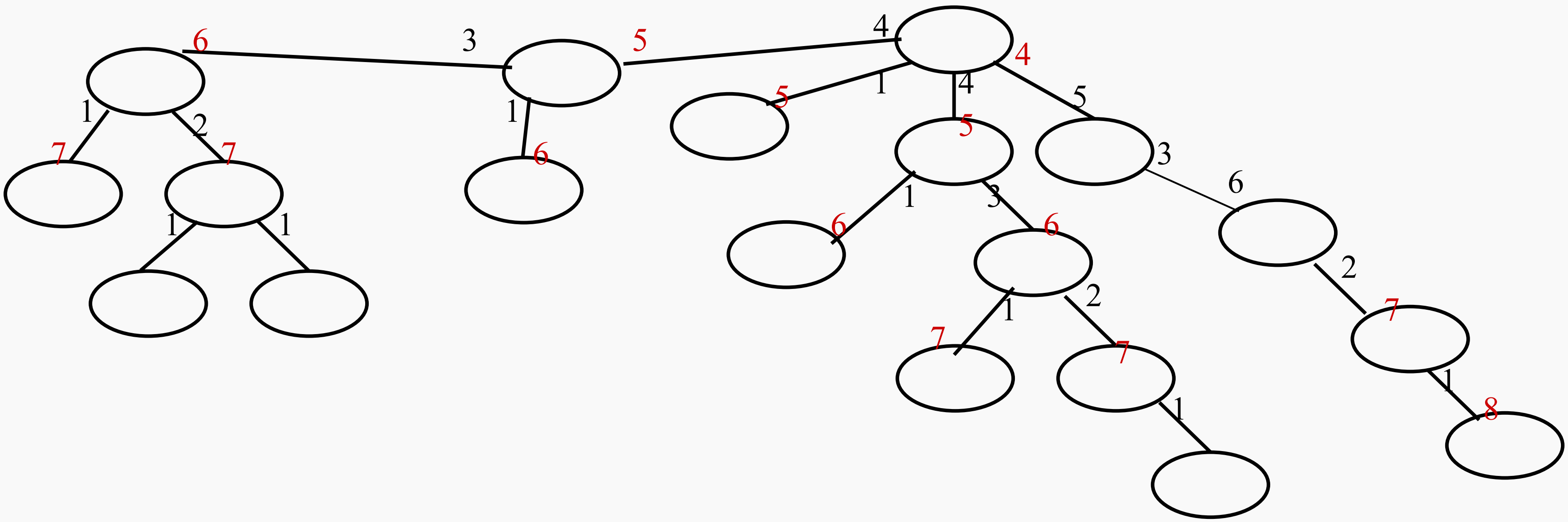

Paola Flocchini

# FIND THE ECCENTRICITY



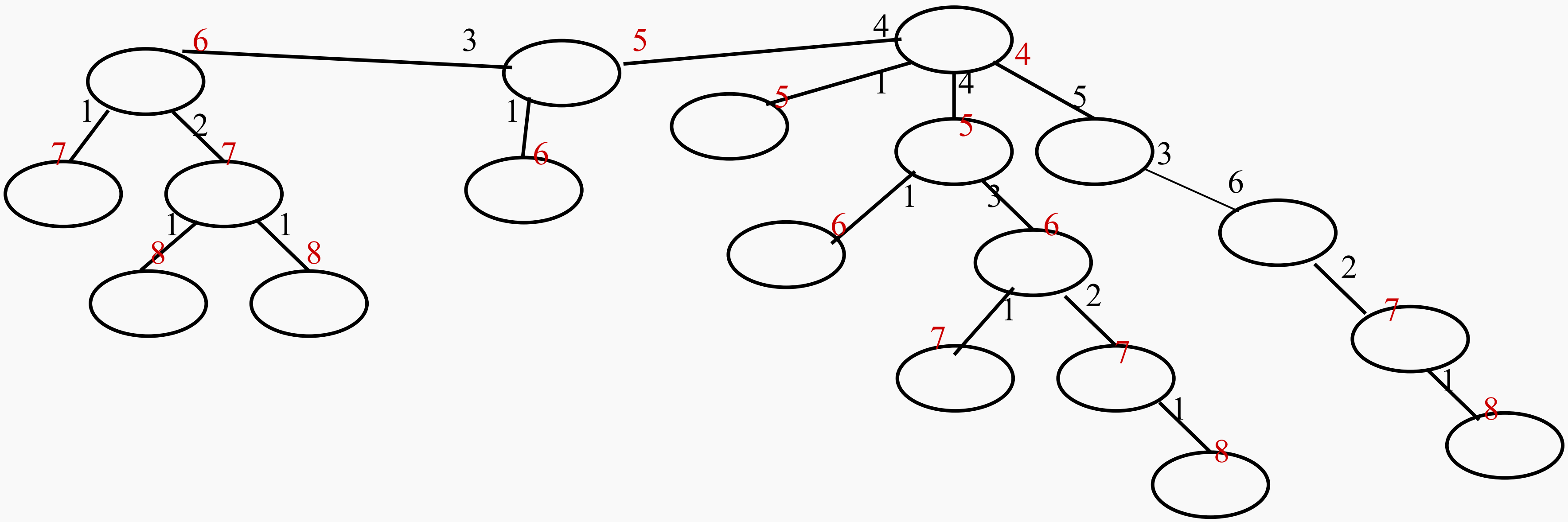

Paola Flocchini

# Find the eccentricity




Paola Flocchini

# Find the eccentricity




Paola Flocchini

define Distance[ ]

**AVAILABLE**

*Spontaneously*

```
        send(Activate) to N(x);
        Distance[x]:= 0;
        Neighbours:=N(x)
        if |Neighbours|=1   then
                maxdist:= 1+ Max{Distance[*]}
                M:=("Saturation", maxdist);
                parent ⇐ Neighbours;
                send(M) to parent;
                become PROCESSING;
        else  become ACTIVE;
```

```
Receiving(Activate)
        send(Activate)to N(x) – {sender};
        Distance[x]:= 0;
        Neighbours:=  N(x);
        if |Neighbours|=1   then
                maxdist:= 1+ Max{Distance[*]}
                M:=("Saturation", maxdist);
                 parent ⇐ Neighbours;
                 send(M)   to  parent;
                 become PROCESSING;
        else  become ACTIVE;
```

**ACTIVE**

*Receiving(M)*

```
Distance[{sender}]:= Received_distance;
Neighbours:= Neighbours - {sender}};
if |Neighbours|=1 then
    maxdist:= 1+ Max{Distance[*]}
    M:=("Saturation", maxdist);
    parent ⇐ Neighbours;
    send(M) to parent;
    become PROCESSING;
```

**PROCESSING**

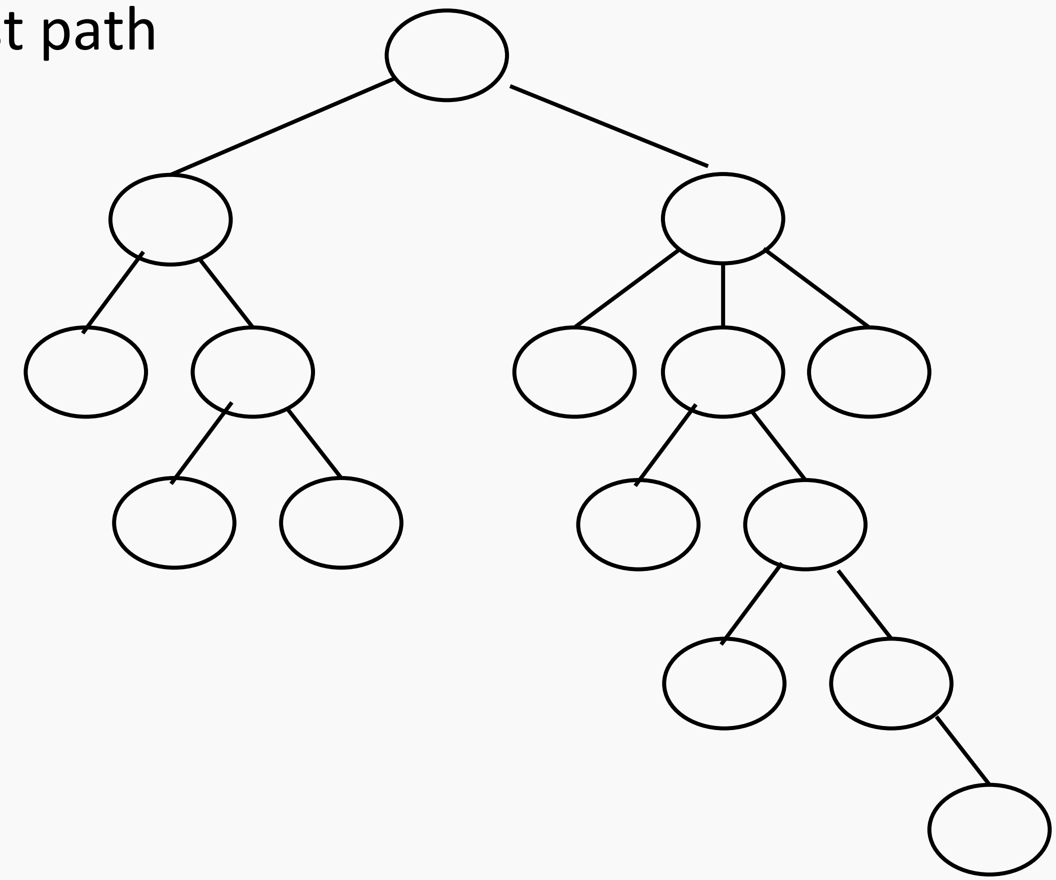
*receiving(M)*

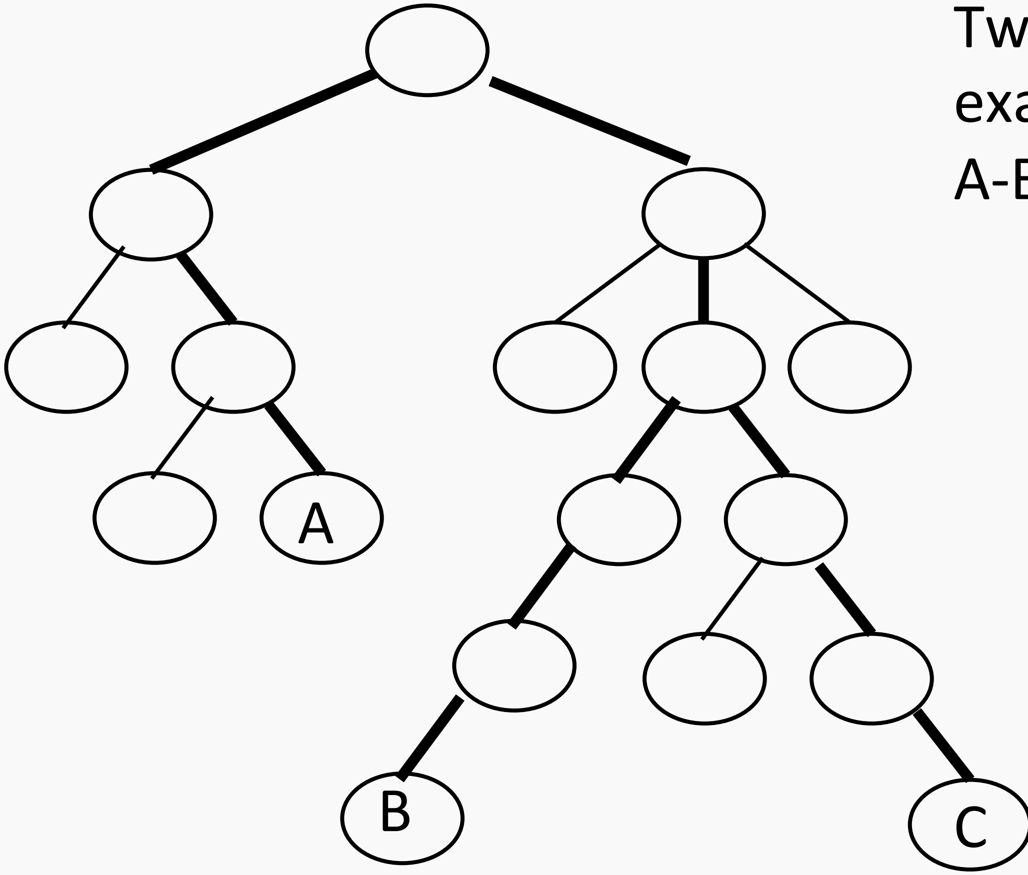
```
        Distance[{ sender}]:= Received_distance;
        r(x):= Max { Distance[z]: z ∈  N(x) }
        for all  y ∈ N(x)-{parent} do
                maxdist:= 1+ Max{Distance[z]:
                                        z∈ N(x)- {y}
                send("Resolution", maxdist)  to y
        endfor
        become DONE
```

# Center Finding

c is the center if r(c) ≤ r(x) for all x belonging to V. Max distance is minimized.

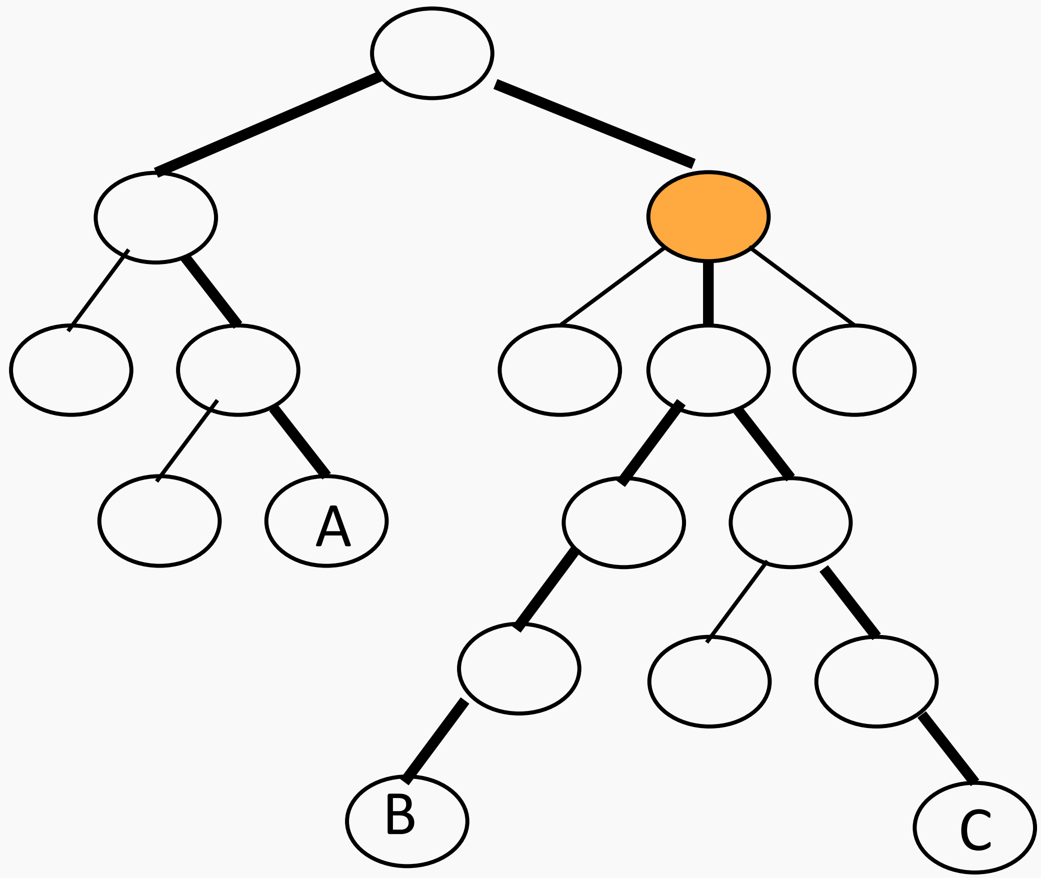
Diametral path: Longest path






Two diameters in this example:
A-B, and A-C

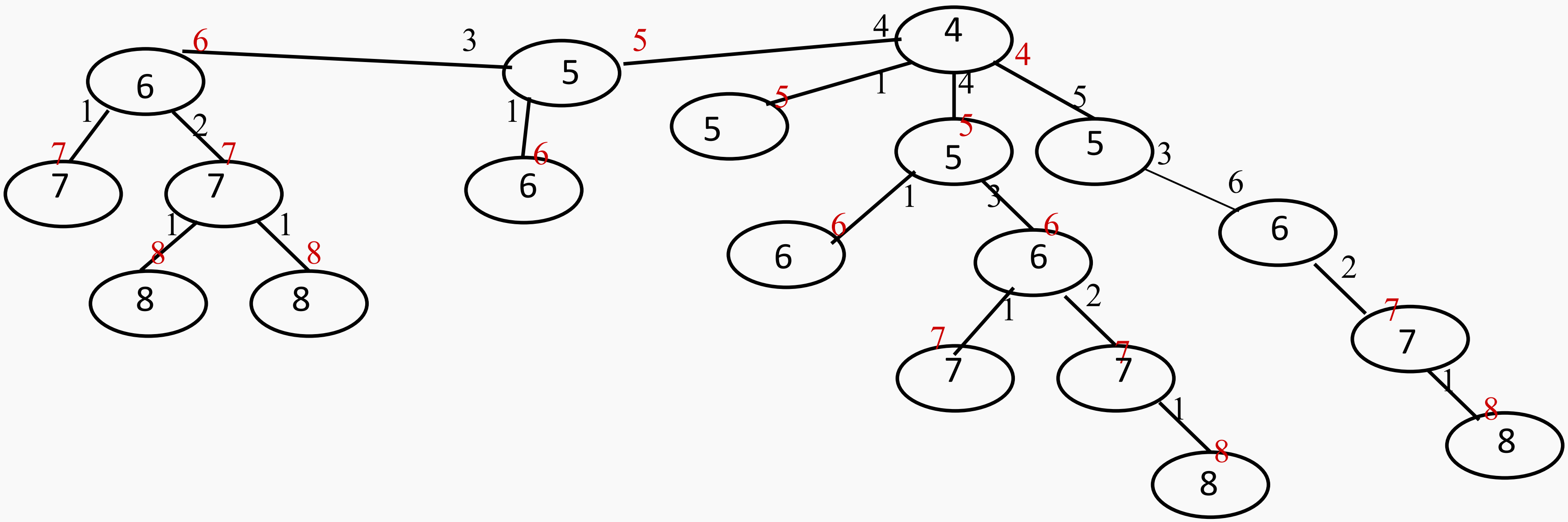
The center is the node with smallest eccentricity




One Idea:

1) Find all the eccentricities  4n-4
2) Find the smallest  2n-2

→ 6n -6

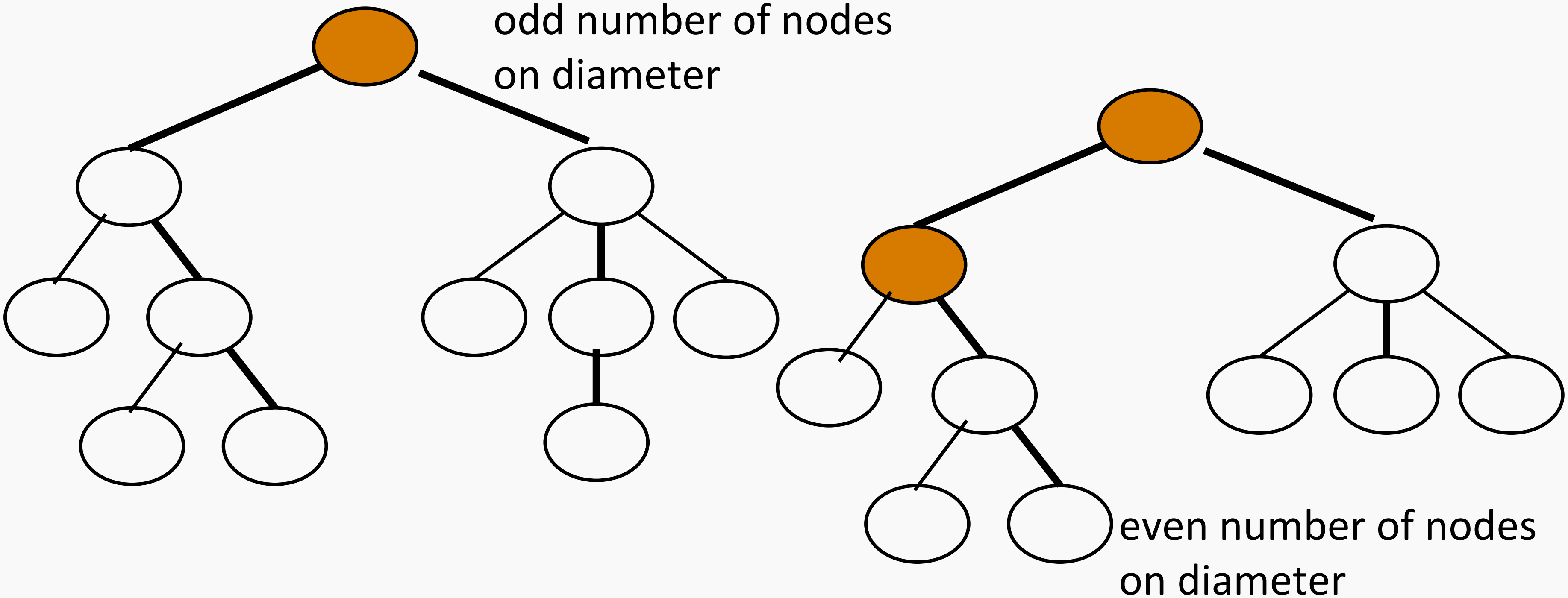



1) Find all the eccentricities

2) Find the smallest

A better strategy can be devised exploiting properties of the center.

**Property 1**: In a tree there is a unique center,
or there are two centers that are neighbours



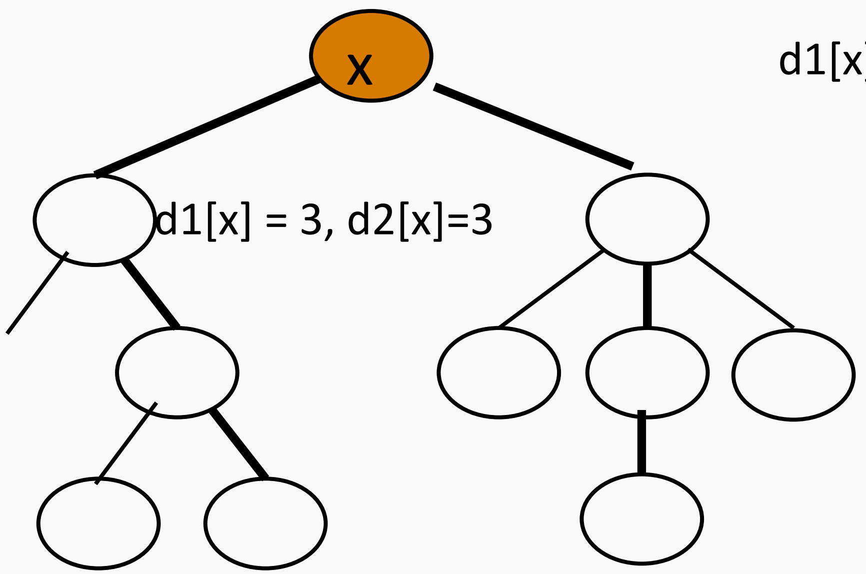

**Proposition 2**: Centers are on diametral paths.

d1[x] = max dist     d2[x] = second max dist  (through different neighbours)

**Property 3**: A node x is a center iff

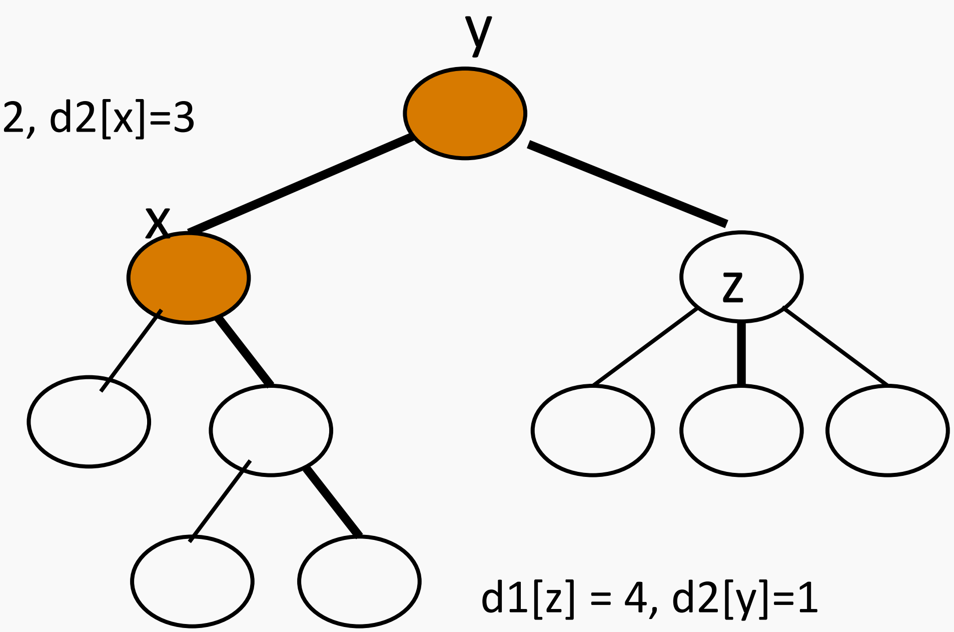
$$d1[x] - d2[x] \leq 1$$

(if d1[x]= d2[x] there is only one center).

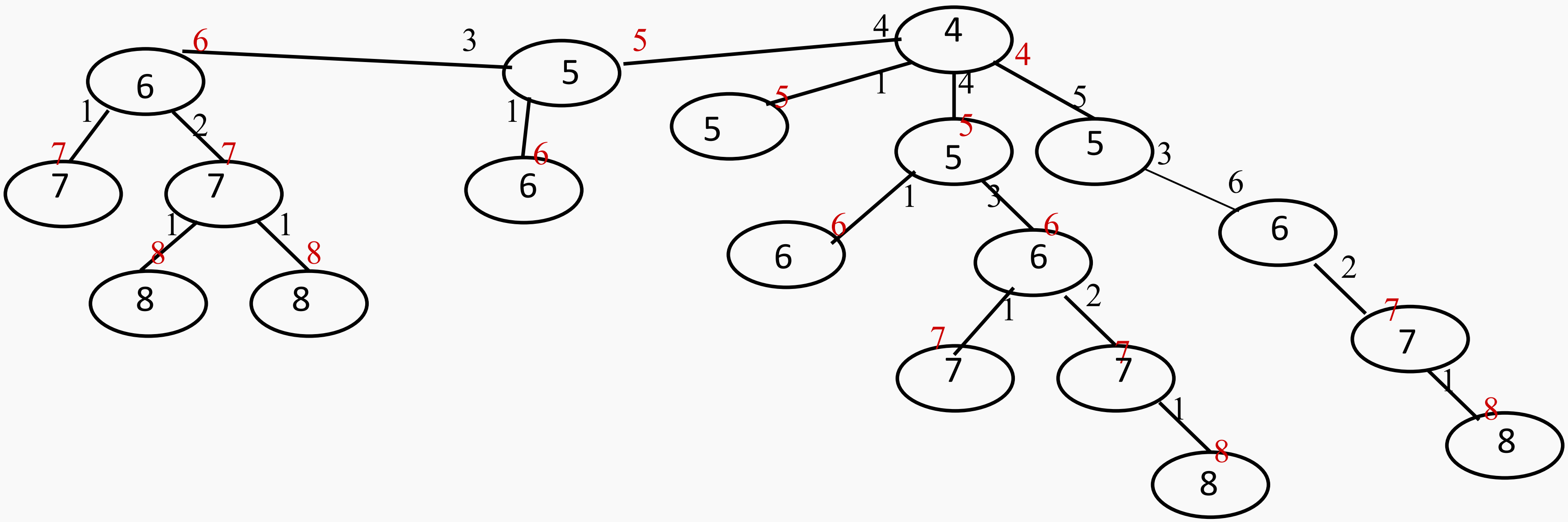



d1[x] = 2, d2[x]=3

d1[y] = 3, d2[y]=2




d1[z] = 4, d2[y]=1




Paola Flocchini

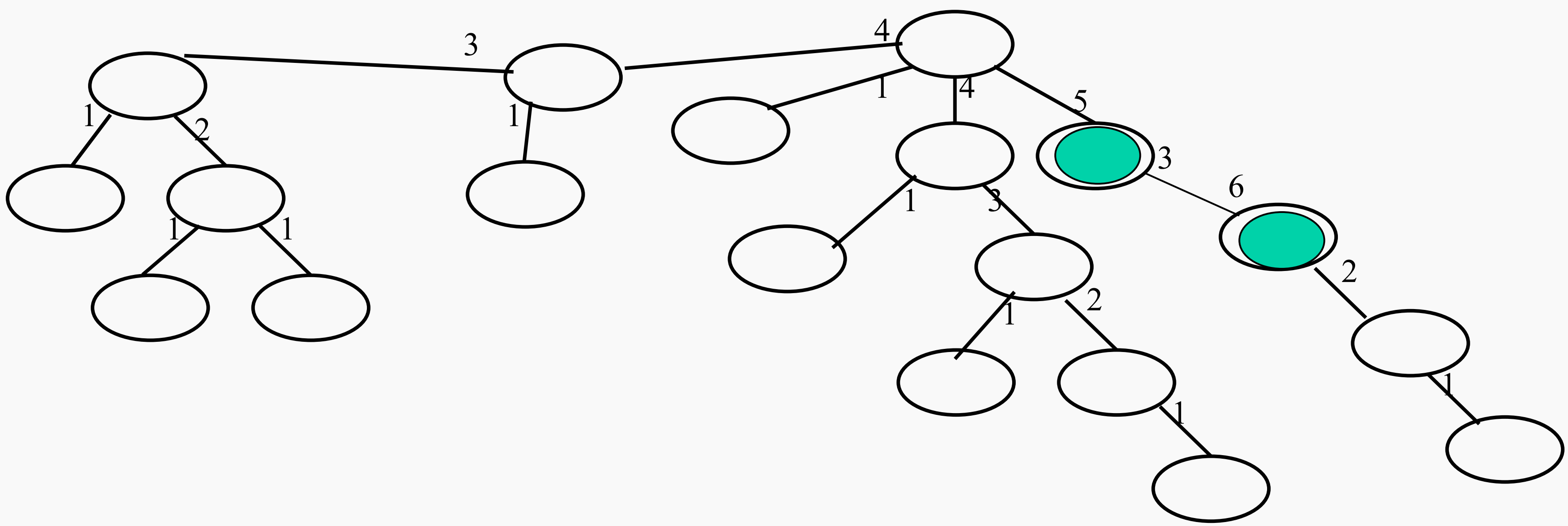
Another Idea:

1) Find all the eccentricities
2) Each node can find out locally whether it is the center or not

4n-4

# Yet another better idea:

1) Find the eccentricities of the saturated nodes




2) Locally Check if I am the center

3) If I am NOT the center ...

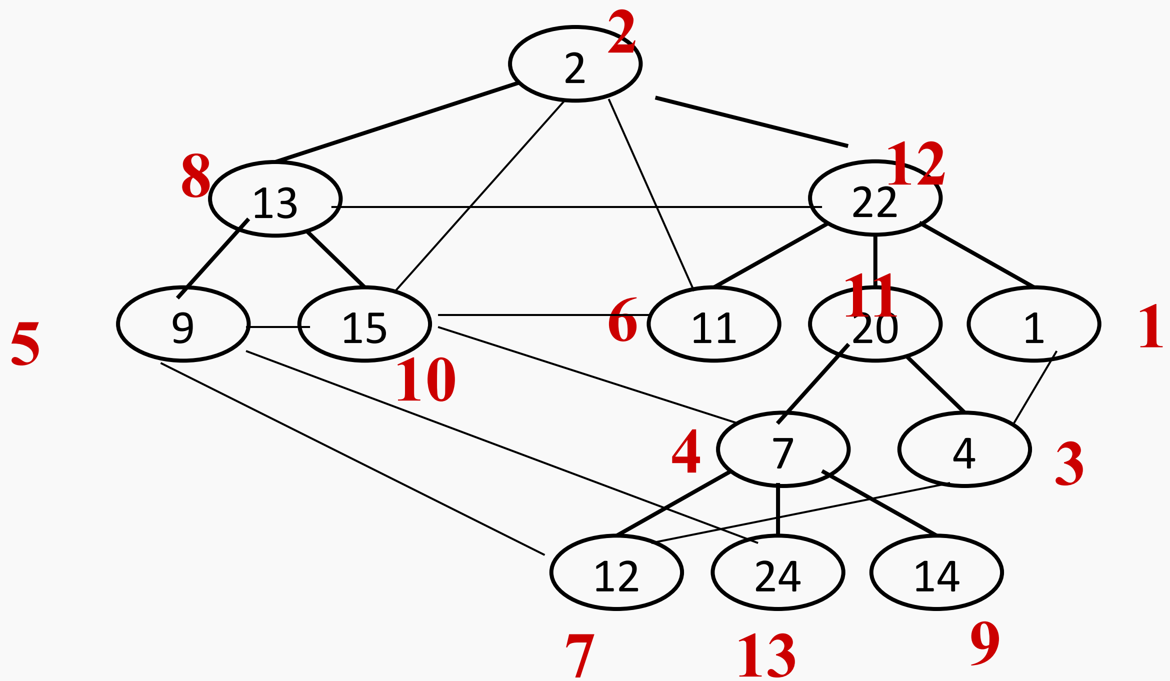
## Yet another better idea:

1) Find the eccentricities of the saturated nodes — 3n-2
2) Locally Check if I am the center (checking largest and second largest)
3) If I am NOT the center, propagate the distance information ONLY in the direction of the center — $\leq$ n/2

How do I know the direction of the center ?

Examples ....... — $\leq$ 3.5 n-2

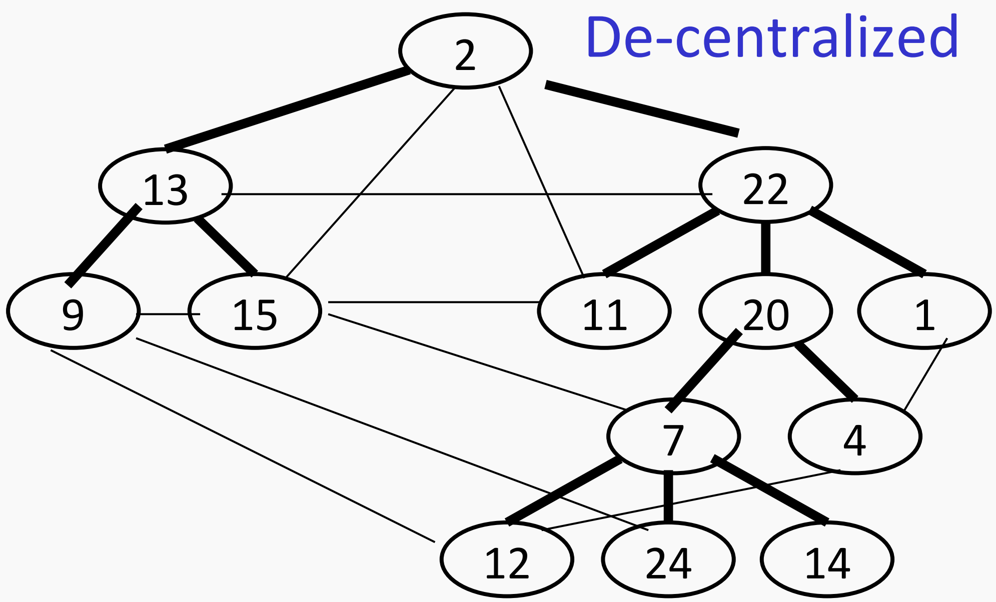
# Ranking



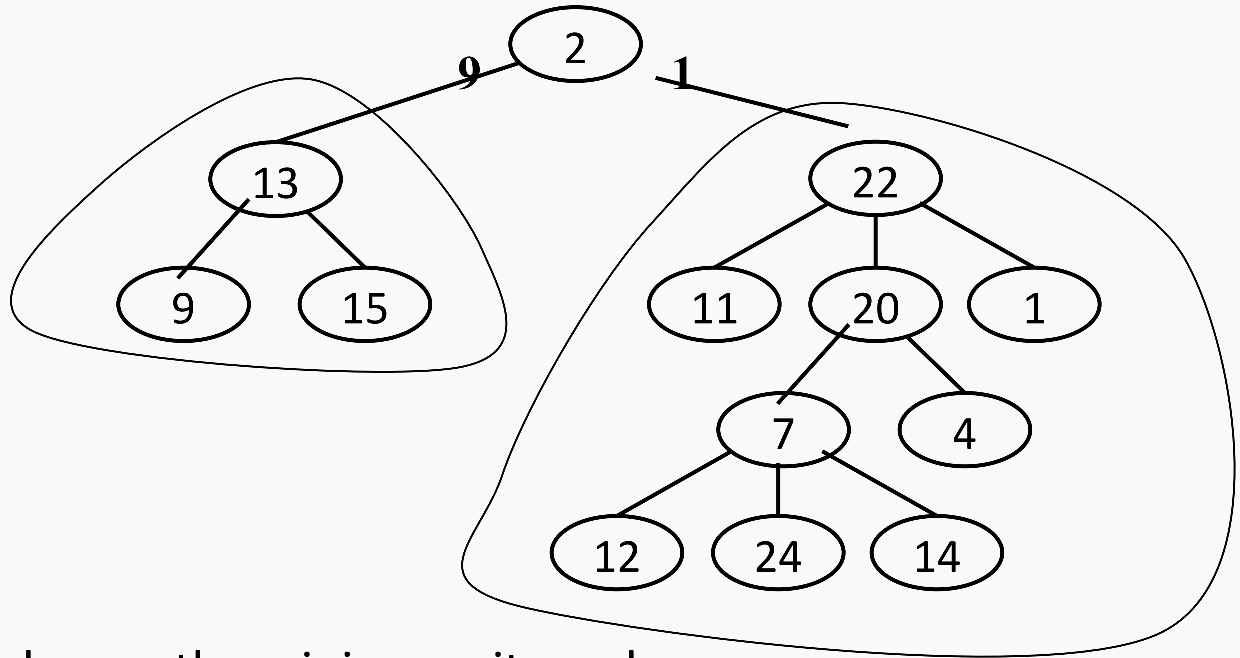
## In an arbitrary network:

1) Find a spanning tree
2) Use saturation+ minimum finding to find a starting node
3) Do-ranking

Phase 3 can be: Centralized
De-centralized



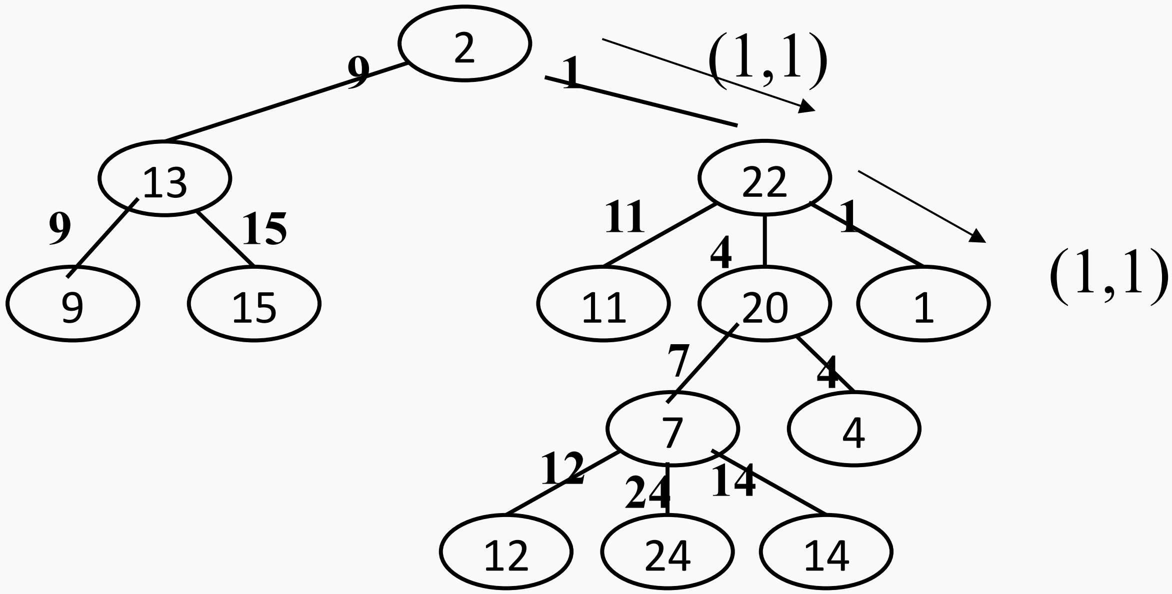
# Centralized Ranking

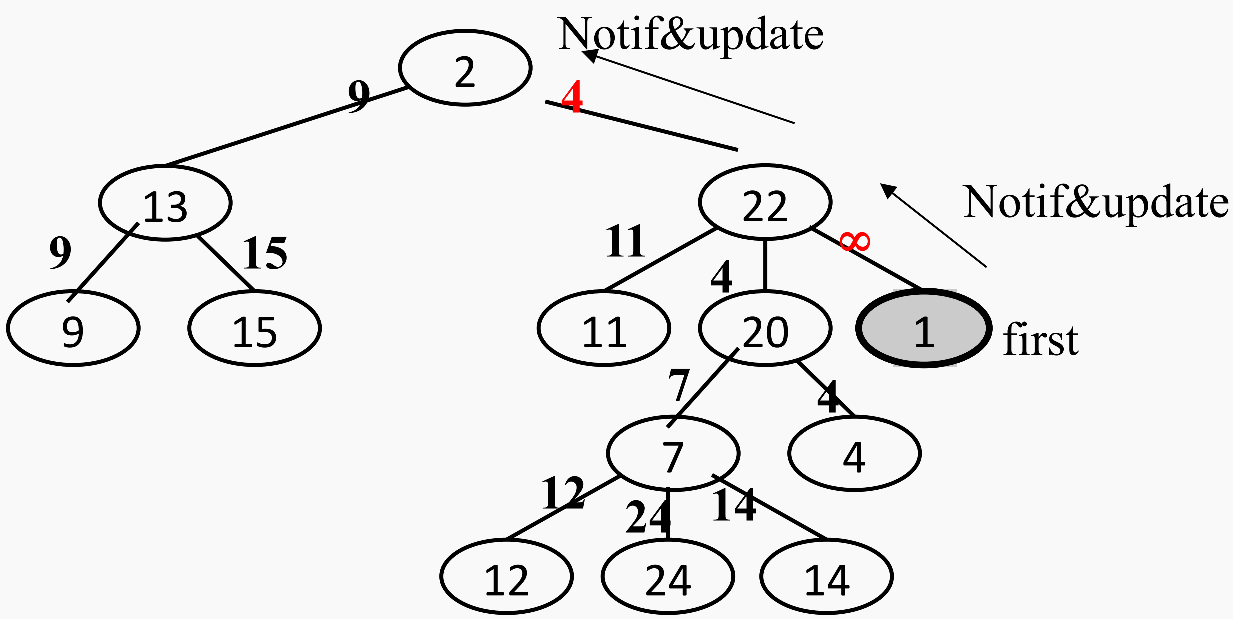


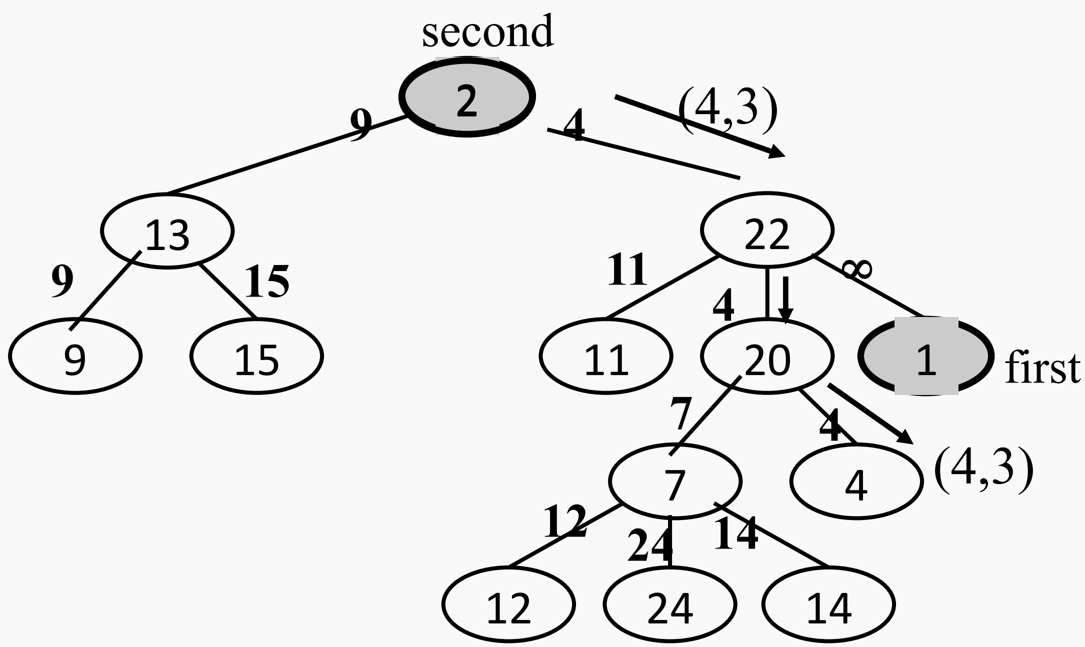

The leader knows the minimum, it sends in that direction a ranking message

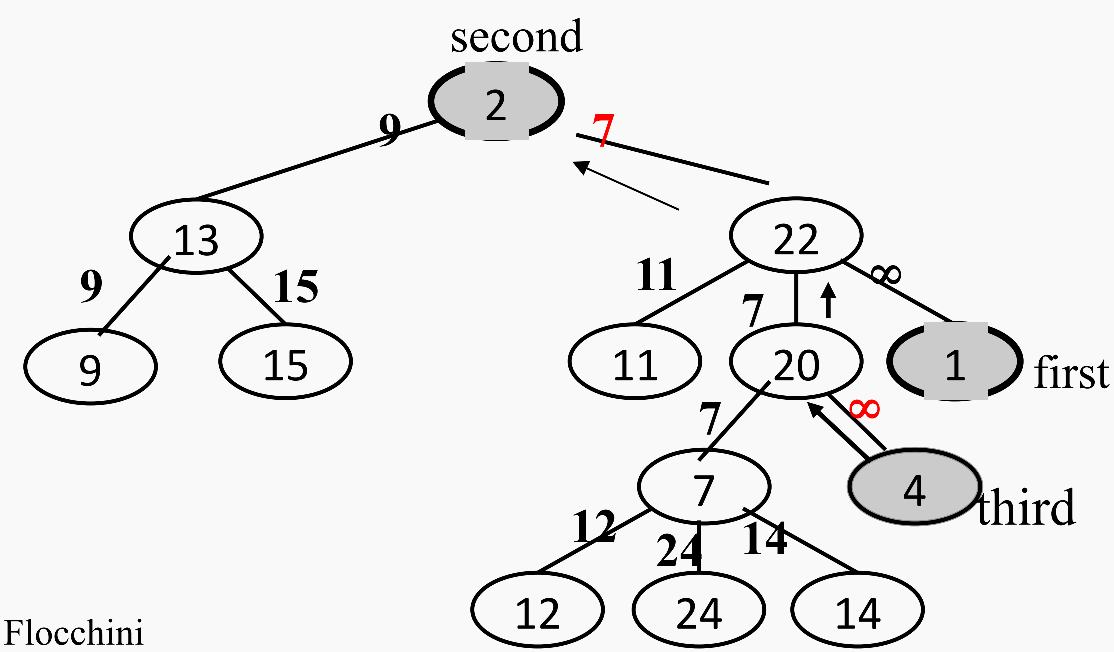
Every node knows the minimum in its subtrees, they can then forward the ranking message in the right direction

When the node to be ranked receives the message It sends up a notification&update message that will travel up to the leader


2
9
1
(1,1)
13
22
9
15
11
4
1
(1,1)
9
15
11
20
1
7
4
7
4
12
24
14
12
24
14


Notif&update
2
9
4
13
22
Notif&update
9
15
11
4
∞
9
15
11
20
1
first
7
4
7
4
12
24
14
12
24
14

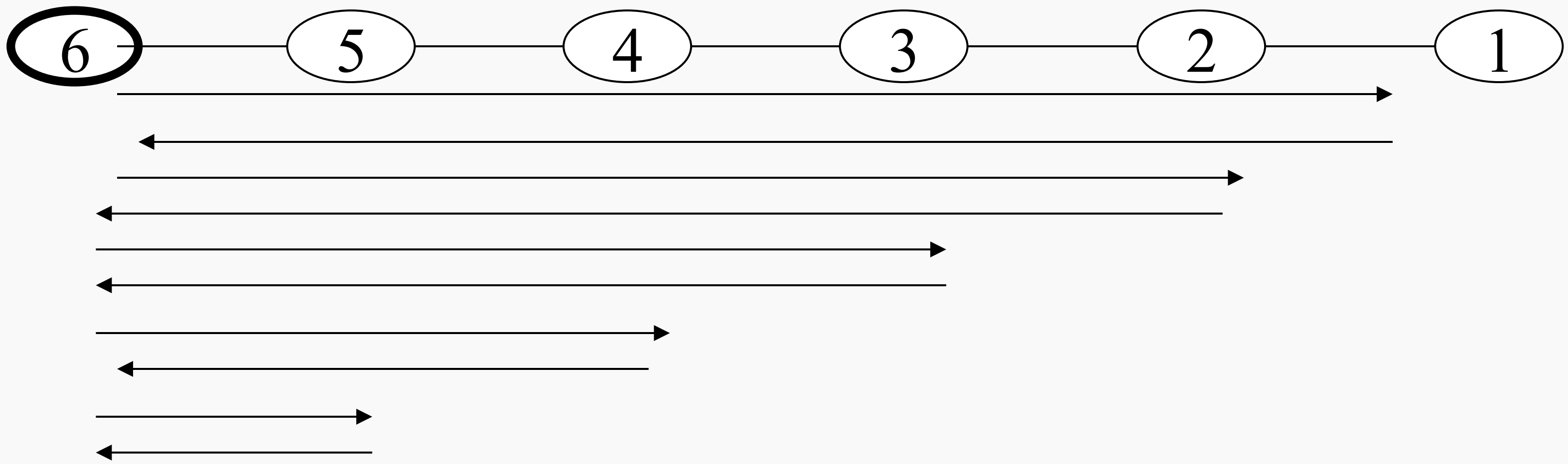

second
2
9
4
(4,3)
13
22
9
15
11
4
∞
9
15
11
20
1
first
7
4
(4,3)
7
4
12
24
14
12
24
14


second
2
9
7
13
22
9
15
11
7
∞
9
15
11
20
1
first
7
∞
7
4
third
12
24
14
12
24
14

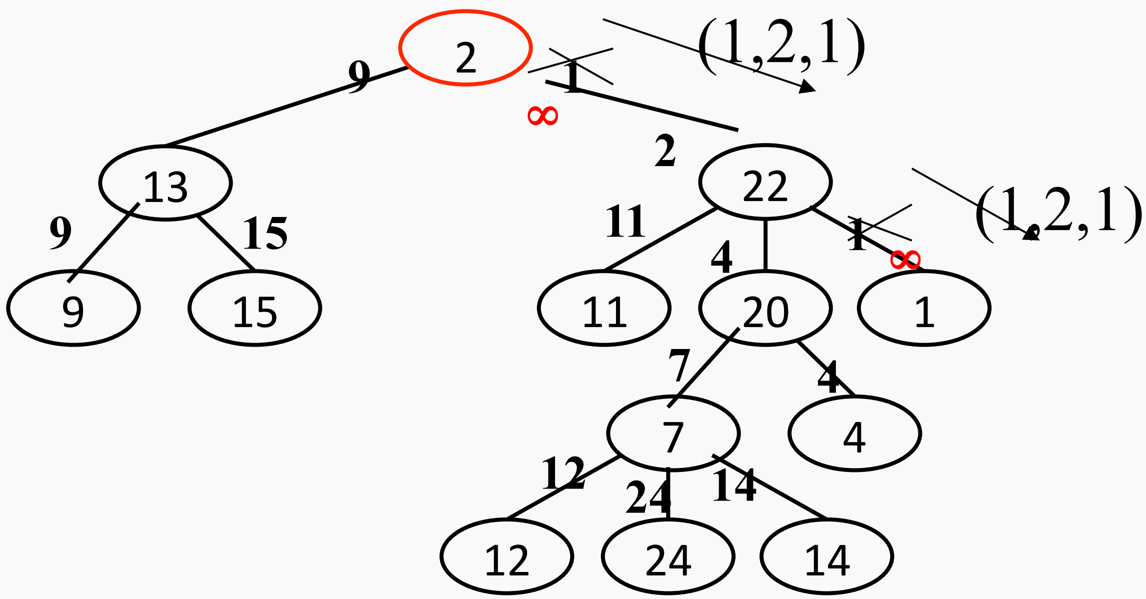


Etc…

## Complexity: worst case



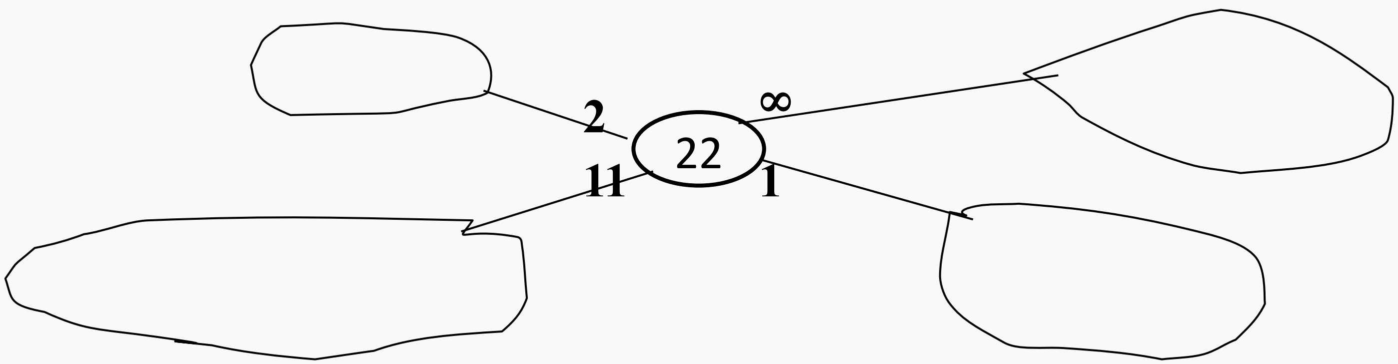
# Decentralized Ranking



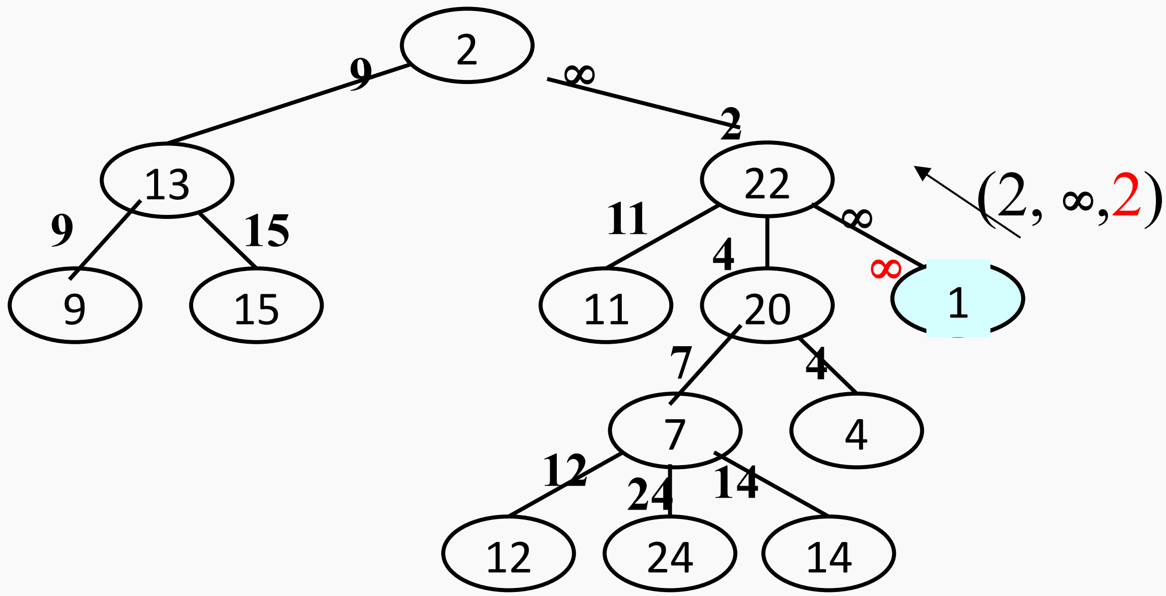

The starter node send a ranking message of the form:
(**first**, **second**,**rank**) in the direction of first.

**first**: smallest value
**second**: second smallest known **SO FAR**
(this is a guess on the value that has to be ranked after first)

The value on a link indicates the SMALLEST value in the corresponding subtree.



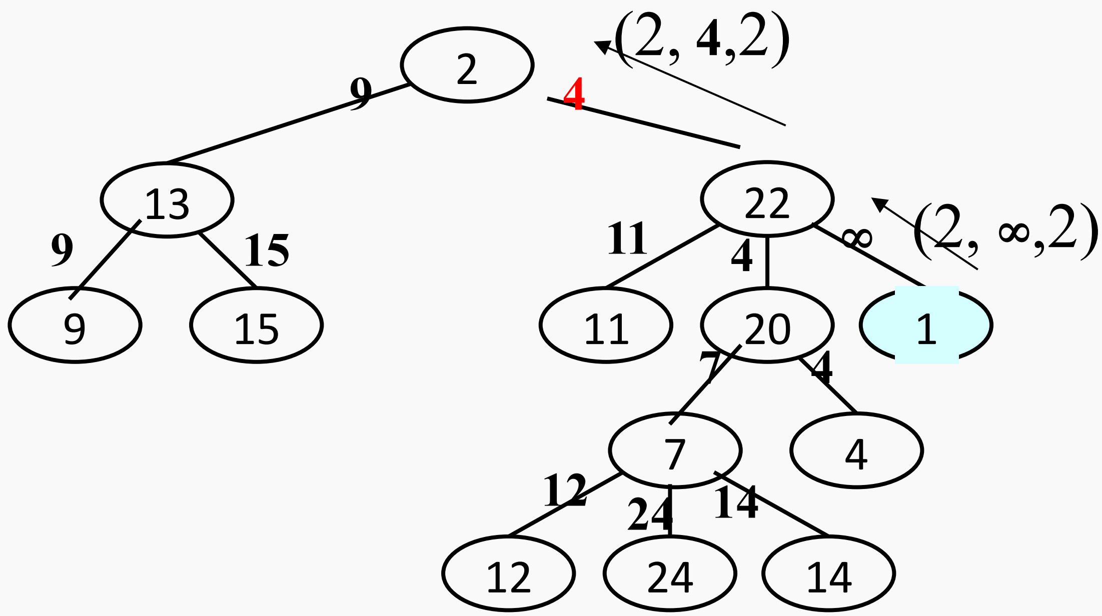

If no value is indicated (or the value is ∞ ) it means that the smallest in the corresponding subtree is unknown (for the moment)

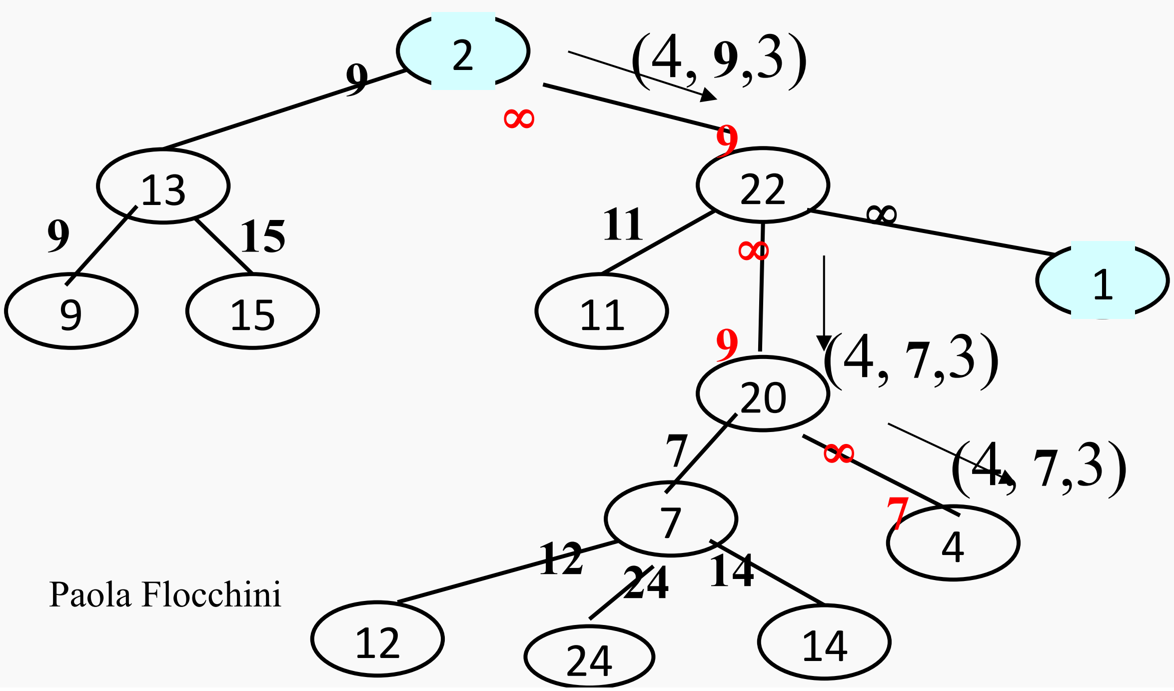


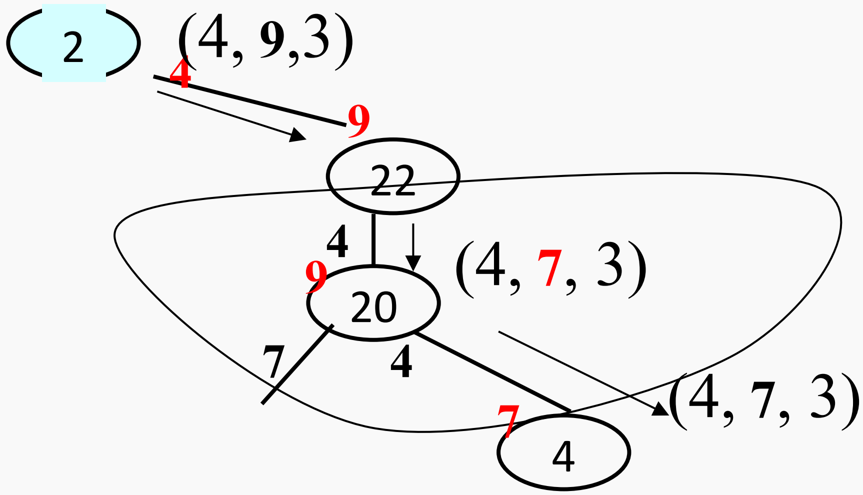

The ranked node attempts to send a ranking message to the next node to be ranked

**Second** might now be unknown, in this case the value ∞ is used

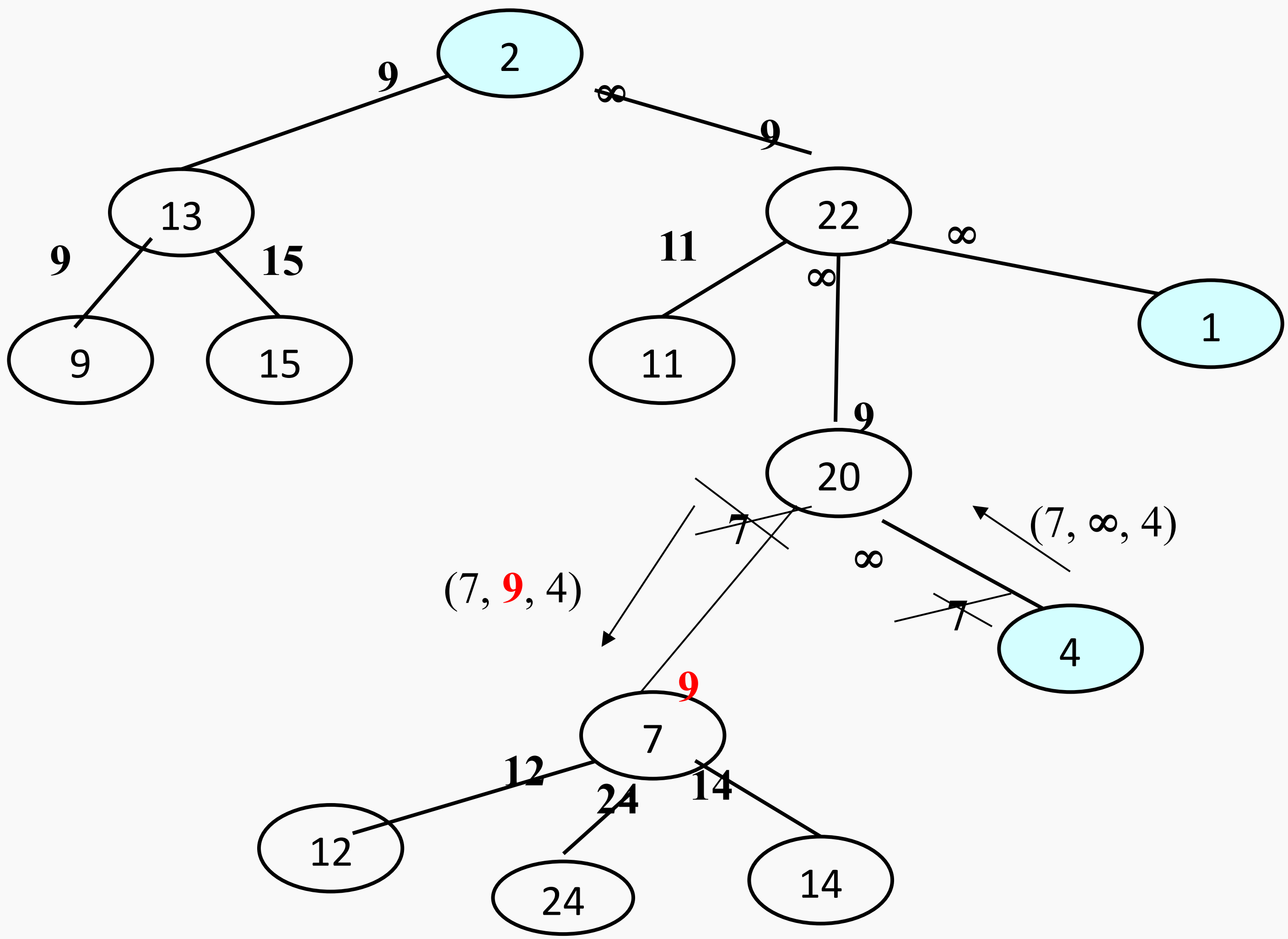
The second variable of the rank message is updated during its travel and the minimum values on the links of the tree are also updates


(2, 4,2)
(2, ∞,2)
2
13
22
9
15
11
20
1
7
4
12
24
14

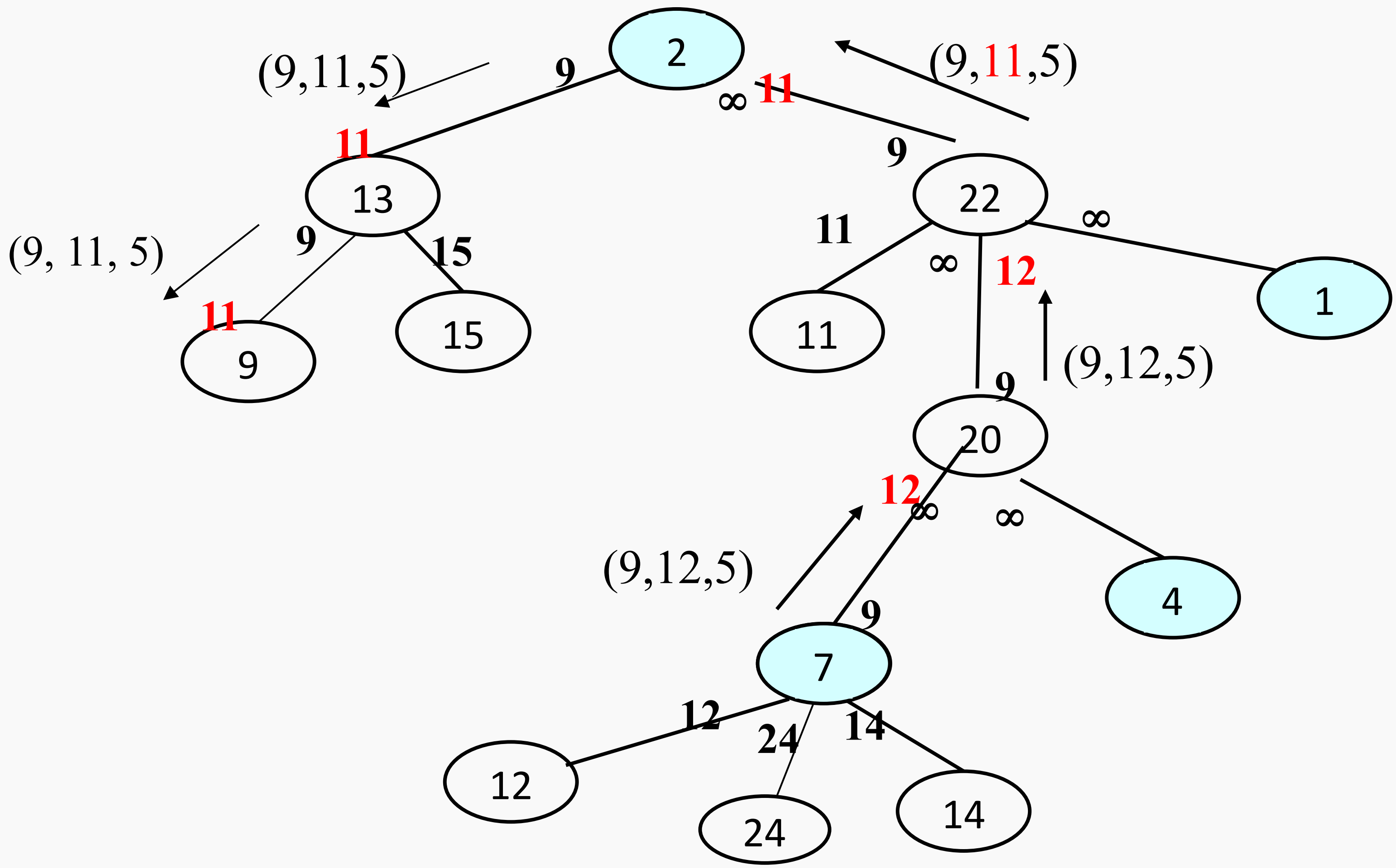

(4, 9,3)
(4, 7,3)
(4, 7,3)
2
13
22
9
15
11
1
20
7
4
12
24
14

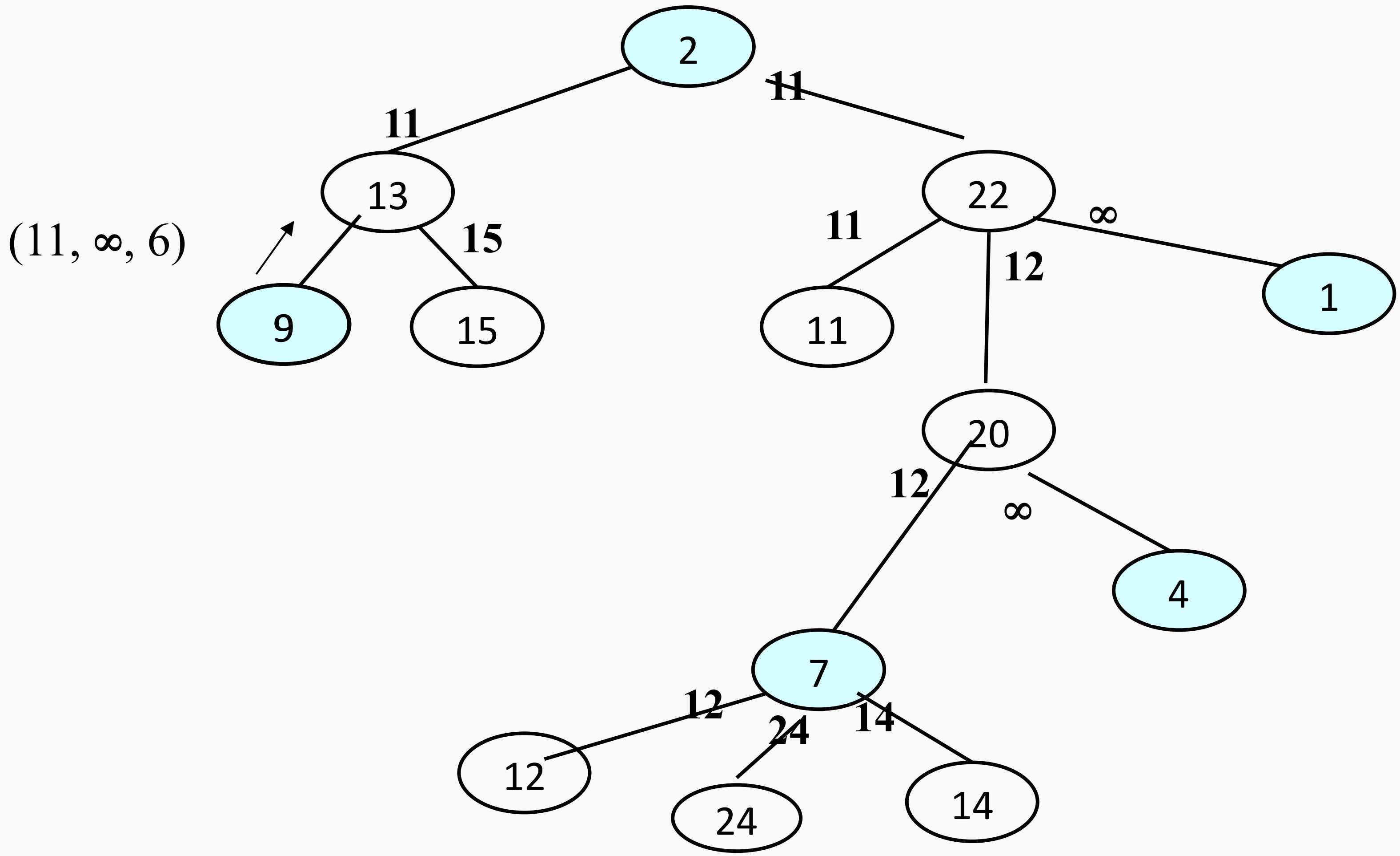



Notice the update




Paola Flocchini


2
(9,11,5)
9
∞ 11
(9,11,5)
11
9
13
22
(9, 11, 5)
9
15
11
∞
∞
12
11
1
9
15
11
(9,12,5)
9
20
12
∞
∞
(9,12,5)
9
4
7
12
24
14
12
24
14




Paola Flocchini

## Complexity: worst case