A note about termination

All entities must terminate in a **TERMINAL STATE**

In a terminal state, **no further actions are possible.**

LOCAL TERMINATION: an entity terminates when its task is completed but the other entities might still be working on solving the problem.

GLOBAL TERMINATION: an entity terminates knowing that the problem has been solved.

Leader Election

Chapter 3

Observations

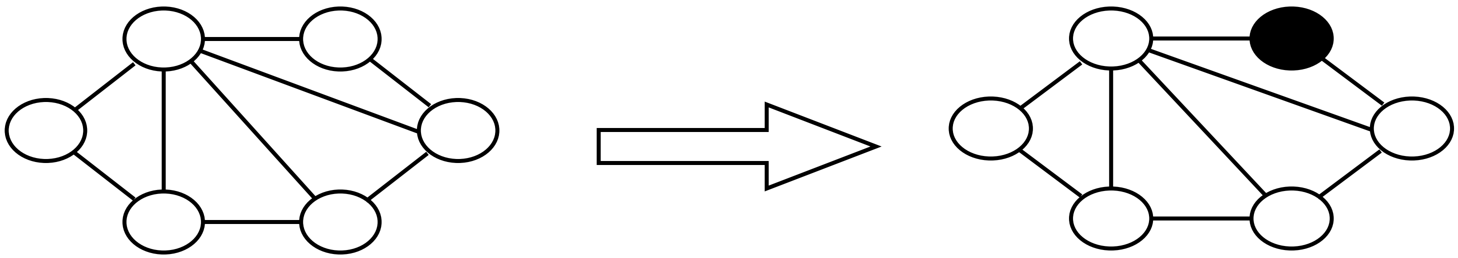
Election in the Ring

Election in the Mesh

Election in the Hypercube

Election in an arbitrary graph

Election



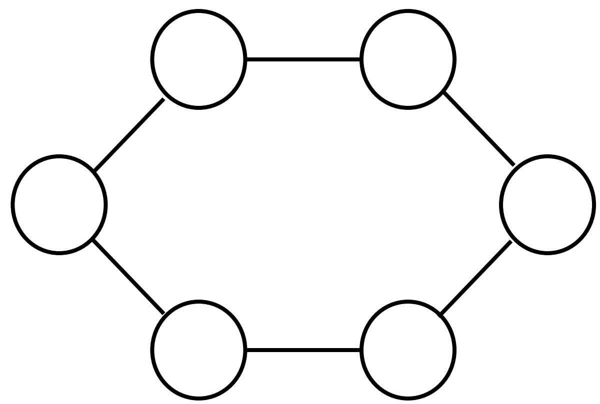
Theorem *[Angluin 80]*

The election problem cannot be generally solved if the entities do not have different identities.

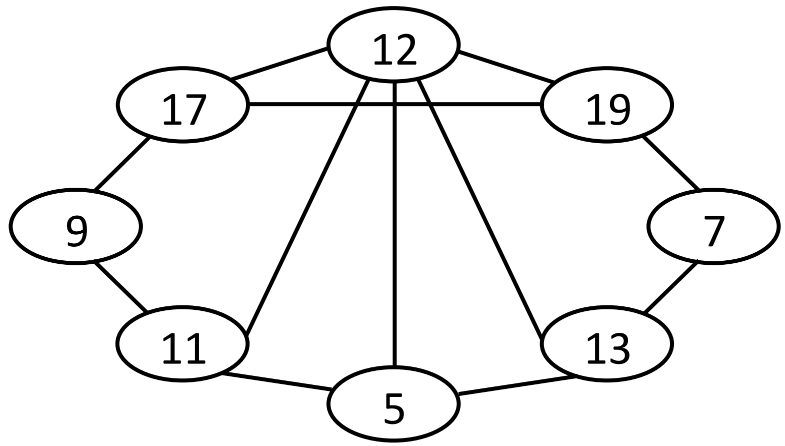
Consider the system where:

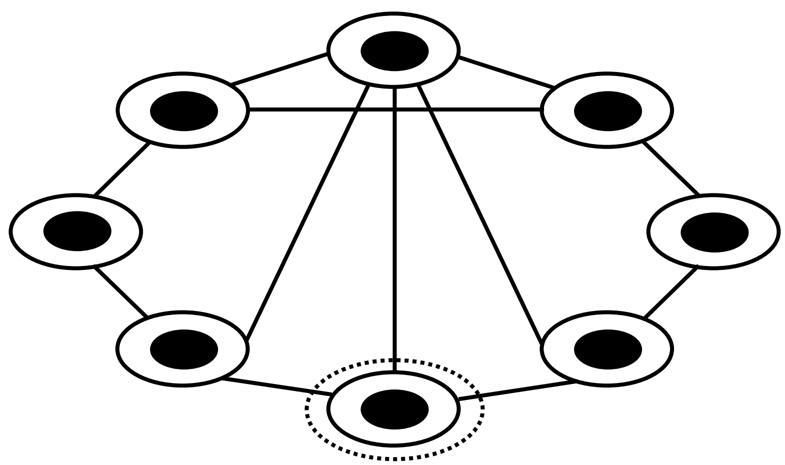
- Unique entities
- Same state
- Anonymous
- Synchronous

At each moment, they are doing the same thing.



Note: with distinct Ids Minimum Finding is an election



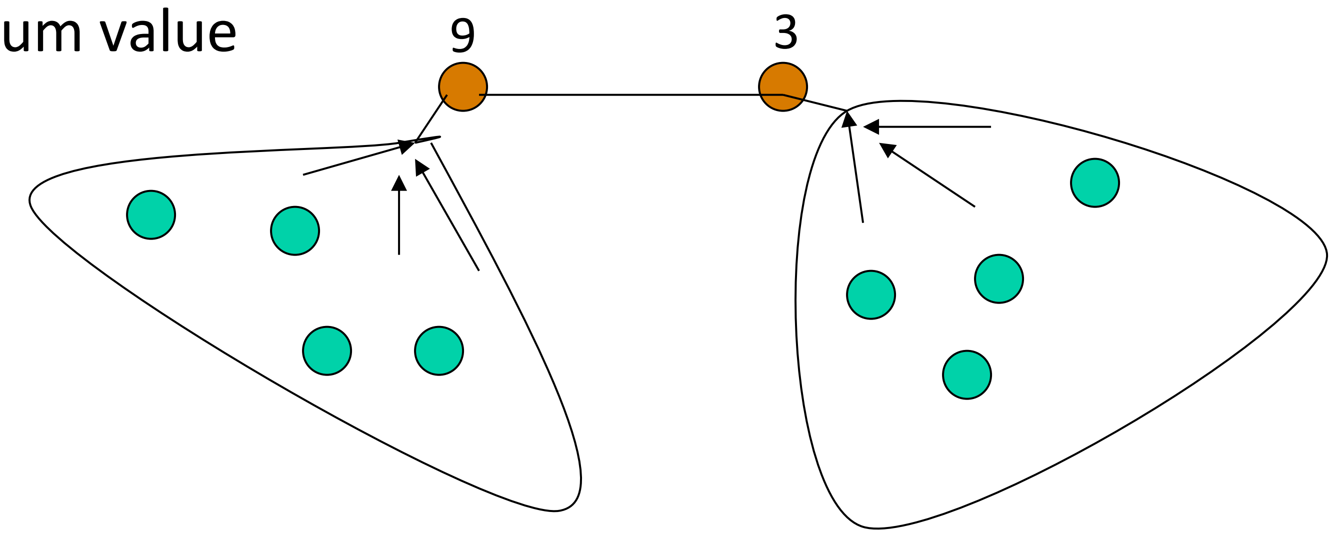


Election in the Tree

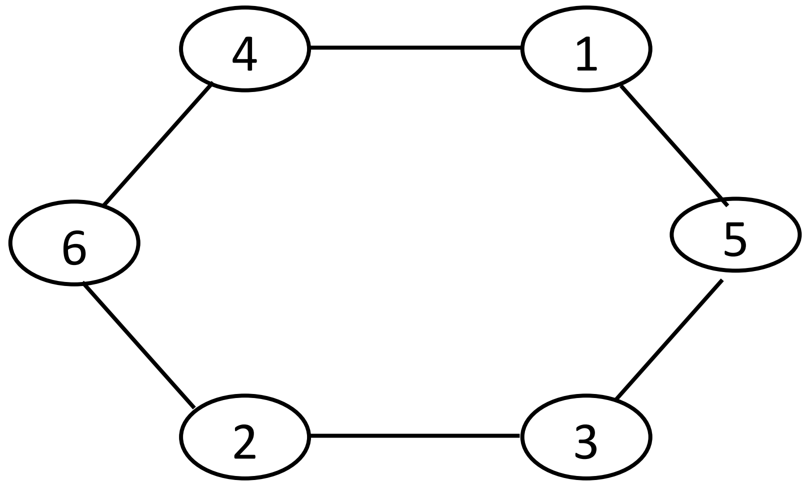
To each node x is associated a distinct identifier v(x)

A simple algorithm:

1) Execute the saturation technique,
2) Choose the saturated node holding the minimum value



Ring

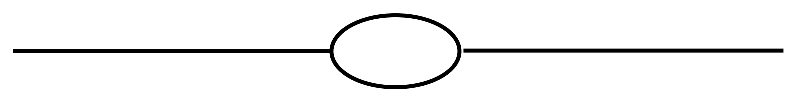


- n entities
- m= n links

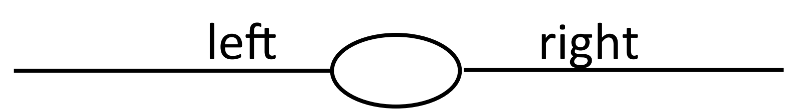
- n. of entities = n. of links

- Symmetric topology

- Each entity has two neighbors



When there is sense of direction:



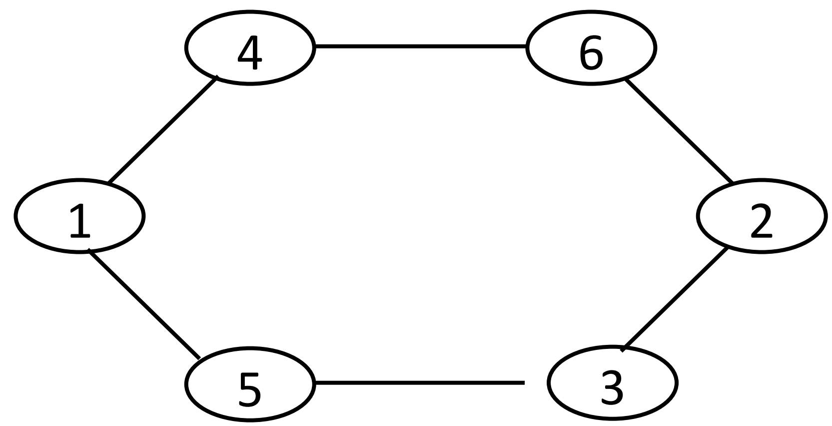
Election Algorithms in Rings

- All the way
- As Far
- Controlled distance
- Electoral stages
  --- bidirectional version
- Alternating steps

ELECTING THE MINIMUM

All the way

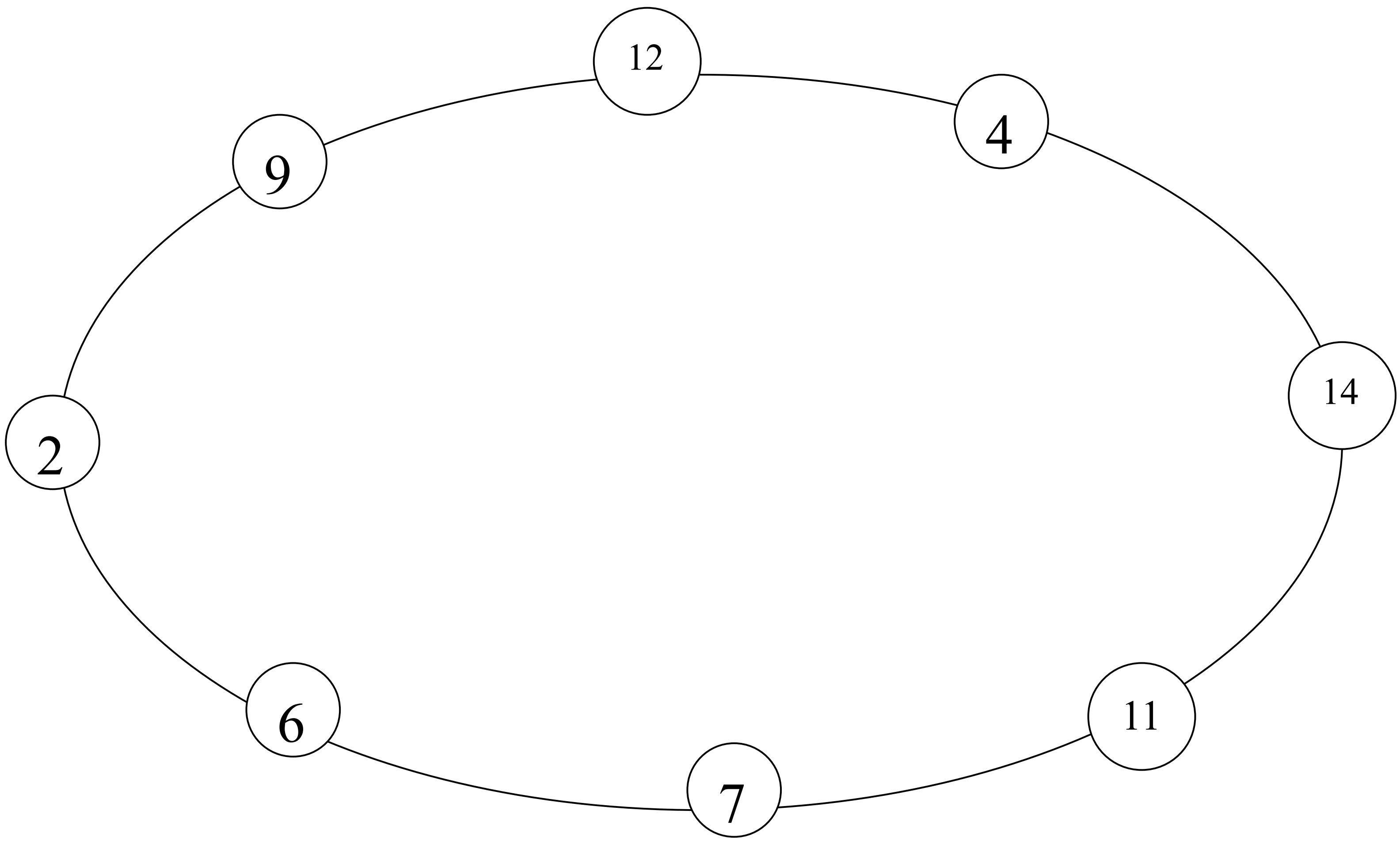
Basic Idea: Each id is fully circulated in the ring.
---> each entity sees all identities.



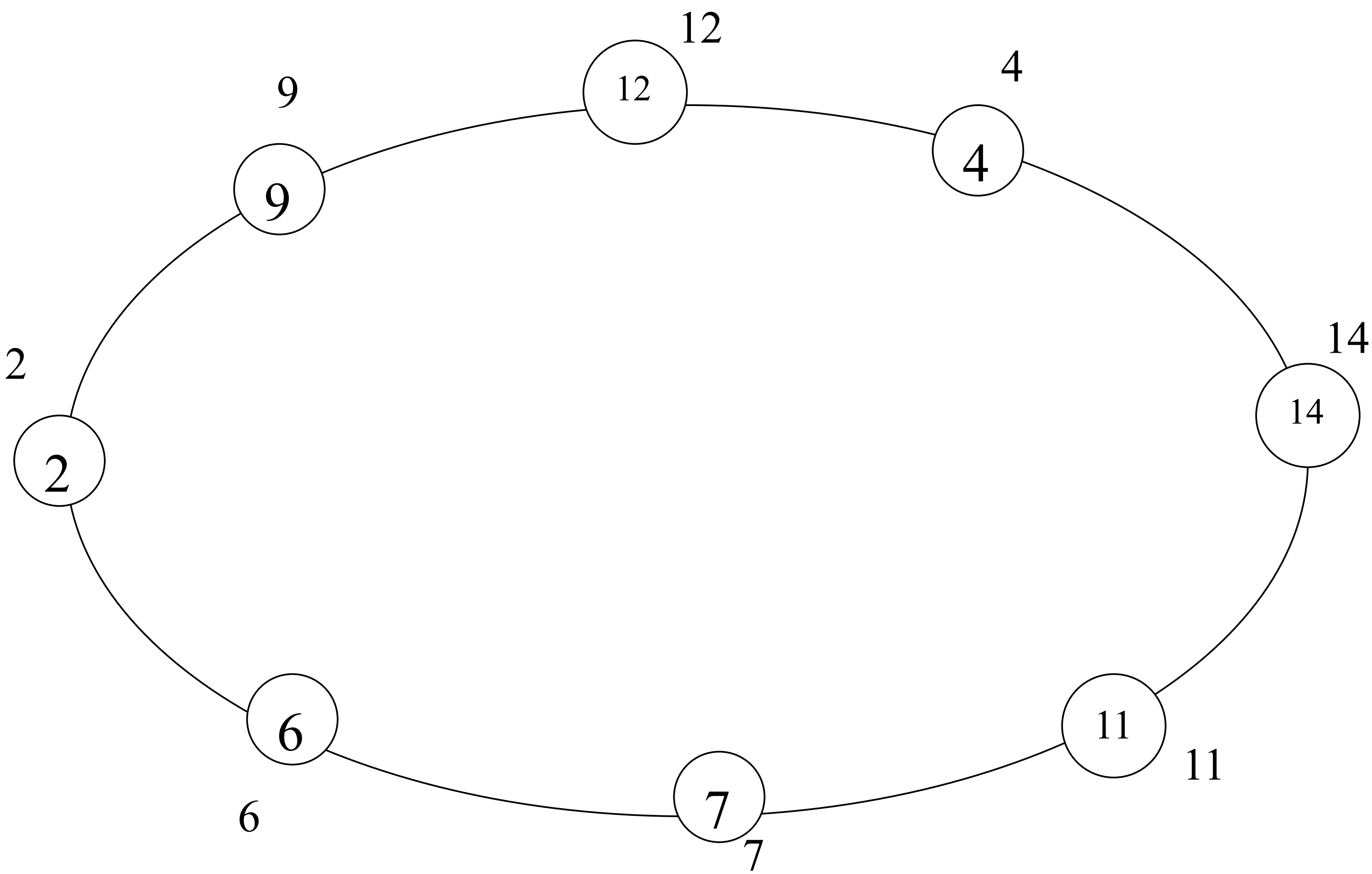
ASSUMPTIONS

- Two versions: unidirectional/bidirectional links.
- Local orientation (i.e. not necessarily a sense of direction)
- Distinct identities.

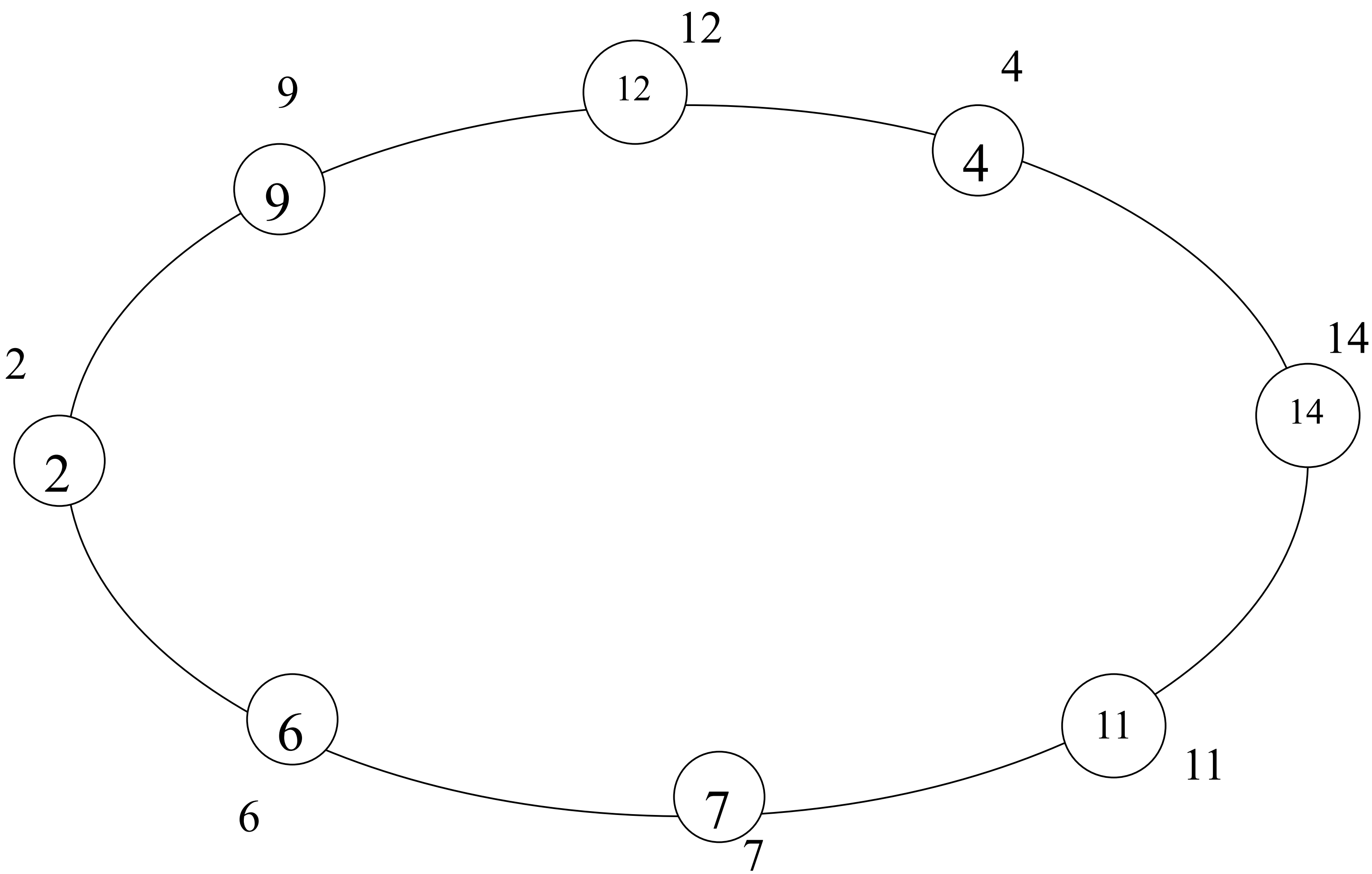
All the way



All the way



All the way



Correctness and Termination

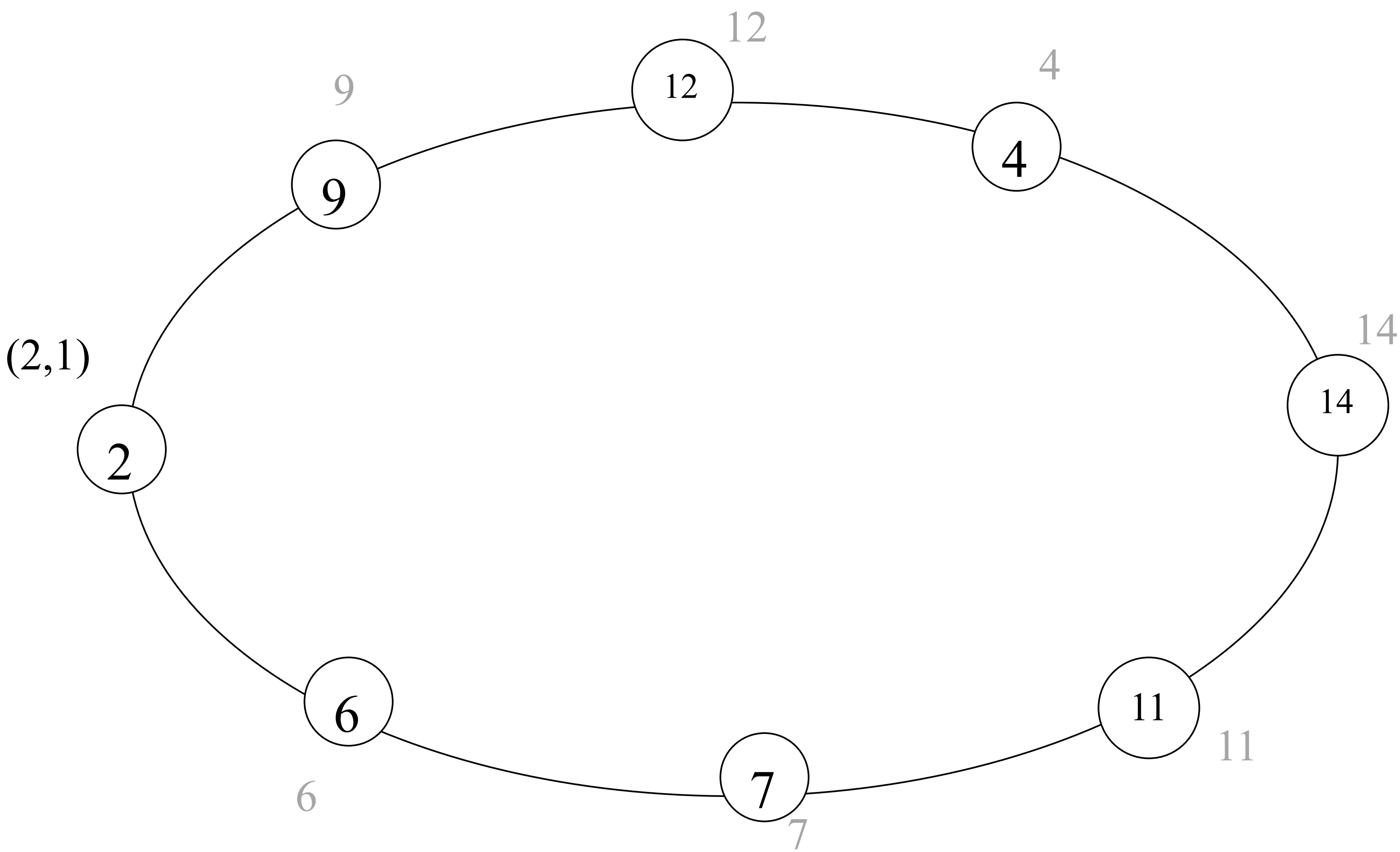
To terminate we need:

either FIFO assumption

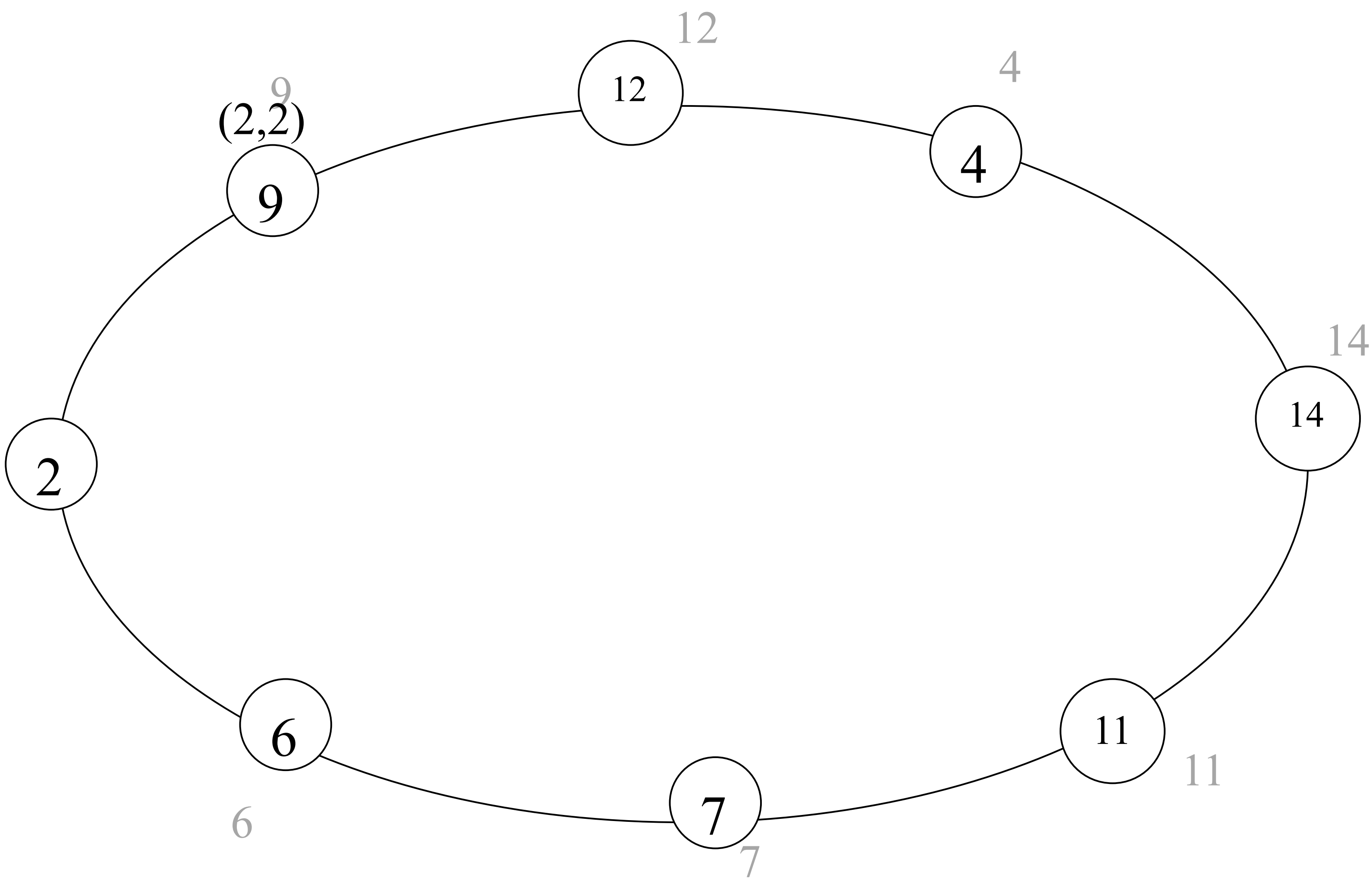
or knowledge of n

Note: knowledge of n can be acquired

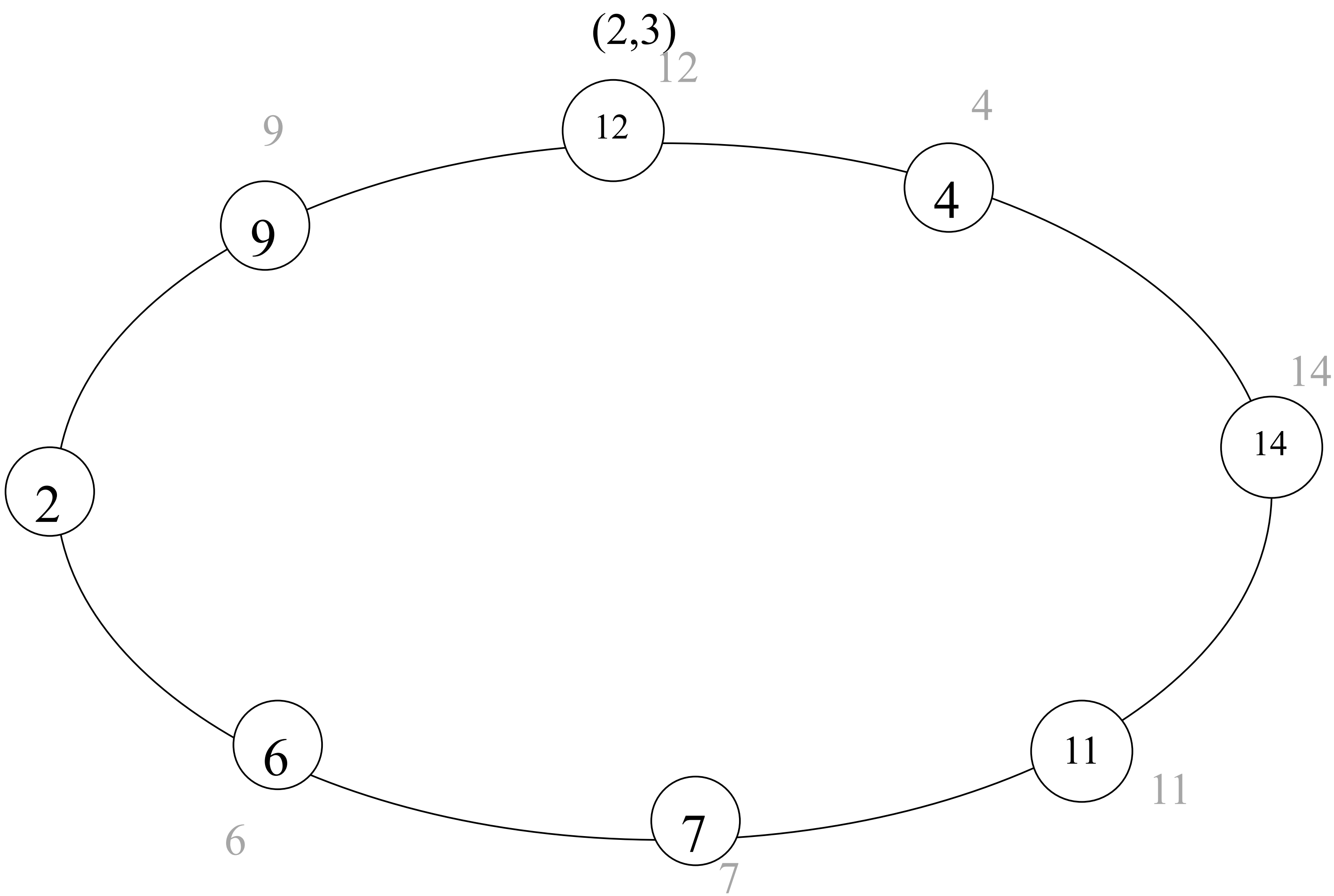
All the way



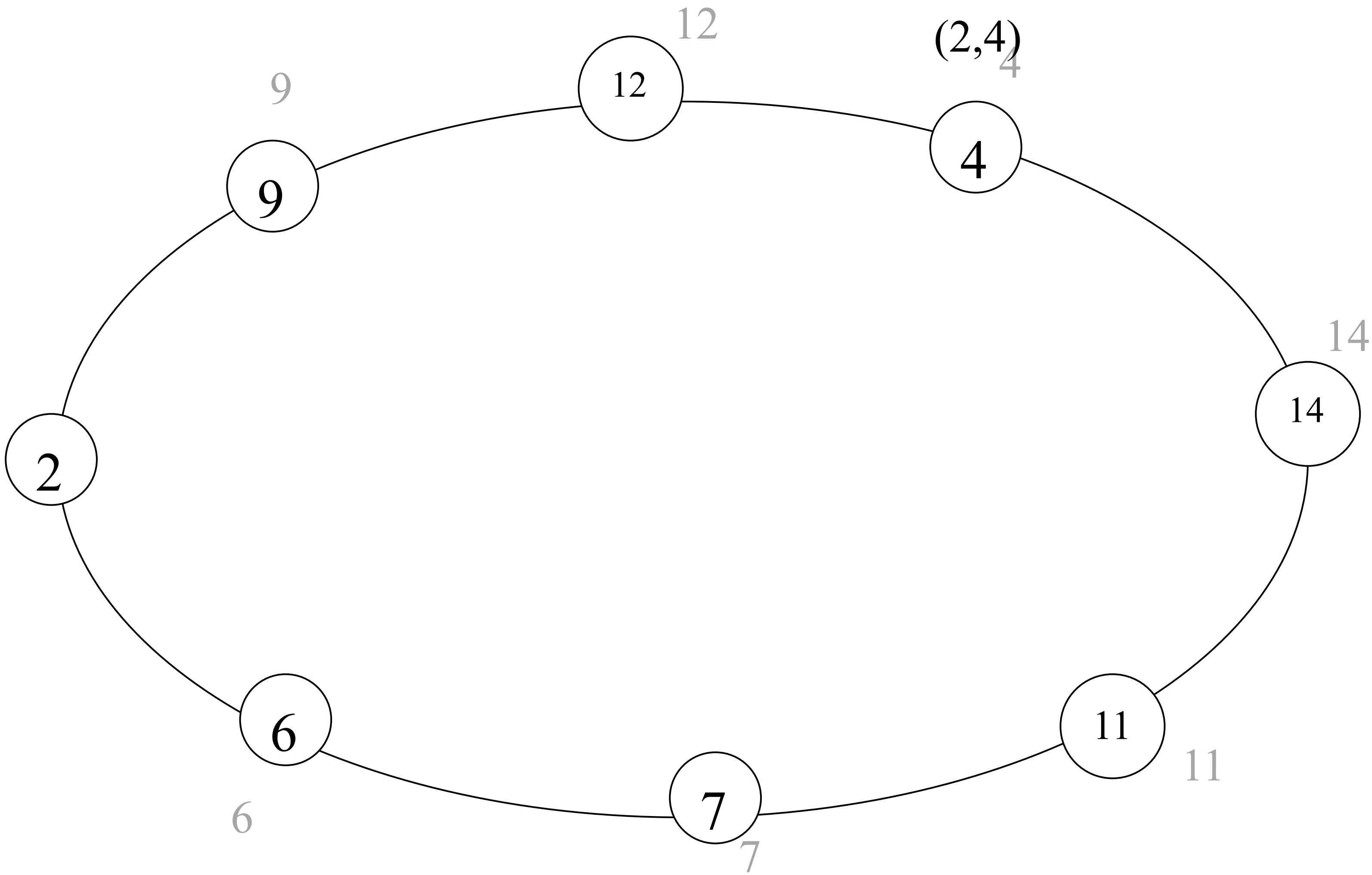
All the way



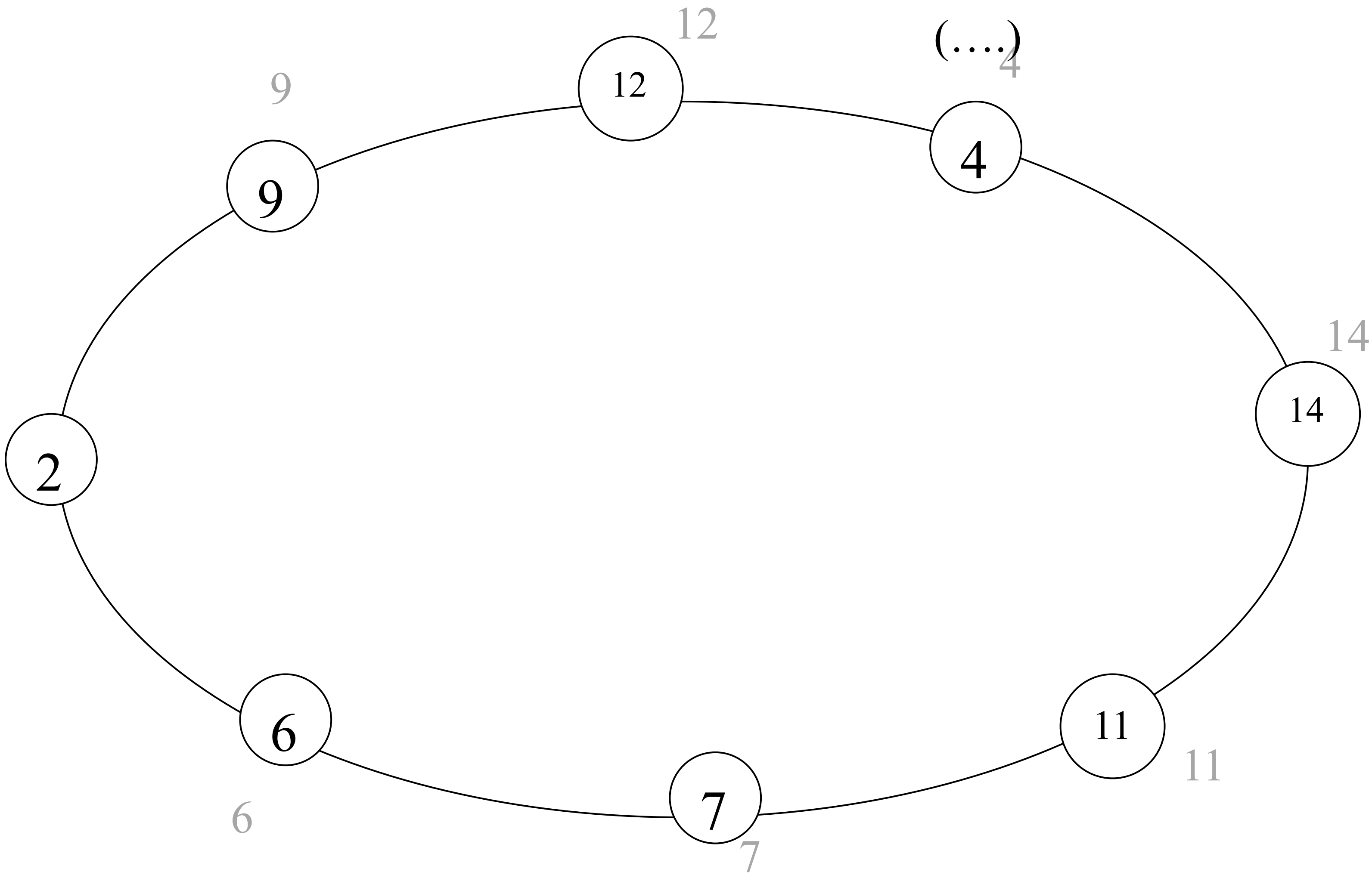
All the way



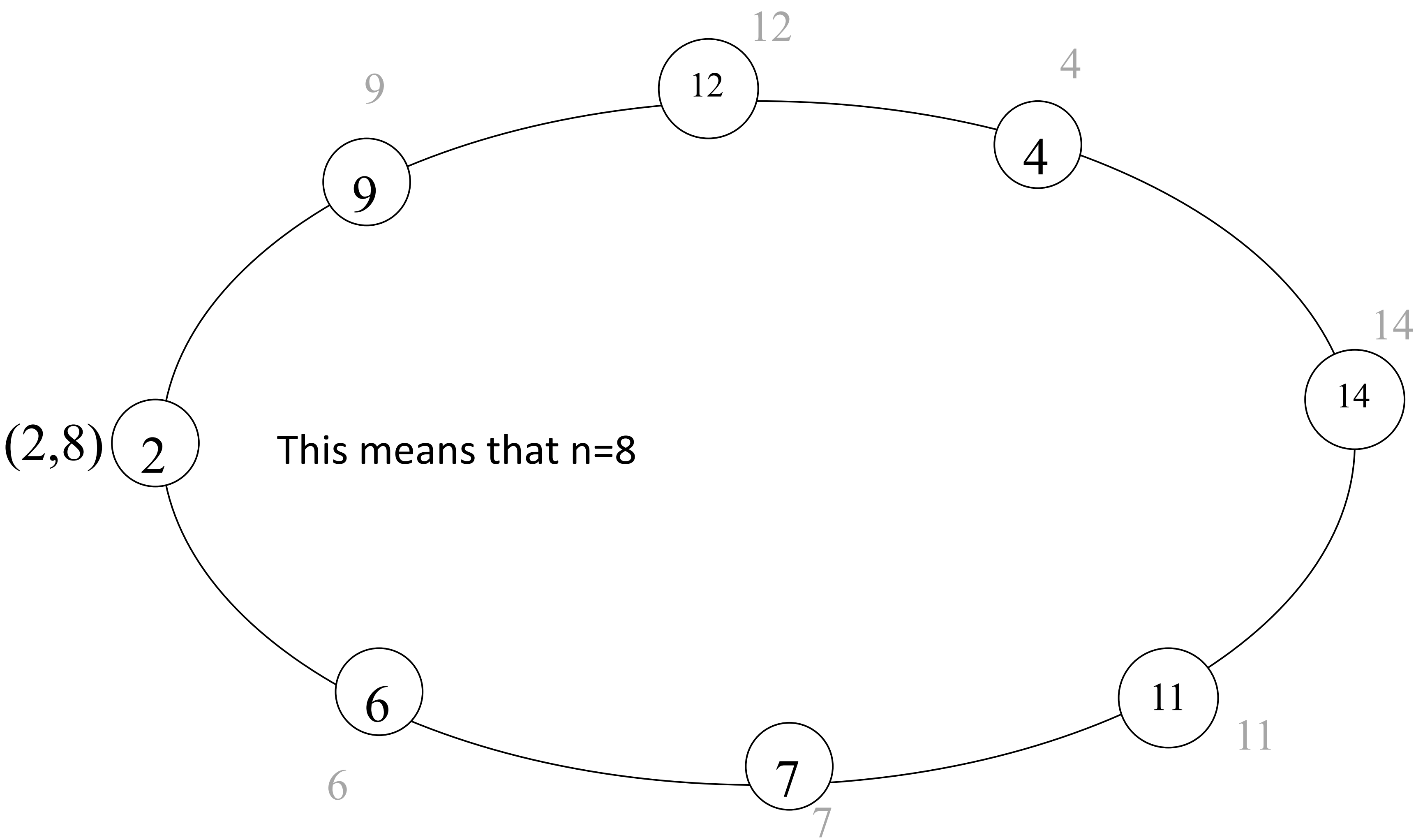
All the way



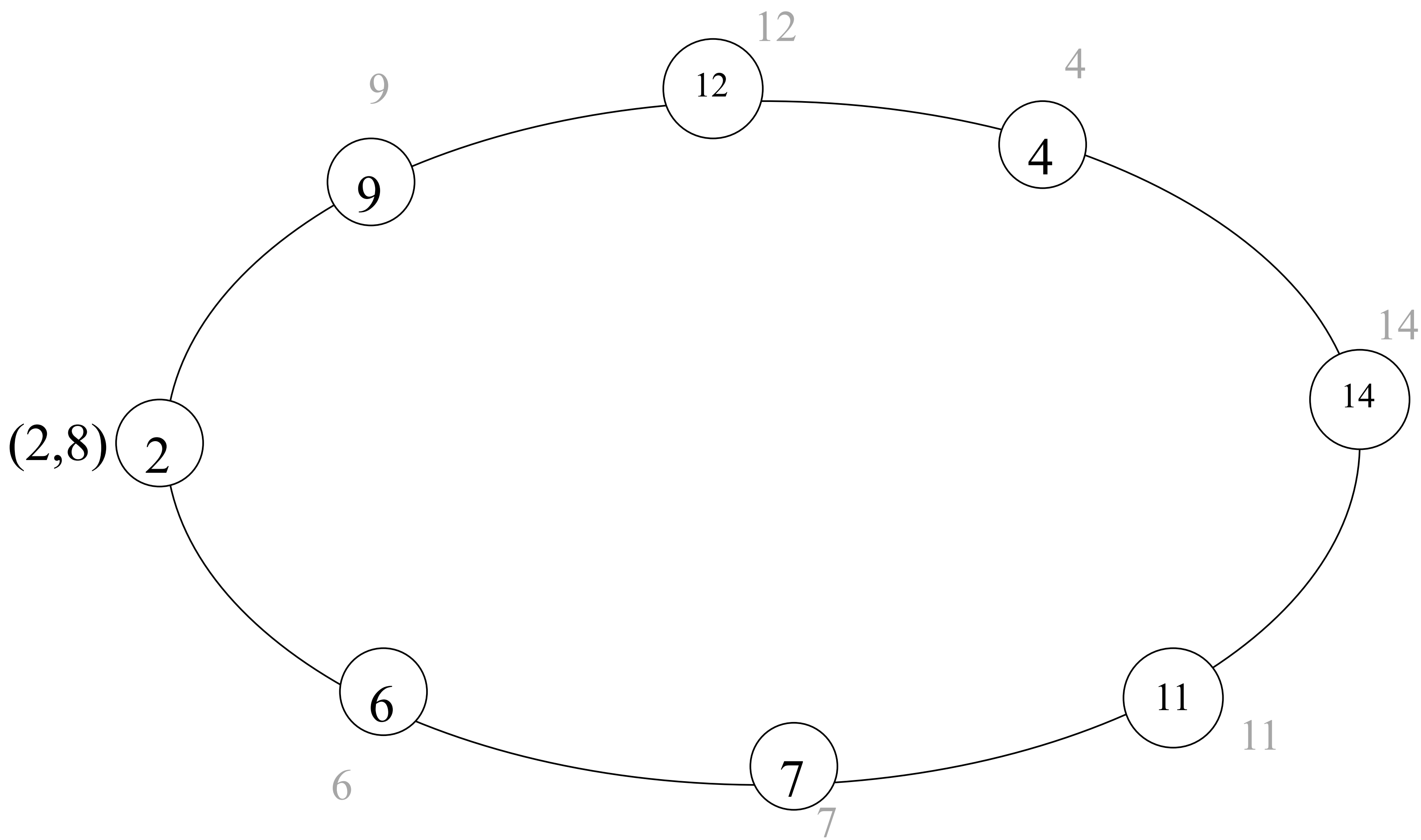
All the way



All the way



All the way



States: S={ASLEEP, AWAKE, FOLLOWER, LEADER}
S INIT={ASLEEP};
S TERM={FOLLOWER, LEADER}.

ASLEEP

Spontaneously (* just send my own Id *)
INITIALIZE
become AWAKE

Receiving(``Election'', value, counter)
INITIALIZE;
send(``Election'', value, counter+1) **to other** (* send my own Id and the Id just received*)
min:= Min{min, value}
count:= count+1
become AWAKE

<u>INITIALIZE</u>

count:= 0
size:= 1
known:= false
send("Election",id(x),size) **to right**;
min:= id(x)

AWAKE

Receiving ("Election", value, counter)

```
If value ≠ id(x) then          (* this is not my Id *)
   send ("Election",value,counter+1)  to  other
   min:= MIN{min,value}
   count:= count+1
   if  known = true   then      (* I already received my Id and
                 CHECK            I am now checking if
                                  I am the leader or not *)
   endif
else   (* this is my Id *)
          ringsize:= counter
          known:=   true
          CHECK
endif
```

<u>CHECK</u>

```
if count = ringsize then
          if  min = id(x) then
                    become LEADER
          else
                    become  FOLLOWER
          endif
endif
```

Complexity

Each identity crosses each link --> n^2

The size of each message is log(*id*)

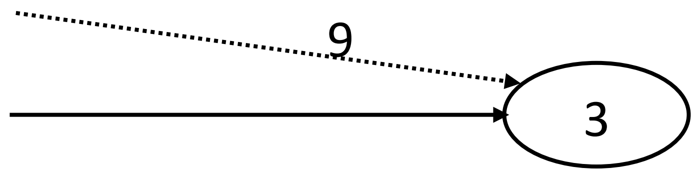
$O(n^2)$ messages
$O(n^2 \log (MaxId))$ bits

Observations:

1. The algorithm also solves the data collection problem.
2. It also works for unidirectional/bidirectional.

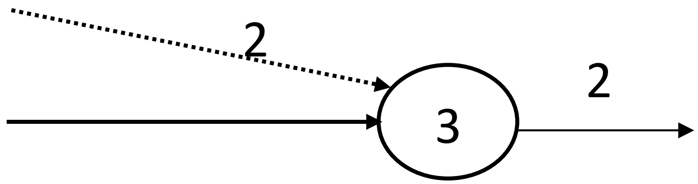
AsFar (as it can)

Basic Idea: It is not necessary to send and receive messages with larger *id*' s than the *id*' s that have already been seen.

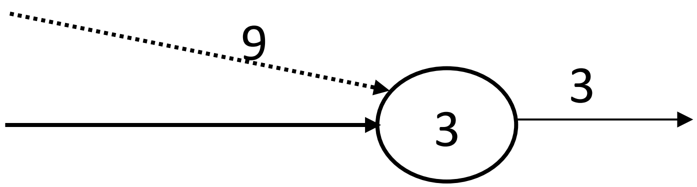


ASSUMPTIONS

- Unidirectional/bidirectional ring
- Different *id*' s
- Local orientation

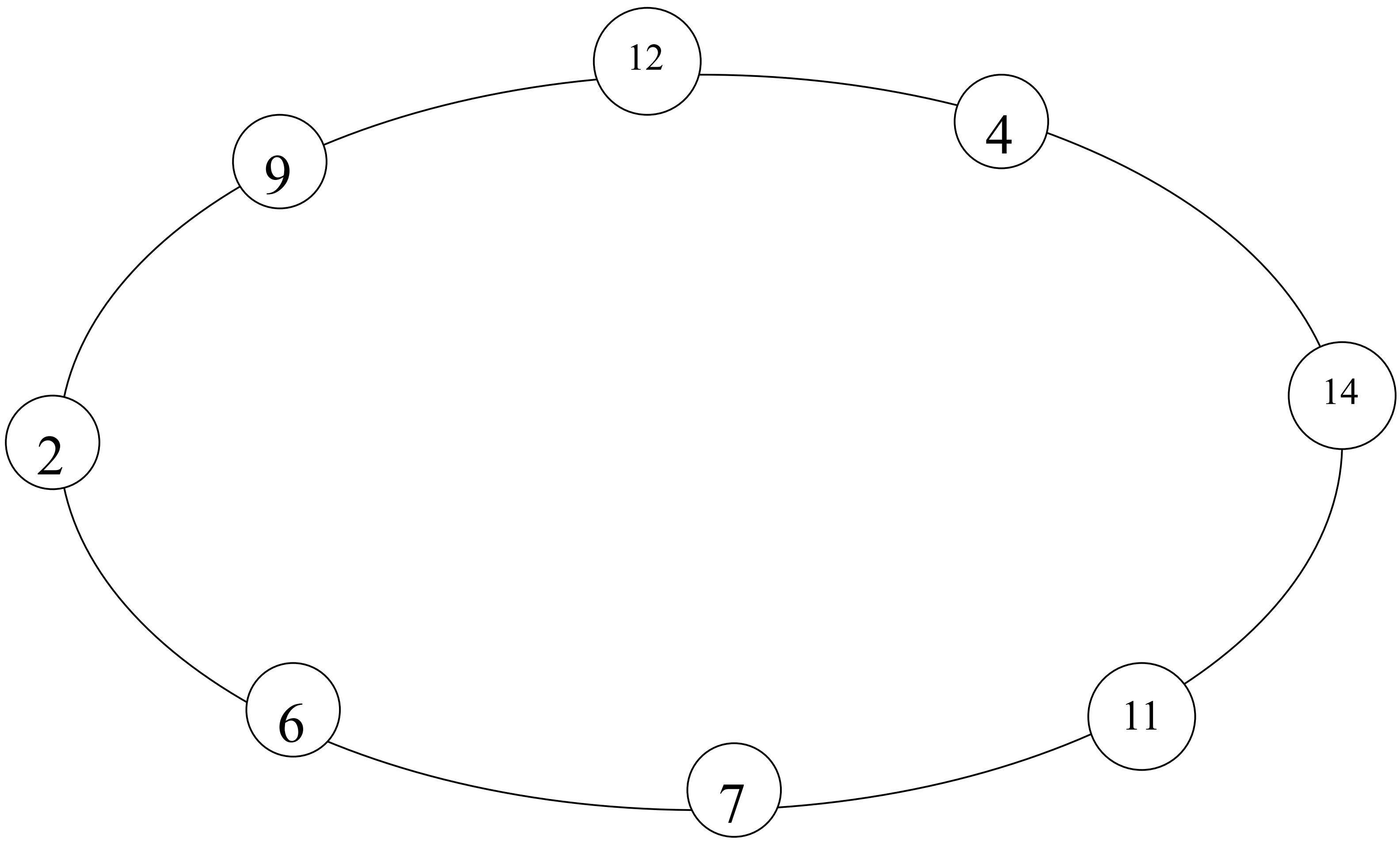


Receiving y smaller-than me
send(y) to other neighbour

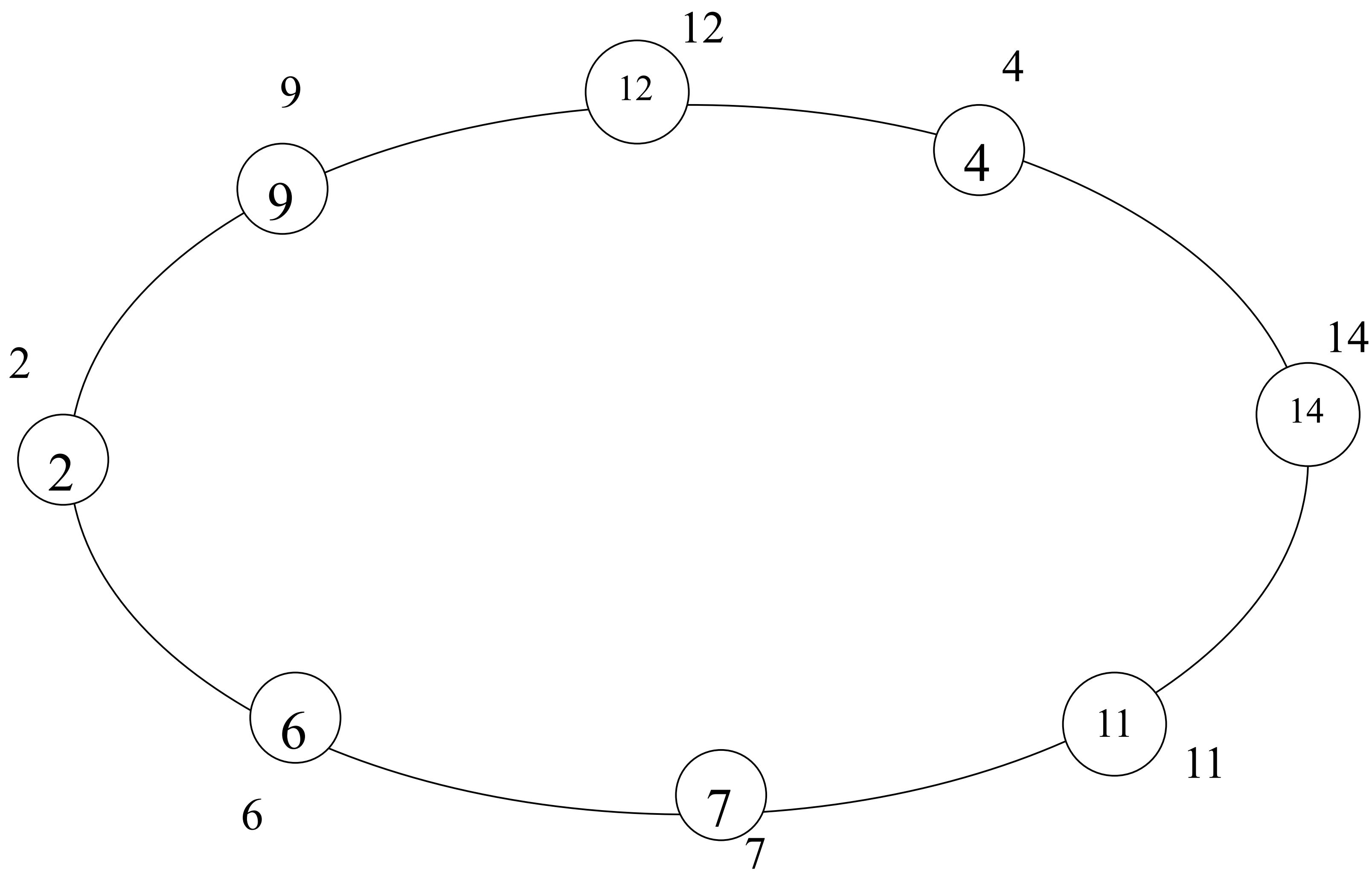


Receiving y bigger-than me
send(x) to other neighbour
(if not sent already)

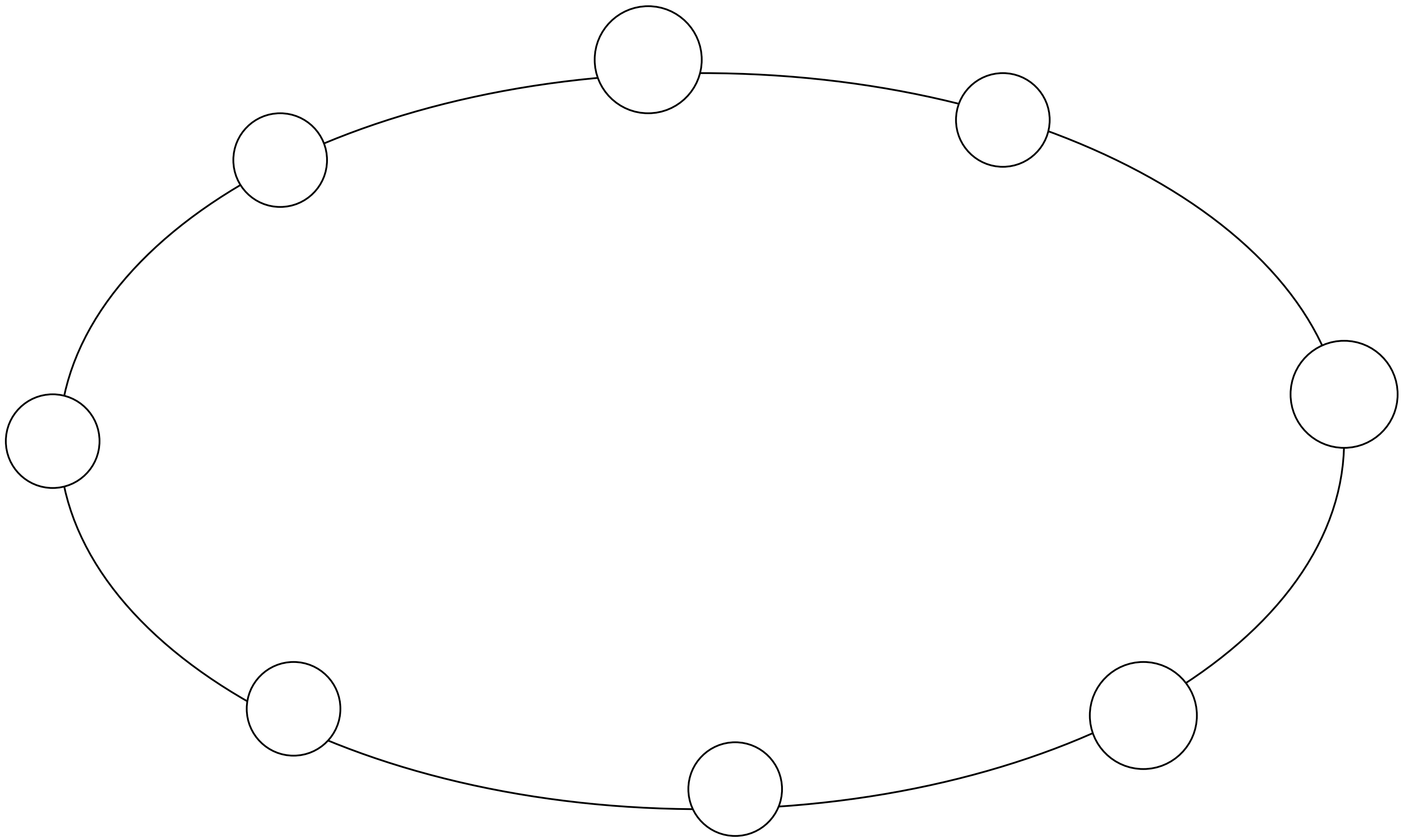
As Far



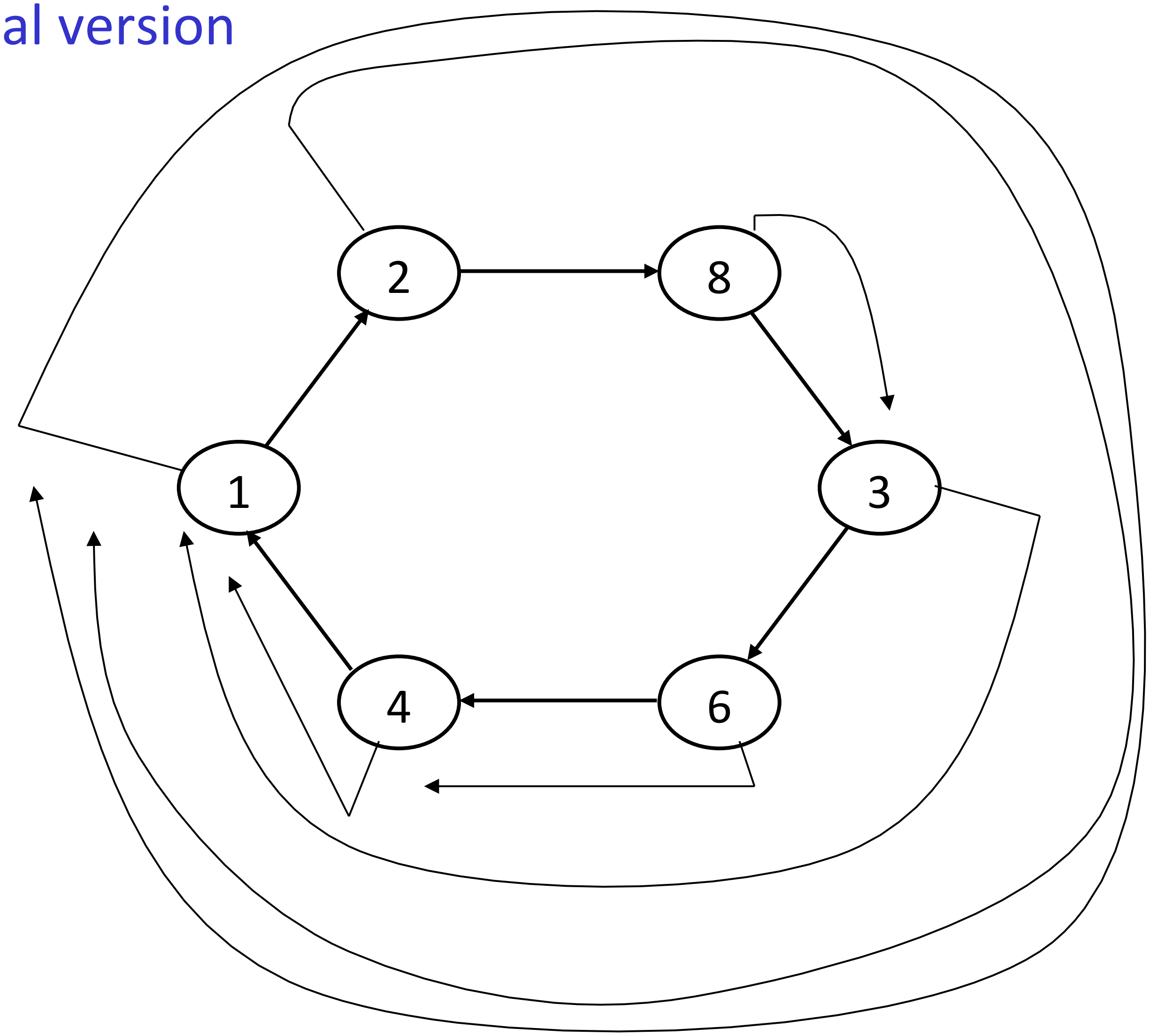
As Far



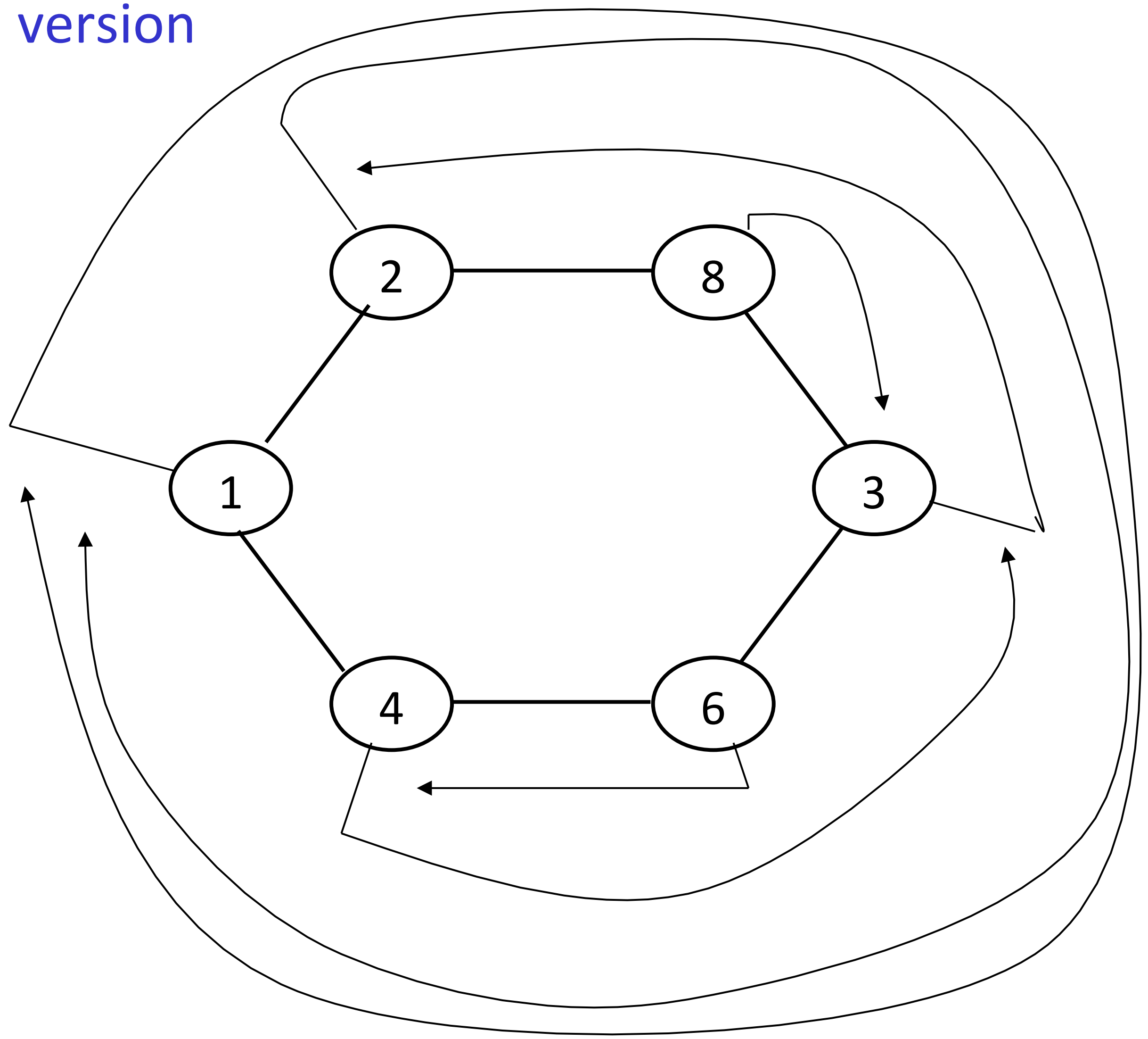
As Far



unidirectional version



bidirectional version



States: S={ASLEEP, AWAKE, FOLLOWER, LEADER}
S_INIT={ASLEEP}
S_TERM={FOLLOWER, LEADER}

--- unidirectional version

ASLEEP

Spontaneously

send("Election",id(x)) **to right**
min:= id(x)
become AWAKE

Receiving("Election", value)

send("Election",id(x)) **to right**
min:= id(x)
If value < min then
send("Election", value) **to other**
min:= value
endif
become AWAKE

(* this could be avoided if id(x)>value *)

AWAKE

```
Receiving("Election '", value)
        if  value < min then                              (*  send only if smaller*)
                send("Election", value)  to other
                min:= value
        else
          If value = min then NOTIFY endif               (*  If I receive my own Id,
        endif                                                 I am the leader *)

Receiving(Notify)
        send(Notify)  to other
        become  FOLLOWER
```

<u>NOTIFY</u>

```
        send(Notify) to  right
        become LEADER
```

Correctness and **Termination**

The leader knows it is the leader when it receives
its message back.

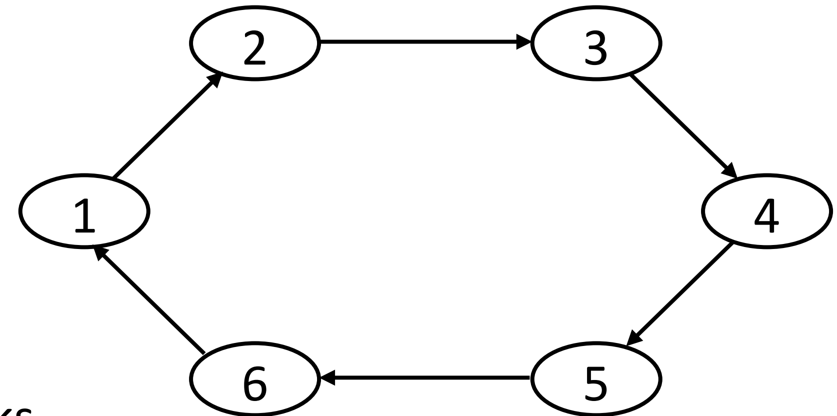
When do the other know ?

Notification is necessary !

Observations:

- Bidirectional version

Worst-Case Complexity (Unidirectional Version)



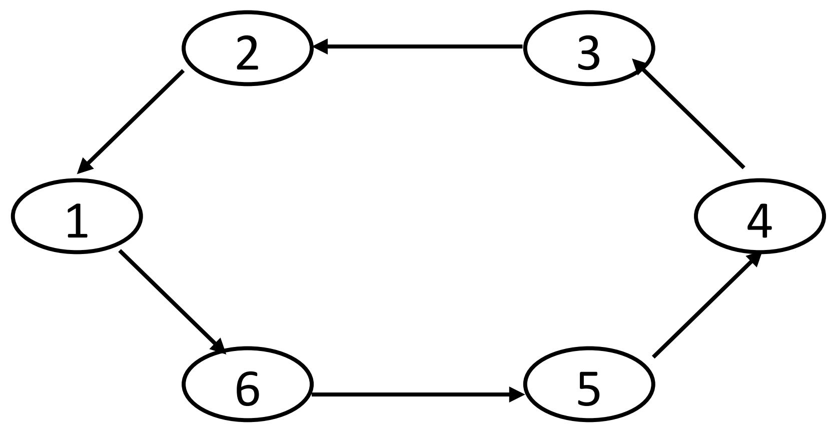
1 ---> n		links
2 ---> n - 1		links
3 ---> n - 2		links
…	…	…
n ---> 1 link		

$$n + (n - 1) + (n - 2) + \dots + 1 = \sum_{i=1}^{n} i = (n+1)(n)/2$$

Total: n (n+ 1) / 2 + n = **$O(n^2)$**

Last n: notification

Best-Case Complexity (Unidirectional Version)



1 ---> n links

for all i ≠ 1 ---> 1 link (--> total = n - 1)

Total: n + (n - 1) + n = **O(n)**

Last n: notification

Average-Case Complexity

Entities are ordered in an equiprobable manner.

J-th smallest id - crosses (n / J) links

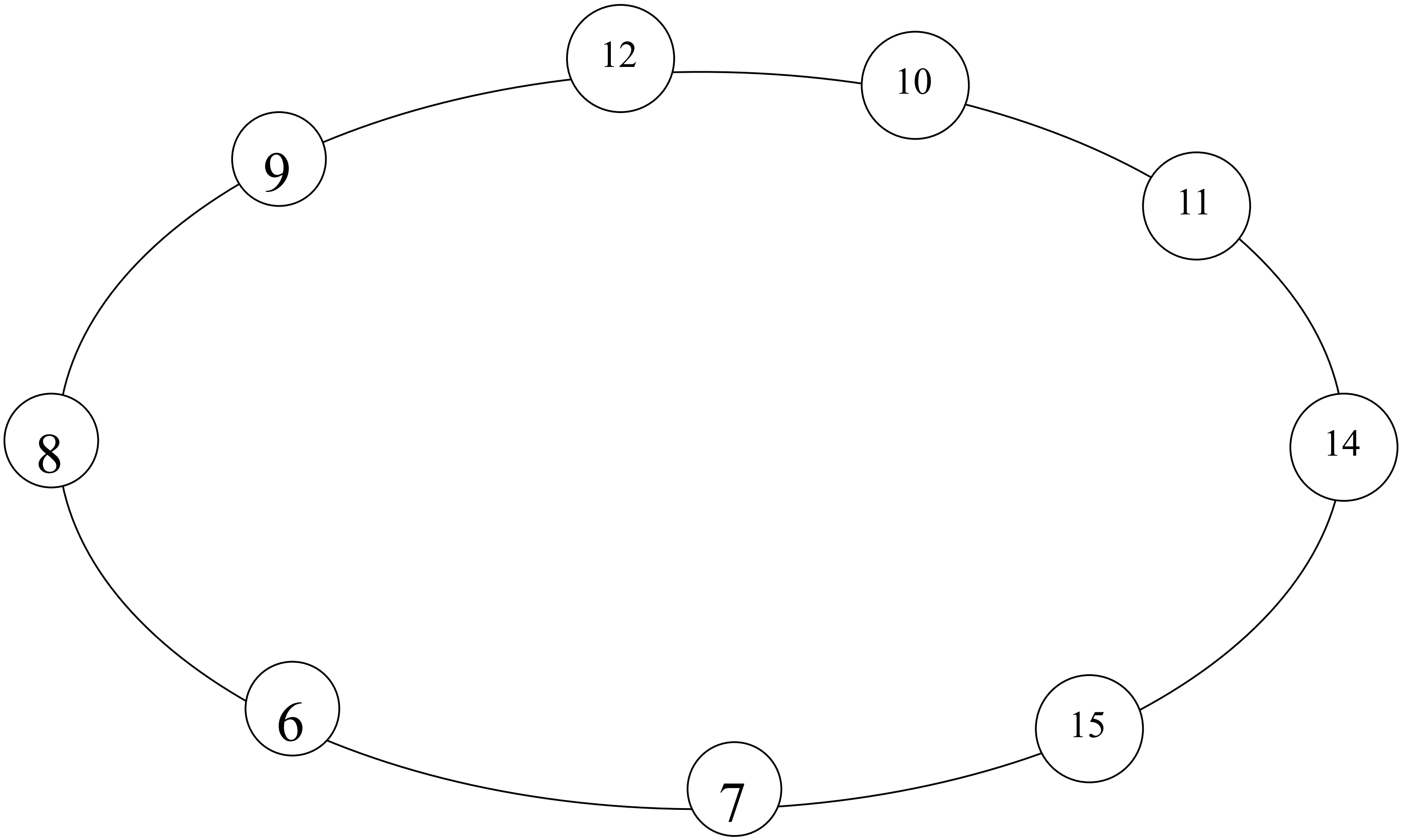
$$\sum_{J=1}^{n} (n / J) = n * Hn$$

Harmonic series of n numbers

(approx. 0.69 log n)

Total: n * Hn + n = 0.69 n log n + O(n) = **O(n log n)**

As Far



Controlled Distance

Basic idea: Operate in stages. An entity maintains control on its own message.

ASSUMPTIONS

- Bidirectional ring
- Different *ids*
- Local orientation

sense of direction only for simplicity - not needed

Ingredients

1) Limited distance (**to avoid big msgs to travel too much**)

Ex: stage i: distance 2^{i-1}

2) Return messages (**if seen something smaller does not continue**)

3) Check both sides

4) Smallest always win (regardless of stage number)

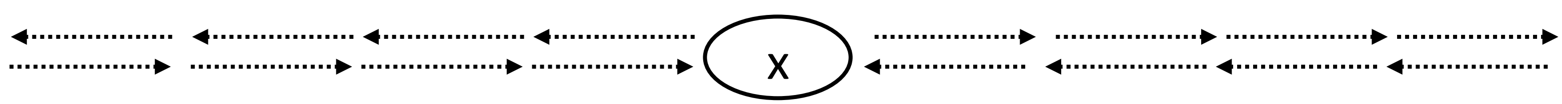
Candidate entities begin the algorithm.

Stage i:

- Each *candidate* entity sends a message with its own *id* in both directions

- the msg will travel until it encounters a smaller Id or reaches a certain distance

- If a msg does not encounters a smaller Id, it will return back to the originator



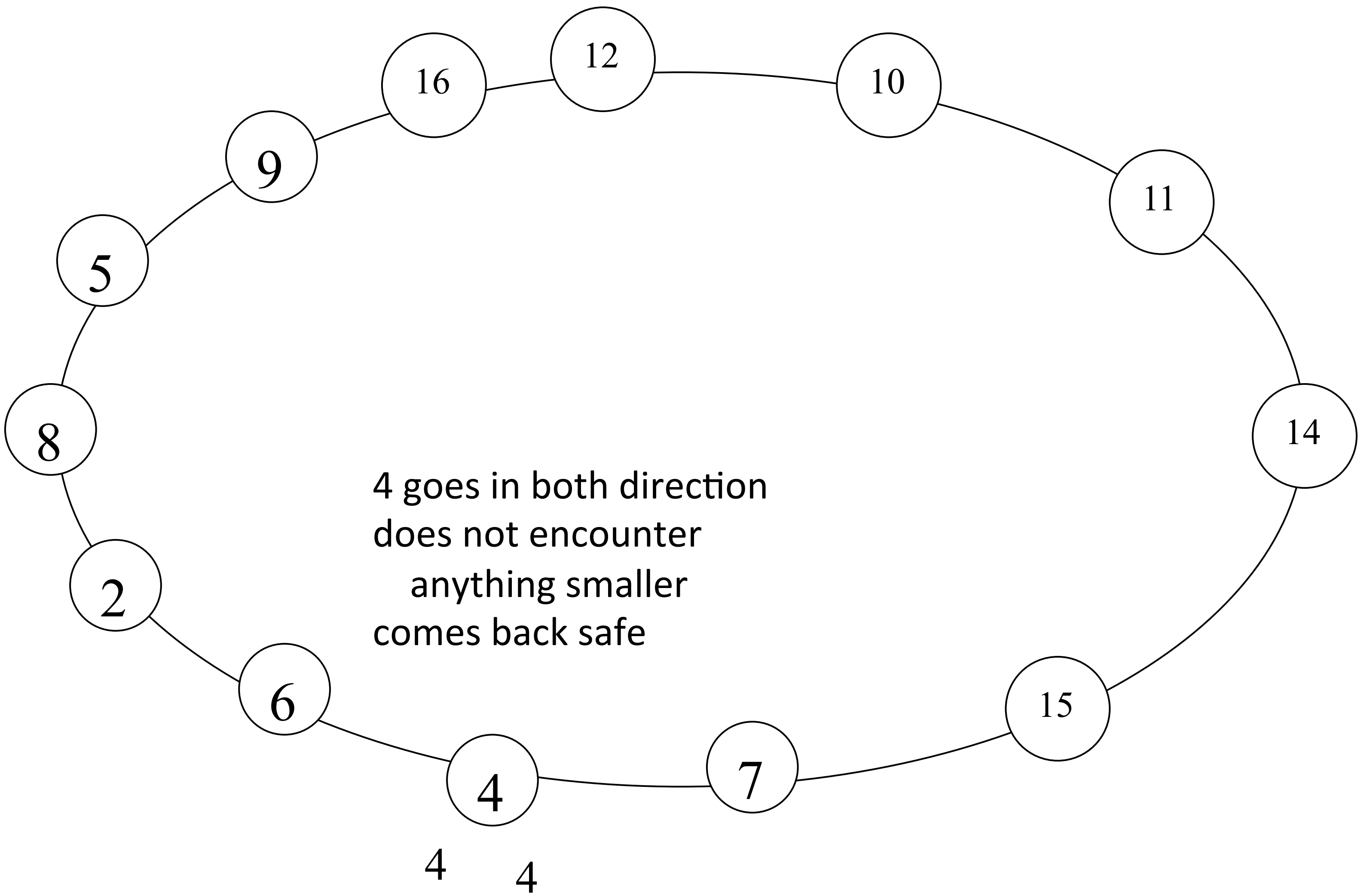
- A candidate receiving its own msg back from both directions survives and start the next stage

Entities encountered along the path read the message and:

- Each entity *i* with a greater identity Id_i becomes defeated (passive).

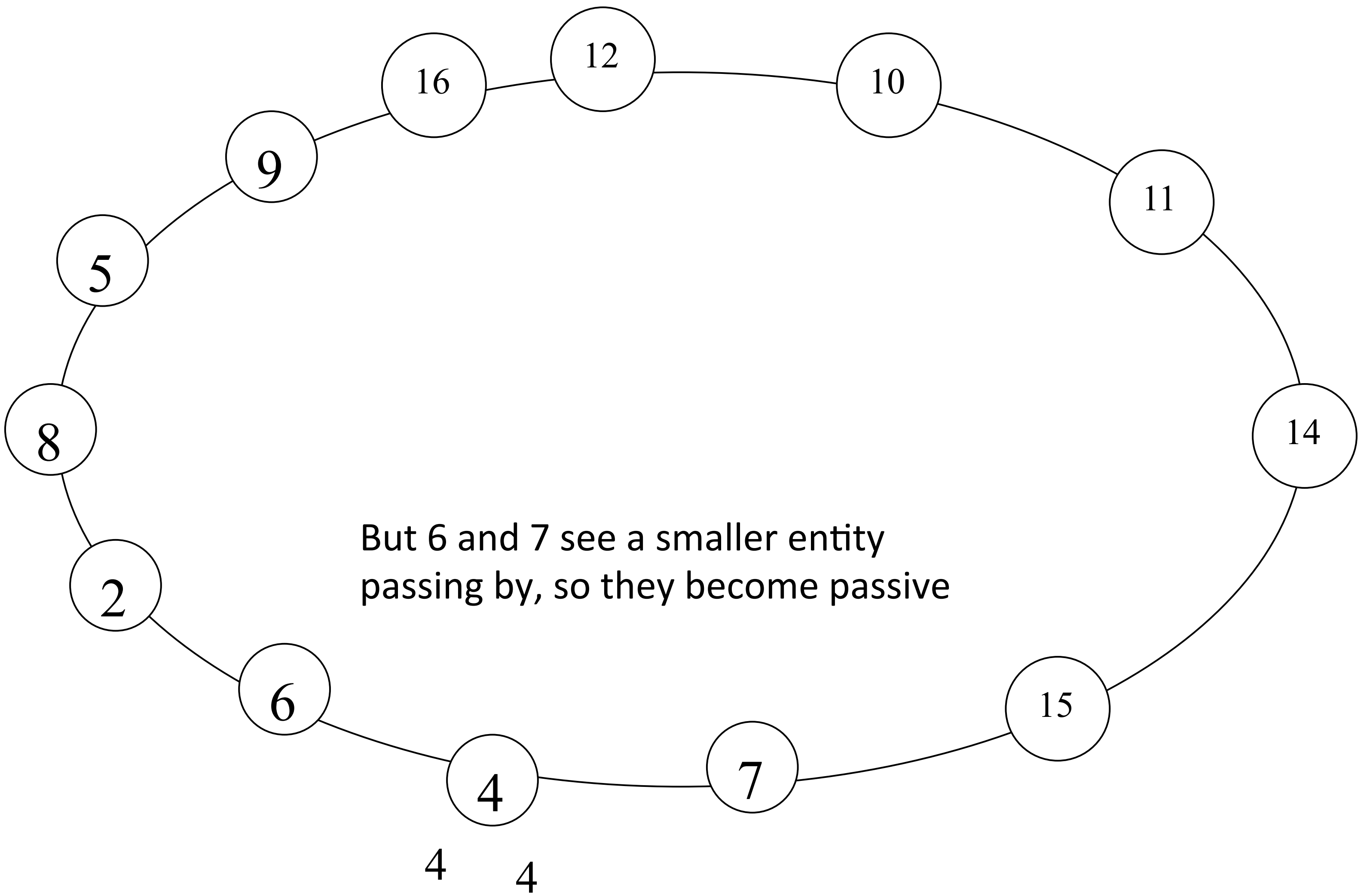
- A defeated entity forwards the messages originating from other entities, if the message is a notification of termination, it terminates

Example



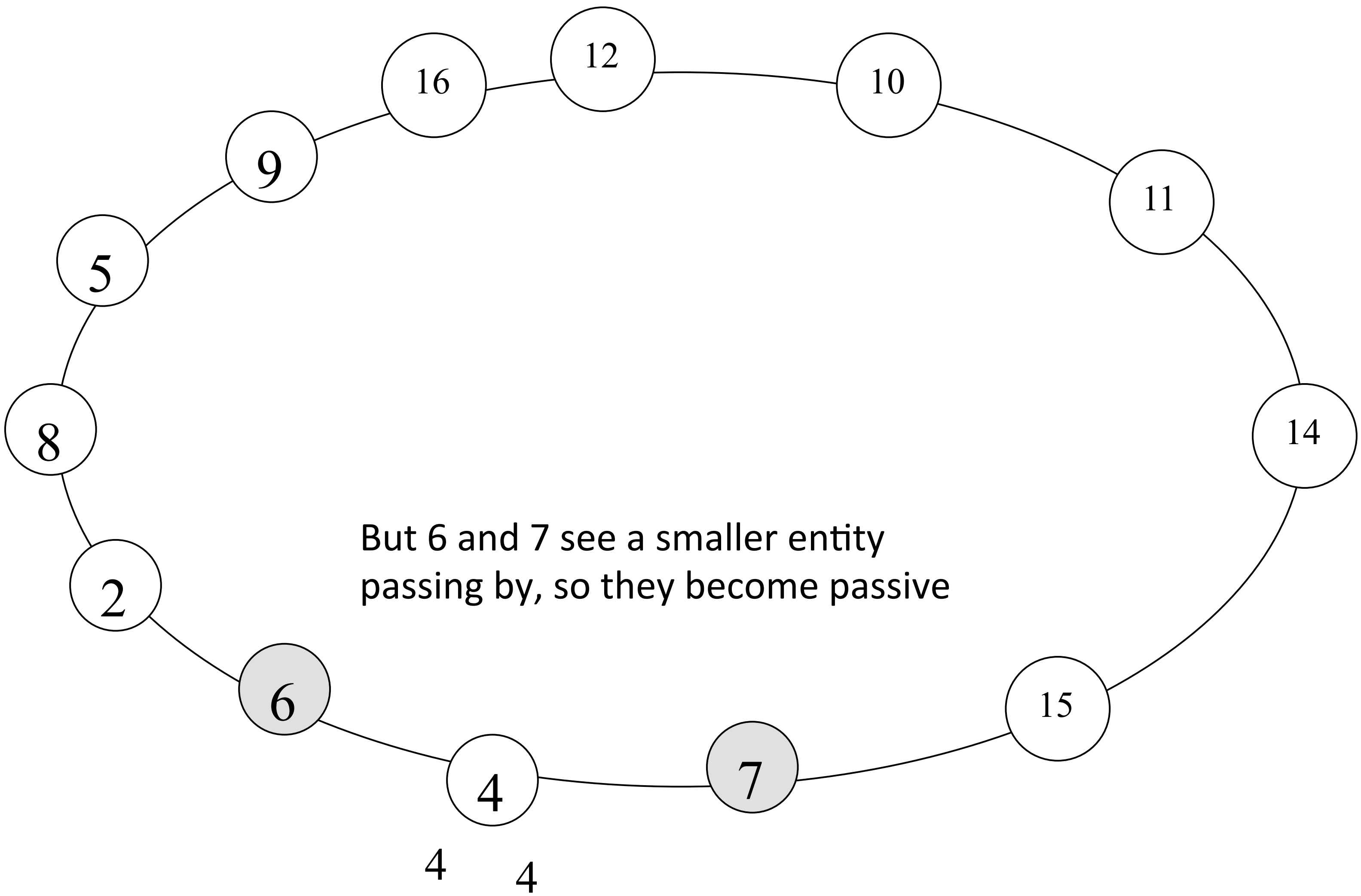
4 goes in both direction
does not encounter
anything smaller
comes back safe

Example



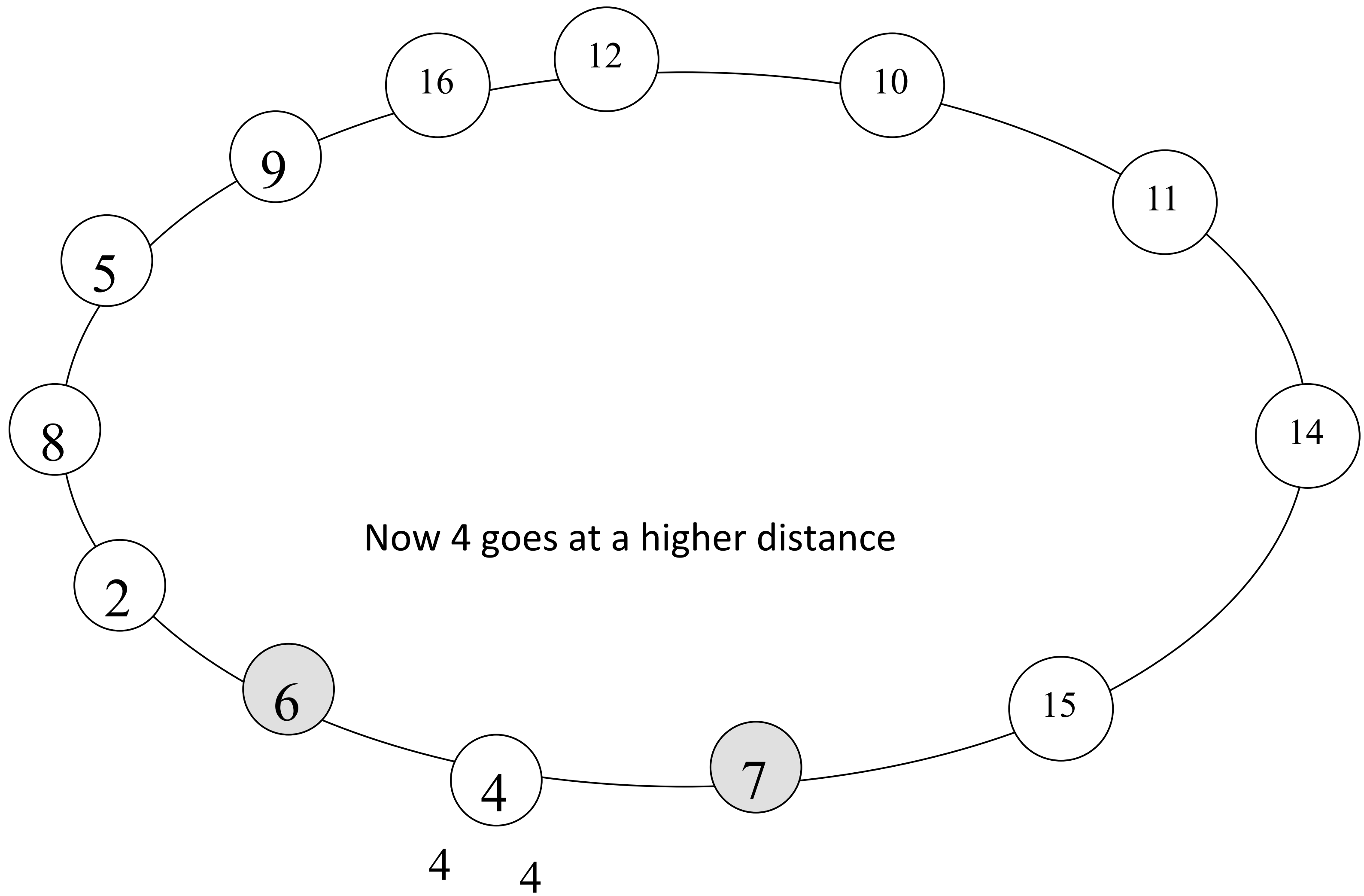
But 6 and 7 see a smaller entity passing by, so they become passive

Example

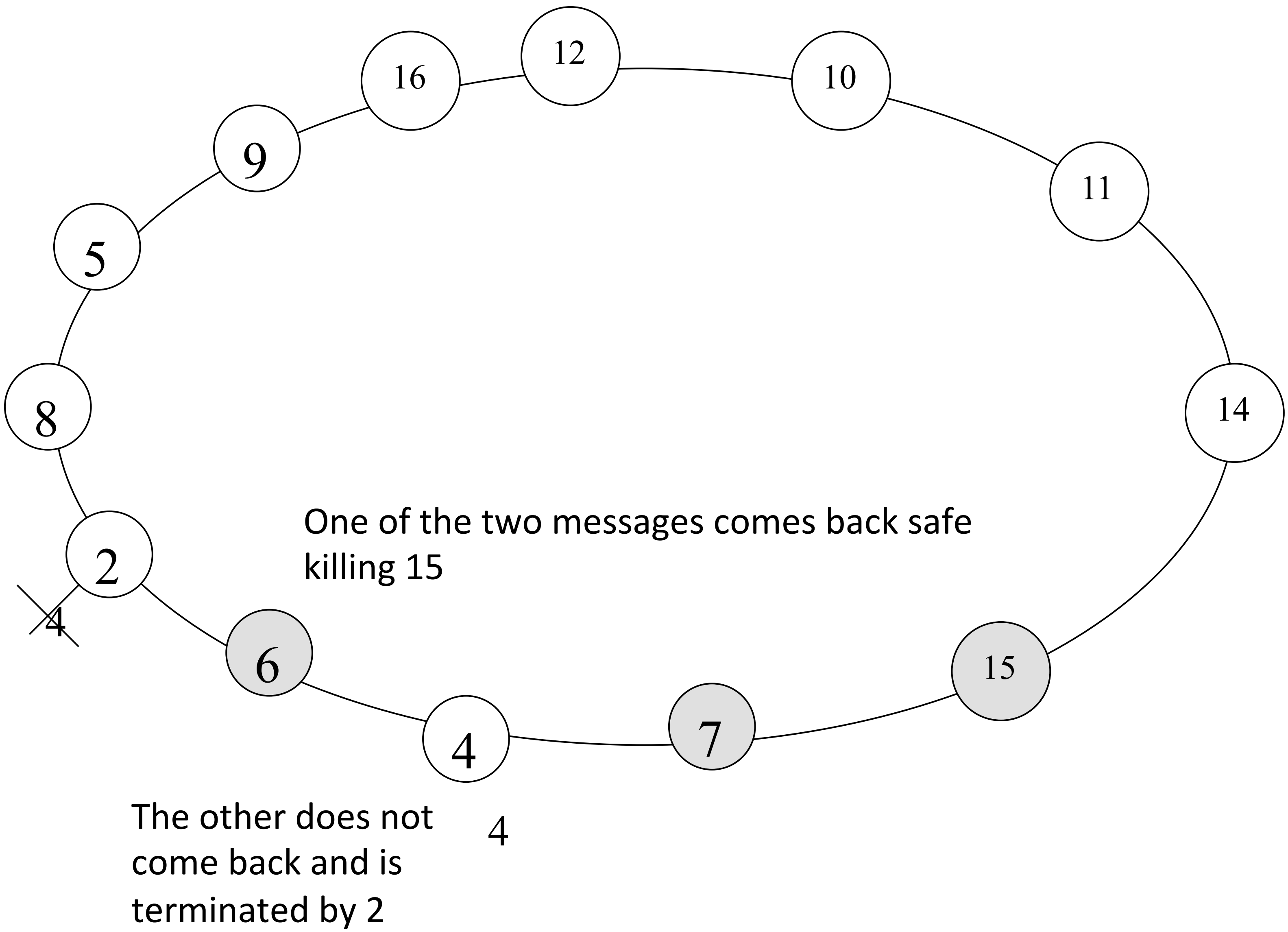


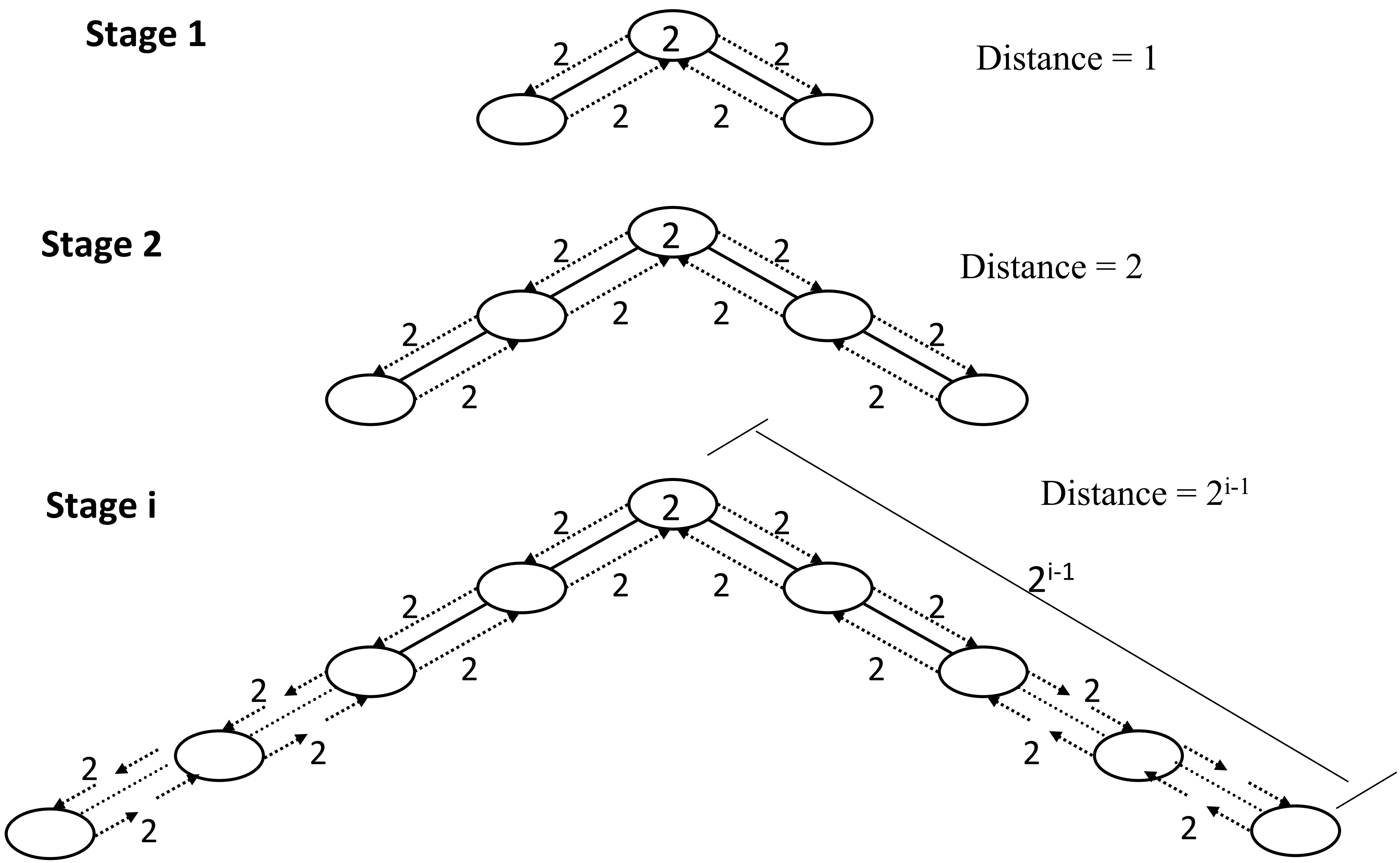
But 6 and 7 see a smaller entity passing by, so they become passive

Example

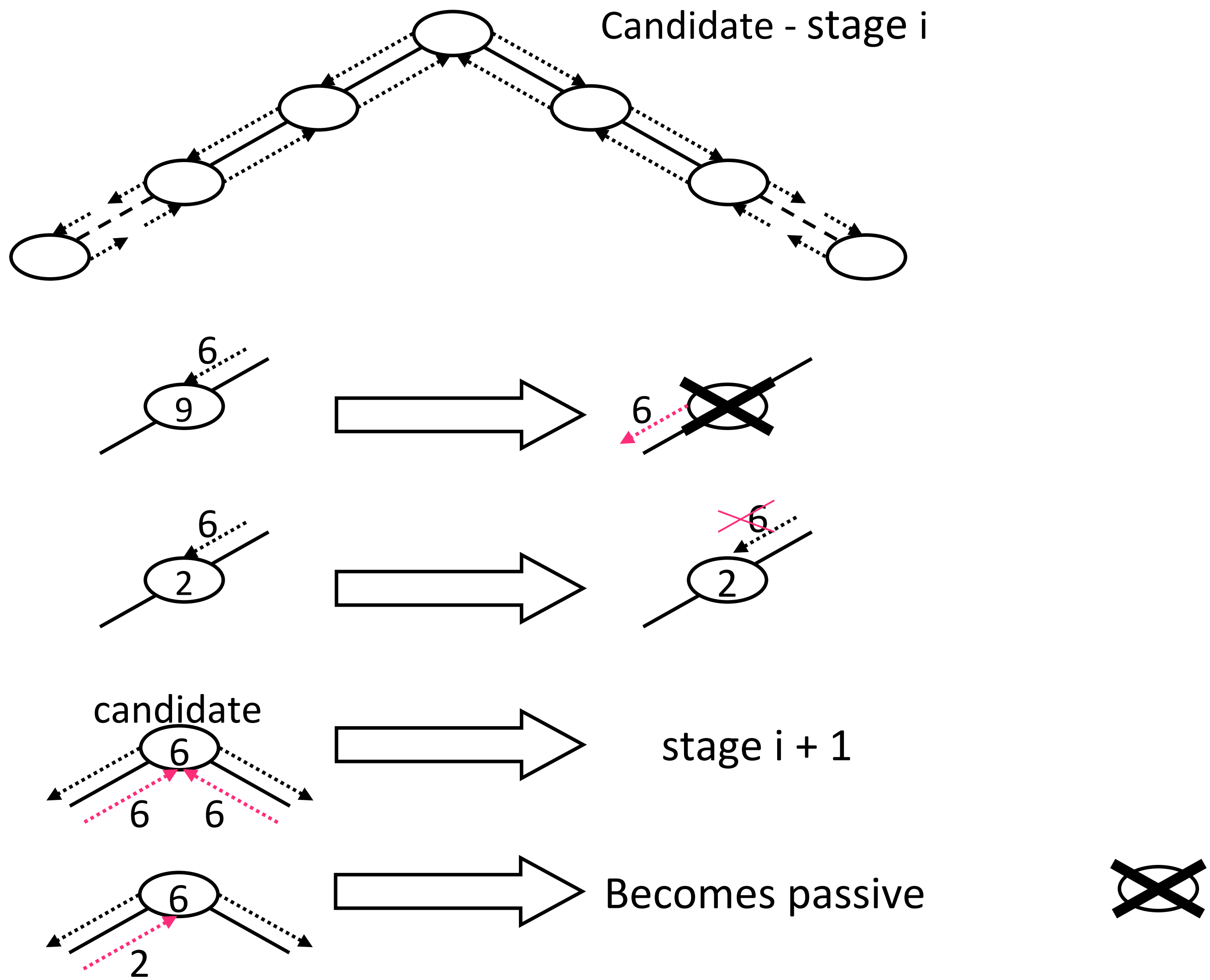


Example

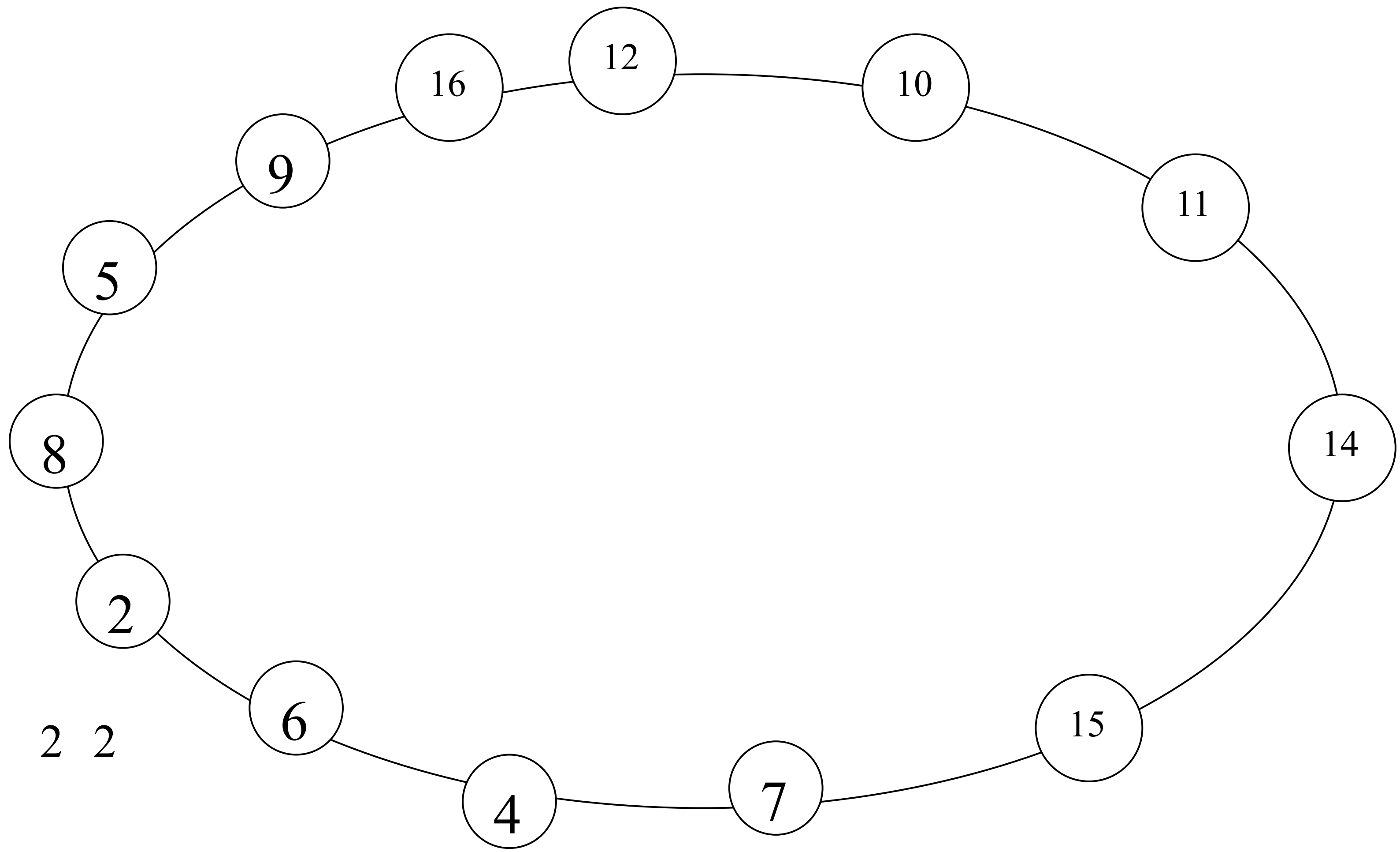

16
12
10
9
11
5
8
14
2
4
6
4
7
15
One of the two messages comes back safe killing 15
The other does not come back and is terminated by 2
4


Stage 1
2
2
2
2
2
Distance = 1
Stage 2
Distance = 2
Stage i
Distance = 2^(i-1)
2^(i-1)

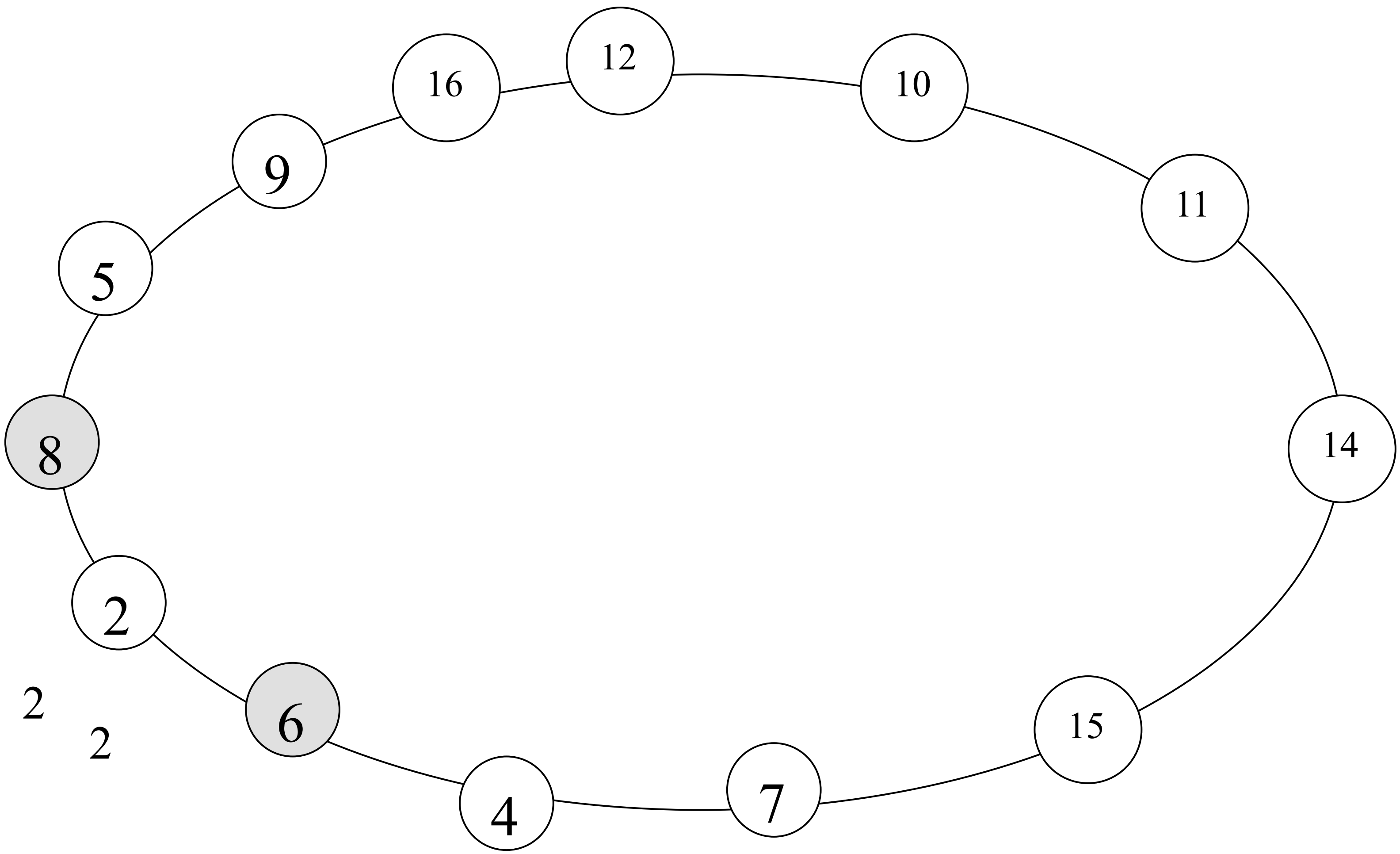
More...


Candidate - stage i
6
9
6
6
2
6
2
candidate
6
6
6
stage i + 1
6
2
Becomes passive

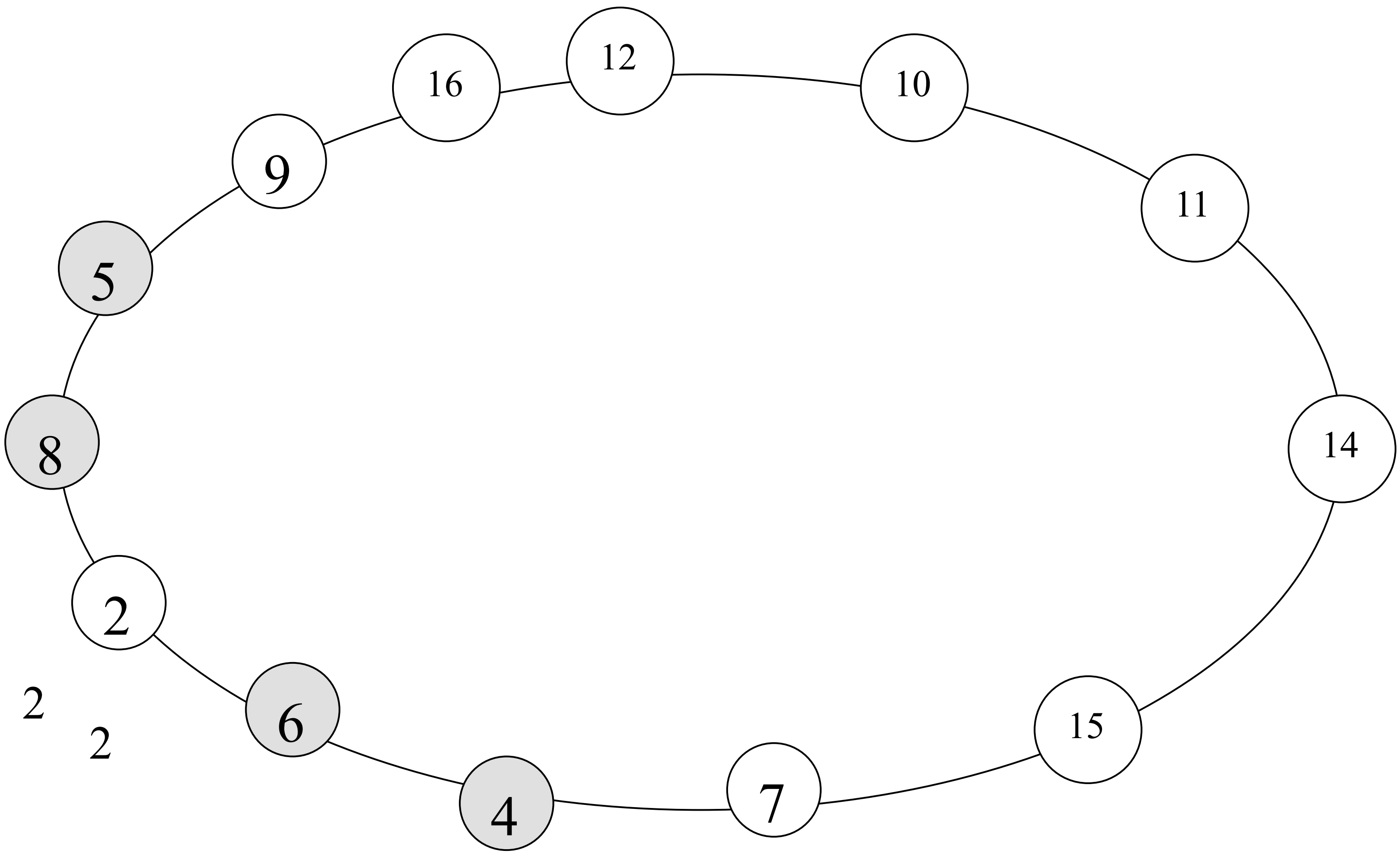
Example



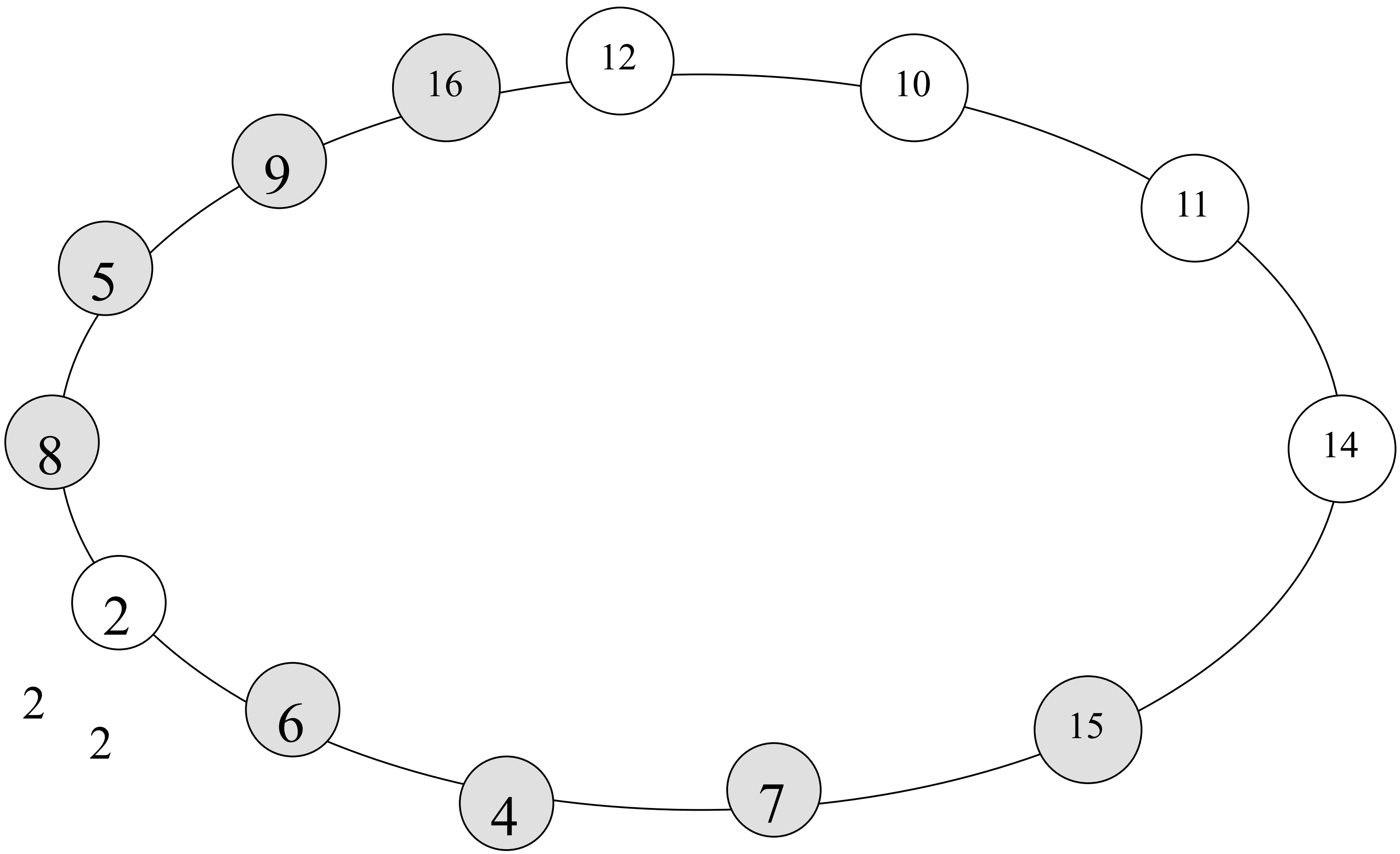
Example



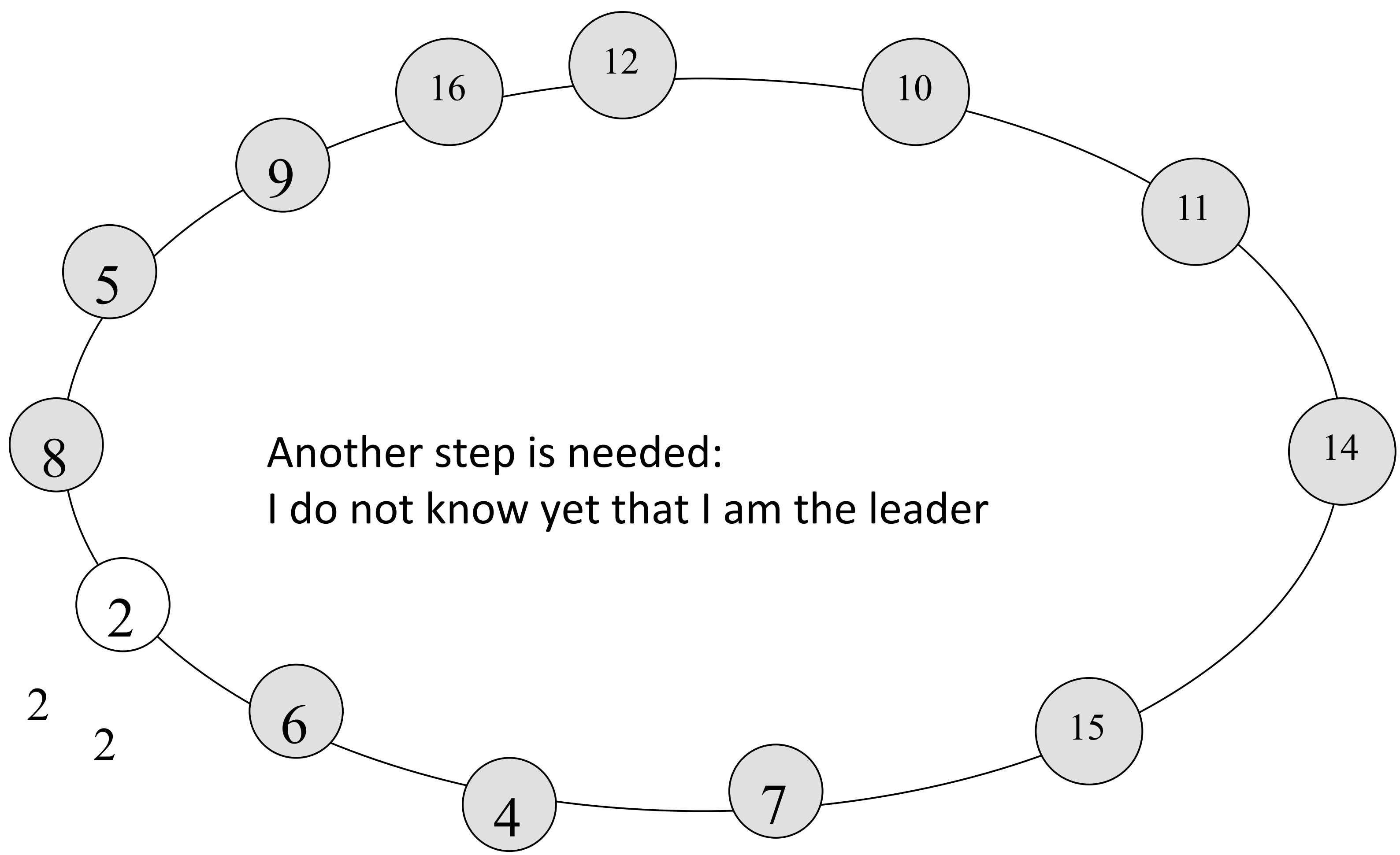
Example



Example

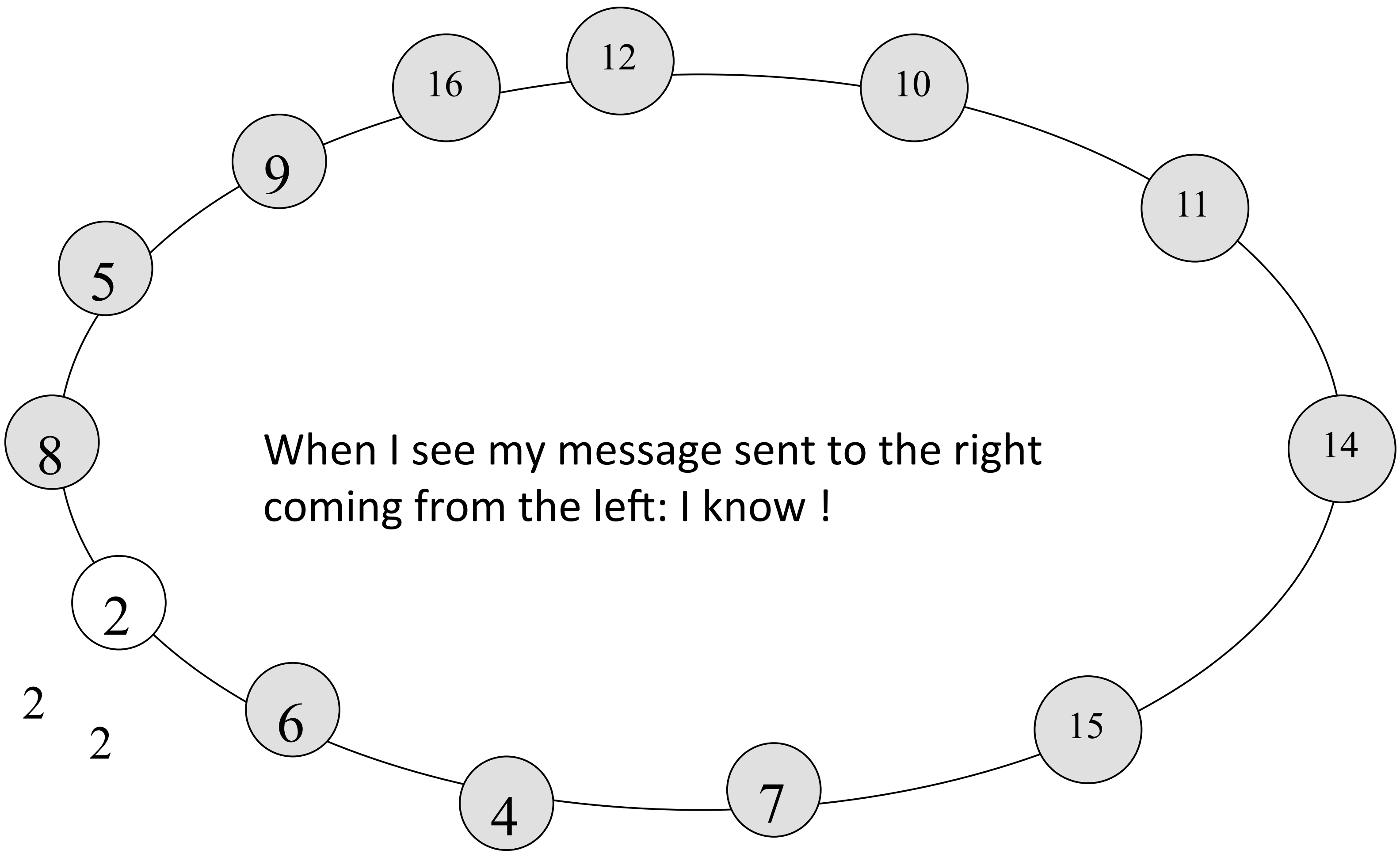


Example



12
16
10
9
11
5
8
14
Another step is needed:
I do not know yet that I am the leader
2
2
2
6
15
4
7

Example



When I see my message sent to the right coming from the left: I know !

Correctness and Termination

If a candidate receives its message from the opposite side it sent it, it becomes the leader and notifies.

-The smallest id will always travel the max distance defeating every entity it encounters

-The distance monotonically increases eventually becoming greater than n

-The leader will eventually receive its message from the opposite directions

Note: we do not need message ordering.

What happens if an entity receives a message from a higher stage ?

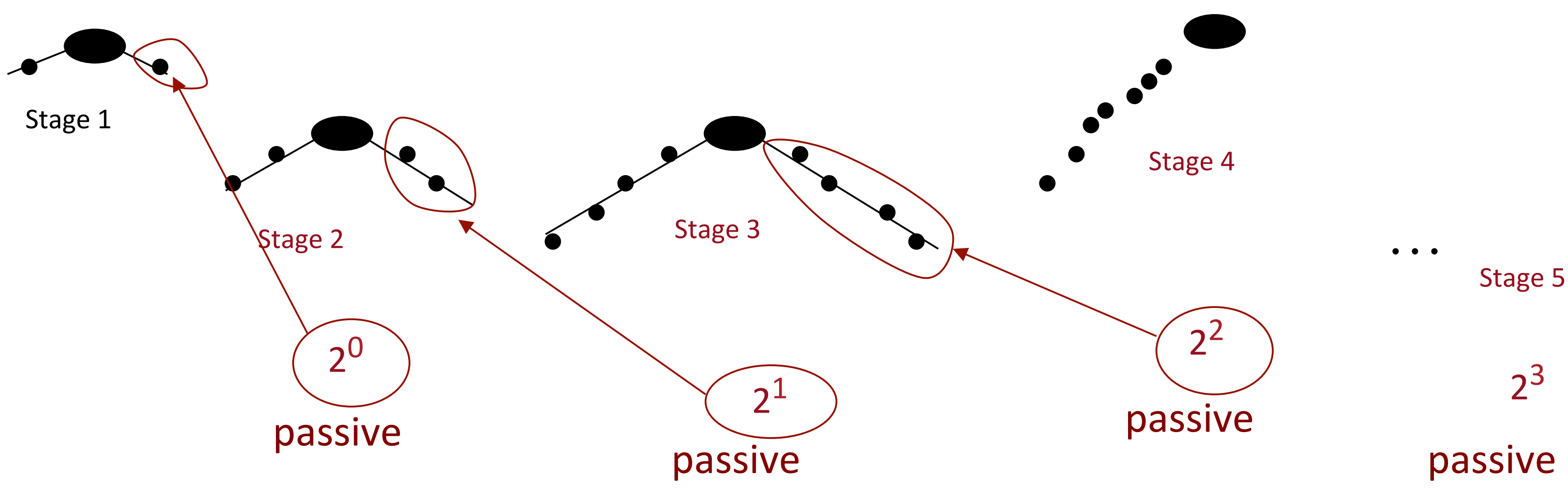
Message Complexity

When the distance is doubled at each stage
i.e., $dis(i) = 2^{i-1}$:

Notion of Logical Stage

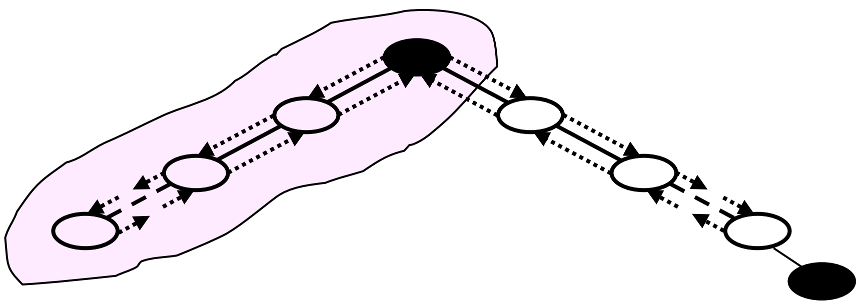
n_i entities start stage i

If x starts stage **i** (i.e., survived stage i-1) the Id of x must be smaller than the Ids of the neighbours at distance up to $\mathbf{2^{i-2}}$ on each side

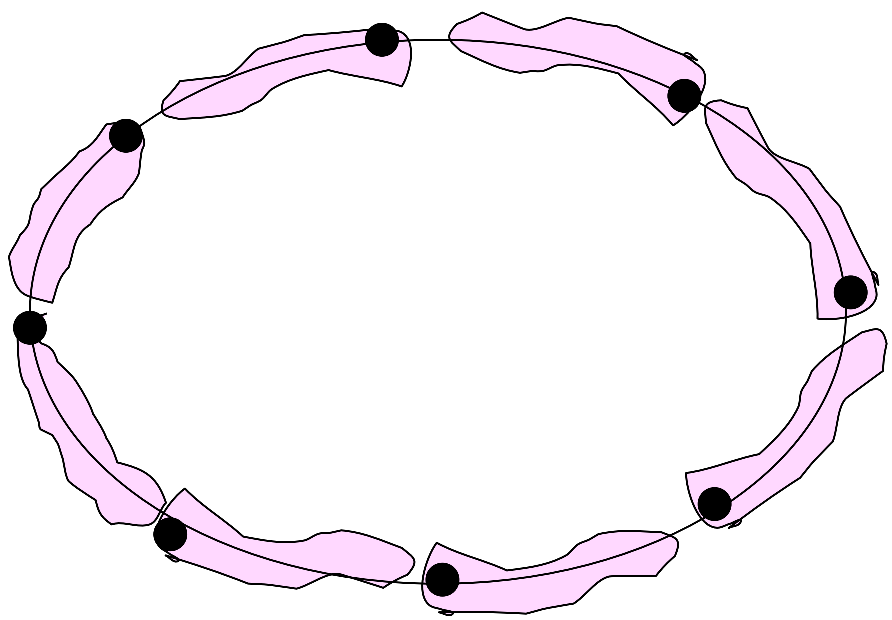


At stage i there are 2^{i-2} passive nodes on the left and 2^{i-2} on the right of the surviving candidate

Within any group of $2^{i-2}+1$ consecutive entities at most one starts stage i (i.e., survives stage i-1).



$$n_i \leq n/(2^{i-2}+1) \leq n/2^{i-2}$$



Total Number of Stages

The ring is fully traversed as soon as
2^{i-1} is greater than or equal to n

$$2^{i-1} \geq n$$

That is, when:

$$i \geq \log n + 1$$

----> $\log n + 1$ stages

Number of messages in stage i:

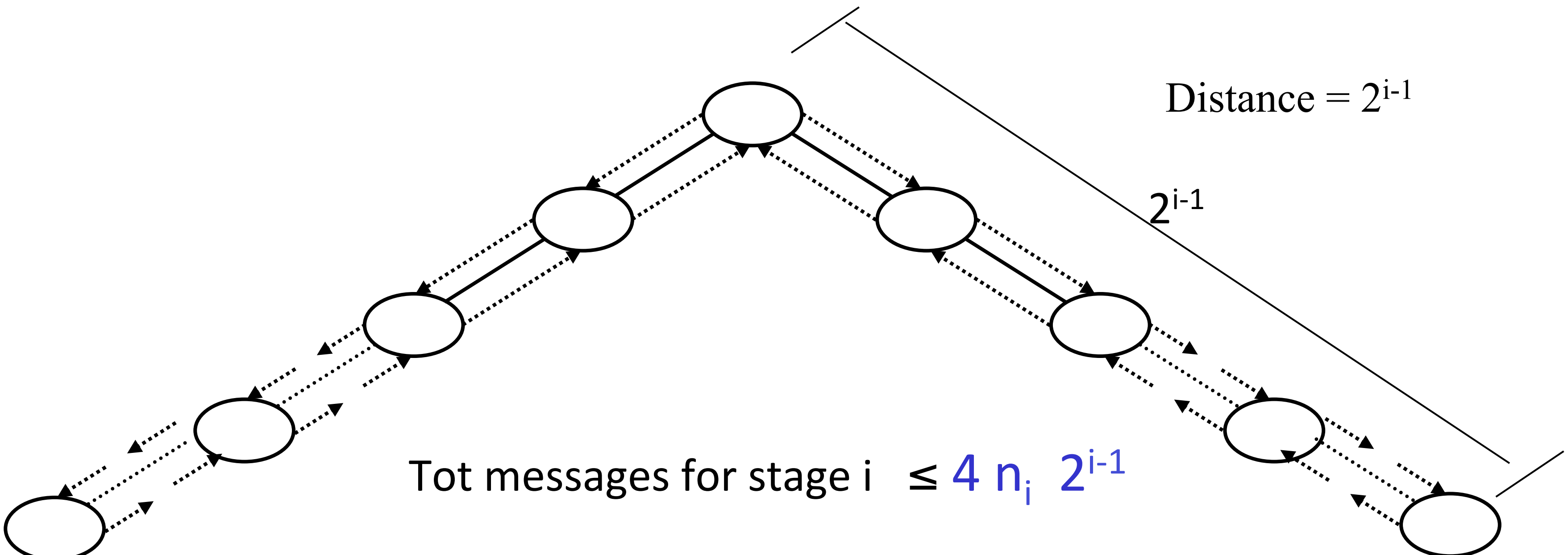
SIMPLIFIED VERSION

n_i are the entities that start stage i

Each of these entities generates:

at most 2^{i-1} "**forth**" messages in both directions

and at most 2^{i-1} "**back**" messages from both directions



Tot messages for stage i $\leq 4\, n_i\; 2^{i-1}$

$= 8\, n_i\; 2^{i-2}$

Number of messages in stage i:

SIMPLIFIED VERSION

Tot messages for stage i $\leq 4\, n_i\, 2^{i-1} = 8\, n_i\, 2^{i-2}$

We know that:

$n_i \leq n/2^{i-2}$

Tot messages for stage i $\leq 8\, \frac{n}{2^{i-2}}\, 2^{i-2} = 8\, n$

Total number of messages

SIMPLIFIED VERSION

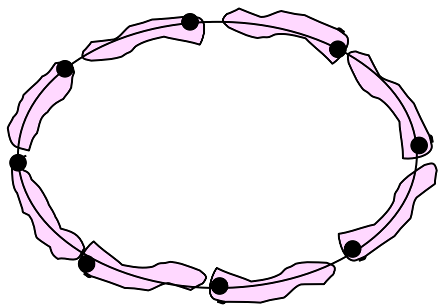
At most 8n messages per stage
At most log n + 1 stages

O(n log n)

SUMMARY

Number of stages: **O(log n)** because 2^{i-1} must become greater than n

Number of Candidates in stage i:



$$n_i \le n/\,(2^{i-2}+1) \;\; \le \mathbf{n/2^{i-2}}$$

Tot number of messages for stage i : $\le 8\, n_i\, 2^{i-2}$

Substituting n_i we obtain

Tot number of messages for stage i : $\le$ 8n

Tot number of messages = n. of stages times messages per stage:

$\le$ 8n log n = O(n log n)

More precise calculation

will lead to 7n log n

Starting stage i:

n_i are the entities that start stage i

"Forth" messages:

each will travel at most 2^{i-1} in both directions

Tot: $2\, n_i \; 2^{i-1}$

"Back" messages:

each survivor will receive one from each side

$2\, n_{i+1} \; 2^{i-1}$

each entity that started the stage but did not survive will receive none or one

$\leq (n_i - n_{i+1})\, 2^{i-1}$

Tot: $2\, n_{i+1} \; 2^{i-1} + (n_i - n_{i+1})\, 2^{i-1}$

stage i>1

Tot: $2\, n_i\ \ 2^{i-1} + 2\, n_{i+1}\ \ 2^{i-1} + (n_i - n_{i+1})\, 2^{i-1}$

$= (3n_i + n_{i+1})\, 2^{i-1}$

$n_i \leq n / (2^{i-2} + 1)$

$\leq (3\lfloor n/(2^{i-2}+1)\rfloor + \lfloor n/(2^{i-1}+1)\rfloor)\, 2^{i-1}$

$$< \frac{3\, n\, 2^{i-1}}{(2^{i-2}+1)} + \frac{n\, 2^{i-1}}{(2^{i-1}+1)}$$

$$< \frac{3\, n\, 2^{i-1}}{2^{i-2}} + \frac{n\, 2^{i-1}}{2^{i-1}}$$

$$= 6n + n = 7n$$

The first stage is a bit different:

If everybody starts:

the survivors $4\ n_2\ 2^0$ 2 "forth", 2 "back"

the others $3\ (n - n_2)\ 2^0$

2 "forth", 1 "back"

$n_2 \leq n / (2^0+1)$

$4\ n_2 + 3\ n - 3n_2 = n_2 + 3\ n$

$= n/2 + 3\ n \quad < 4n$

first stage

$$\text{TOT} \le \sum_{i=1}^{\log n} 7n + O(n)$$

$$= n \sum_{i=2}^{\log n} 7 = 7\, n \log n + O(n)$$

O(n log n)

Conjecture:

In unidirectional rings,
the worst case complexity is (n^2);
to have a complexity of $O(n \log n)$ messages,
bidirectionality is necessary.

We will see that this is not true

Stages

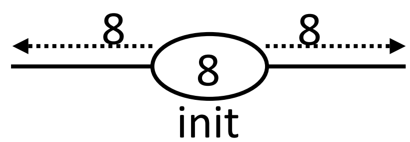
Basic idea:

A message will travel until it reaches another candidate
A candidate will receive a message from both sides

ASSUMPTIONS

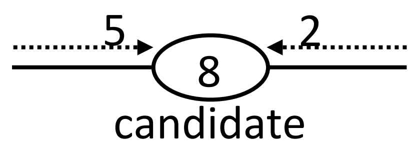
- Distinct *id*' s
- Bidirectional ring (+ unidirectional version)
- Local orientation
- Message ordering (for simplicity only: not needed)

Each *candidate* sends its own *Id* in both directions.



When a *candidate i* receives two messages Id_j (from the right) and Id_k (from the left), it determines if it becomes *passive* (= it is not the smallest), or if it remains *candidate* (= it is the smallest).

When a *candidate i* receives two messages
Id_j (from the right) and Id_k (from the left),



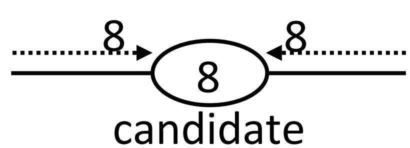
After receiving the first message:
`close-port` (enqueue messages possibly arriving later)

After receiving the second message, perform the action
and re-**`open-port`**

Correctness and termination

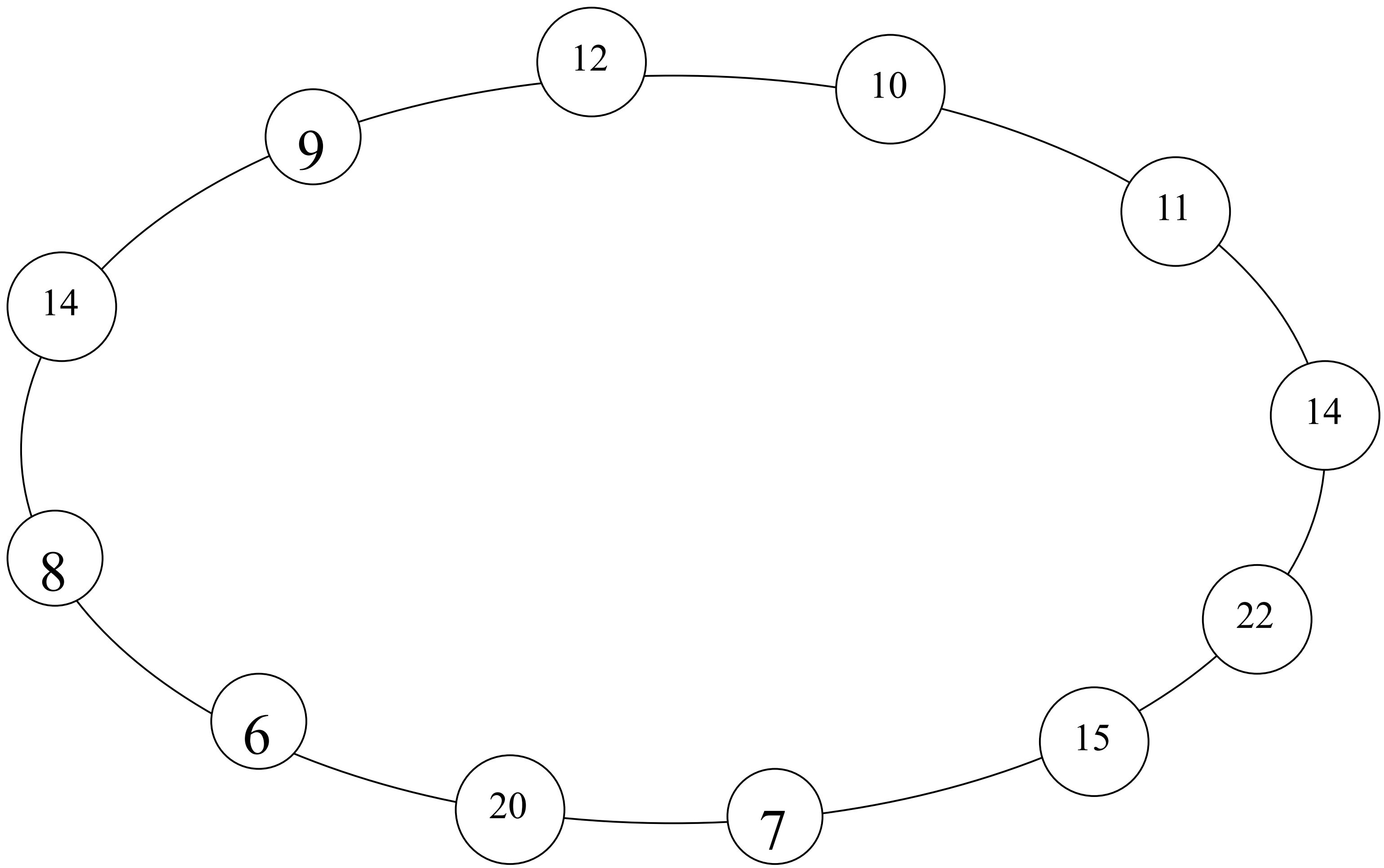
The minimal entity will never cease to send messages.

When an entity knows that it is the *leader*

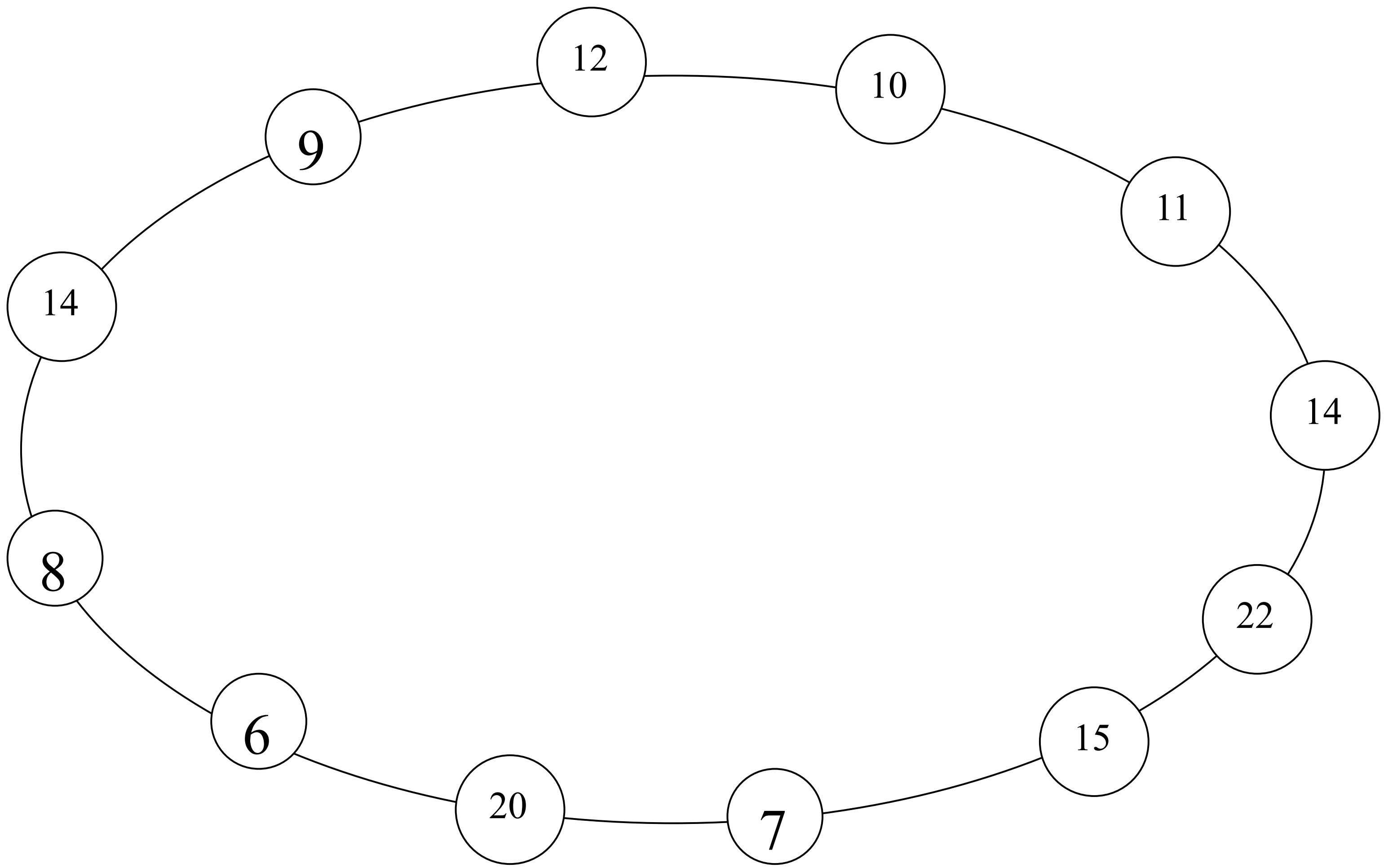


it sends a *notification* message which travels around the ring.

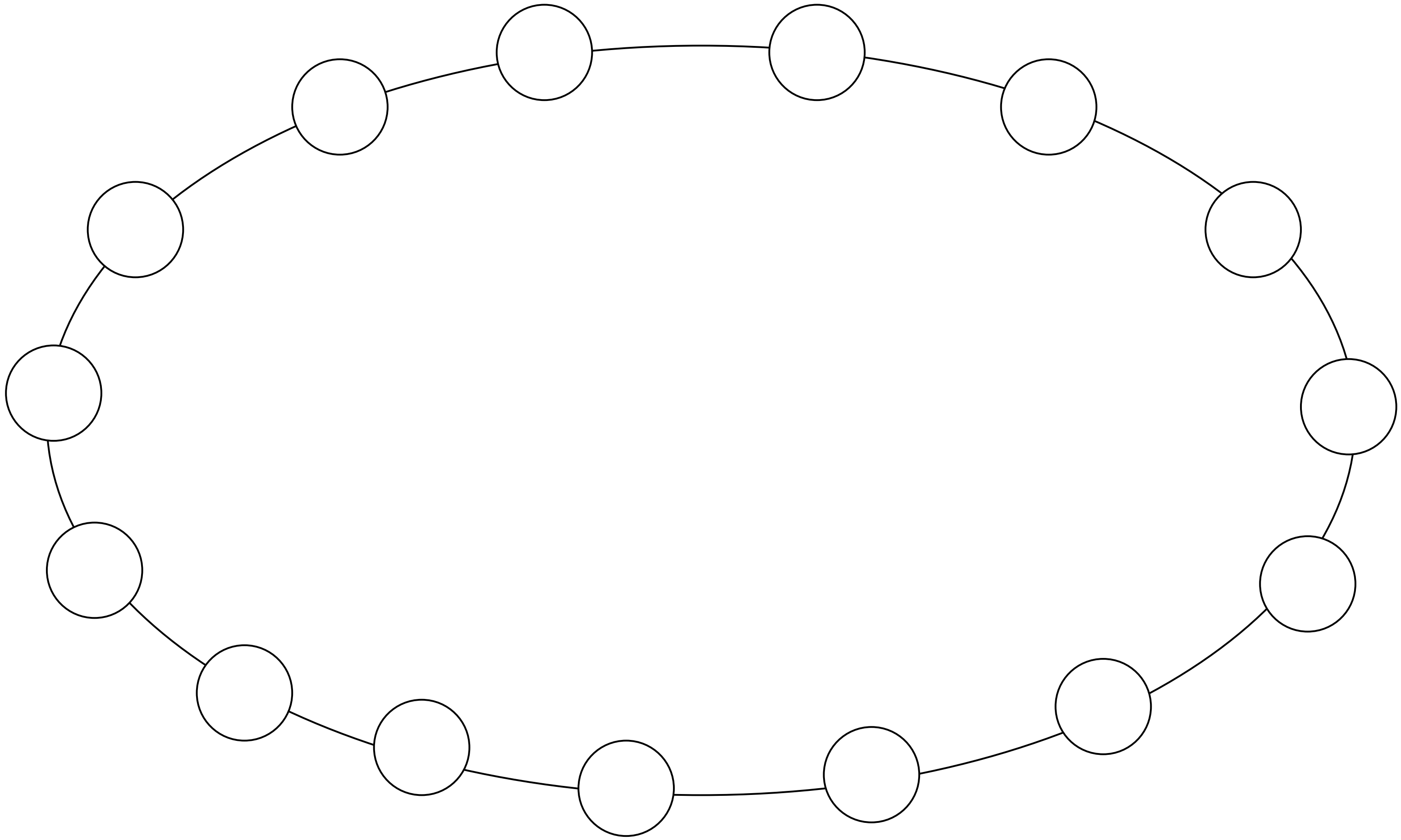
Stages



Stages



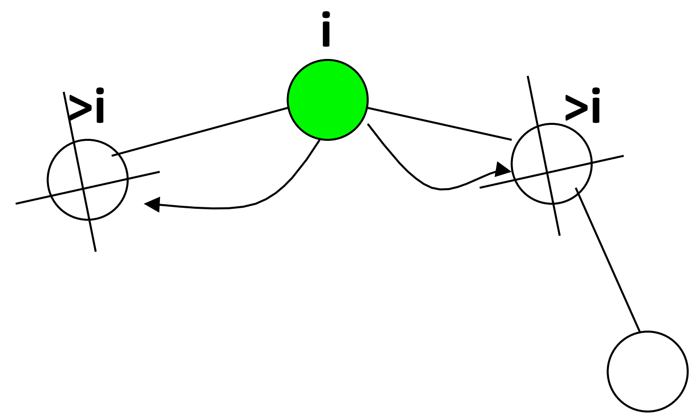
Stages



Complexity - Worst Case

At each step: At least half the entities became passive.

$$n_{i+1} \le \frac{n_i}{2}$$



$$\left.\begin{array}{l} n_0 = n \\ n_1 \le n/2 \\ \vdots \\ n_i \le n/2^i \end{array}\right\} \Rightarrow$$

$n/2^k \le 1$

when

$k \ge \log n$

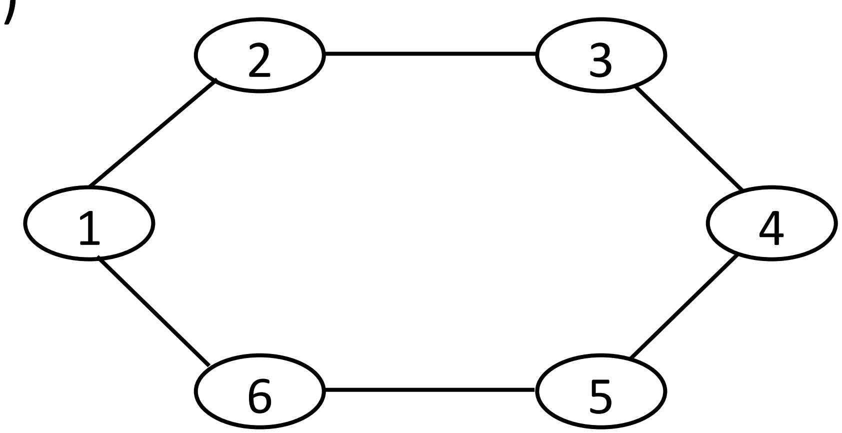
 # steps: At most ⌊ (log *n*) ⌋

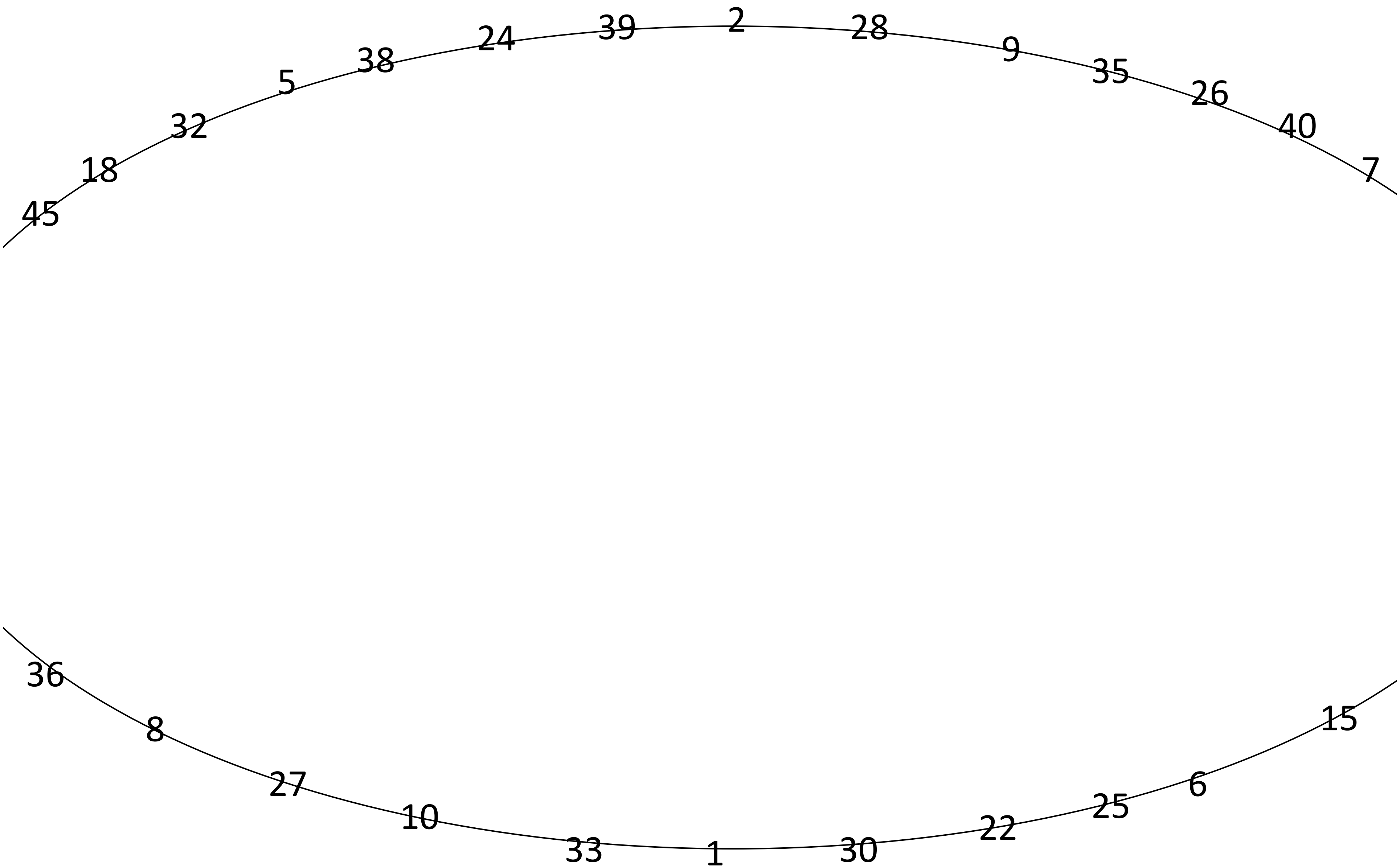
Each entity sends or resends 2 messages.
 # messages: 2n
 # bits: 2*n* * ⌊(log *n*) ⌋

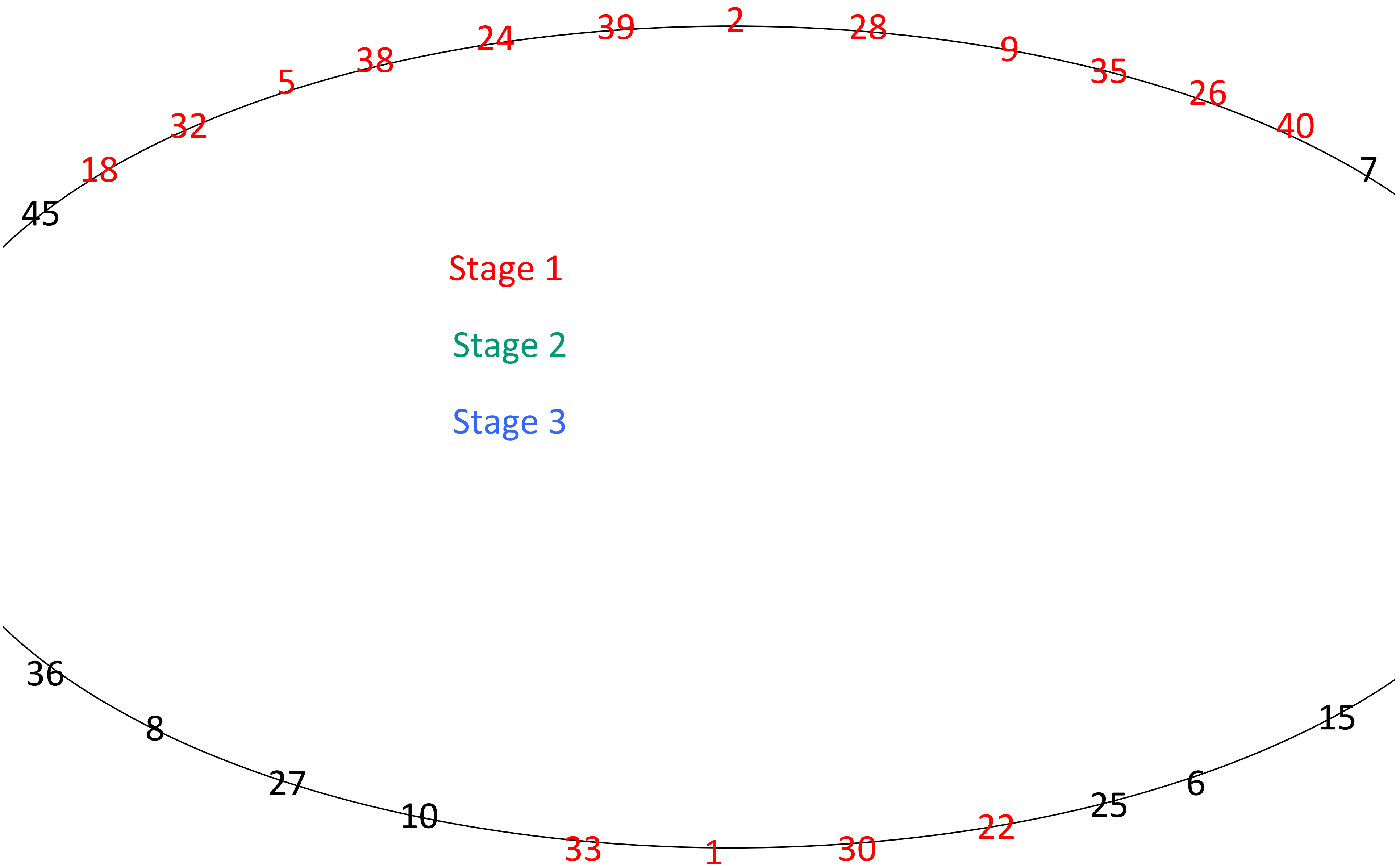
Last entity: 2n messages to understand that it is the last active entity, then n notification messages.

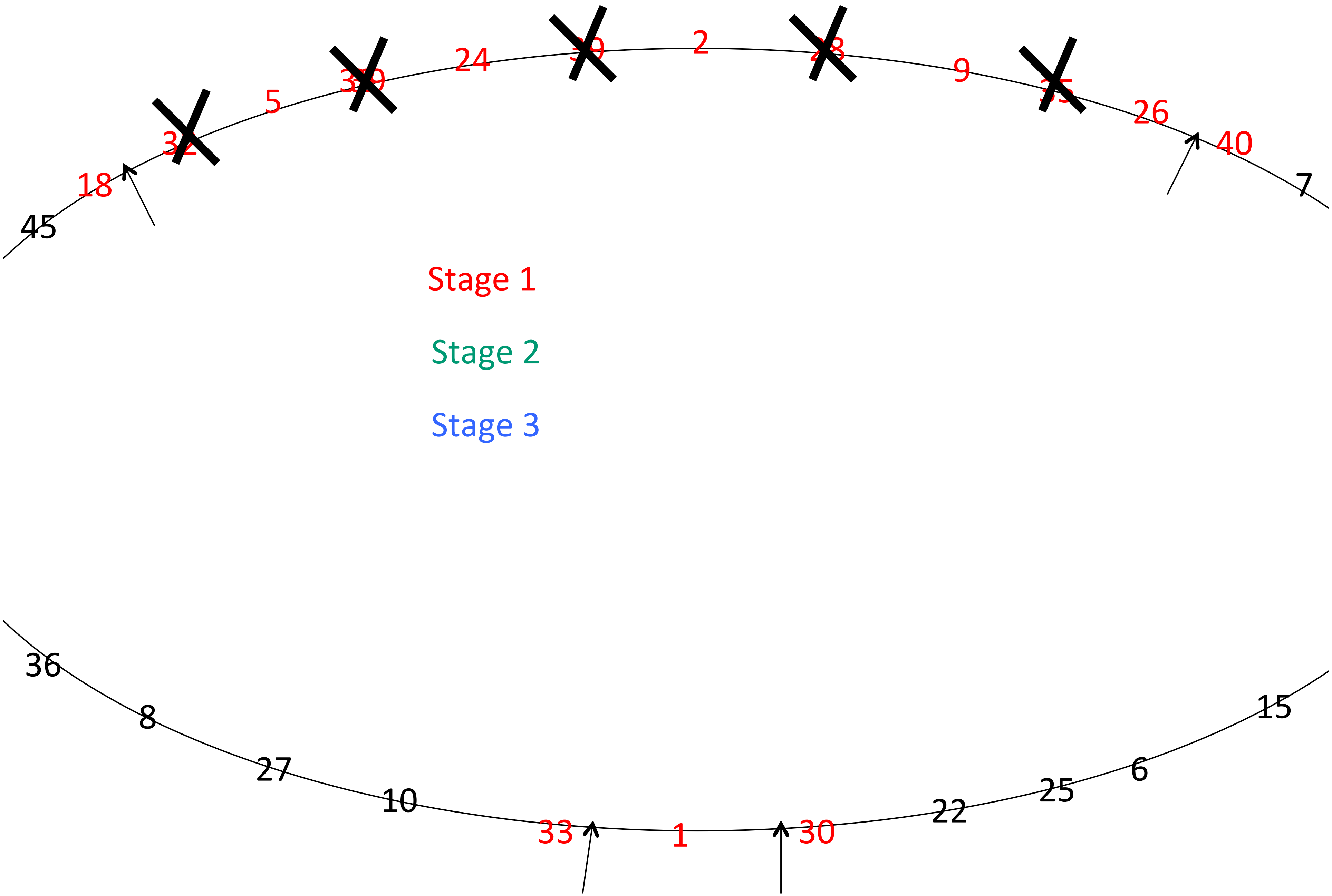
Total: 2n * ⌊(log *n*) ⌋ + 3 n = O(n log n)

Best Case ?

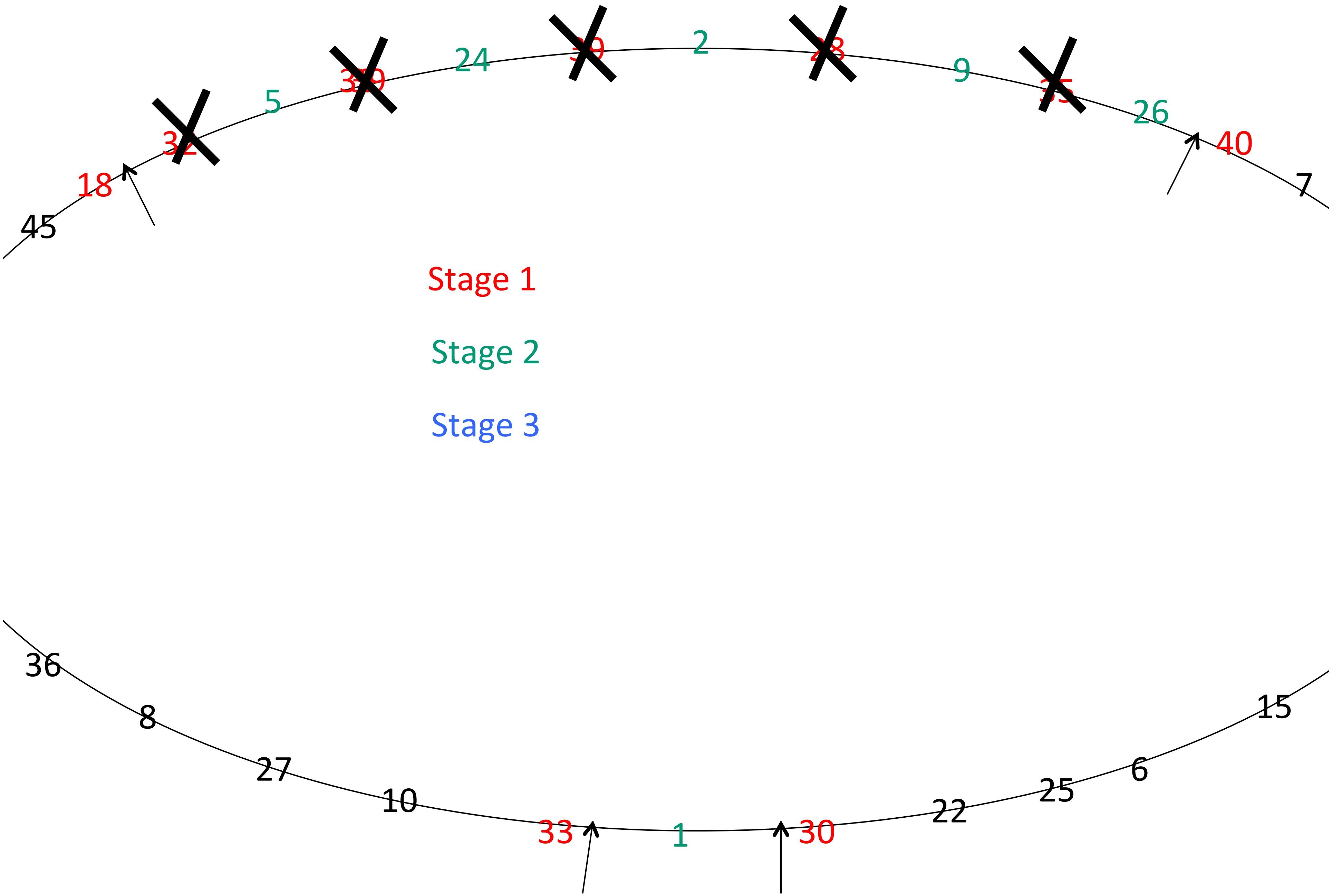


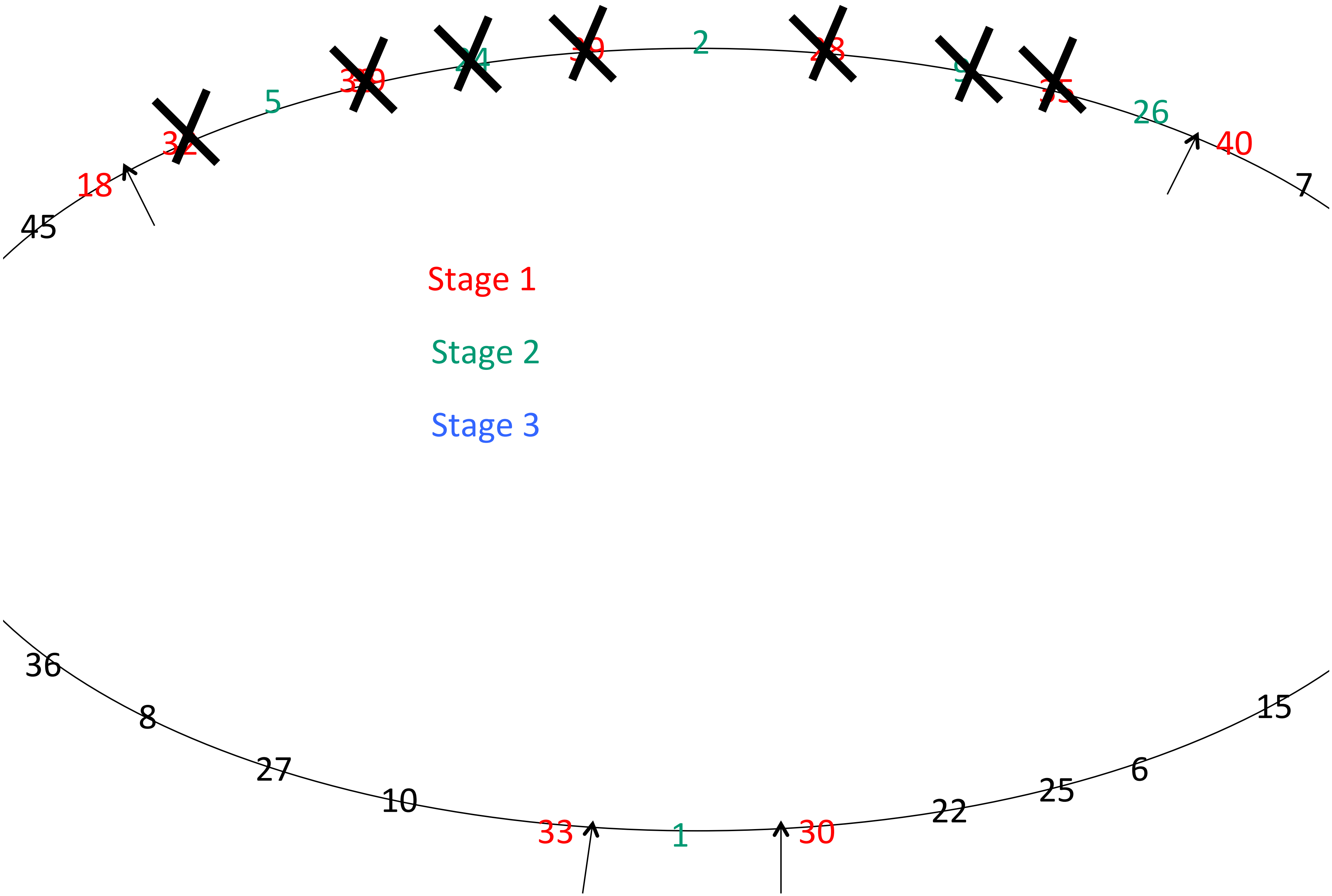

45
18
32
5
38
24
39
2
28
9
35
26
40
7
36
8
27
10
33
1
30
22
25
6
15

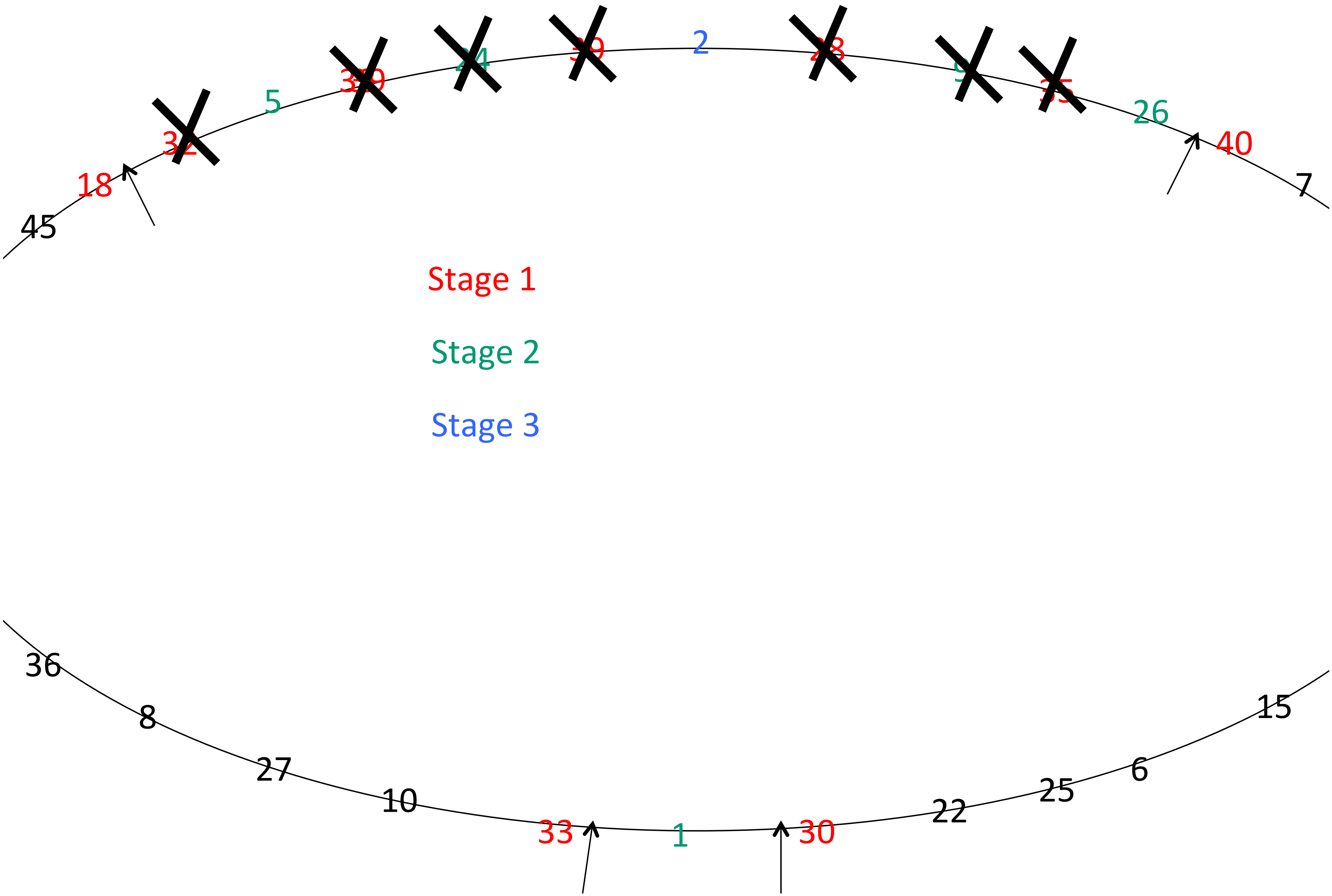

2
39
28
24
9
38
35
5
26
32
40
18
7
45
Stage 1
Stage 2
Stage 3
36
15
8
27
6
25
10
22
33
1
30

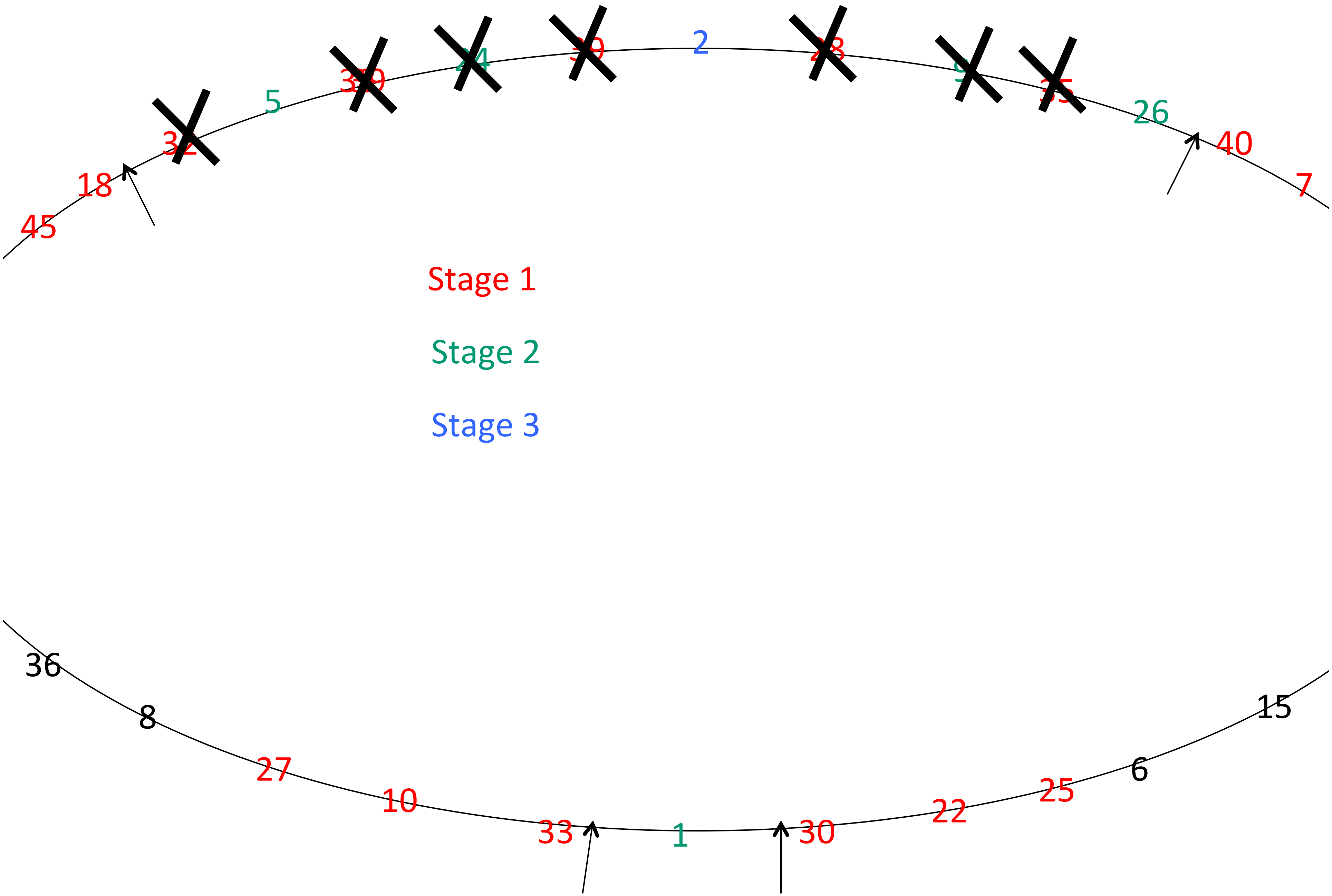


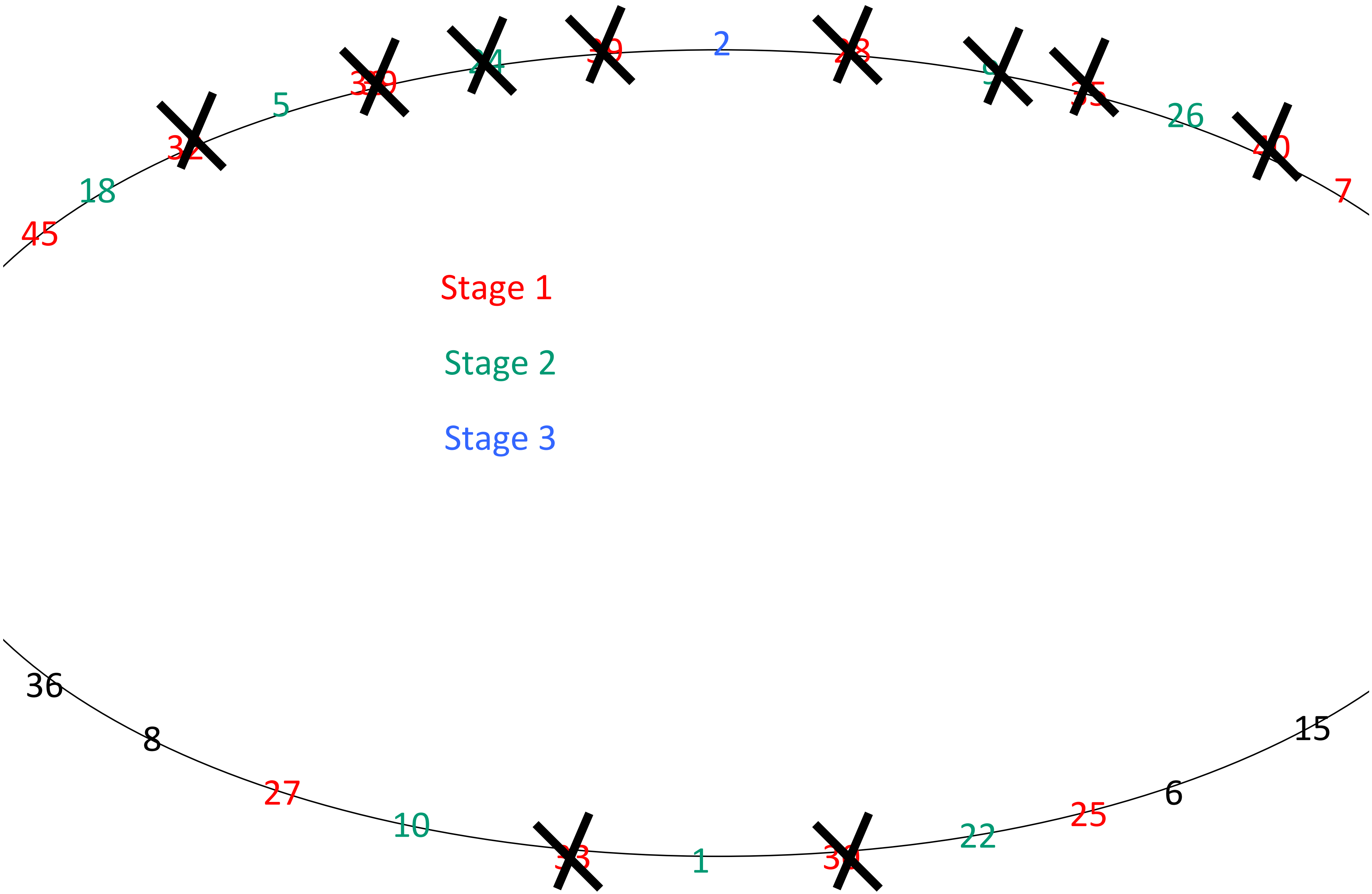
45
18
32
5
24
2
9
26
40
7
Stage 1
Stage 2
Stage 3
36
8
27
10
33
1
30
22
25
6
15


45
18
32
5
24
2
9
26
40
7
Stage 1
Stage 2
Stage 3
36
8
27
10
33
1
30
22
25
6
15


45
18
32
5
24
2
9
26
40
7
Stage 1
Stage 2
Stage 3
36
8
27
10
33
1
30
22
25
6
15


45
18
5
2
26
40
7
Stage 1
Stage 2
Stage 3
36
8
27
10
33
1
30
22
25
6
15

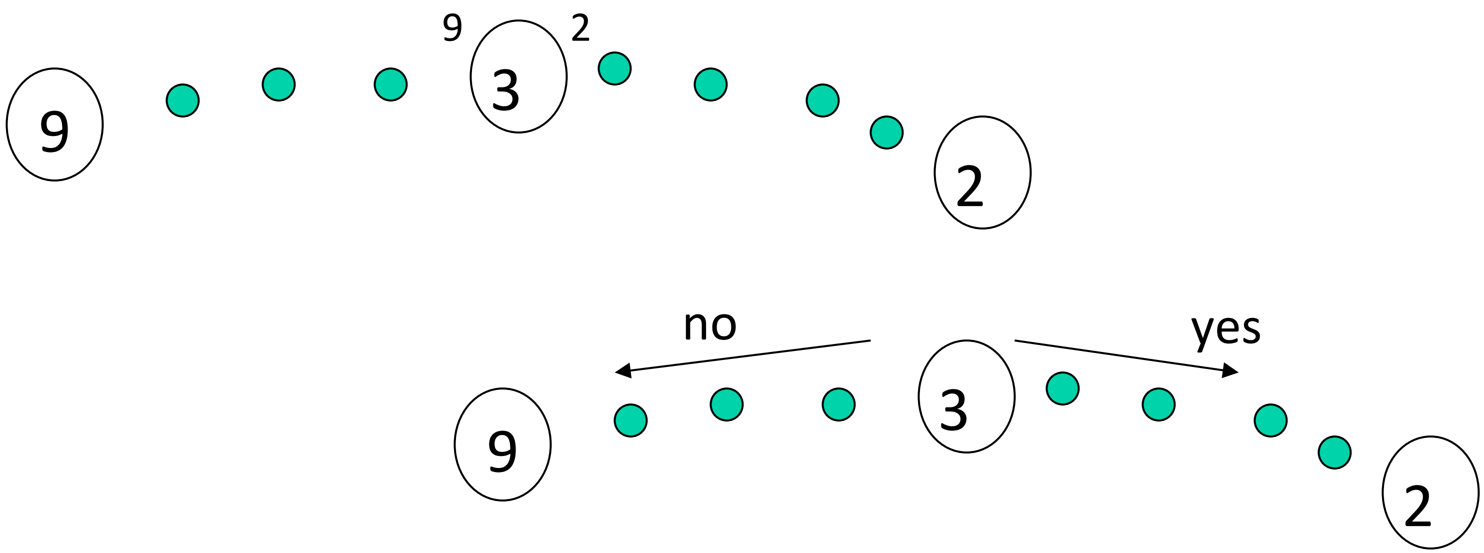

45
18
32
5
24
2
26
40
7
Stage 1
Stage 2
Stage 3
36
8
27
10
33
1
30
22
25
6
15



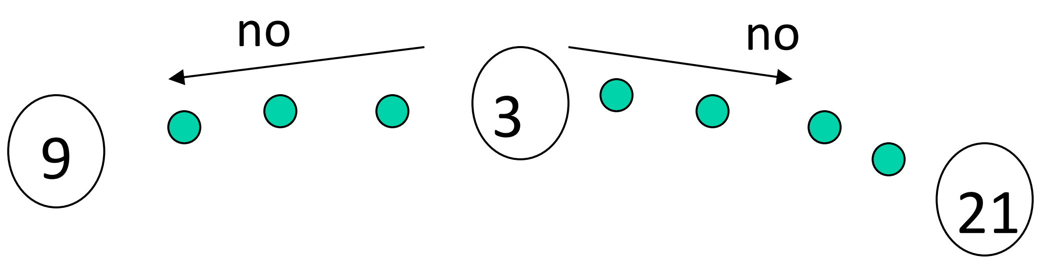
45
18
5
2
26
7
Stage 1
Stage 2
Stage 3
36
8
27
10
1
22
25
6
15

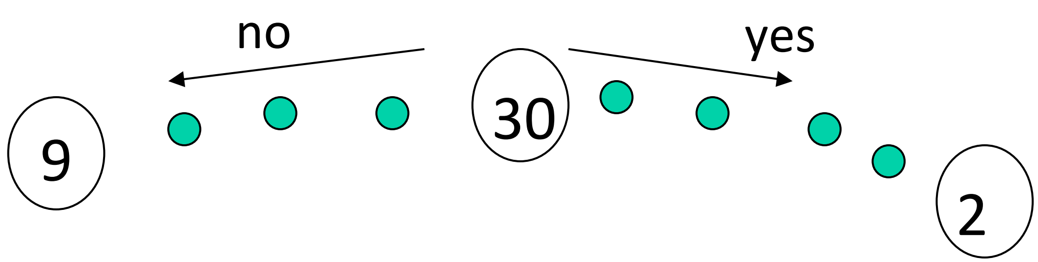
Stages with Feedback

A feedback is sent back to the originator of the message



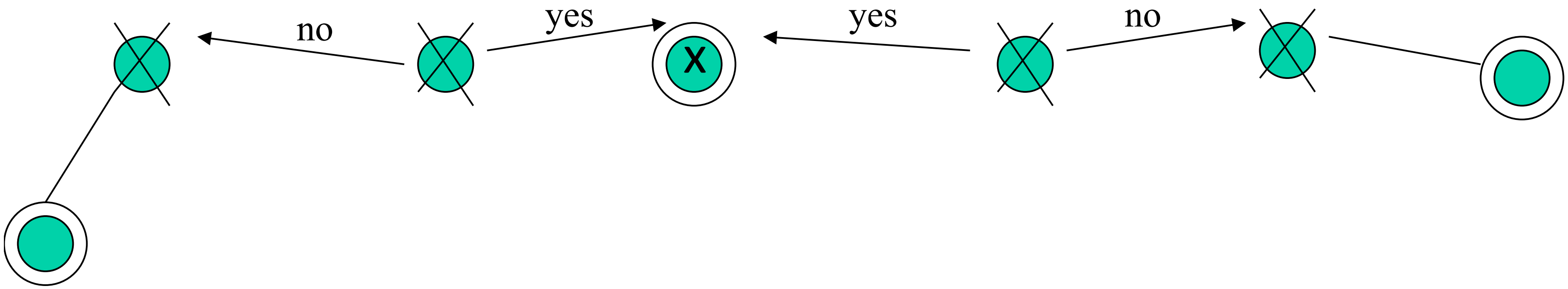
send YES to the smallest of the two IF it is smaller than me
(otherwise send NO)
send NO to the other



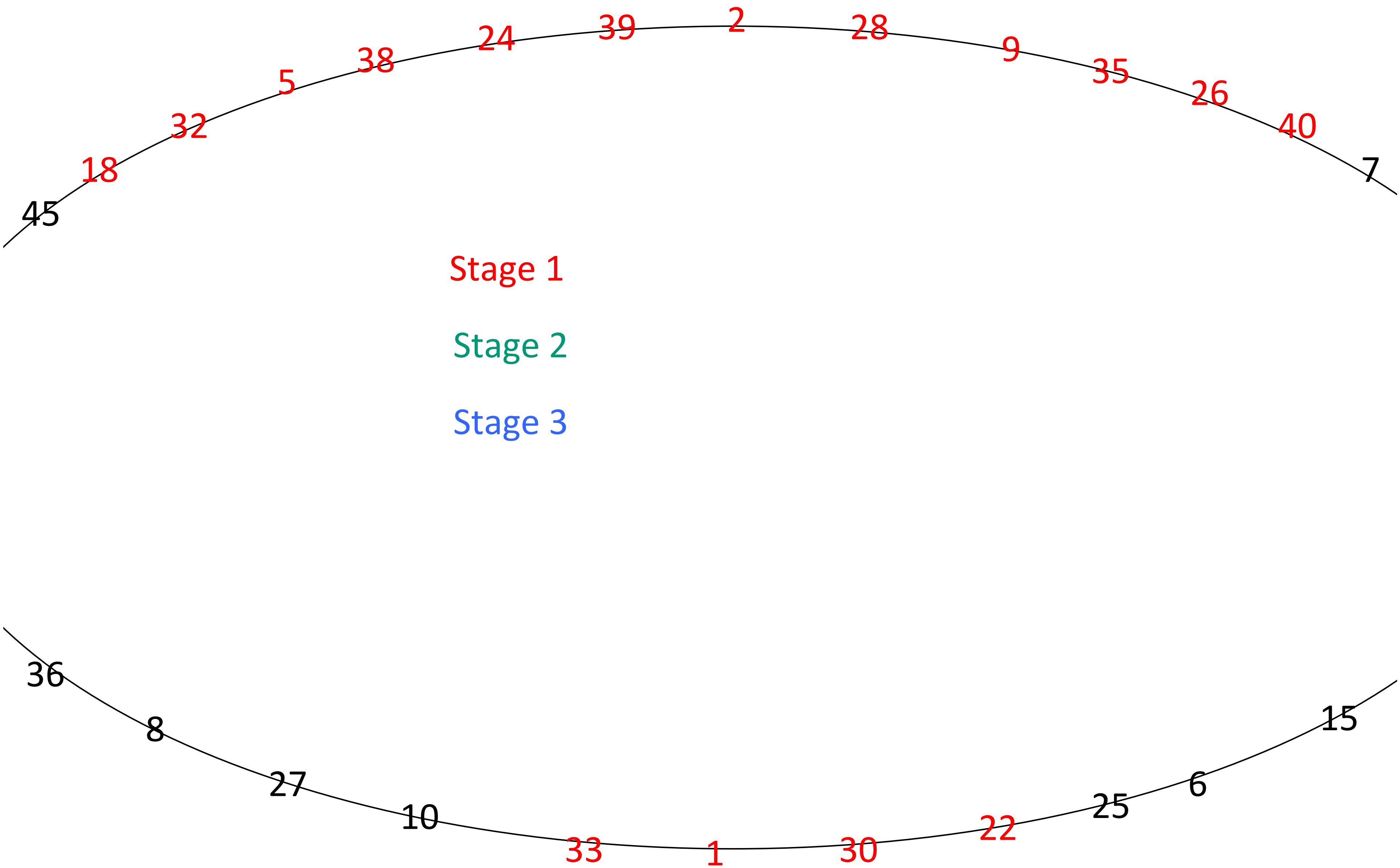


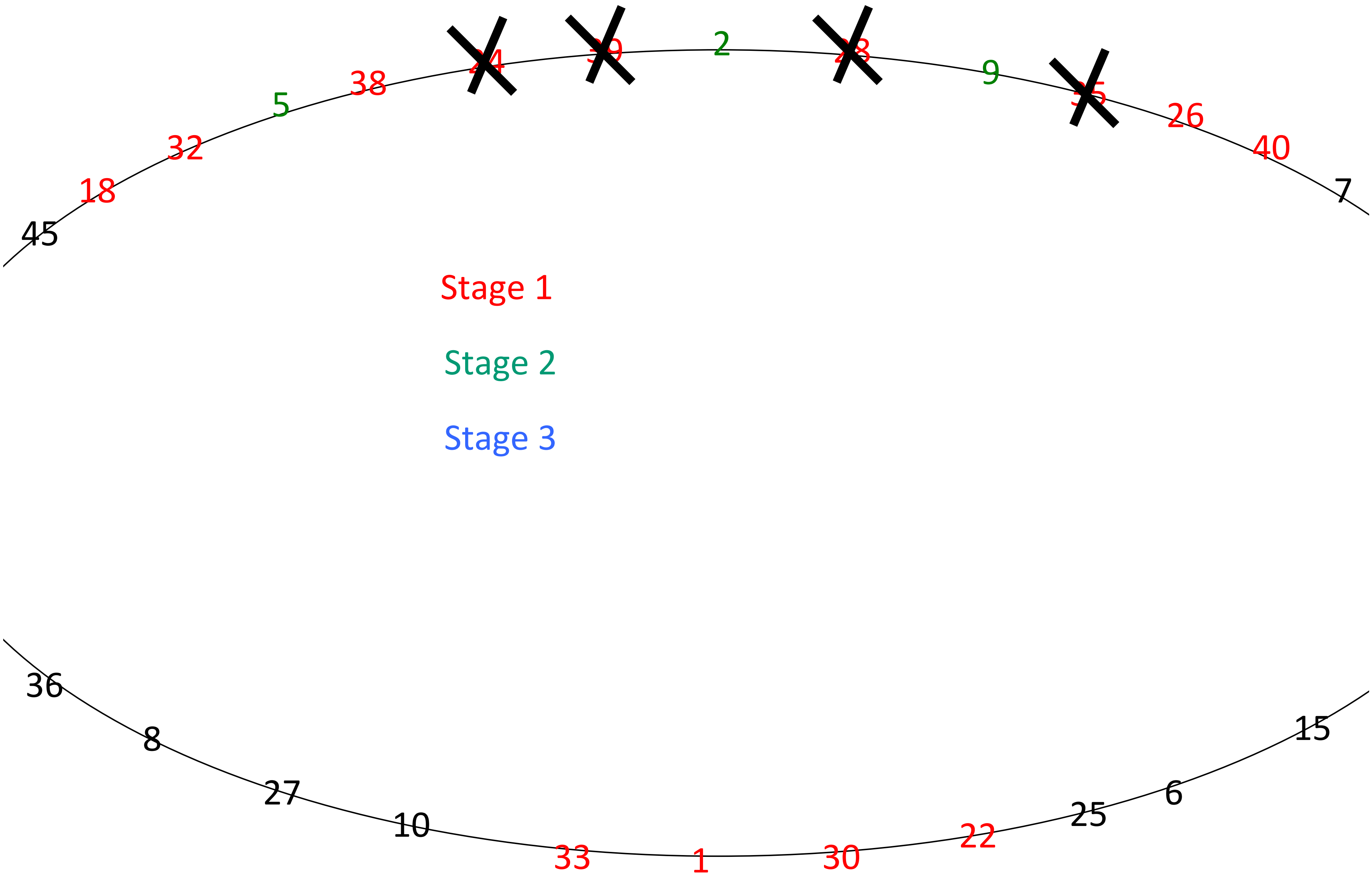
A node survives if it receives two YES feedbacks

If x survives, it must have received a feedback from both neighbouring candidates …



$$n_{i+1} \leq \frac{n_i}{3}$$


45
18
32
5
38
24
39
2
28
9
35
26
40
7
Stage 1
Stage 2
Stage 3
36
8
27
10
33
1
30
22
25
6
15


45
18
32
5
38
2
9
26
40
7
Stage 1
Stage 2
Stage 3
36
8
27
10
33
1
30
22
25
6
15

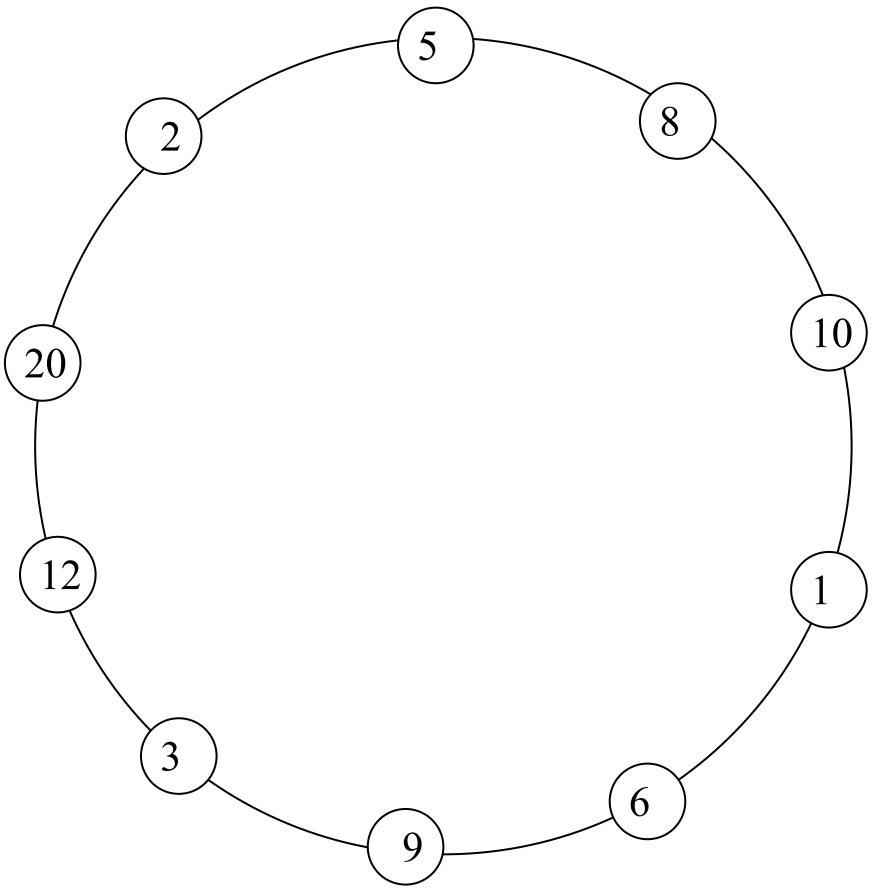
Unidirectional version

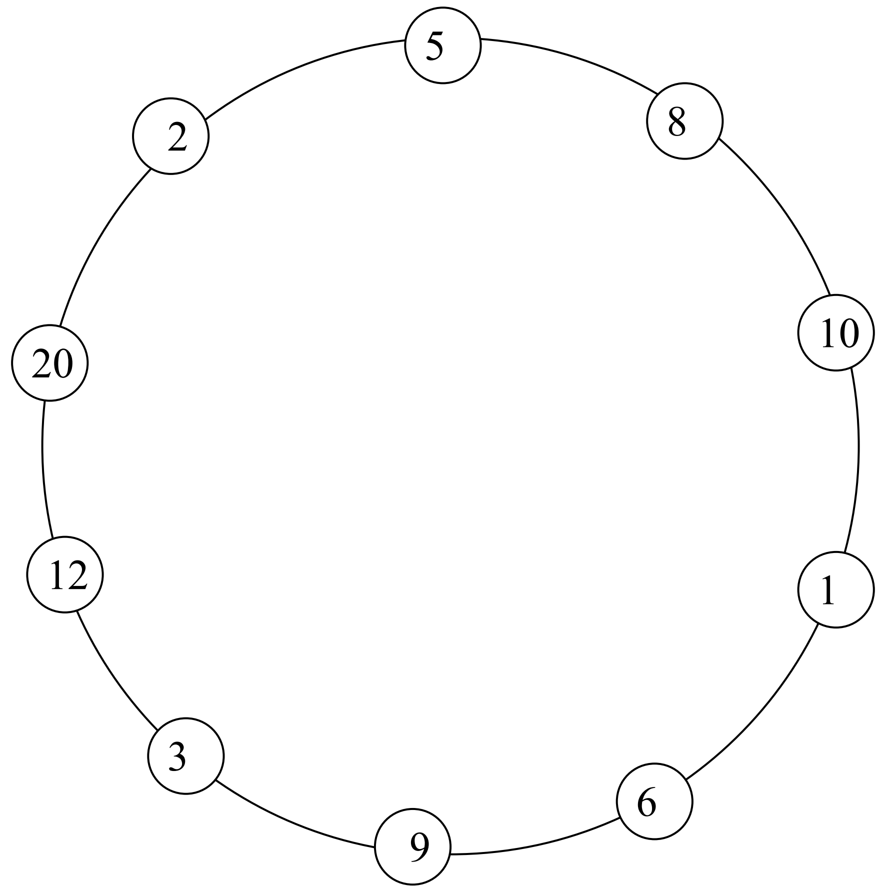
Simulation of the bidirectional algorithm with the same complexity.

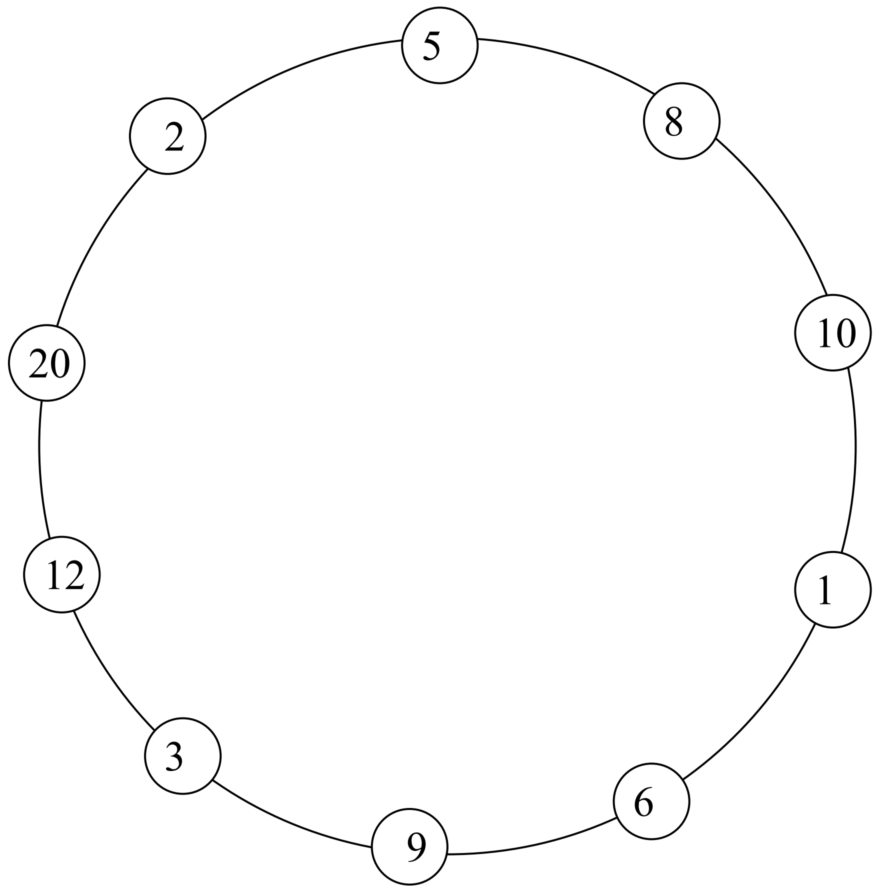
Examples ….

The Conjecture is false.

Unidirectional version






5
8
10
1
6
9
3
12
20
2

Alternating Steps

Basic idea: Alternating directions.

- Different *id*' s.
- Bidirectional ring and sense of direction.
- Local orientation.
- Message ordering.

send-left
begin-to-defeat (if possible)
send-right

Algorithm:

1. **Each entity** sends a message **to its right**. This message contains the entity' s own *id*.

2. Each entity compares the *id* it received from its left to its own *id*.

3. If its own *id* is greater than the received *id*, the entity becomes passive.

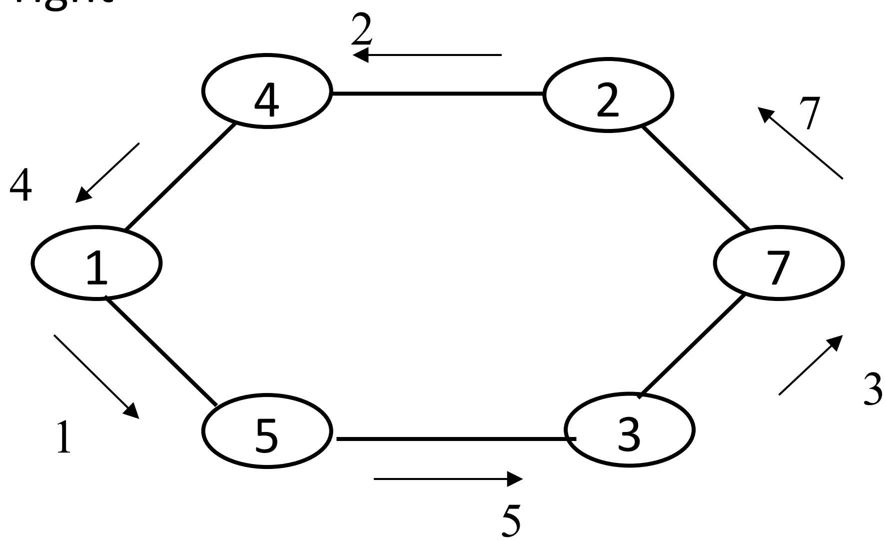
4. All entities that remained active (surviving) send their *id*s **to their left**.

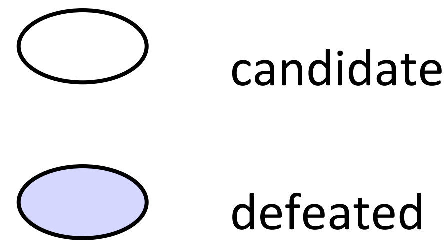
5. A surviving entity compares the *id* it received from its right with its own *id*.

6. If its own *id* is greater than the *id* it received, it becomes passive.

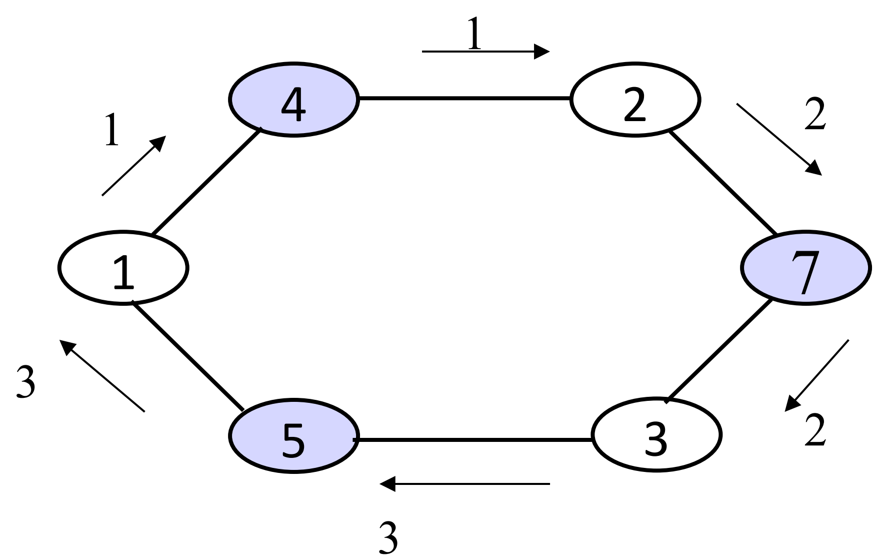
7. Go back to step 1 and repeat until an entity receives its own *id* and becomes *leader*.

Step 1: right



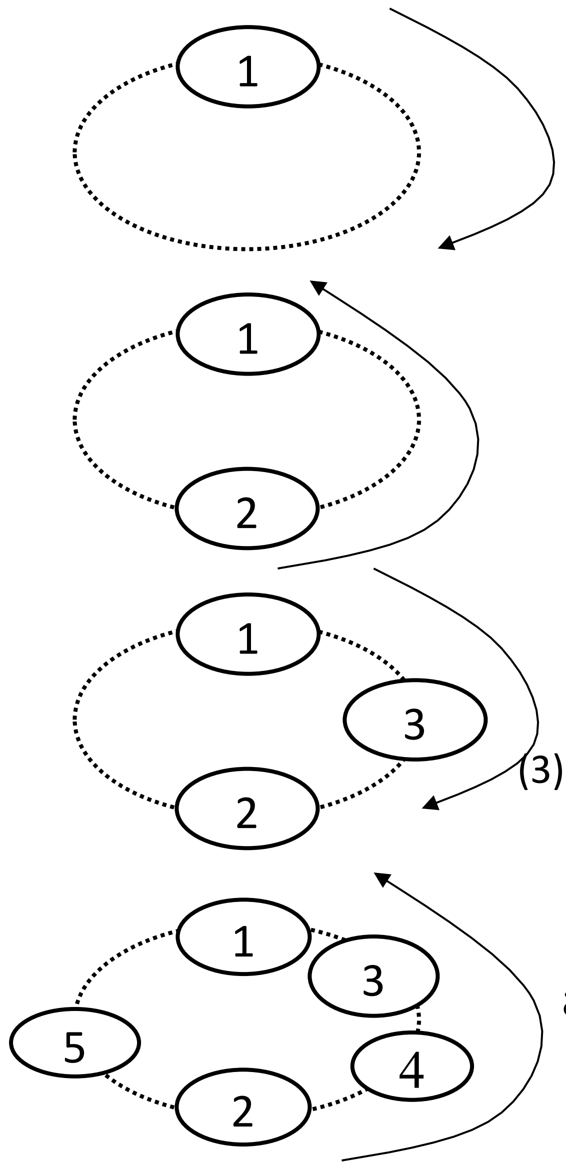


Step 2: left



Complexity

Analyze # of steps in worst case:



Last phase k — 1 active entity

Phase k - 1 — at least 2 active entities

(2) will become passive at the next step.

Phase k - 2 — at least 3 active entities

(3) must be there; otherwise, (2) would be killed.

Phase k - 2 — at least 5 active entities

1 2 3 5 8 13 21 ....

 # steps =

index of the lowest Fibonacci number >= n

F_1 = 1
F_2 = 2
F_3 = 3
F_4 = 5
F_5 = 8
. . .

F_k = i = ?
= approx. 1.45 $\log_2 n$

 # Messages = n for each step

Total = approx. 1.45 n $\log_2 n$

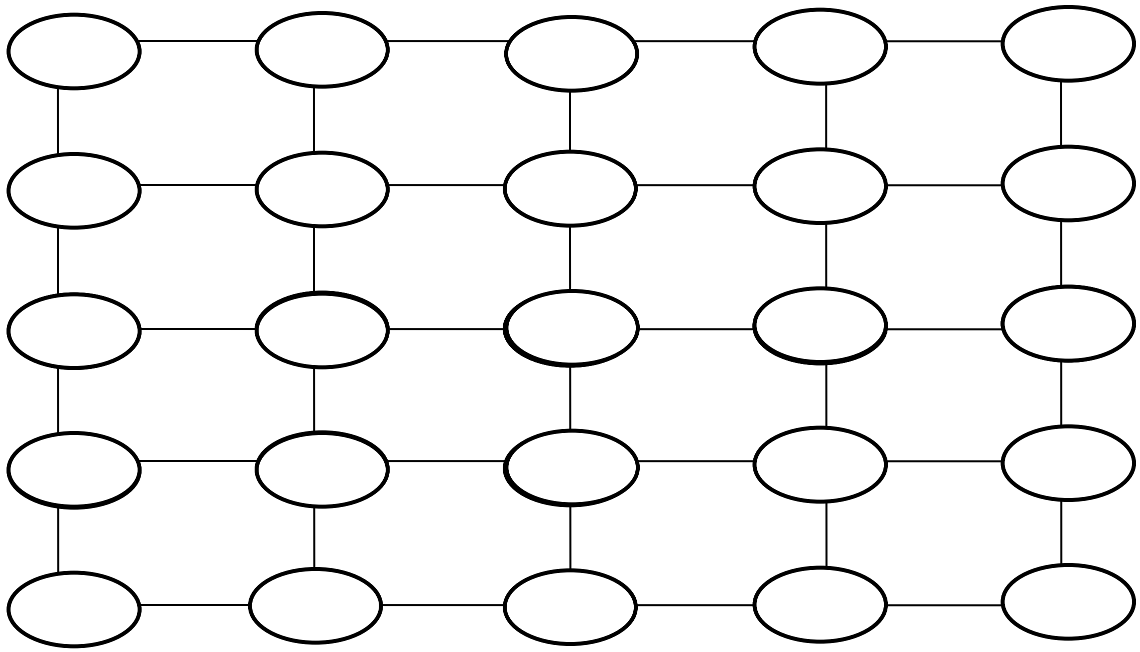
upper bounds

Bidirectional		Unidirectional	
LeLann (1977) "All the way"	n^2	LeLann (1977) Unidirectional simulation	n^2
Chang & Roberts (1979) "As far as you can" average case n log n	n^2	Chang & Roberts	n^2
Hirshberg & Sinclair (1980) stages message control	$7n \log n$		
Franklin (1982) stages	$2n \log n$	Dolev, Klawe & Rodeh Unidirectional simulation	$2n \log n$
Peterson (1982) Alternate	$1.44n \log n$	Peterson 1982 Unidirectional simulation	$1.44n \log n$
		Dolev, Klawe & Rodeh (1982)	$1.36n \log n$
		Higham, Przytycka (1984)	$1.22n \log n$

lower bounds

Burns	$0.5n \log n$
Pachl, Korach Rotem (1984)	$0.69n$ log

Mesh



If it is square mesh: n nodes = $n^{½} \times n^{½}$

m = O(n)

Asymmetric topology

corners

border

internal

Idea: Elect as a leader one of the four corners

Three phases:

1) Wake up

2) Election (on the border) among the corners

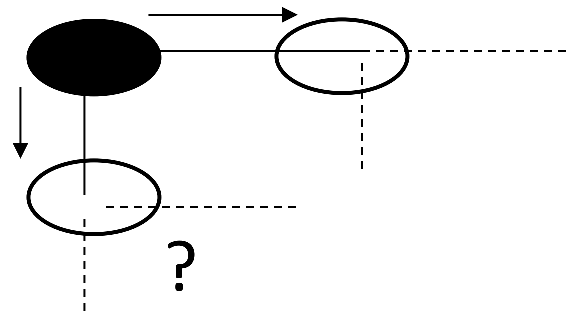
3) Notification

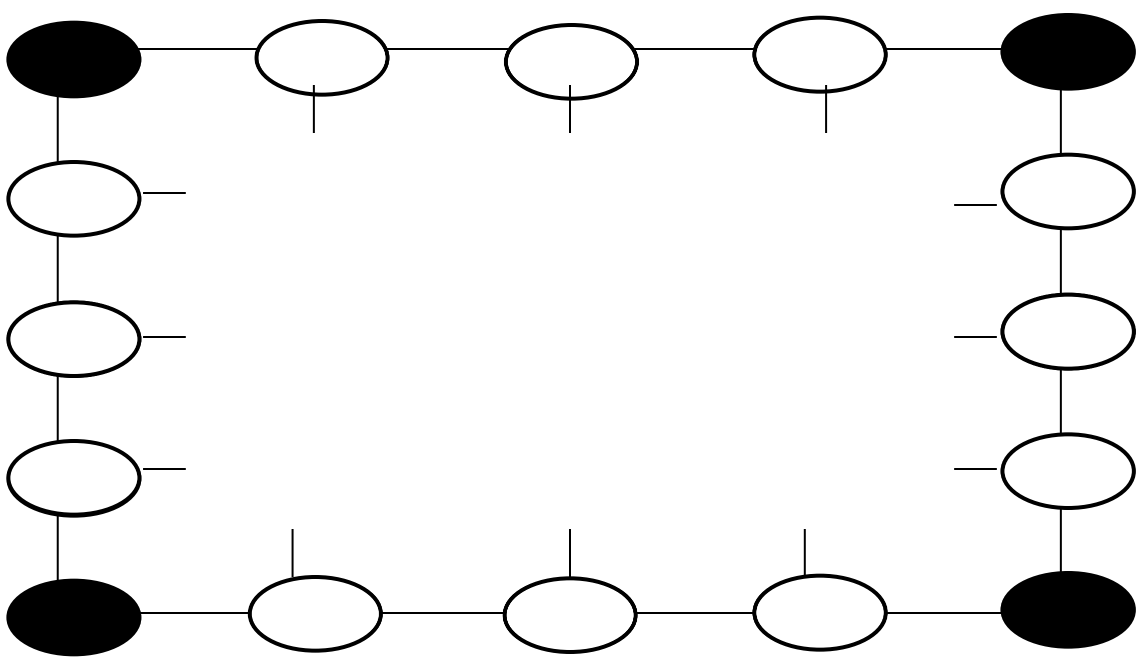
1) Wake up

- Each initiator send a wake-up to its neighbours
- A non-initiator receiving a wake up, sends it to
  its other neighbours

$$O(m) = O(n)$$

2) Election on the border started by the corners





$O(\sqrt{n})$

3) Notification

by flooding

$O(m) = O(n)$

TOT: $O(n)$

Torus