The Model & Basic Computations

Chapter 1 and 2

The Model

Broadcast

Spanning Tree Construction

Traversal

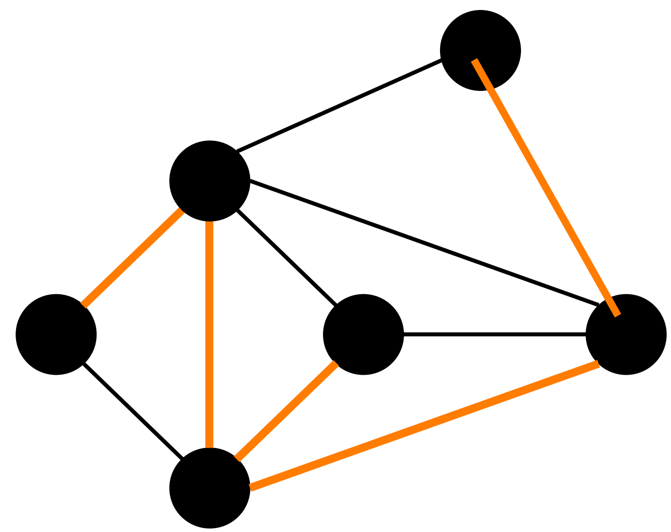
Wake-up

Spanning Tree Construction

A spanning tree T of a graph G = (V,E) is an acyclic subgraph of G such that T=(V,E') and E' $\subset$ E.

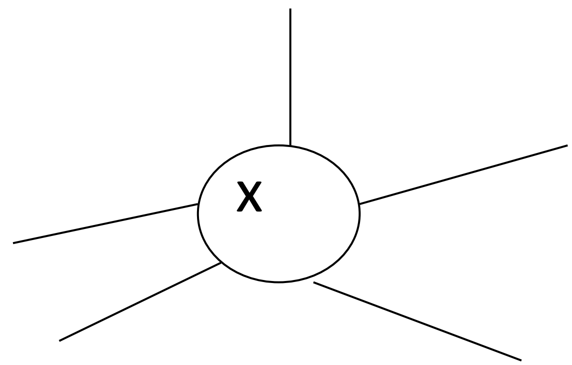
Assumptions:

single initiator
bidirectional links
total reliability
G connected



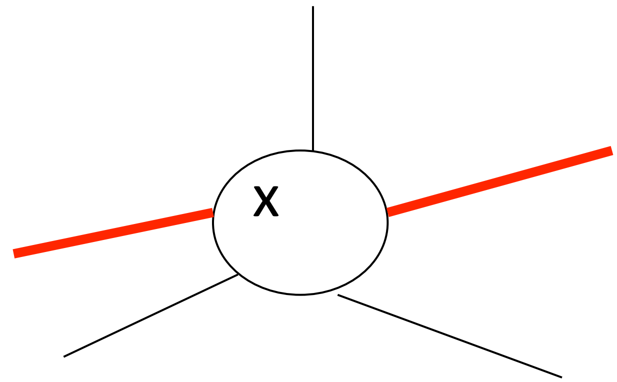
Protocol SHOUT

Initially: ∀ x, Tree-neighbors(x) = { }

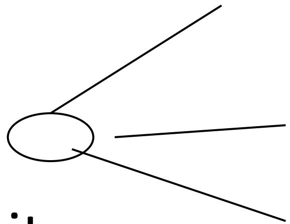


At the end:

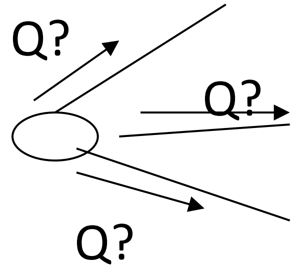
∀ x, Tree-neighbors(x) = {links that belong to the spanning tree }



1.

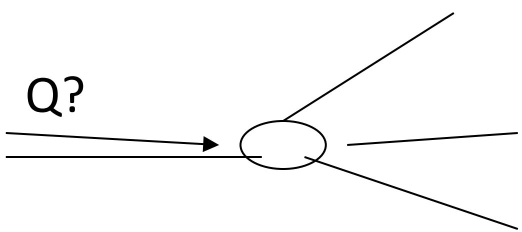


init

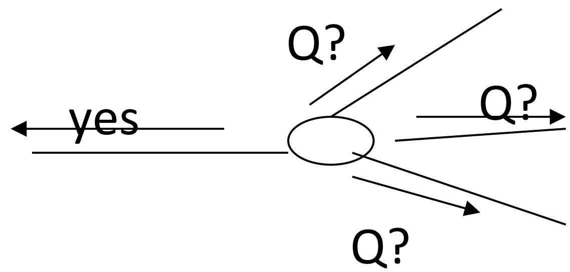


Q? = do you want to be my neighbour in the spanning tree ?

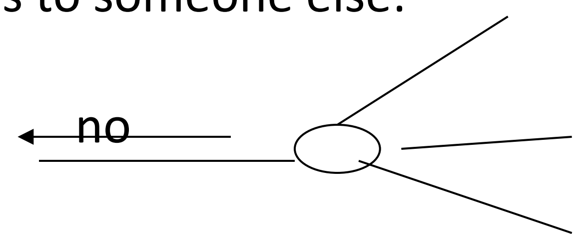
2.



If it is the first time:



If I have already answerd yes to someone else:



Example

States S={INITIATOR, IDLE, ACTIVE, DONE}
Sinit = {INITIATOR, IDLE}
Sterm = {DONE}

INITIATOR
Spontaneusly

```
        root:= true
        Tree-neighbours := { }
        send(Q) to N(x)
        counter:=0
        become ACTIVE
```

IDLE
receiving(Q)

```
        root:= false
        parent := sender
        Tree-neighbours := {sender}
        send(yes) to sender
        counter := 1
        if counter = |N(x)| then
                become DONE
        else
                send(Q) to N(x) – {sender}
                become ACTIVE
```

ACTIVE

receiving(Q)

send(no) to sender

receiving(yes)

Tree-neighbours:=

Tree-neighbours $\cup$ sender

counter := counter +1

if counter = |N(x)|

become DONE

receiving(no)

counter := counter +1

if counter = |N(x)|

become DONE

Note:

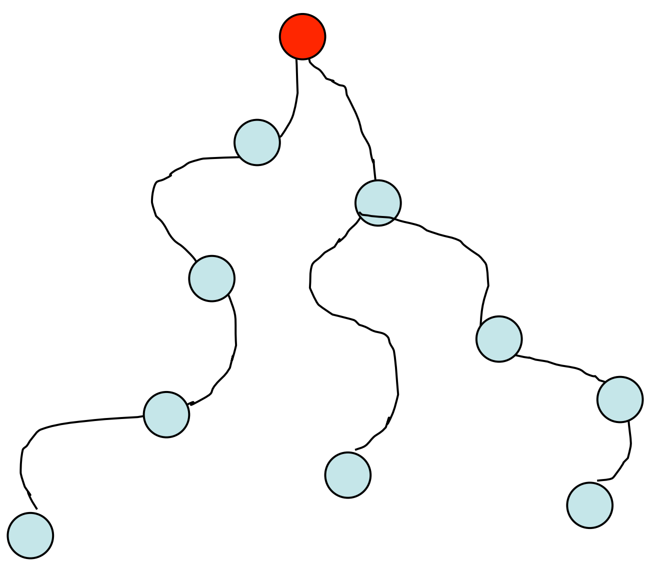
SHOUT = FLOOD +REPLY

Correctness and Termination

- If x is in Tree-neighbours of y, y is in Tree-neighbours of x

- If x sends YES to y, then x is in Tree-neighbour of y
  and is connected to the initiator by a chain of YES

- Every x (except the initiator) sends exactly one YES



Correctness and Termination

- If x is in Tree-neighbours of y, y is in Tree-neighbours of x

- If x sends YES to y, then x is in Tree-neighbour of y
  and is connected to the initiator by a chain of YES

- Every x (except the initiator) sends exactly one YES

The spanning graph defined by the Tree-neighbour relation is connected and contains all the entities

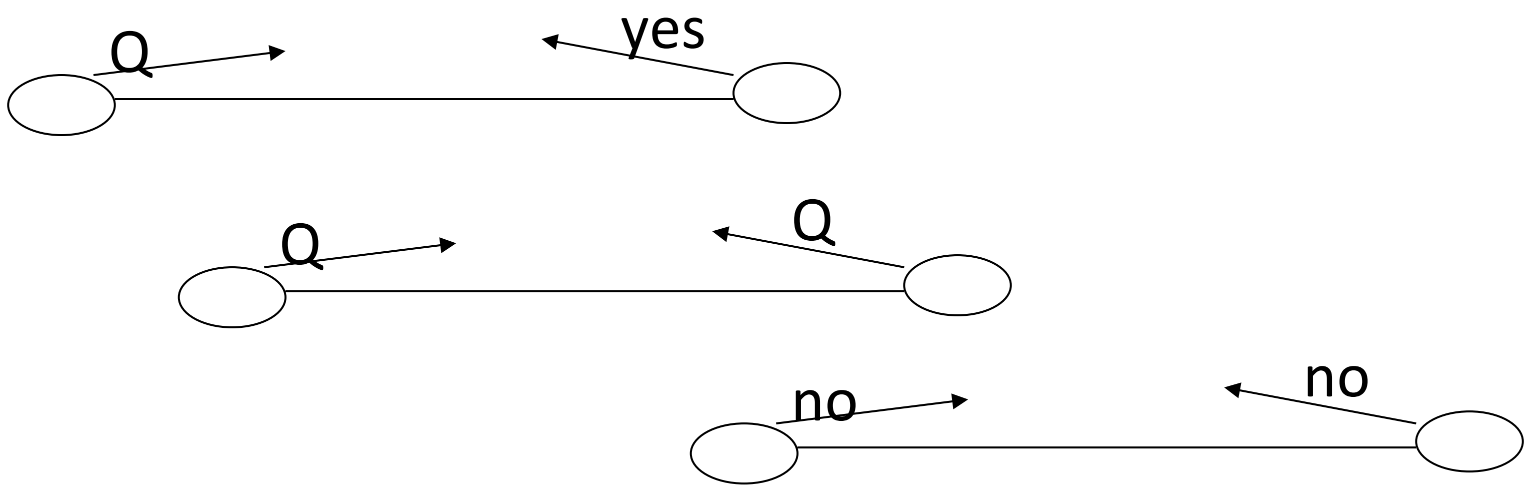
Note: local termination

Message Complexity

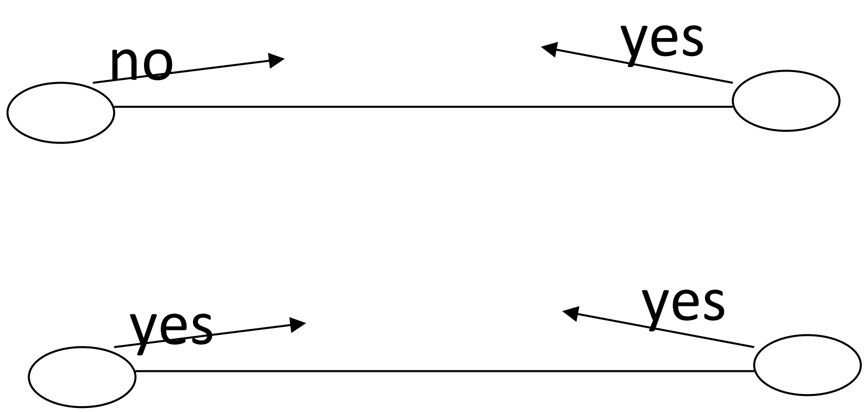
SHOUT = FLOOD + REPLY

Messages(SHOUT) = 2 M(FLOOD)

Possible situations



Impossible situations

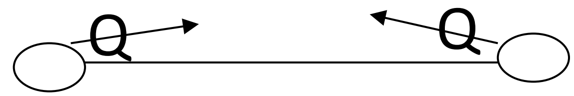


Message Complexity - worst case

Total n. of Q:

(n-1)

only one Q on the ST links



m -(n-1)

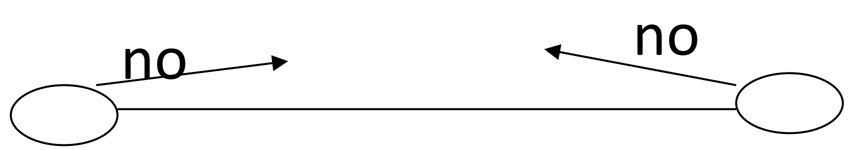
on the other links

Total: $2(m - (n-1)) + (n-1)$

$= 2m - n + 1$

Message Complexity - worst case

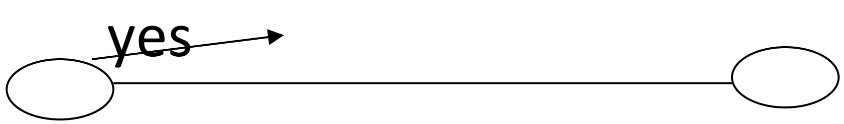
Total n. of NO:



as many as Q---Q

2(m - (n-1))

Total n. of YES:



Exactly: (n-1)

Message Complexity - worst case

$2m - n + 1 + 2(m - (n-1)) + n-1$

$= 2m - n + 1 + 2m - 2n + 2 + n - 1$

$= 4m - 2n + 2$

$$\boxed{\text{Messages(SHOUT)} = 4m - 2n + 2}$$

In fact: M(SHOUT) = 2 M(FLOOD) = $2(2m-n+1)$

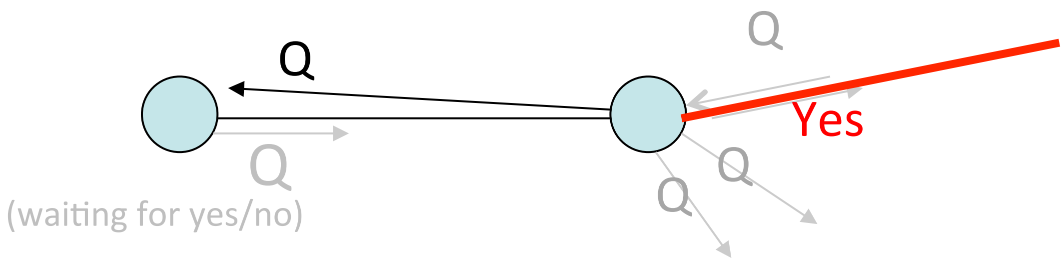
$\Omega(m)$ is a lower bound also in this case

Spanning Tree Construction

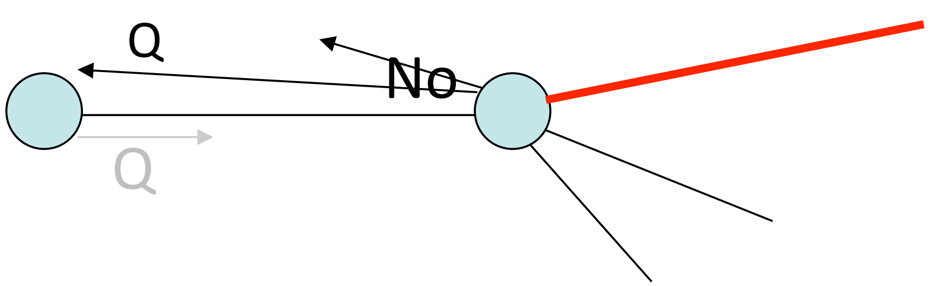
Without "NO"

Protocol SHOUT+

What does the reception of a question mean ?



It means that to my question, you will **certainly** reply **NO**



No need to send negative answers:
I can understand you cannot be part of the ST simply by your **Question**

States S={INITIATOR, IDLE, ACTIVE, DONE}
Sinit = {INITIATOR, IDLE}
Sterm = {DONE}

INITIATOR
Spontaneusly

```
        root:= true
        Tree-neighbours := { }
        send(Q) to N(x)
        counter:=0
        become ACTIVE
```

IDLE
receiving(Q)

```
        root:= false
        parent := sender
        Tree-neighbours := {sender}
        send(yes) to sender
        counter := 1
        if counter = |N(x)| then
                become DONE
        else
                send(Q) to N(x) – {sender}
                become ACTIVE
```

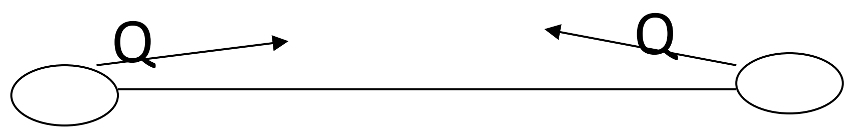
ACTIVE

receiving(Q) **(to be interpreted as NO)**

counter := counter +1
if counter = |N(x)|
become DONE

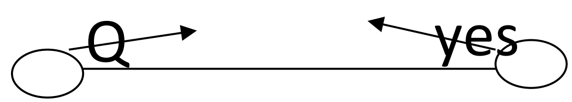
receiving(yes)
Tree-neighbours:=
Tree-neighbours ∪ {sender}
counter := counter +1
if counter = |N(x)|
become DONE

On each link there will be exactly 2 messages:



either

or



Messages(SHOUT+) = 2m

Much better than:

Messages(SHOUT) = 4m -2n + 2

Spanning Tree Construction

With Notification

States S={INITIATOR, IDLE, ACTIVE, DONE}
Sinit = {INITIATOR, IDLE}
Sterm = {DONE}

INITIATOR
Spontaneusly

```
root:= true
Tree-neighbours := { }
send(Q) to N(x)
counter:= 0
ack-counter:= 0
become ACTIVE
```

IDLE
receiving(Q)

```
root:= false
parent := sender
Tree-neighbours := {sender}
send(yes) to sender
counter := 1
ack-counter:= 0
if counter = |N(x)| then
        CHECK
else
        send(Q) to N(x) – {sender}
become ACTIVE
```

ACTIVE

receiving(Q)

counter := counter +1

if counter = |N(x)| and not root then

CHECK

receiving(yes)

Tree-neighbours:=

Tree-neighbours ∪ {sender}

counter := counter +1

if counter = |N(x)| and not root then

CHECK

ACTIVE (cont)

receiving(Ack)

ack-counter:= ack-counter +1

if counter = |N(x)| /* indicate tree-neighbors is done
if root then

if ack-counter = |Tree-neighbours|

send(Terminate) to Tree-neighbours

become DONE

else if ack-counter = |Tree-neighbours| - 1

send(Ack) to parent

receiving(Terminate)

send(Terminate) to Children

become DONE

CHECK

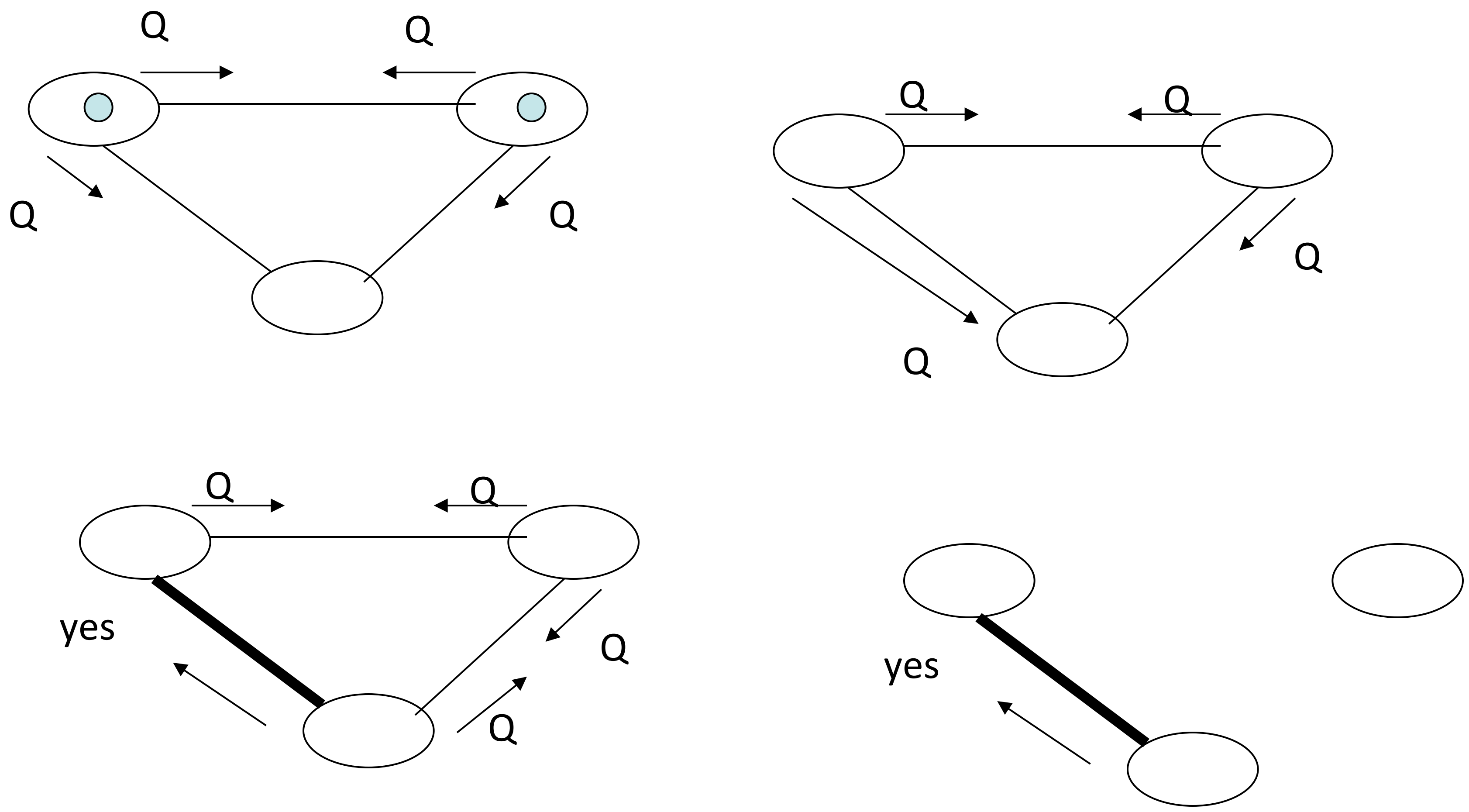
If I am a leaf

(* that is: Children:= Tree-neighbours – {parent}

if Children = emptyset *)

send(Ack) to parent

What happens if there are multiple initiators ?



An election is needed to have a unique initiator.

or

Another protocol has to be devised.

NOTE: Election is impossible if the nodes do not have distinct IDs