Distributed Computing
Xidian 2021

Prof. Mirosław Kutyłowski

8: Byzantine agreement

Consensus with adversarial nodes

A Byzantine node can behave in an arbitrary way, not following the protocol.

Consensus value – must be the value proposed by ONE of correct nodes.

One Byzentine node only

Algorithm 17.9 Byzantine Agreement with $f = 1$.

Code for node u, with input value x:

Round 1

Byzantine

Send $\texttt{tuple}(u, x)$ to all other nodes

Receive $\texttt{tuple}(v, y)$ from all other nodes v

Store all received $\texttt{tuple}(v, y)$ in a set S_u

Round 2

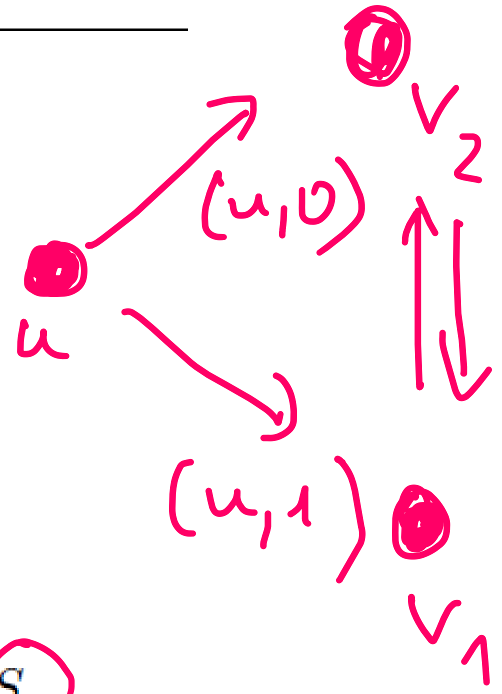
Send set S_u to all other nodes

Receive sets S_v from all nodes v

T = set of $\texttt{tuple}(v, y)$ seen in at least two sets S_v, including own S_u

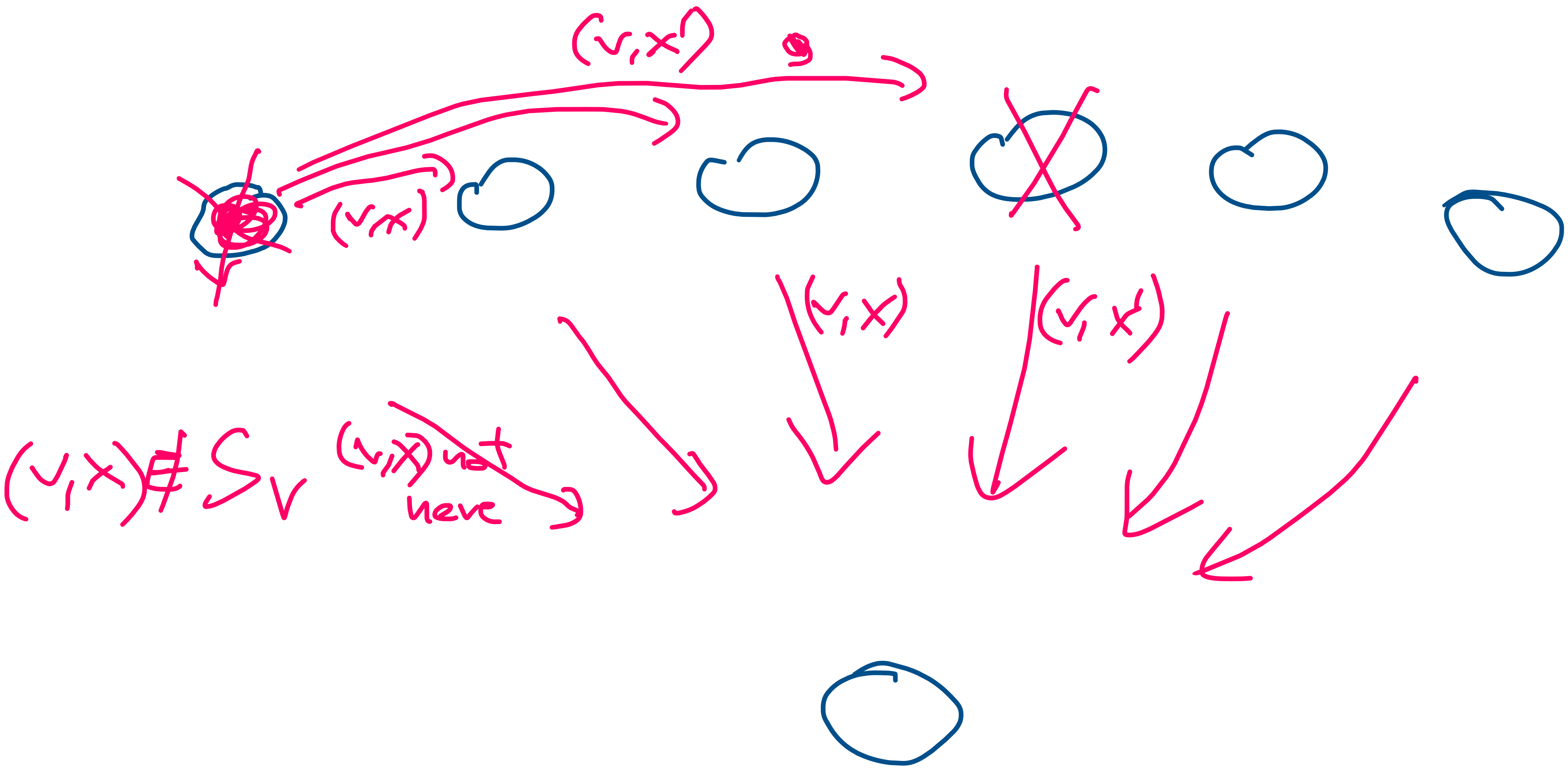
Let $\texttt{tuple}(v, y) \in T$ be the tuple with the smallest value y

Decide on value y



tuple $q = (v, x)$

honesty: but not in two S_u


(v,x')
(v,x)
(v,x)
(v,x')
$(v,x) \notin S_v$
(v,x) not here

Property for n≥4

All nodes hold the same T:

- If there are 3 correct nodes, then 2 correct values appear in 2 sets S_u
- The values sent by the Byzantine node are not in T if it sends different values.

Corollary: one value will be chosen

For n=3 one cannot reach agreement

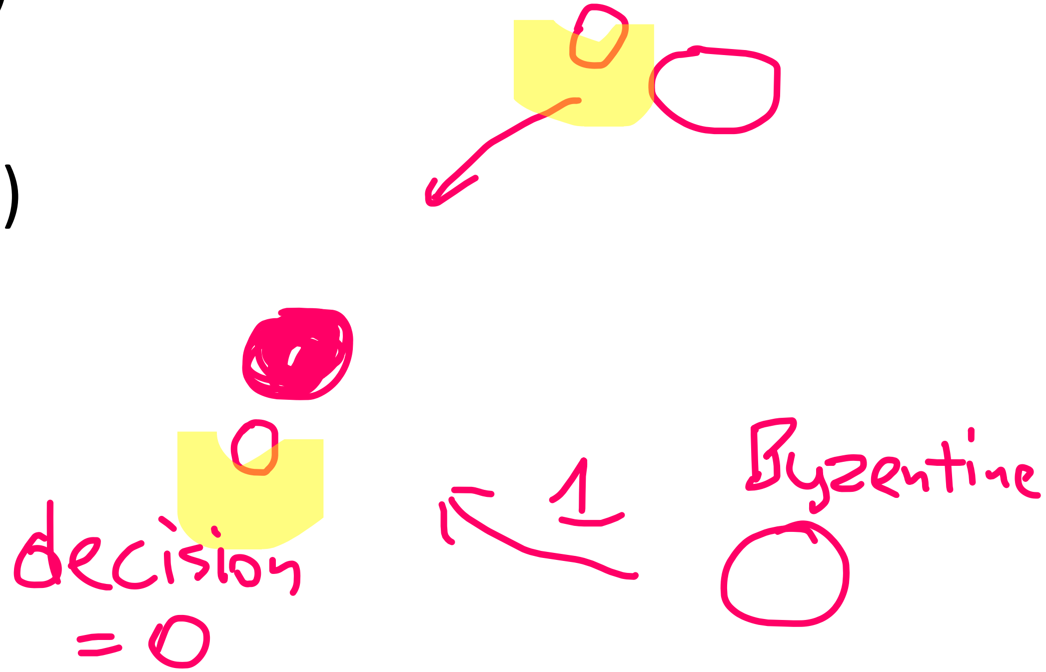
Property. A correct node u must decide on its value if node w supports it (while v disagrees – v might be byzantine)

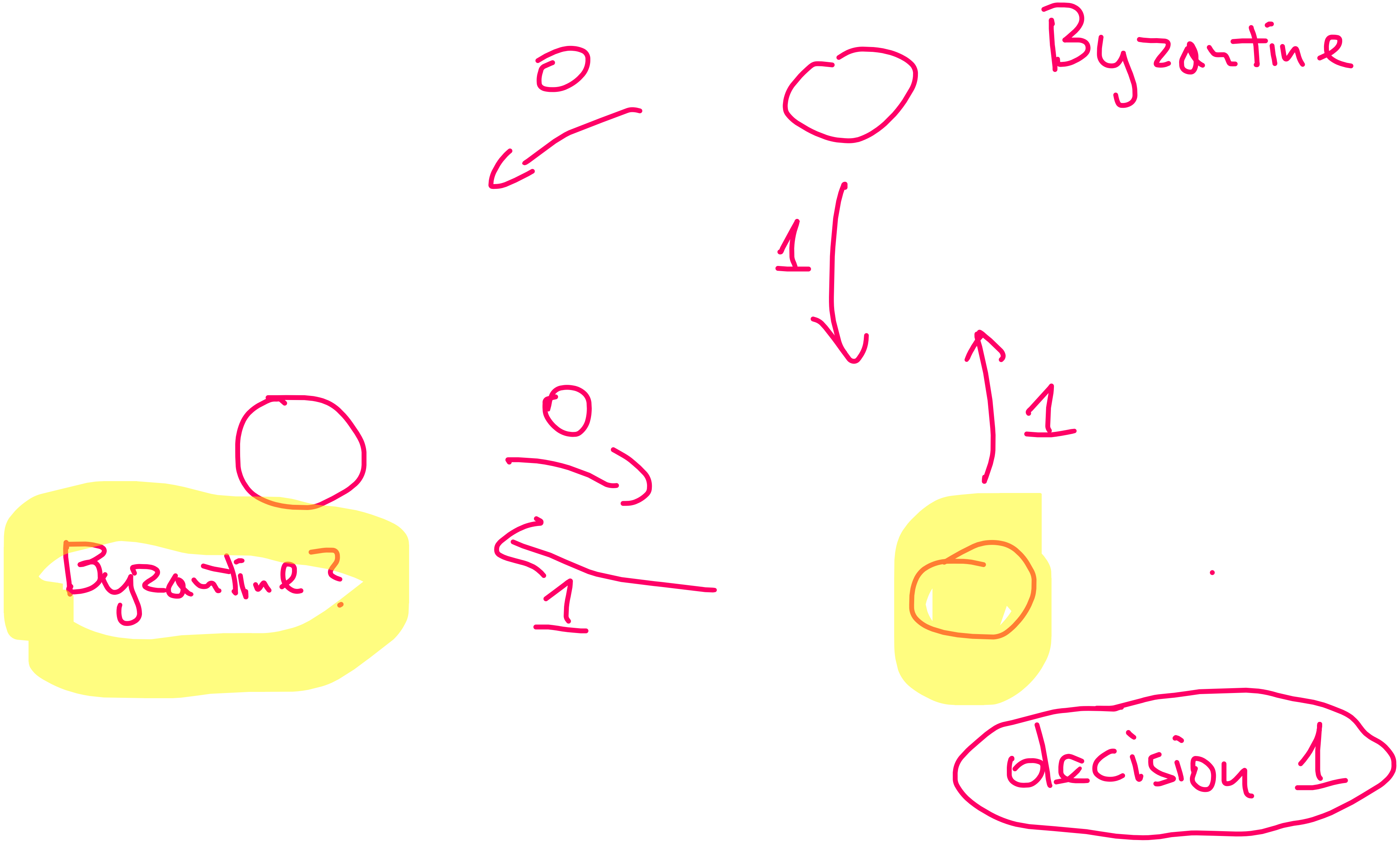
Honest: u (with input 0) and v (with input 1)

Byzantine: w says 0 to u

w says 1 to w

u decides on 0, v decides on 1



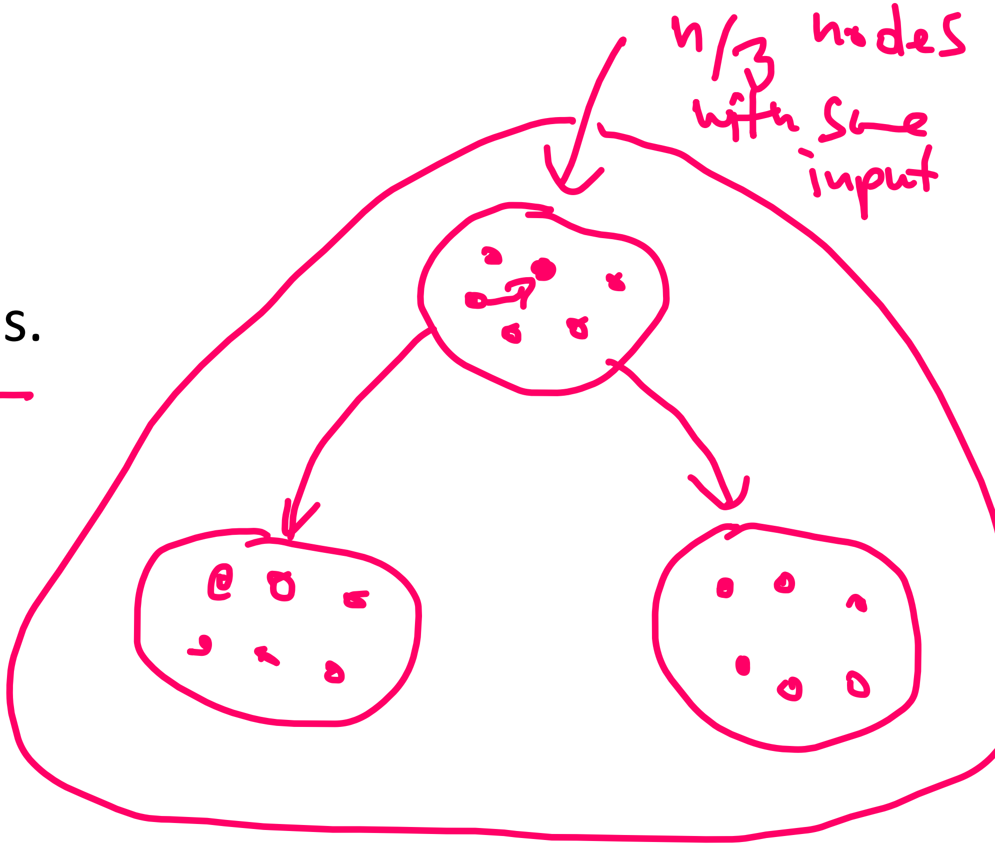

0
Byzantine
1
1
0
Byzantine?
1
decision 1

Fundamental theorem

Theorem 17.13. *A network with n nodes cannot reach byzantine agreement with $f \geq n/3$ byzantine nodes.*

Proof.
Assume there is such agreement protocol A for n nodes.

Emulate it by 3 nodes – each emulating n/3 nodes



King's algorithm

Algorithm 17.14 King Algorithm (for $f < n/3$)

x = my input value
for phase = 1 to $f + 1$ **do**
Round 1
Broadcast `value`(x)
Round 2
if some `value`(y) at least $n - f$ times **then**
Broadcast `propose`(y)
end if
if some `propose`(z) received more than f times **then**
$x = z$
end if
Round 3
Let node v_i be the predefined king of this phase i
The king v_i broadcasts its current value w
if received strictly less than $n - f$ `propose`(x) **then**
$x = w$
end if
end for

King 1
King 2
⋮
King($f+1$)

Same value input

Algorithm 17.14 King Algorithm (for $f < n/3$)

```
x = my input value
for phase = 1 to f + 1 do
    Round 1
    Broadcast value(x)
    Round 2
    if some value(y) at least n − f times then
        Broadcast propose(y)
    end if
    if some propose(z) received more than f times then
        x = z
    end if
    Round 3
    Let node v_i be the predefined king of this phase i
    The king v_i broadcasts its current value w
    if received strictly less than n − f propose(x) then
        x = w
    end if
end for
```

The king's value will never be accepted.

Nodes stick to their input values

$x = 0$ for all

value(0) from honest nodes

$n - f > f$

$n \geq 2f$

Proposing values

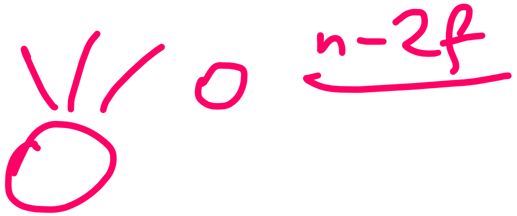
Lemma 17.16. *If a correct node proposes x, no other correct node proposes y, with $y \neq x$, if $n > 3f$.*

Proof

A correct node must receive n-f messages, so at least n-2f from correct nodes

Two different proposal require the following number of nodes:

2(n-2f) correct nodes, together 2n-3f>n nodes !?



Kings

At least one king correct

Lemma: after the round with the correct king the nodes do not change their values anymore.

Proof

Easy case: all correct nodes take the king's value

Crucial case: line 12 executed by only SOME correct nodes

is the King's value the same as for other correct nodes??

Crucial case

- thereby we assume that some node received a proposal at least n-f times
- … so from at least n-$2f$ correct nodes
- as n-$2f>f$ every node received a proposal from more than f nodes
- … and therefore every correct node will change the value according to the proposal

Lower bound – no algorithm with f rounds determining minimum

- u_1 sends to u_2 at round 1 and then crashes
- u_2 sends to u_3 at round 2 and then crashes
- …
- u_{f-1} sends to u_f at round f and then crashes

- Only u_f will see the minimum value

Byzantine agreement, asynchronous model

Algorithm 17.21 Asynchronous Byzantine Agreement (Ben-Or, for $f < n/9$)

```
1:  x_i ∈ {0,1}            ◁ input bit
2:  r = 1                  ◁ round
3:  decided = false
4:  Broadcast propose(x_i,r)
5:  repeat
6:    Wait until n − f propose messages of current round r arrived
7:    if at least n − 2f propose messages contain the same value x then
8:      x_i = x, decided = true
9:    else if at least n − 4f propose messages contain the same value x then
10:     x_i = x
11:   else
12:     choose x_i randomly, with Pr[x_i = 0] = Pr[x_i = 1] = 1/2
13:   end if
14:   r = r + 1
15:   Broadcast propose(x_i,r)
16: until decided (see Line 8)
17: decision = x_i
```