Description

Each phase has two rounds, round 1 and round 2.
$R[p, t]$ is a *multi-set* of received values in round t at phase p.
$\text{MAJ}(R)$ = the number of occurrence of the majority non-? value in a multi-set R.
$\text{MAJV}(R)$ = the majority non-? value in a multi-set R.
Line 33 makes sure that every node flips the coin same number of times.

(i, p, 1, *) == round-1 messages == State messages
(i, p, 2, *) == round-2 messages == Vote messages

The verbose version of Weak-MVC and its helper function

Algorithm 4 IDK-BC: Code for node i

Local Variables: /* These variables can be accessed and modified by any thread at i. */

x	▷input at node i, $\{0, 1\}$
$v[t, 1]$	▷a vector of local state at round 1, $\{0, 1\}$
$v[t, 2]$	▷a vector of local state at round 2, $\{0, 1, ?\}$
p	▷phase, integer

```
/* All messages tagged with seq */
When IDK-BC(v, seq) is invoked:
/* Initial Phase */
1: Send (i, 0, 1, v) to all
2: wait until |R[0, 1]| ≥ n − f
3: if MAJ(R[0, 1]) ≥ ⌊n/2⌋ + f + 1 then
4:    Send (i, MAJV(R[0, 1]), DEC) to all
5:    Return MAJV(R[0, 1])
6: else if MAJ(R[0, 1]) ≥ ⌊n/2⌋ + 1 then
7:    v[0, 2] ← 1
8: else
9:    v[0, 2] ←?
An optimization for increasing 1's,
 which might not be necessary.
10: Send (i, 0, 2, v[0, 2]) to all
11: wait until |R[0, 2]| ≥ n − f
12: if MAJ(R[0, 2]) ≥ f + 1 then
13:    m ← FindReturnValue(0, 2)
14:    Send (i, m, DEC) to all
15:    Return m
16: else if MAJ(R[0, 2]) ≥ 1 then
17:    v[1, 1] ← MAJV(R[0, 2])
18: else
19:    v[1, 1] ← 0

/* Same as BO-BC except for the return step*/
20: p ← 1
21: while TRUE do
      /* Round 1 */
22:   Send (i, p, 1, v[p, 1]) to all
23:   wait until |R[p, 1]| ≥ n − f
24:   if MAJ(R[p, 1]) ≥ ⌊n/2⌋ + f + 1 then
25:      m ← FindReturnValue(p, 1)
26:      Send (i, m, DEC) to all
27:      Return m
28:   else if MAJ(R[p, 1]) ≥ ⌊n/2⌋ + 1 then
29:      v[p, 2] ← MAJV(R[p, 1])
30:   else
31:      v[p, 2] ←?

      /* Round 2 */
32:   Send (i, p, 2, v[p, 2]) to all
33:   randBit ← COMMONCOINFLIP()
34:   wait until |R[p, 2]| ≥ n − f
35:   if MAJ(R[p, 2]) ≥ f + 1 then
36:      m ← FindReturnValue(p, 2)
37:      Send (i, m, DEC) to all
38:      Return m
39:   else if MAJ(R[p, 2]) ≥ 1 then
40:      v[p + 1, 1] ← MAJV(R[p, 2])
41:   else
42:      v[p + 1, 1] ← randBit
43:   p ← p + 1          ▷Proceed to next phase

/* Helper procedure */
Procedure FindReturnValue(p, r)
44: if MAJV(R[p, r]) = 1 then
45:    if MAJ(R[0, 1]) ≥ ⌊n/2⌋ + 1 then
46:       m ← MAJV(R[0, 1])
47:    else
48:       m ← proposal from j who has 1 in R[0, 2]
49:    Return m
50: else
51:    Return ⊥

/* Event handler: executing in background */
Upon receiving (j, p, t, b) from node j:
52: Add b to R[p, t]
Upon receiving (j, b, DEC) from node j:
53: Return b
```