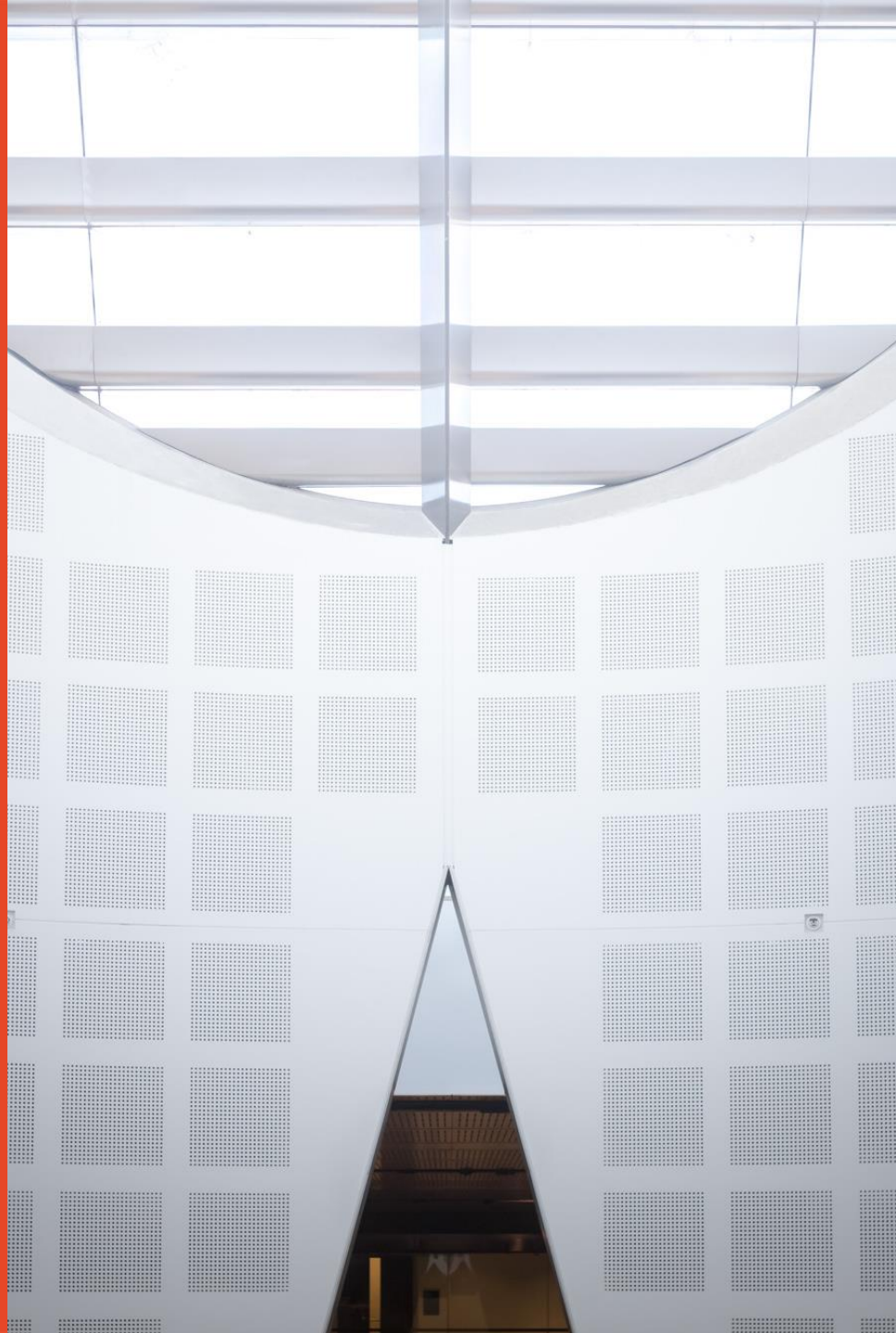
COMP3221: Distributed Systems

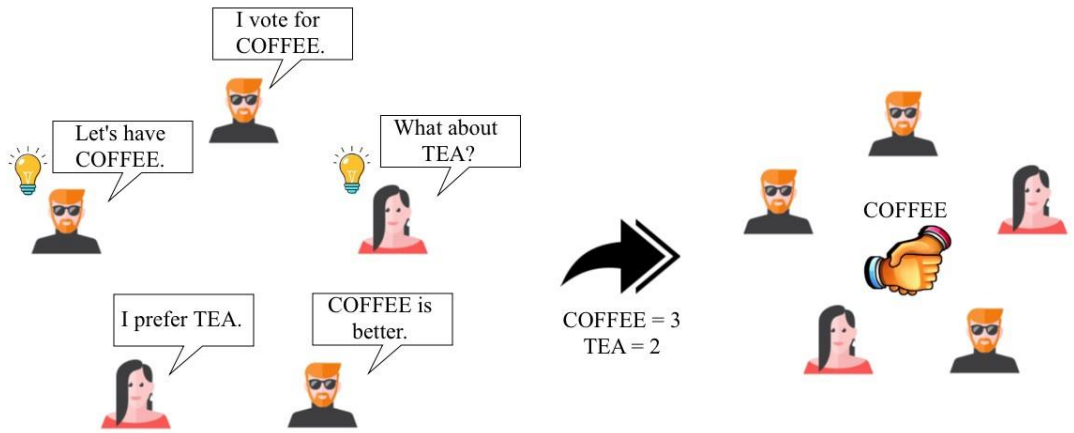
Lab 8: Concensus

Long Tan Le

THE UNIVERSITY OF SYDNEY



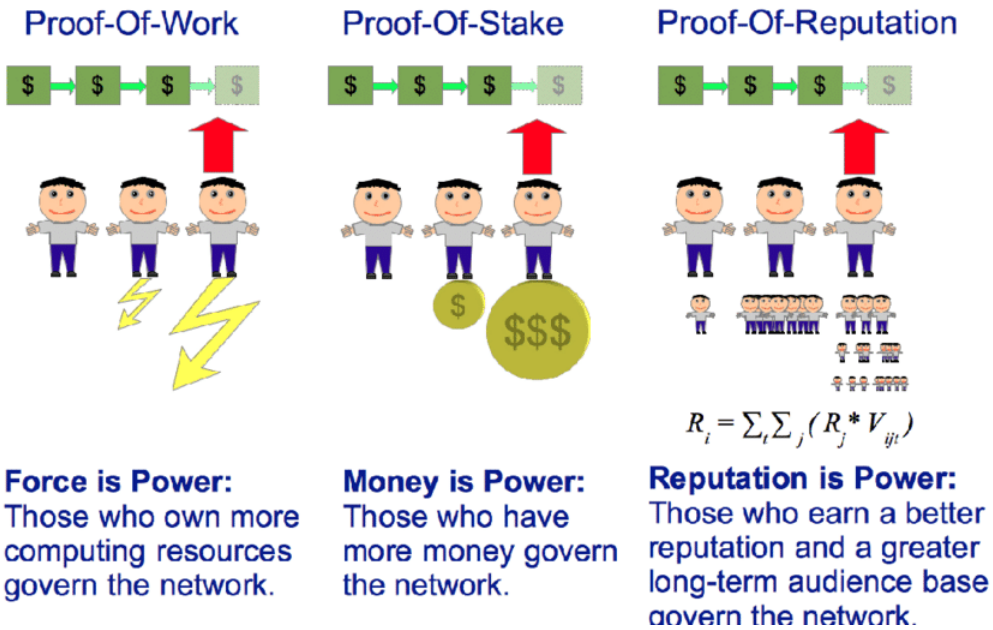
Consensus in Distributed Systems



- Consensus is a general agreement on a decision made by the majority of those involved.
- Three conditions to achieve distributed consensus
  - **Agreement**: All non-faulty processes must agree on the same value.
    - *Every person who didn't leave the conversation (non-faulty process) agrees to drink coffee.*
  - **Termination**: Every process will eventually decide some value.
    - *Everyone in the group promises to make a decision after discussing it for no more than 10 mins.*
  - **Validity**: All the processes must decide on one of the values that were proposed at the beginning.
    - *If they eventually decide on a drink, it'll be one that someone in the group originally proposed.*

Consensus in Blockchain

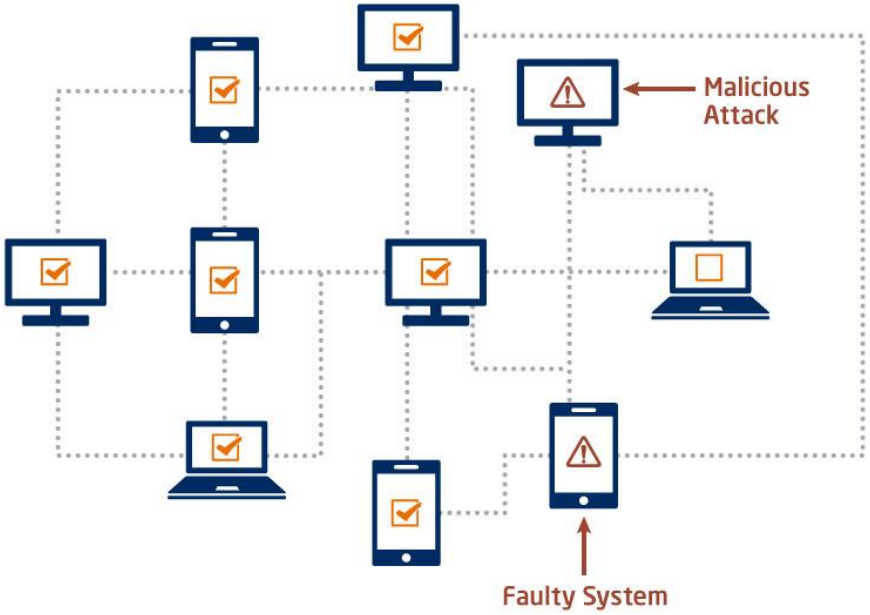
- Decentralised Blockchain need a way for users to agree on the current state of the blockchain
- Several different blockchain consensus mechanisms have been proposed
  - Proof of Work:
  - Proof of Stake
  - Delegate Proof of Stake
  - Practical Byzantine Fault Tolerance
  - Direct Acyclic Graph
  - Proof of Elapsed Time



Consensus Problems

- Some of nodes might get failed (**Crash fault**) or starts behaving abnormally (**Byzantine Fault**)
  - There are n processes, m of which may be faulty (m < n)
  - The task is to make all the Non-faulty processes agree on some value(s) even **in the presence of the faulty processes.**

- Solutions:
  - Consensus Without Any Fault (Failure-Free Consensus)
  - **Consensus With at most f Crash Faults (Crash-Tolerant Consensus)**
  - Consensus With at most f Byzantine Faults



Consensus With at most f Crash Faults

1. Algorithm tolerates at most **f** failures, out of **n** nodes (f < n)
2. Each process maintains the set of values **V** proposed by other processes
3. In every round a process: Sends to all other processes the values from **V** that it has not sent before
4. **After f+1 rounds** each process decides on the minimum value in V

P_i::
var
V: set of values initially $\{v_i\}$;

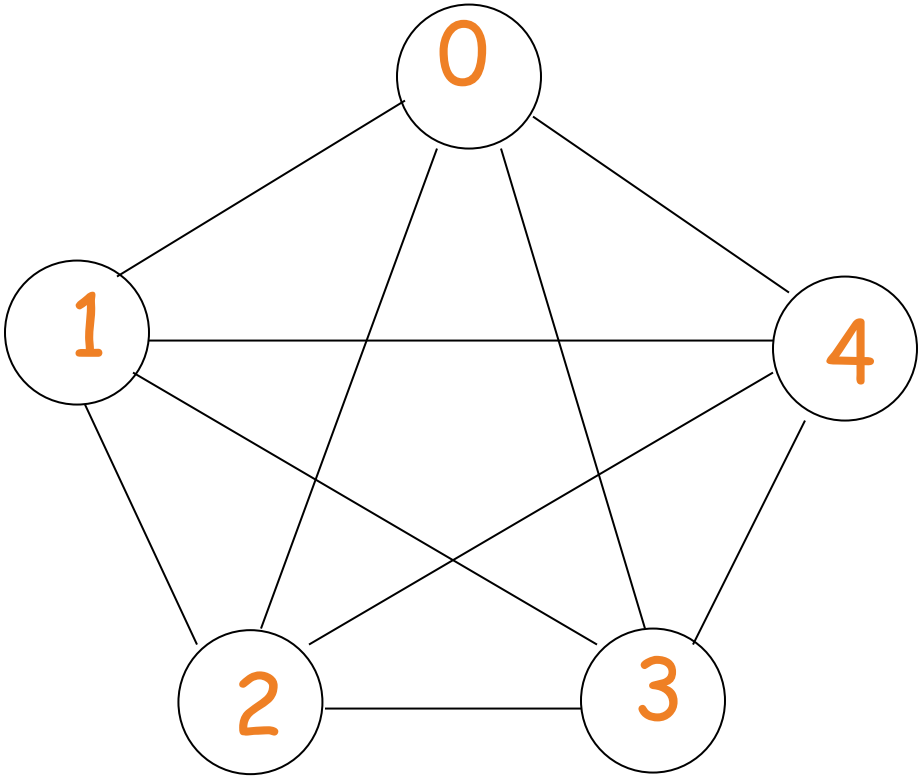
for $k := 1$ to $f + 1$ **do**
send $\{v \in V \mid P_i$ has not already sent $v\}$ to all;
receive S_j from all processes $P_j, j \neq i$;
$V := V \cup S_j$;
endfor;

$y := min(V)$;

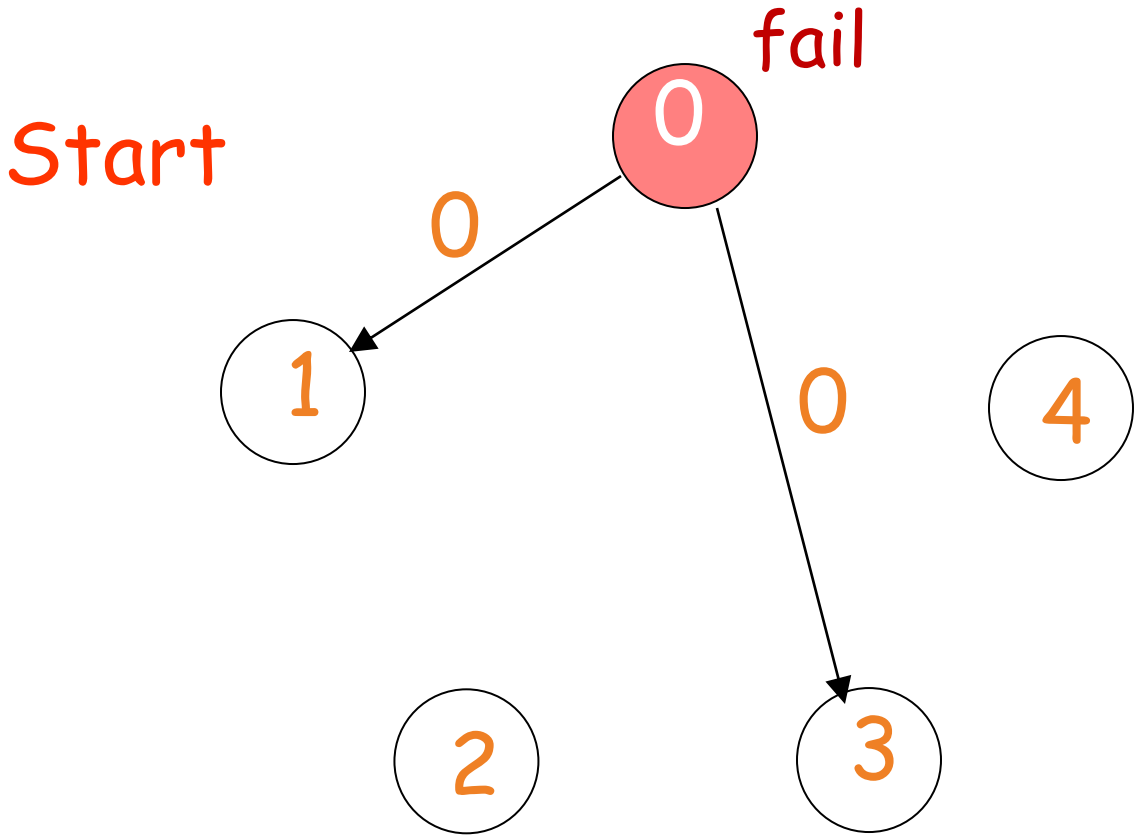
- **Agreement:** Non-faulty processes agree on the same value (min(V))
- **Termination:** By the code, finish in round f+1
- **Validity:** All processes decide on one of the values that were proposed (in V)

Example: f=1 failures, f+1 = 2 rounds needed

Start

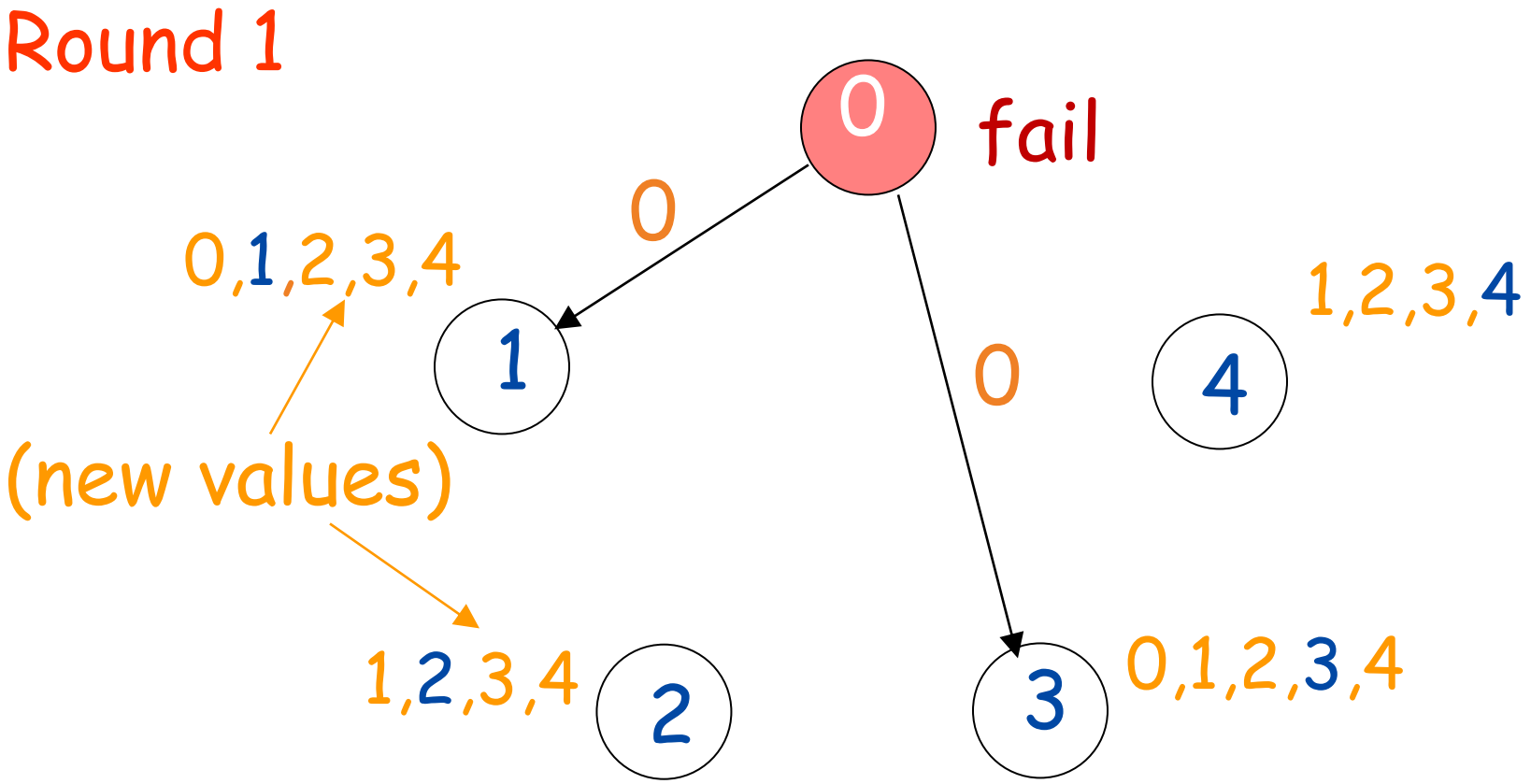


Example: f=1 failures, f+1 = 2 rounds needed



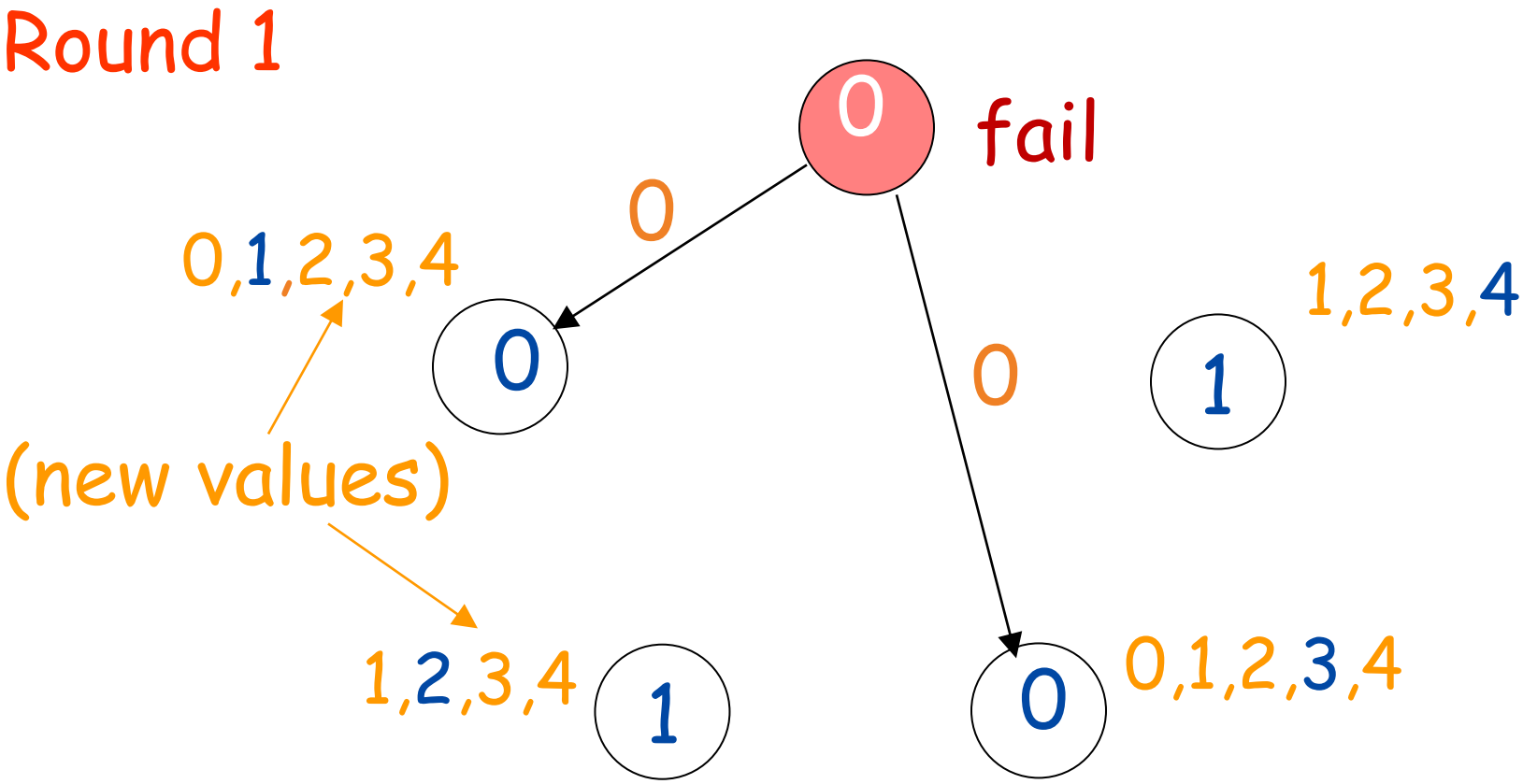
The failed processor doesn't broadcast its value to all processors

Example: f=1 failures, f+1 = 2 rounds needed



Broadcast all new values to every process

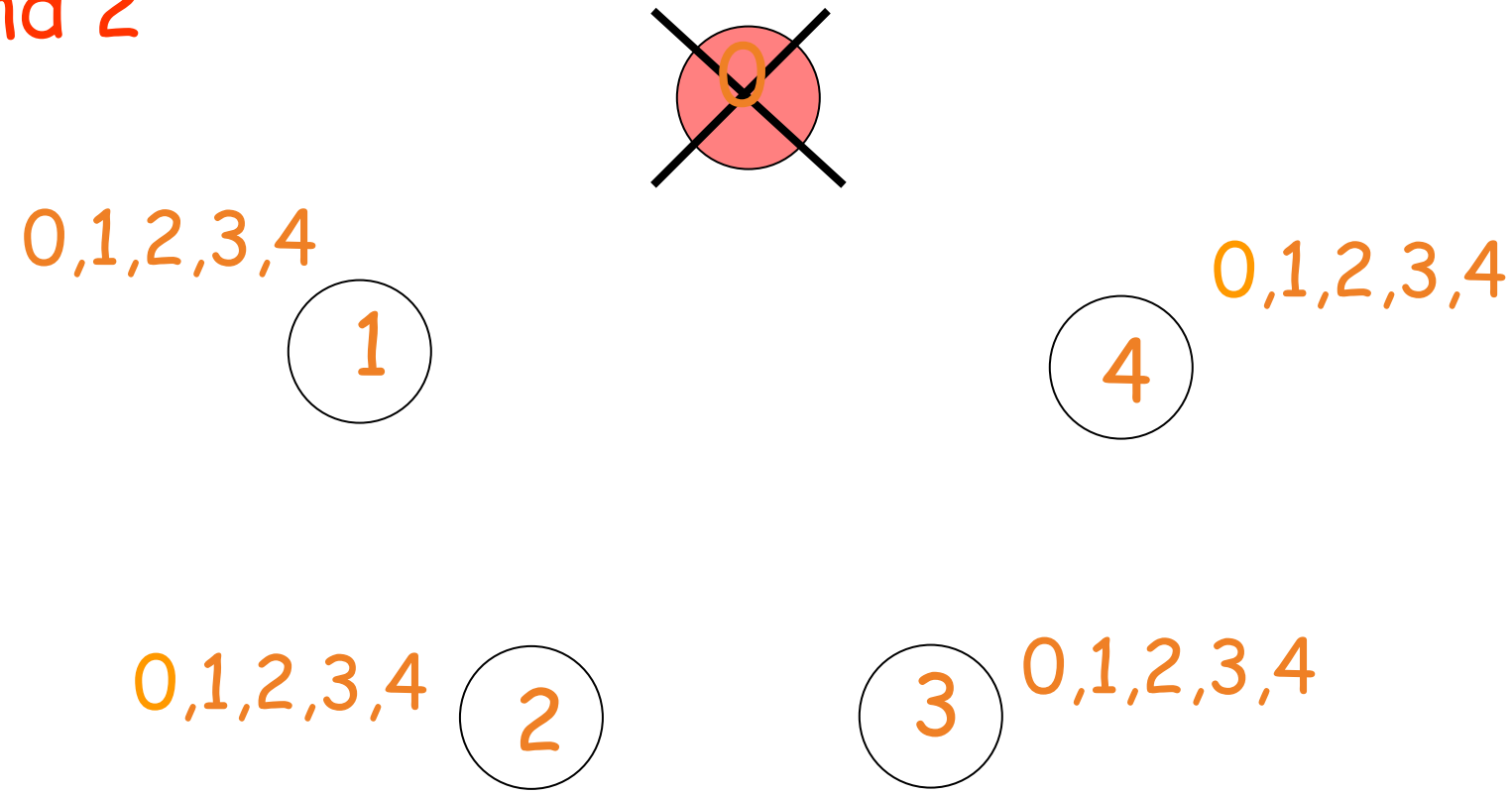
Example: f=1 failures, f+1 = 2 rounds needed



Decide on minimum: two 1s, two 0s -> No agreement

Example: f=1 failures, f+1 = 2 rounds needed

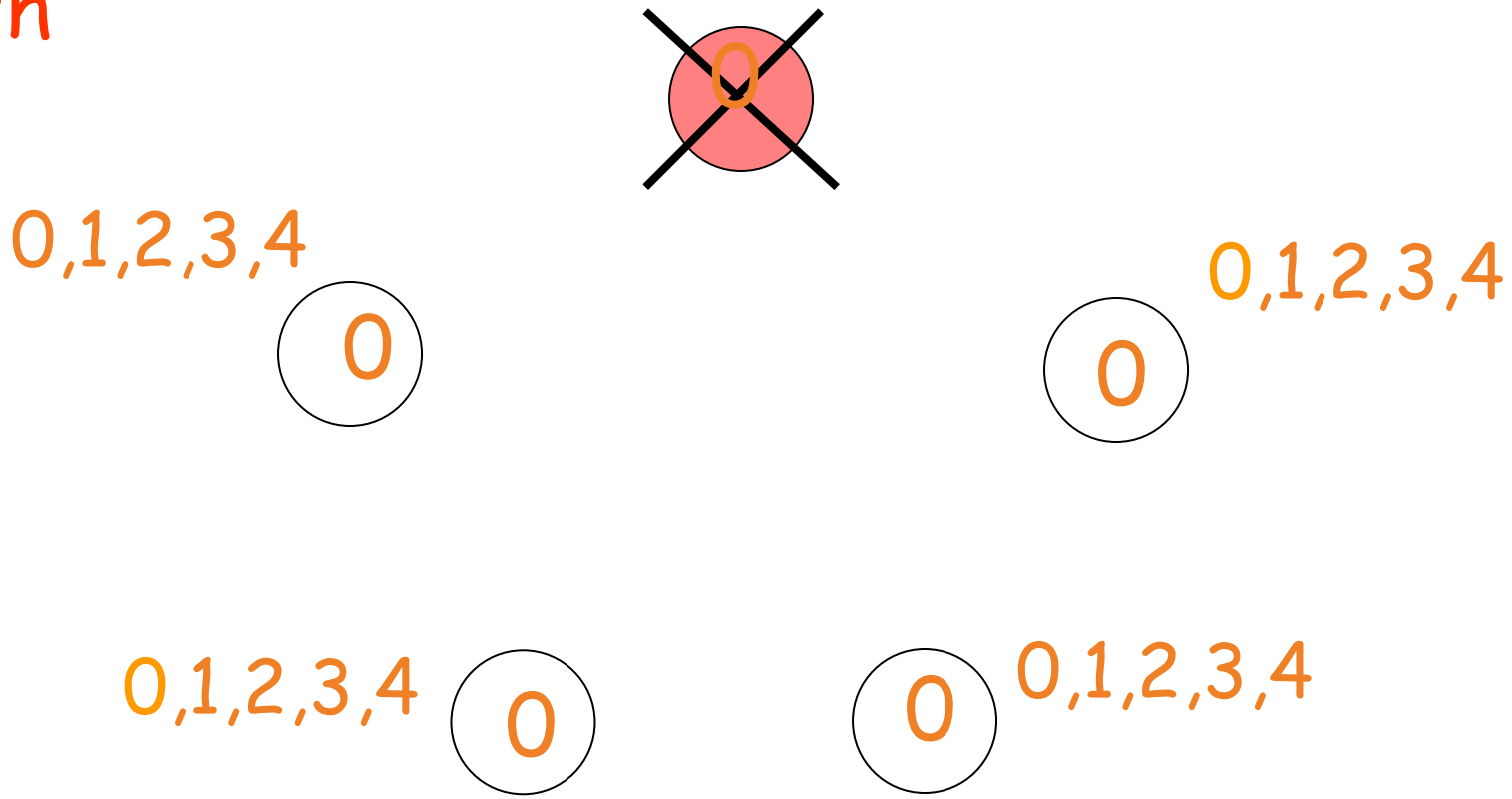
Round 2



Broadcast all new values to everybody

Example: f=1 failures, f+1 = 2 rounds needed

Finish



Decide on minimum: all 0s -> Agreement