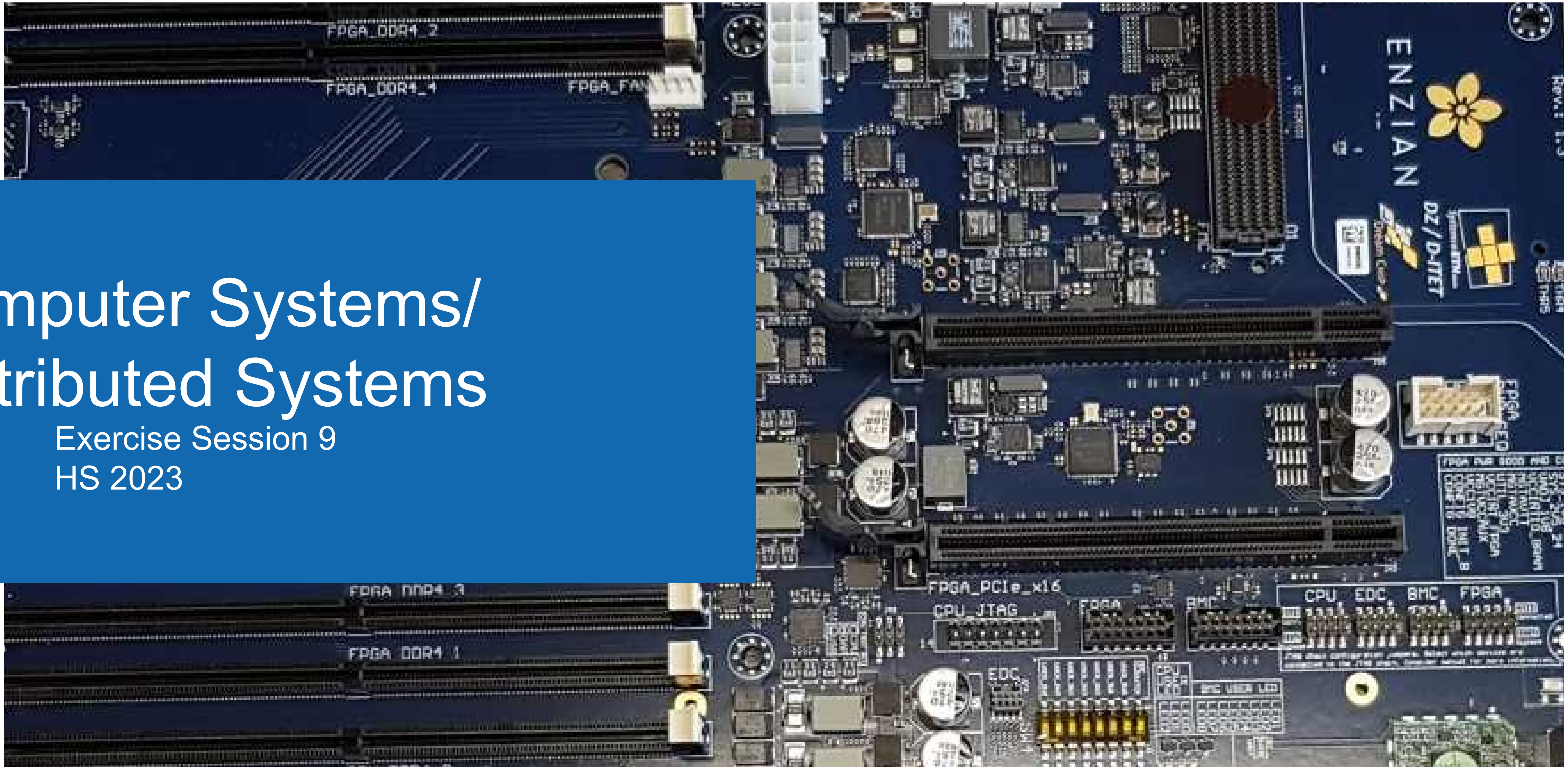
Slides last updated:
20.11.2023



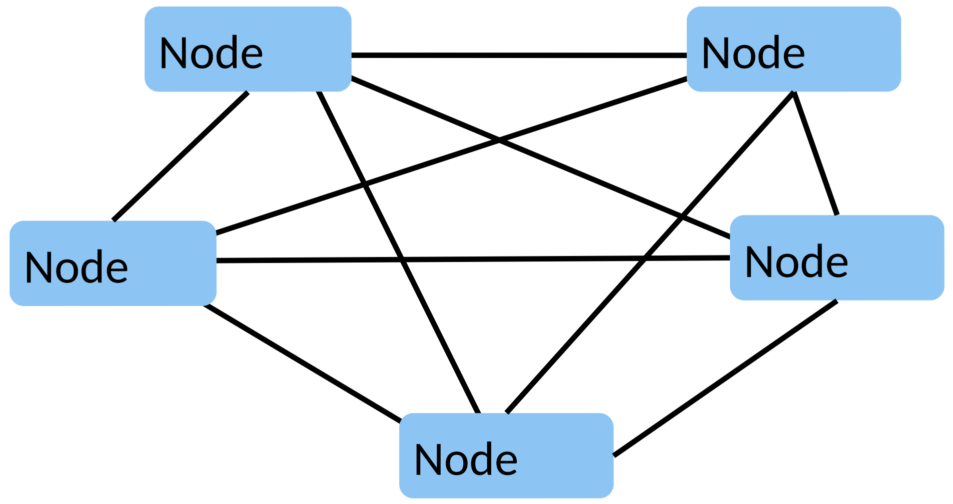
Computer Systems/ Distributed Systems

Exercise Session 9
HS 2023

Program:

1. Lecture Recap
    a) Byzantine Agreement
    b) King’s Algorithm
    c) Ben-Or’s Algorithm
    d) Reliable Broadcast
2. Quiz
3. Assignment Preview

Set-Up



Node: single actor in a distributed system

Previous Challenges in Consensus

- Messages can get lost
- Nodes may crash
- Messages can have various delay

New Challenge in Byzantine Agreement

- *Byzantine nodes = Nodes can have arbitrary behaviour*

Byzantine Agreement

We want:

1. Agreement:
   All (correct) nodes decide on the same value.

Byzantine Agreement

We want:

1. Agreement:
   All (correct) nodes decide on the same value.
2. Termination:
   All (correct) nodes terminate.

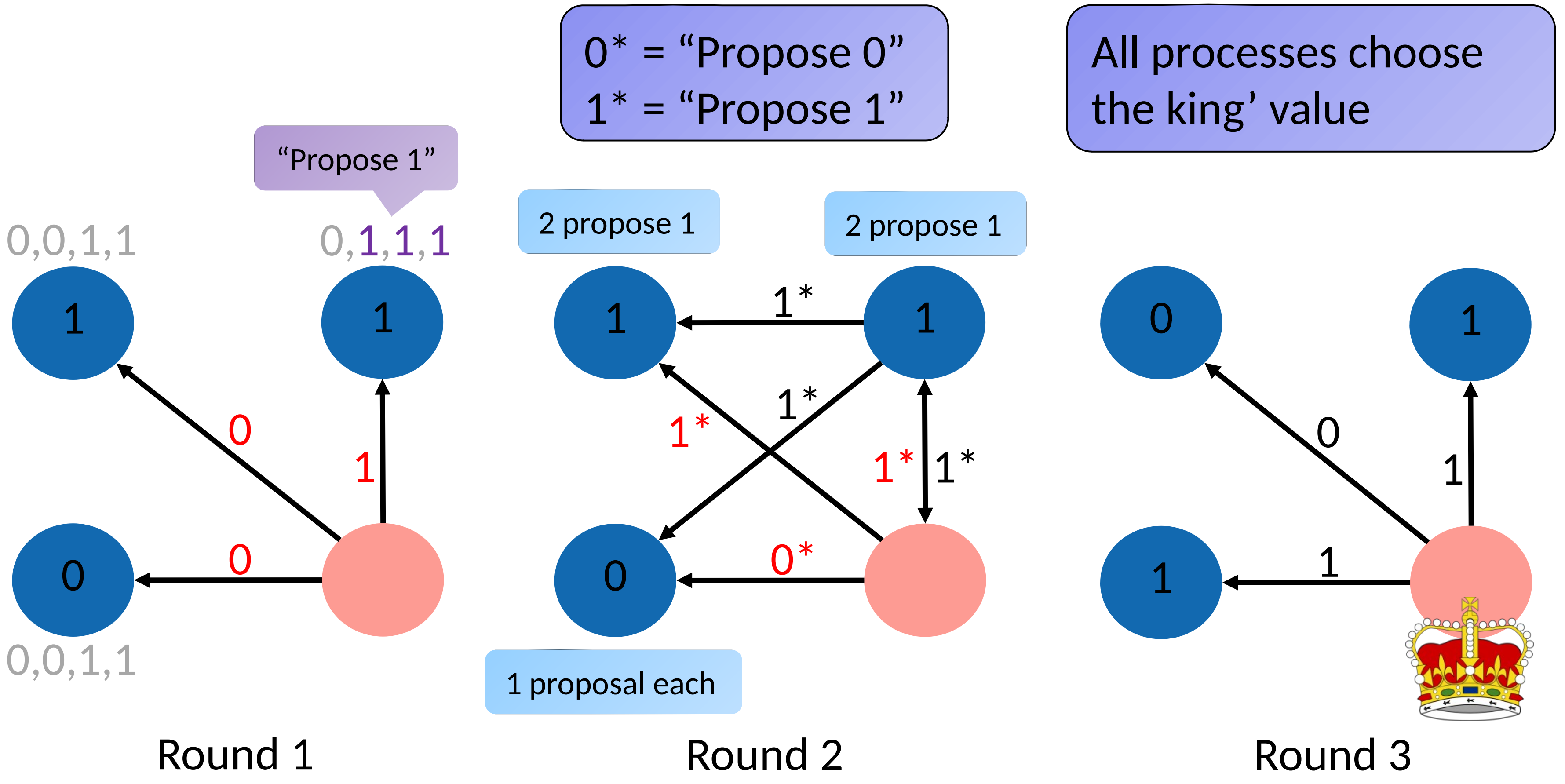
Byzantine Agreement

We want:

1. Agreement:
   All (correct) nodes decide on the same value.
2. Termination:
   All (correct) nodes terminate.
3. All-same Validity:
   If all correct nodes start with the same value “v”, the decision value must be “v”.

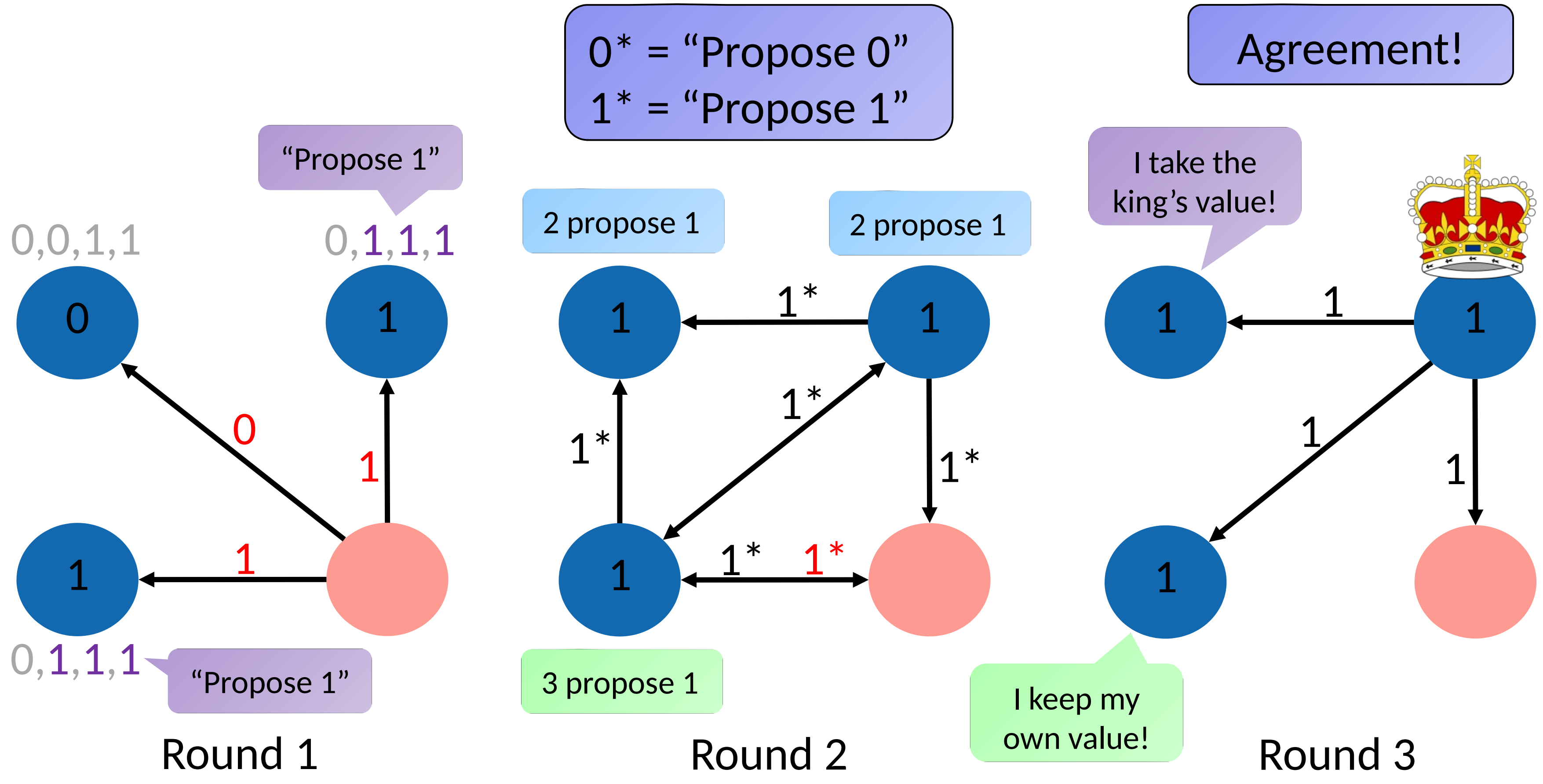
(Synchronous) King's Algorithm

- Example: *n* = 4, *f*=1
- Phase 1:



(Synchronous) King’s Algorithm

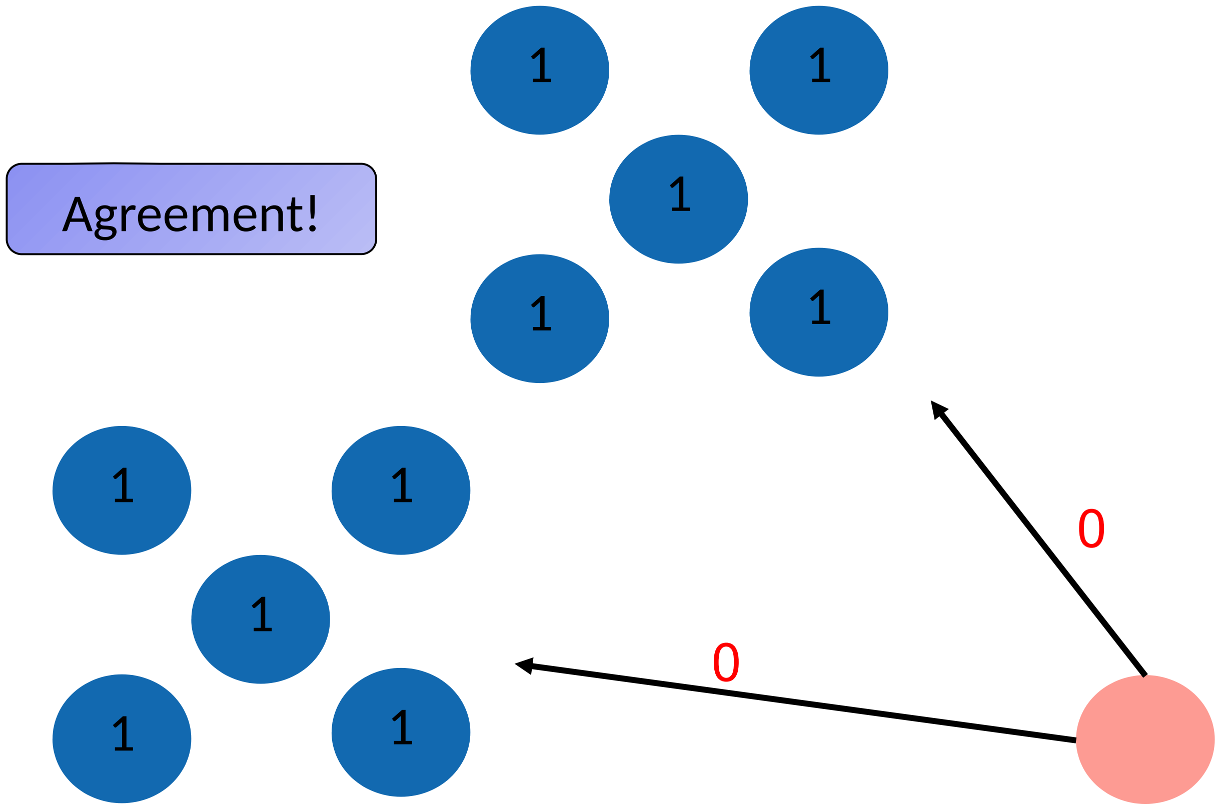
- Example: *n* = 4, *f*=1
- Phase 2:



ETH zürich Distributed Computing

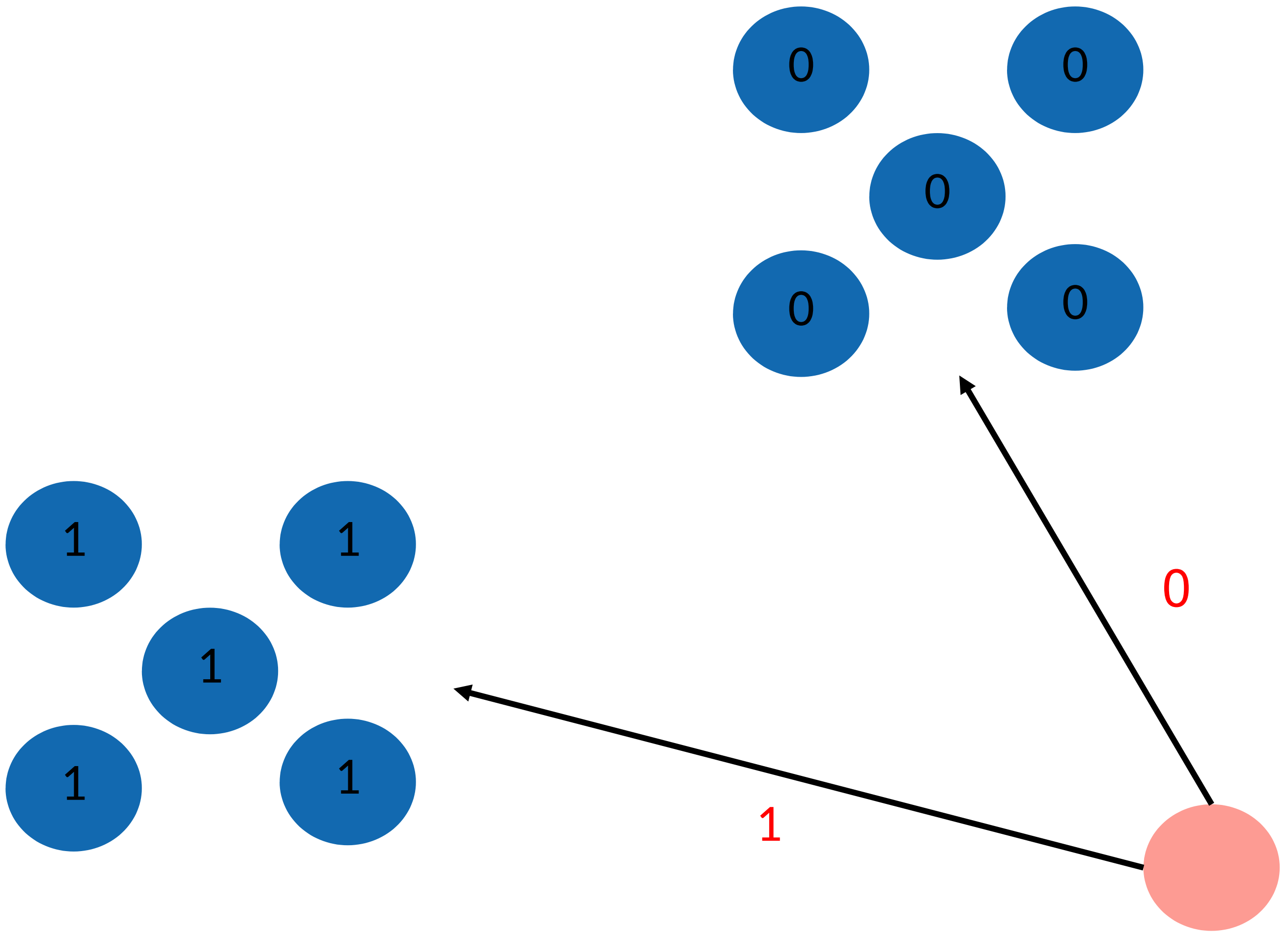
(Asynchronous) Ben-Or's Algorithm

- Example: n = 11, f=1
- Byzantine node has no power



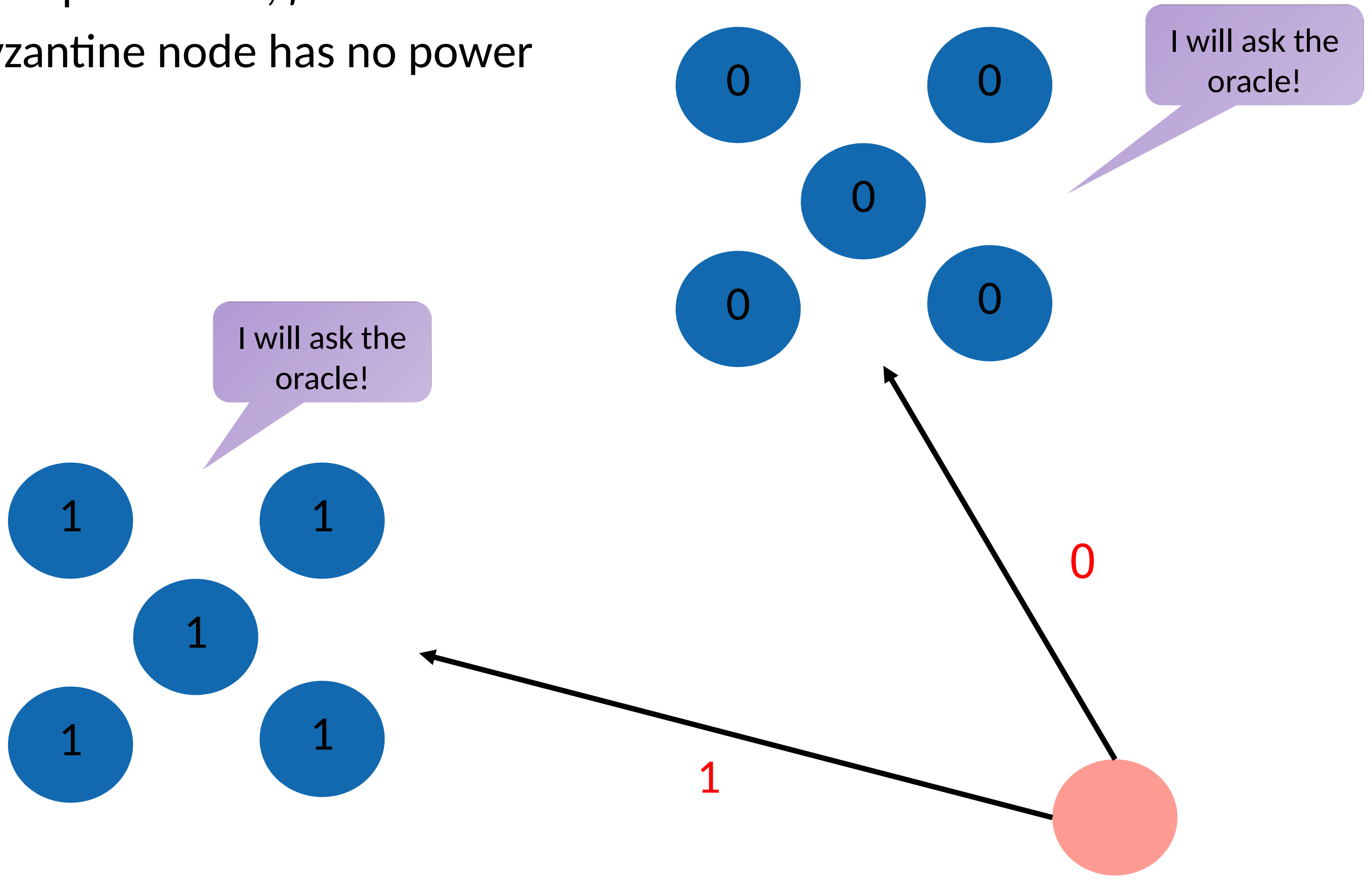
(Asynchronous) Ben-Or's Algorithm

- Example: *n* = 11, *f*=1



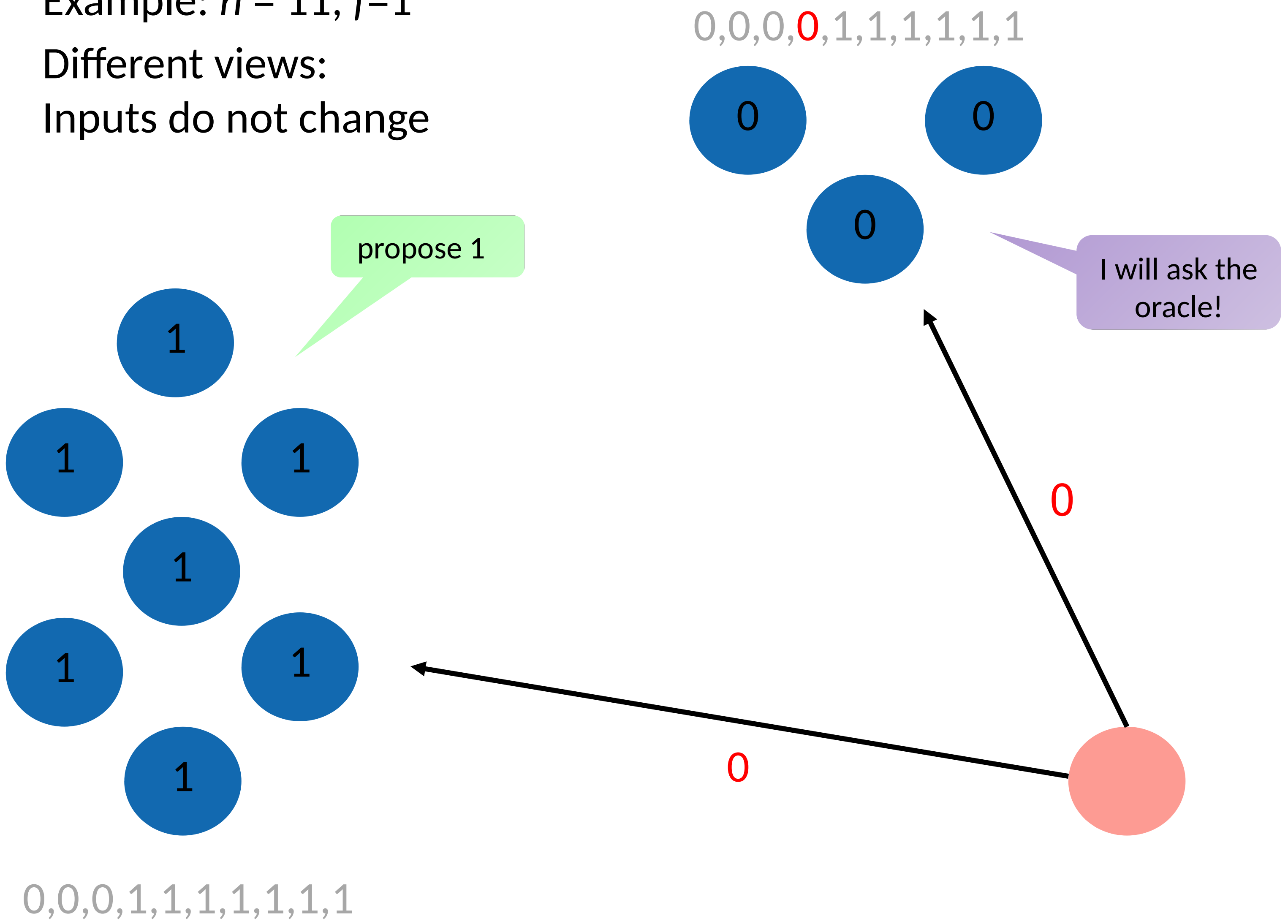
(Asynchronous) Ben-Or's Algorithm

- Example: n = 11, f=1
- Byzantine node has no power



(Asynchronous) Ben-Or's Algorithm

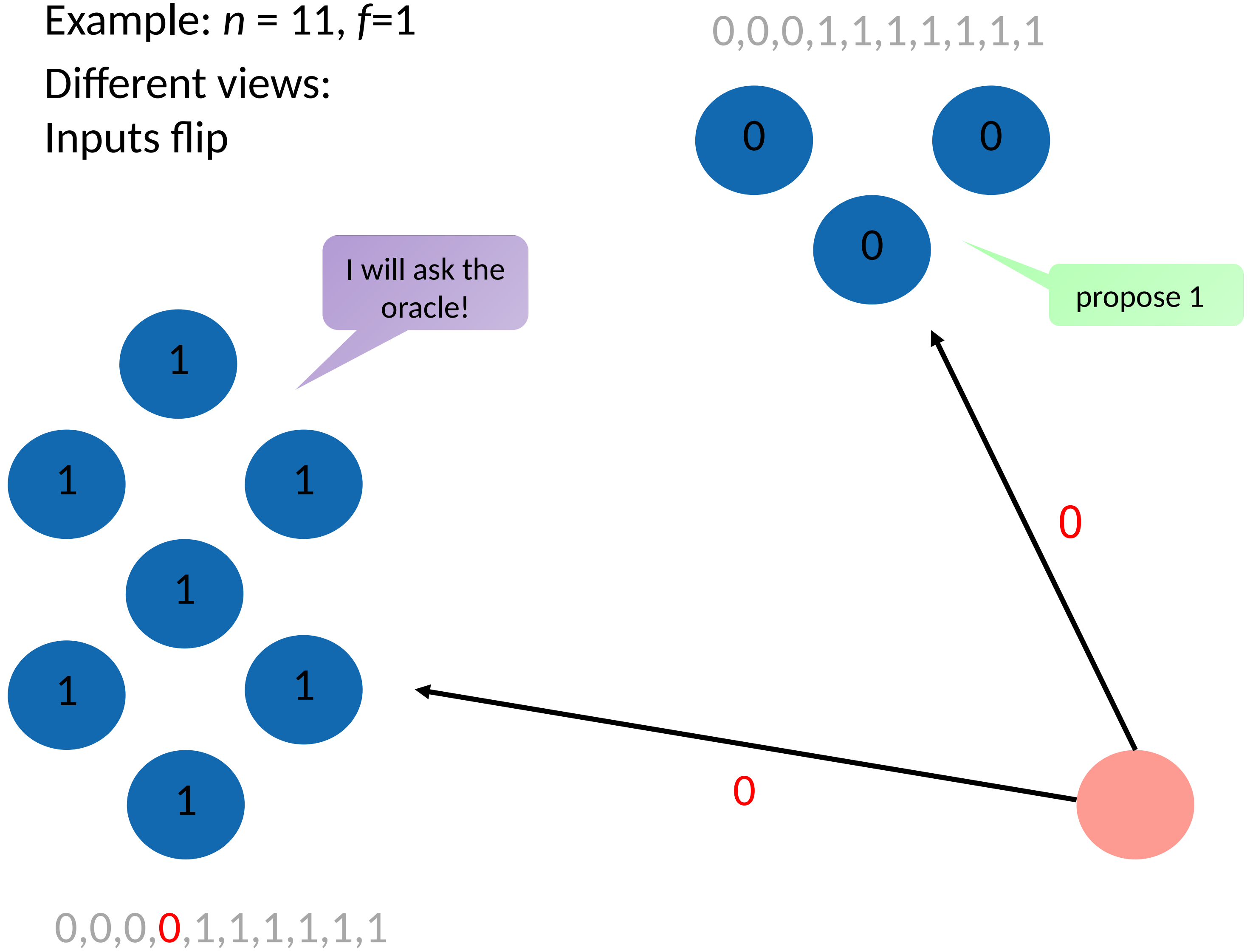
- Example: n = 11, f=1
- Different views:
  Inputs do not change



ETH zürich Distributed Computing

(Asynchronous) Ben-Or's Algorithm

- Example: n = 11, f=1
- Different views:
  Inputs flip

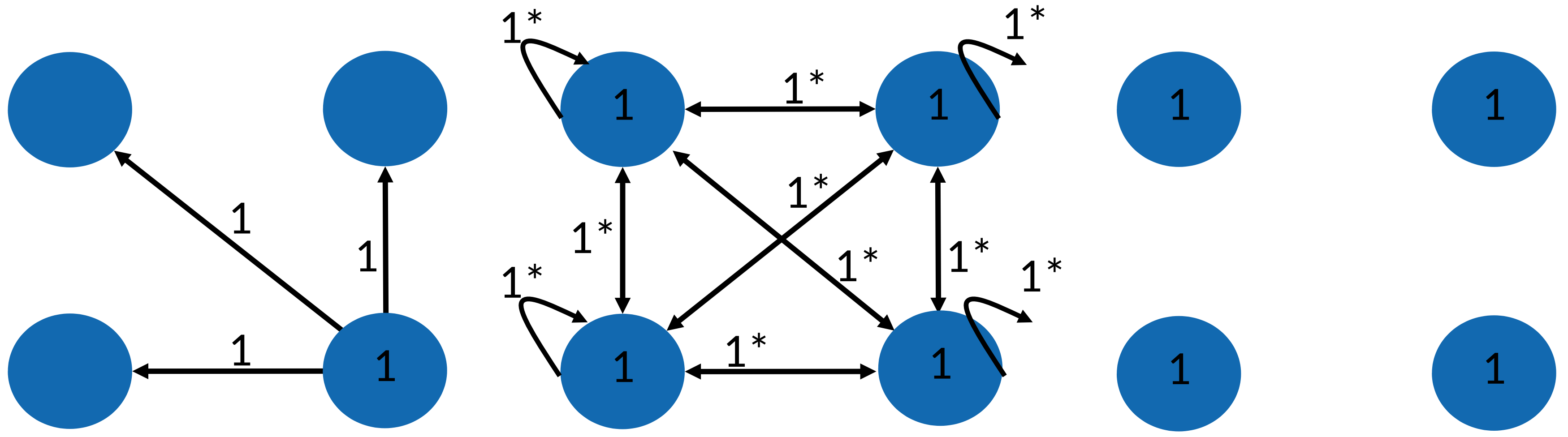


(Asynchronous) Reliable Broadcast

- Example: *n* = 4, *f*=1

1* = “Echo 1”

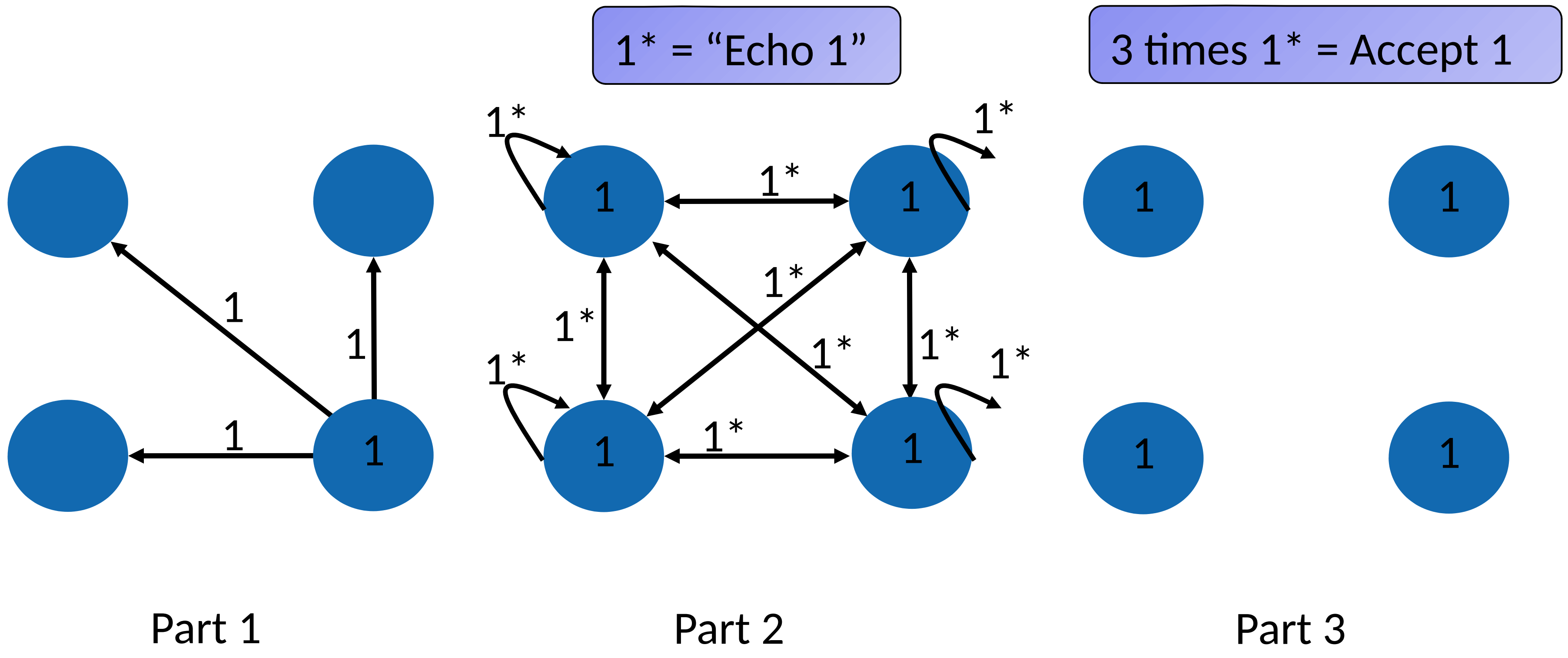
3 times 1* = Accept 1



Part 1 Part 2 Part 3

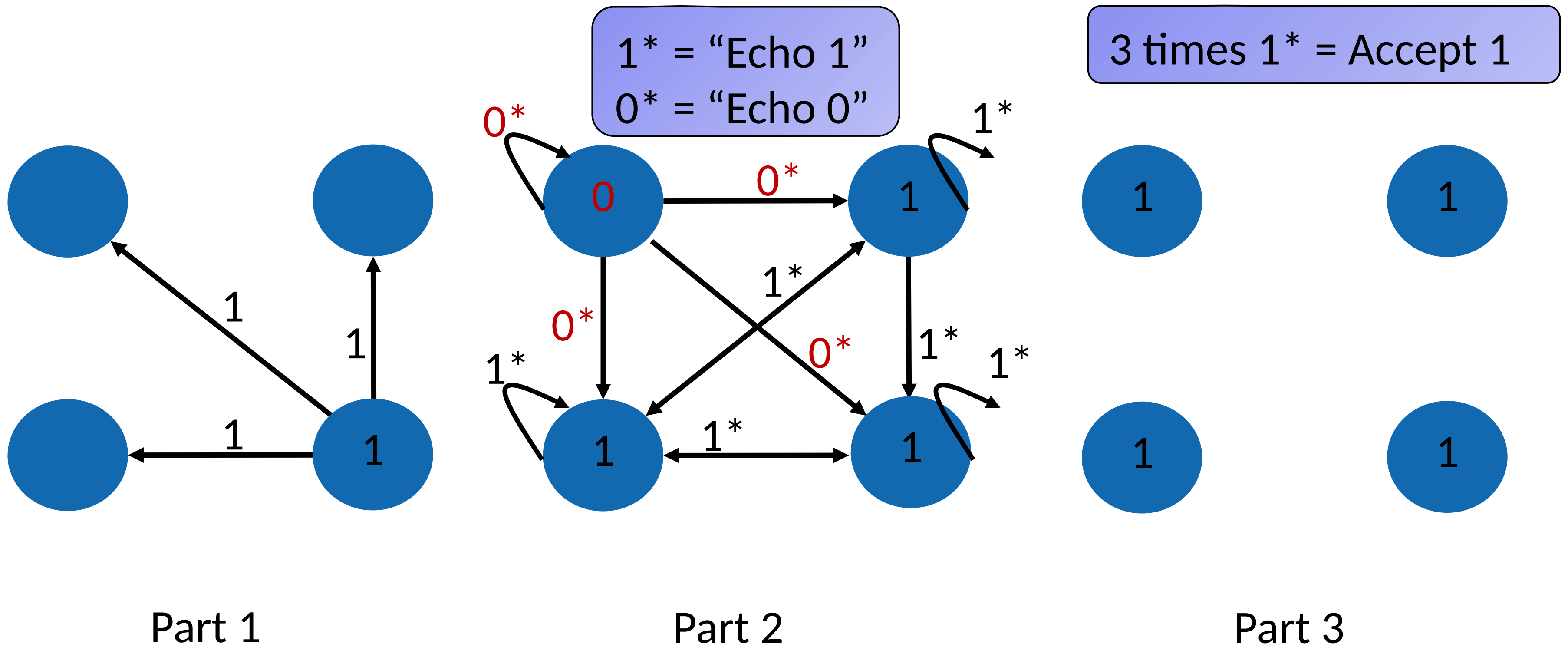
(Asynchronous) Reliable Broadcast

- Example: n = 4, f=1
- If a correct node broadcasts, it will eventually be accepted by every correct node



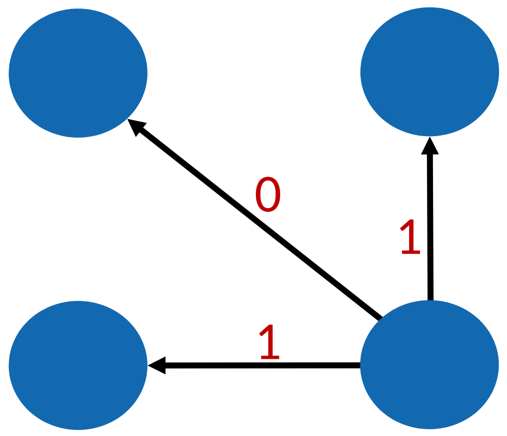
(Asynchronous) Reliable Broadcast

- Example: *n* = 4, *f*=1
- If a correct node broadcasts, it will eventually be accepted by every correct node
- If a correct node hasn't broadcast a message, will not be accepted by any other correct node.



(Asynchronous) Reliable Broadcast

- Example: n = 4, f=1
- If a correct node broadcasts, it will eventually be accepted by every correct node
- If a correct node hasn't broadcast a message, will not be accepted by any other correct node.
- Sender corrupted?

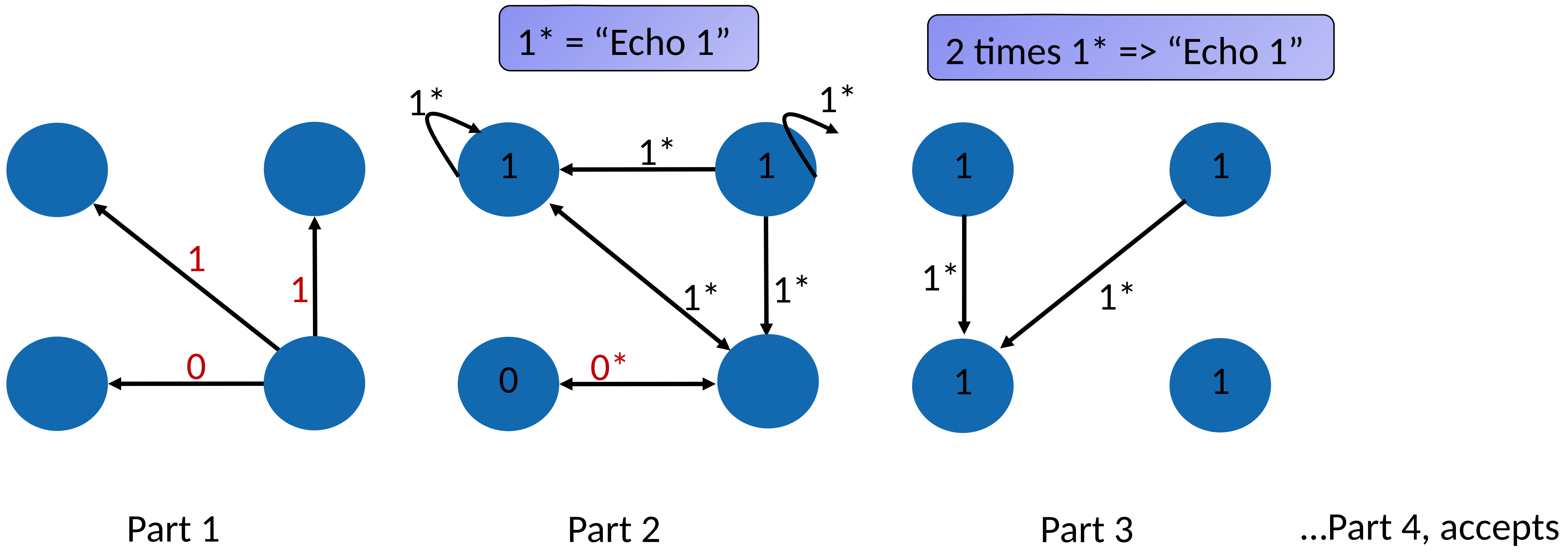


Part 1 ?

ETH zürich Distributed Computing

(Asynchronous) Reliable Broadcast

- Example: $n = 4$, $f=1$
- If a correct node broadcasts, it will eventually be accepted by every correct node
- If a correct node hasn't broadcast a message, will not be accepted by any other correct node.
- If a correct node accepts a message, it will be eventually accepted by every correct node



Quiz questions (choose the right answer)

1. No algorithm satisfies byzantine agreement with n = 3f. T/F?

2. A single byzantine node can prevent any deterministic algorithm from reaching agreement in asynchronous model. T/F?

3. If in the first phase of the King algorithm the first king is honest, then:
   1. Every honest node will decide for the initial input of the king. T/F?
   2. An honest node will propose a value in the first phase. T/F?
   3. The king himself can change its own value in the first phase. T/F?
   4. The nodes might change the value in future phases if there are other honest kings later. T/F?

4. In reliable broadcast:
   1. All honest nodes accept at most one message. T/F?
   2. All honest nodes accept at least one message. T/F?
   3. All honest nodes echo at least one message. T/F?
   4. It might happen that no honest node accepts a message. T/F?

Quiz questions (choose the right answer)

1. No algorithm satisfies byzantine agreement with n = 3f. [✓ T]/F

2. A single byzantine node can prevent any deterministic algorithm from reaching agreement in asynchronous model. [✓ T]/F

3. If in the first phase of the King algorithm the first king is honest, then:
    1. Every honest node will decide for the initial input of the king. T/[✓ F]
    2. An honest node will propose a value in the first phase. T/[✓ F]
    3. The king himself can change its own value in the first phase. [✓ T]/F
    4. The nodes might change the value in future phases if there are other honest kings later. T/[✓ F]

4. In reliable broadcast:
    1. All honest nodes accept at most one message. T/[✓ F]
    2. All honest nodes accept at least one message. T/[✓ F]
    3. All honest nodes echo at least one message. T/[✓ F]
    4. It might happen that no honest node accepts a message. [✓ T]/F

ETH zürich Systems@ETH zürich Distributed Computing

Assignment Preview

1 Synchronous Model

Quiz

1.1 Synchronous Consensus in a Grid

In the lecture you learned how to reach consensus in a fully connected network where every process can communicate directly with every other process. Now consider a network that is organized as a 2-dimensional grid such that every process has up to 4 neighbors. The width of the grid is w, the height is h. Width and height are defined in terms of edges: A 2×2 grid contains 9 nodes! The grid is big, meaning that $w + h$ is much smaller than $w \cdot h$. We use the synchronous time model; i.e., in every round every process may send a message to each of its neighbors, and the size of the message is not limited.

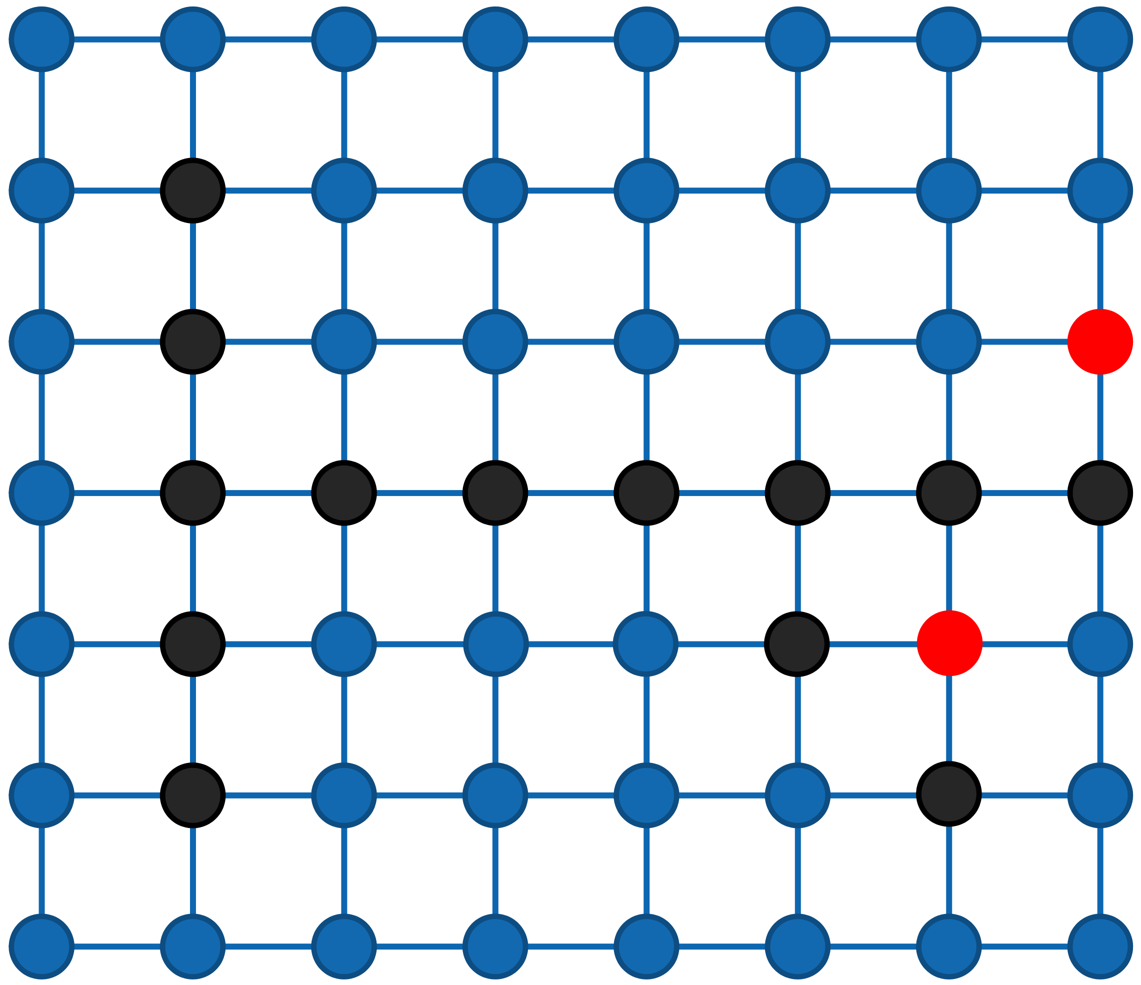
a) Assume every node knows w and h. Write a short protocol to reach consensus.

b) From now on the nodes do not know the size of the grid. Write a protocol to reach consensus and optimize it according to runtime.

c) How many rounds does your protocol from **b)** require?

Assume there are Byzantine nodes and that you are the adversary who can select which nodes are Byzantine.

d) What is the smallest number of Byzantine nodes that you need to prevent the system from reaching agreement, and where would you place them?



ETH zürich Systems@ETH zürich Distributed Computing

Assignment Preview

1.2 Synchronous Consensus in a Grid - Crash Failures

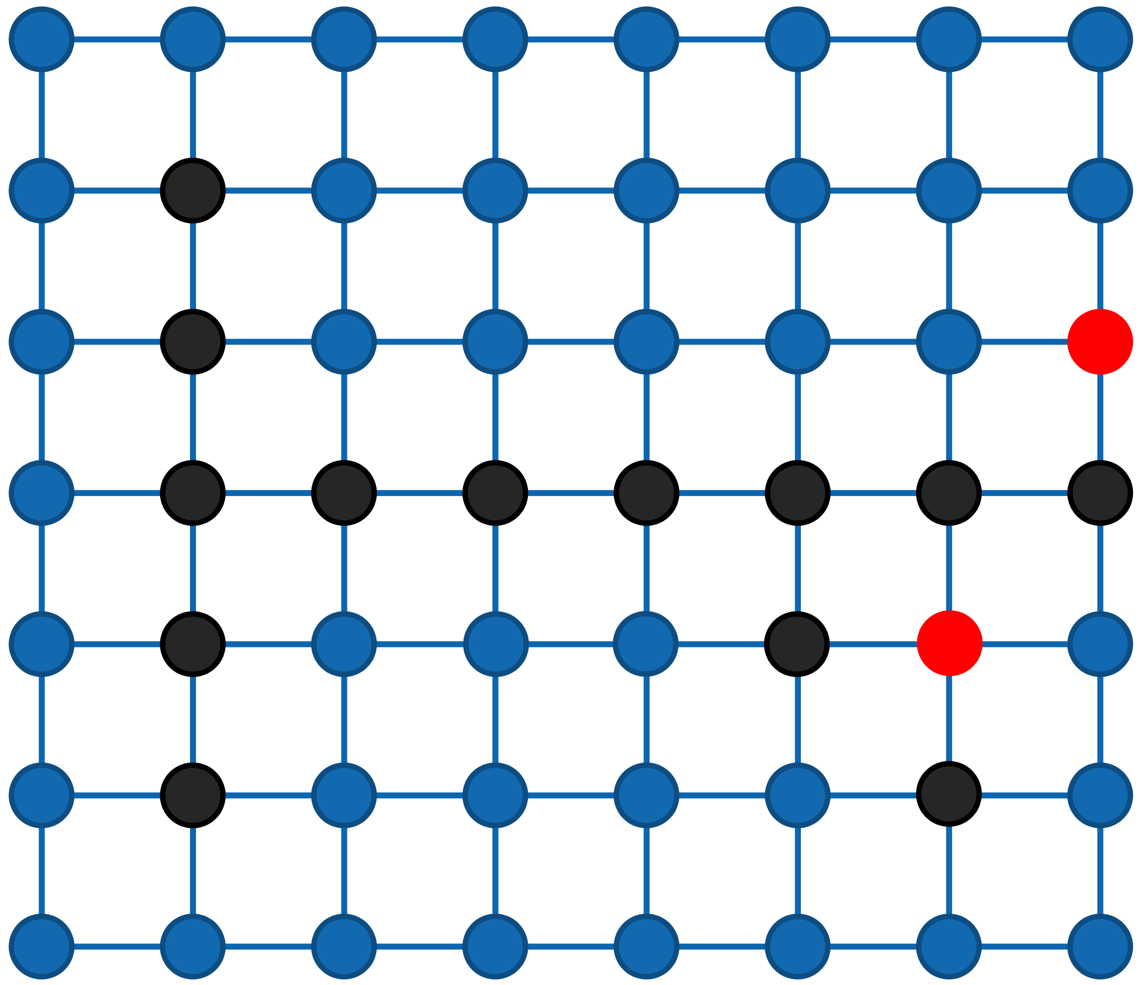
Consider the same network as in Question 1.1. Assume that some of the nodes crashed at the beginning of the algorithm such that any two correct processes are still connected through at least one path of correct processes.

Let l be the length of the longest shortest path between any pair of nodes in the grid; i.e., l is the number of edges between those two nodes which are "farthest away" from each other. If there are no failures, l is the distance between two corners, i.e. $l = w + h$.

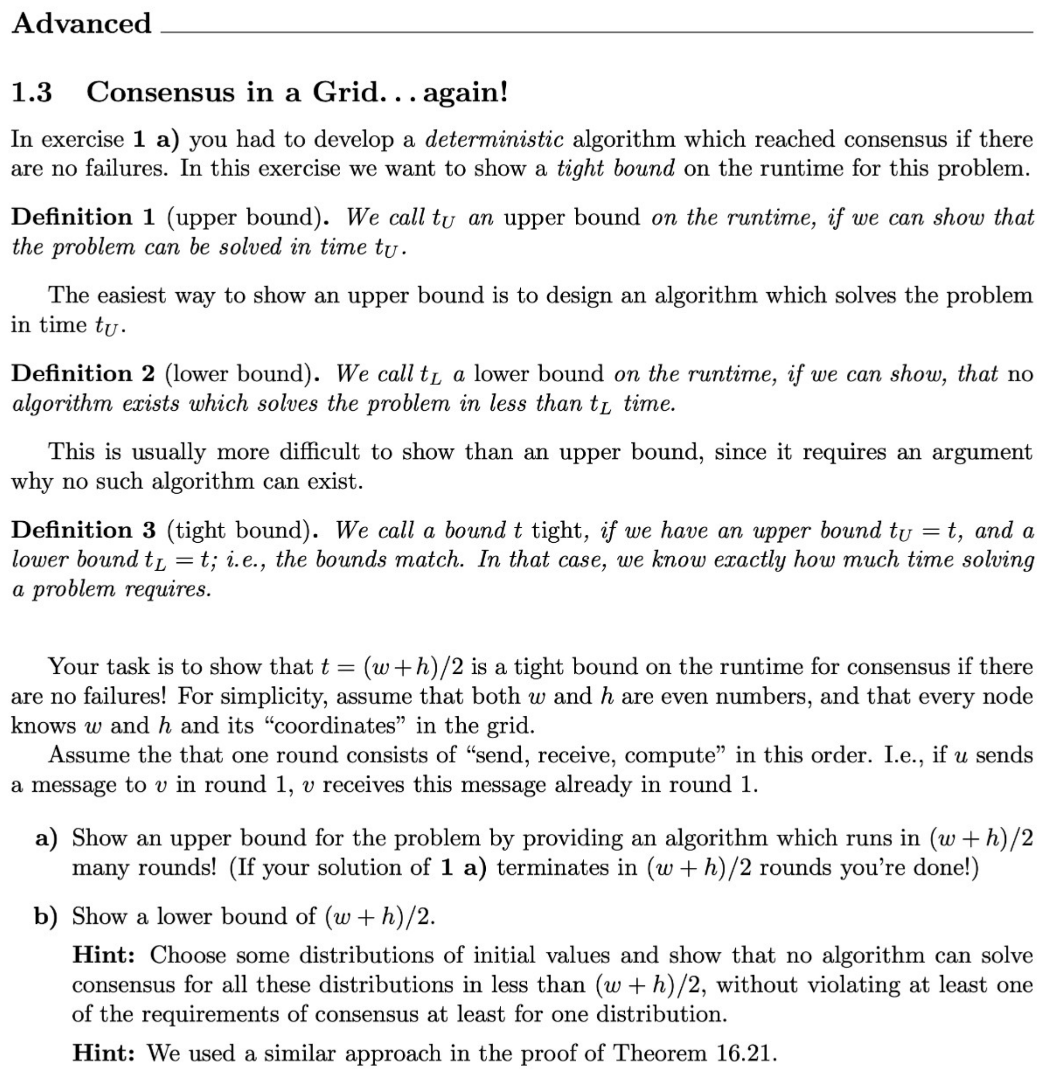
a) Modify the algorithm from 1.1**b)** to solve consensus in $l + 1$ rounds with this special type of crash failures. Show that your algorithm works correctly; i.e., a node does not terminate before it learned the initial value of all nodes.

b) As an adversary you are allowed to crash up to $w + h$ many nodes at the beginning of the algorithm. Let $w = 7, h = 6$. What is the largest l you can achieve?

c) Assume that you run the algorithm with any type of crash failures; i.e, nodes can crash at any time during the execution. Show that with such failures the algorithm does not always work correctly anymore, by giving an execution and a failure pattern in which some nodes terminate too early!



Assignment Preview



Advanced

1.3 Consensus in a Grid... again!

In exercise **1 a)** you had to develop a *deterministic* algorithm which reached consensus if there are no failures. In this exercise we want to show a *tight bound* on the runtime for this problem.

Definition 1 (upper bound)**.** *We call t_U an* upper bound *on the runtime, if we can show that the problem can be solved in time t_U.*

The easiest way to show an upper bound is to design an algorithm which solves the problem in time t_U.

Definition 2 (lower bound)**.** *We call t_L a* lower bound *on the runtime, if we can show, that* no *algorithm exists which solves the problem in less than t_L time.*

This is usually more difficult to show than an upper bound, since it requires an argument why no such algorithm can exist.

Definition 3 (tight bound)**.** *We call a bound t* tight, *if we have an upper bound $t_U = t$, and a lower bound $t_L = t$; i.e., the bounds match. In that case, we know exactly how much time solving a problem requires.*

Your task is to show that $t = (w+h)/2$ is a tight bound on the runtime for consensus if there are no failures! For simplicity, assume that both w and h are even numbers, and that every node knows w and h and its "coordinates" in the grid.

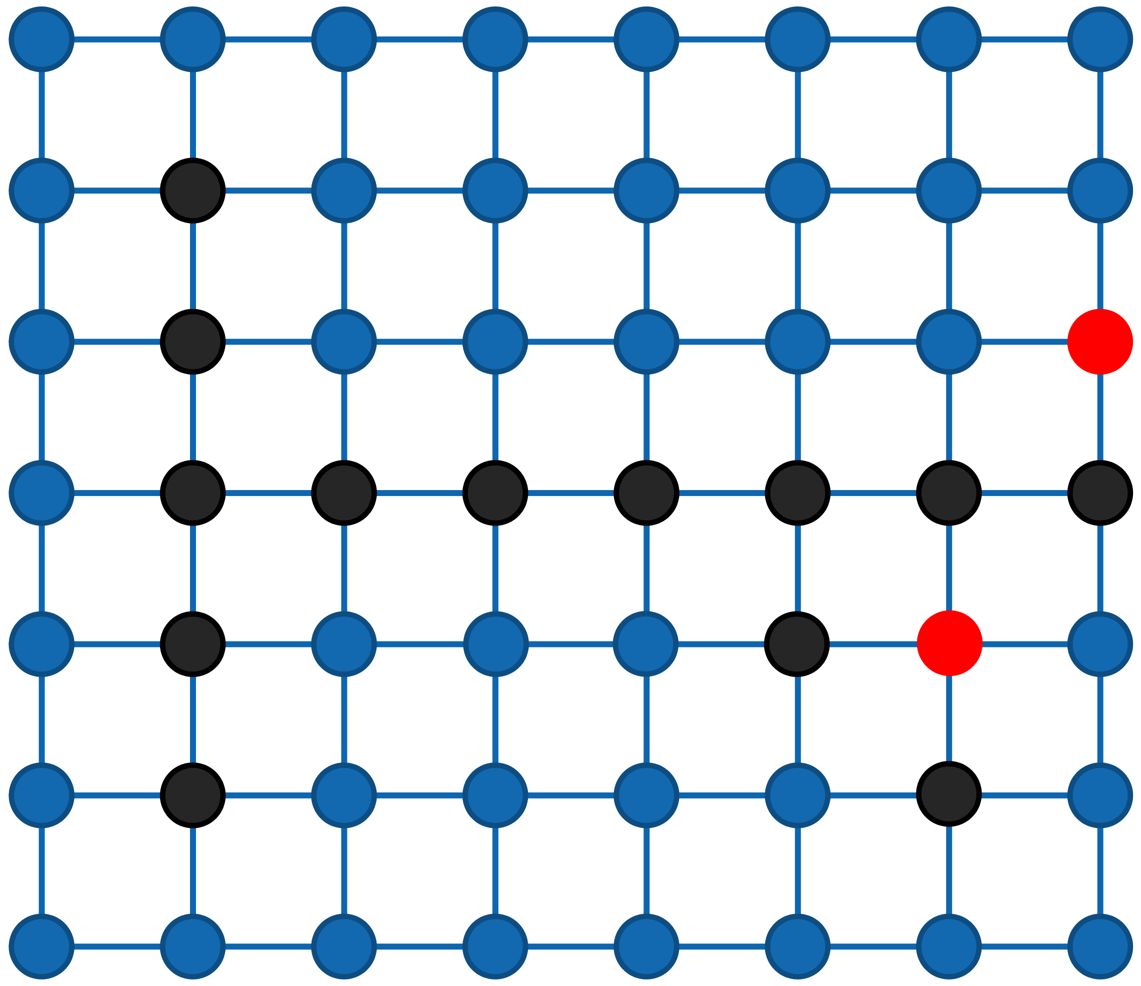
Assume the that one round consists of "send, receive, compute" in this order. I.e., if u sends a message to v in round 1, v receives this message already in round 1.

a) Show an upper bound for the problem by providing an algorithm which runs in $(w+h)/2$ many rounds! (If your solution of **1 a)** terminates in $(w+h)/2$ rounds you're done!)

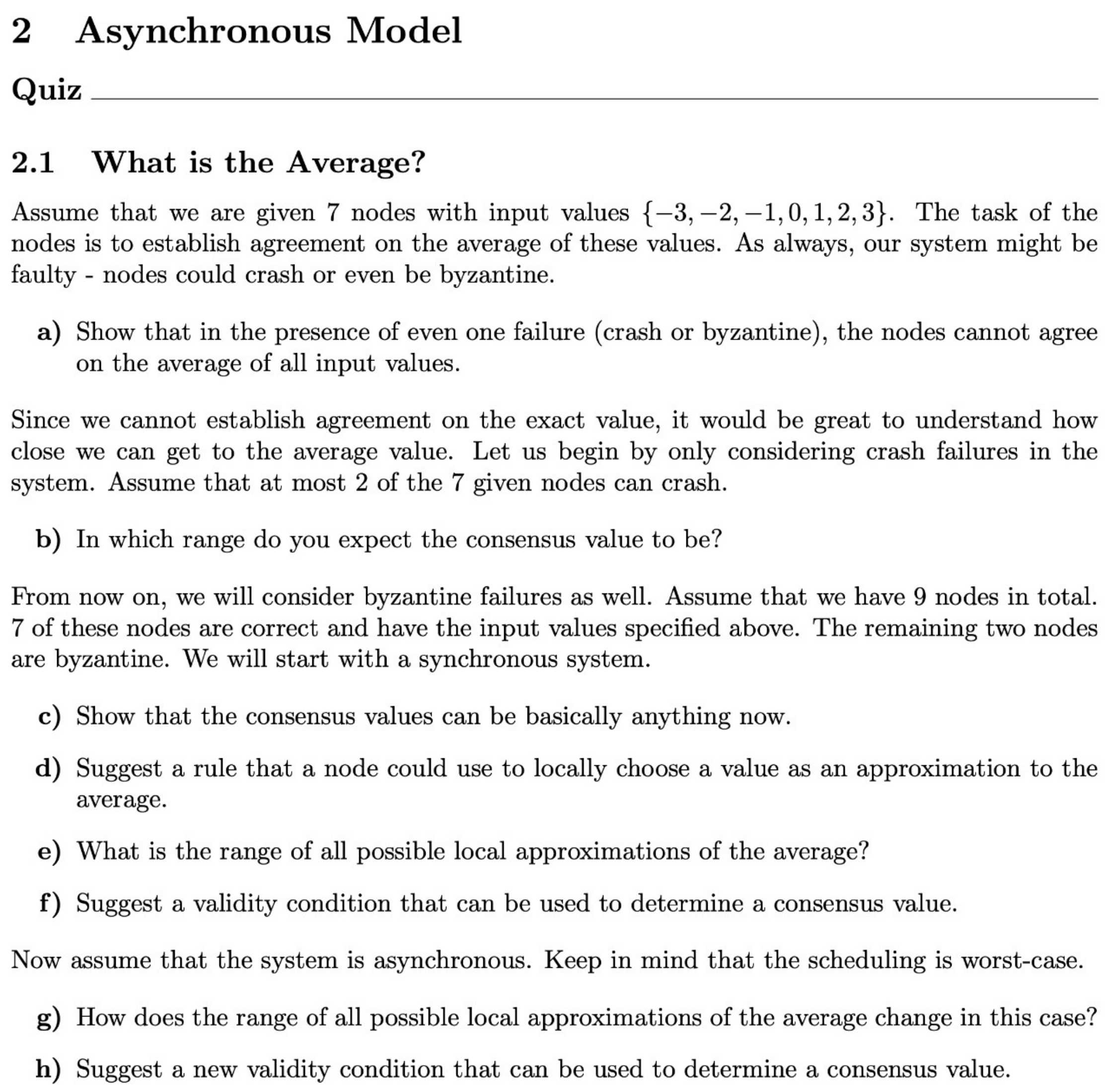
b) Show a lower bound of $(w+h)/2$.

Hint: Choose some distributions of initial values and show that no algorithm can solve consensus for all these distributions in less than $(w+h)/2$, without violating at least one of the requirements of consensus at least for one distribution.

Hint: We used a similar approach in the proof of Theorem 16.21.



Assignment Preview



2 Asynchronous Model

Quiz

2.1 What is the Average?

Assume that we are given 7 nodes with input values $\{-3, -2, -1, 0, 1, 2, 3\}$. The task of the nodes is to establish agreement on the average of these values. As always, our system might be faulty - nodes could crash or even be byzantine.

a) Show that in the presence of even one failure (crash or byzantine), the nodes cannot agree on the average of all input values.

Since we cannot establish agreement on the exact value, it would be great to understand how close we can get to the average value. Let us begin by only considering crash failures in the system. Assume that at most 2 of the 7 given nodes can crash.

b) In which range do you expect the consensus value to be?

From now on, we will consider byzantine failures as well. Assume that we have 9 nodes in total. 7 of these nodes are correct and have the input values specified above. The remaining two nodes are byzantine. We will start with a synchronous system.

c) Show that the consensus values can be basically anything now.

d) Suggest a rule that a node could use to locally choose a value as an approximation to the average.

e) What is the range of all possible local approximations of the average?

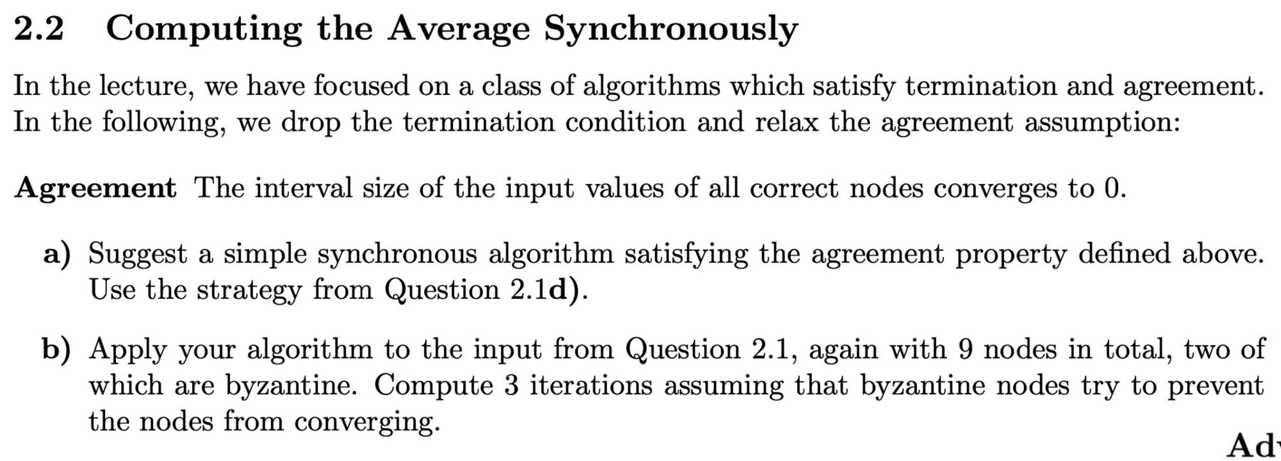
f) Suggest a validity condition that can be used to determine a consensus value.

Now assume that the system is asynchronous. Keep in mind that the scheduling is worst-case.

g) How does the range of all possible local approximations of the average change in this case?

h) Suggest a new validity condition that can be used to determine a consensus value.

Assignment Preview



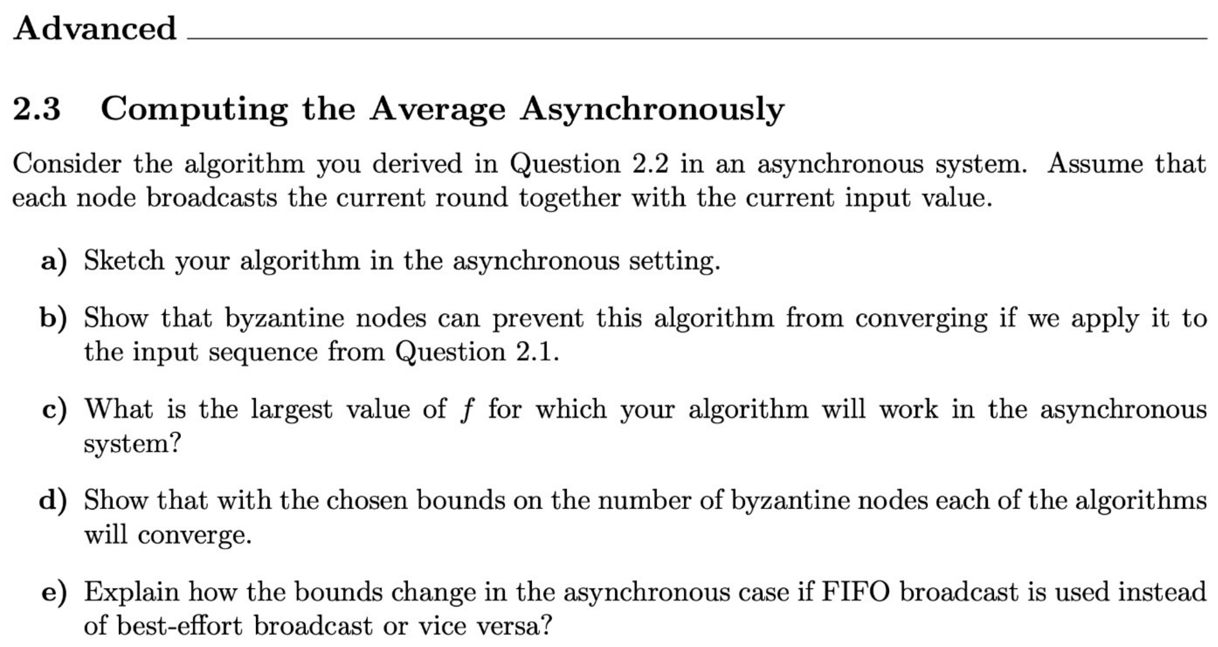
2.2 Computing the Average Synchronously

In the lecture, we have focused on a class of algorithms which satisfy termination and agreement. In the following, we drop the termination condition and relax the agreement assumption:

Agreement The interval size of the input values of all correct nodes converges to 0.

a) Suggest a simple synchronous algorithm satisfying the agreement property defined above. Use the strategy from Question 2.1**d)**.

b) Apply your algorithm to the input from Question 2.1, again with 9 nodes in total, two of which are byzantine. Compute 3 iterations assuming that byzantine nodes try to prevent the nodes from converging.



Advanced

2.3 Computing the Average Asynchronously

Consider the algorithm you derived in Question 2.2 in an asynchronous system. Assume that each node broadcasts the current round together with the current input value.

a) Sketch your algorithm in the asynchronous setting.

b) Show that byzantine nodes can prevent this algorithm from converging if we apply it to the input sequence from Question 2.1.

c) What is the largest value of f for which your algorithm will work in the asynchronous system?

d) Show that with the chosen bounds on the number of byzantine nodes each of the algorithms will converge.

e) Explain how the bounds change in the asynchronous case if FIFO broadcast is used instead of best-effort broadcast or vice versa?