SYNCHRONOUS BYZANTINE AGREEMENT

DISTRIBUTED SYSTEMS

Master of Science in Cyber Security

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

MOTIVATION

- "Byzantine Generals Problem" [Lamport, Pease, Shostak] –
- Faulty processes may exhibit "arbitrary behavior":

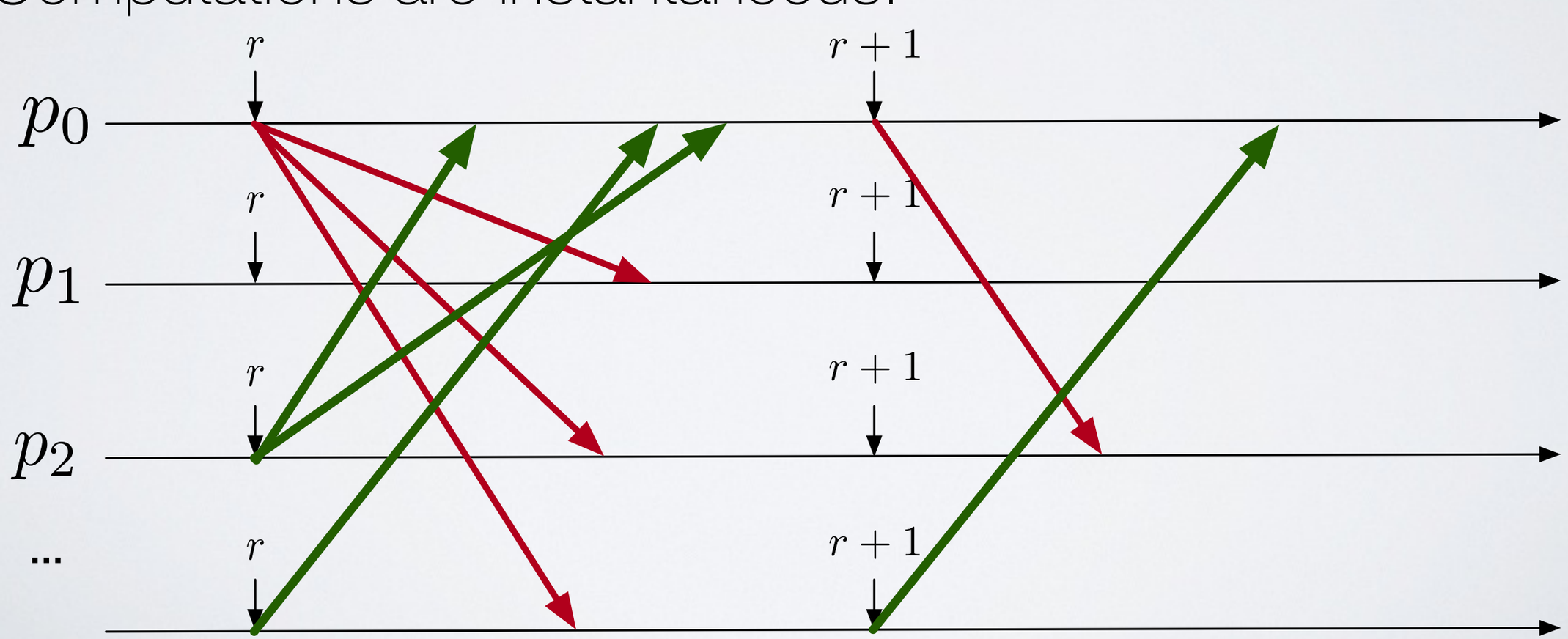
– Can start in arbitrary states, send arbitrary messages, perform arbitrary transitions.

To make flying safer, researchers studied possible failures of various sensors and machines used in airplanes. Failing machines did not just crash, instead they sometimes showed an unusual behavior before stopping completely. (SIFT project)

SYNCHRONOUS MODEL RECAP

We assume a synchronous round model:

- Fixed known delay of messages.
- If a correct sends a message at round 0, every correct destination receives it by the end of round 0. (No delay of messages)
- Computations are instantaneous.



CONSENSUS WITH CRASH FAILURES

Module 5.1: Interface and properties of (regular) consensus

Module:

Name: Consensus, **instance** c.

Events:

Request: $\langle\ c, \mathit{Propose} \mid v\ \rangle$: Proposes value v for consensus.

Indication: $\langle\ c, \mathit{Decide} \mid v\ \rangle$: Outputs a decided value v of consensus.

Properties:

C1: *Termination:* Every correct process eventually decides some value.

C2: *Validity:* If a process decides v, then v was proposed by some process.

C3: *Integrity:* No process decides twice.

C4: *Agreement:* No two correct processes decide differently.

A MEANINGFUL VALIDITY

- (Any-Input validity) If a process decides v, then v was proposed by some process.
  - What if v is proposed by a Byz. Think about altimeters on airplanes.
- (Correct-Input validity) If a process decides v, then v was proposed by some **correct** process.
  - but… what if the Byz behaves correctly but for faking its input value. In many cases, is impossible to distinguish from correct.

Any-Input validity has applications in real life:
Adding entries to a distributed log

A MEANINGFUL VALIDITY

- (All-same validity, also weak validity). If all correct processes start with value v, then the decision value must be v.
  - It makes senses in some application: think about sensors, if all correct sensors read the same value we have to decide that.

Applications in
Mission critical sensors systems:
Industrial Systems, Airplanes, Satellites, …

A MEANINGFUL VALIDITY

- (All-same validity, also weak validity). If all correct processes start with value v, then the decision value must be v.
  - It make senses in some application: think about sensors, if all correct sensors read the same value we have to decide that.

The AGREEMENT remains the same!

BIZANTINE AGREEMENT PROBLEM

Events:

- Propose(V)
- Decide=V

Properties:

- Termination: Every correct eventually decides
- All-same validity (Weak validity): If all correct processes propose v, the only decision is v.
- Integrity: No correct decides twice.
- Agreement: All correct decides the same value

LECTURE OVERVIEW

- We will show the formal proof for the impossibility of solving Byzantine Agreement (BA) when n=3f (when there are no signatures).
- We will show the king algorithm that solves BA as long as n=3f+1.
- **IMPORTANT: We assume authenticated channels (MAC) but no signatures.**

IMPOSSIBILITY OF CONSENSUS IN SYNCHRONOUS SYSTEMS WITH AUTHENTICATED CHANNELS

WHEN N=3F (OR LESS)

When signatures are available you can tollerate f<N/2 failures

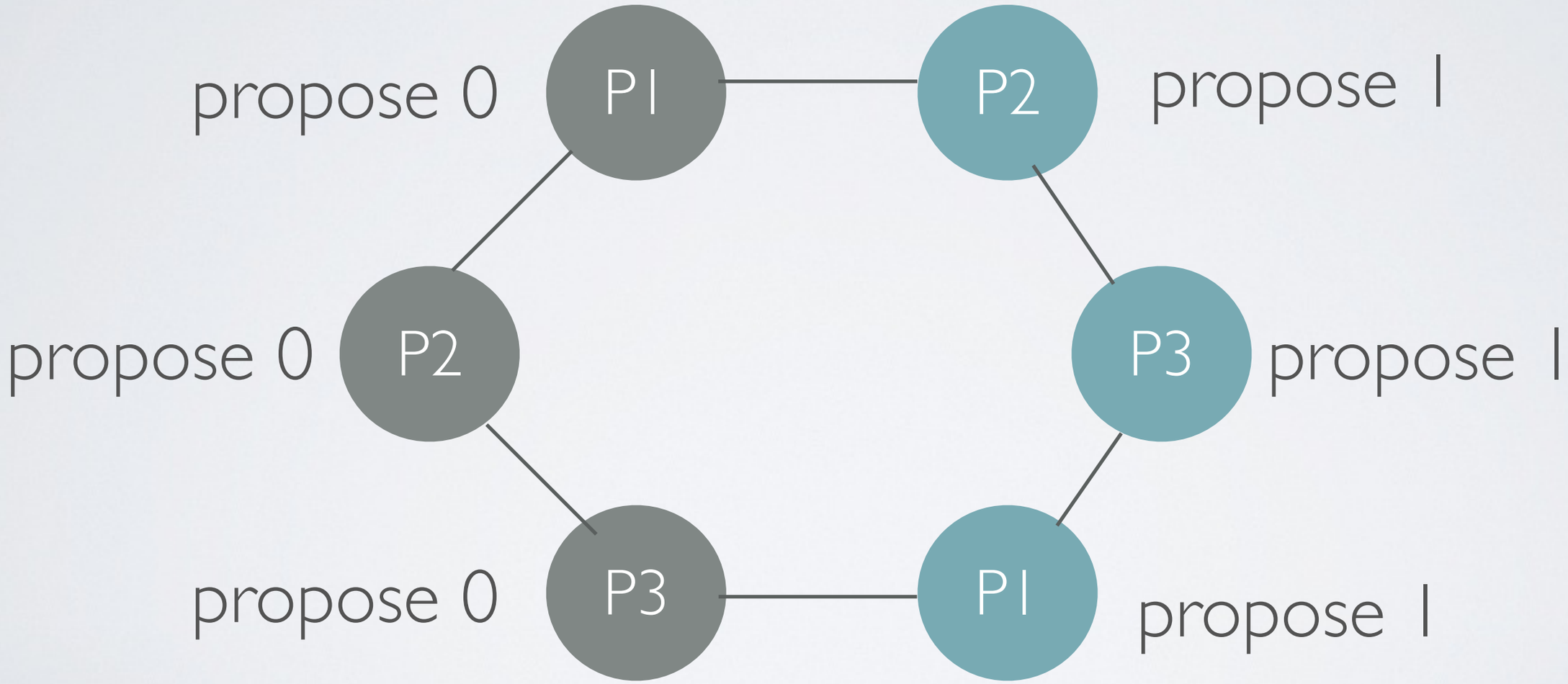
IMPOSSIBILITY FOR 3 PROCESSES IF ONE BIZ

Statement: Consider a synchronous system with authenticated channels composed by 3 processes. It is not possible to solve consensus if one of the processes is byzantine.

Proof: The proof is by contradiction we assume that an algorithm A exists and tolerates 1 bz failure in a system of 3 process. We will show that A is not able to satisfy all the properties of the specification.
The proof uses a “scenario argument”

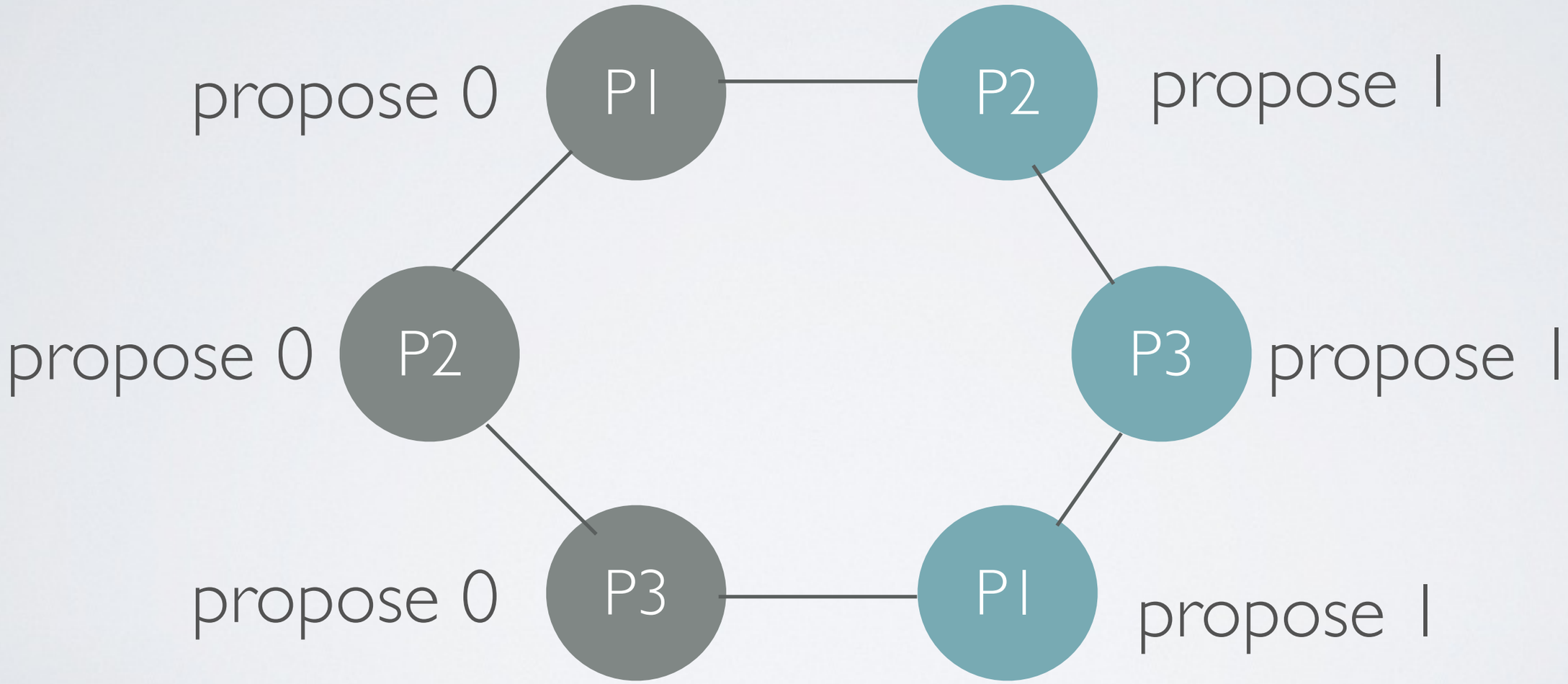
SCENARIO - 6 PROCESSES

Step 1: Take algorithm A and let it run on a system S of 6 processes arranged in a ring (see figure below). All processes in **S are correct.**



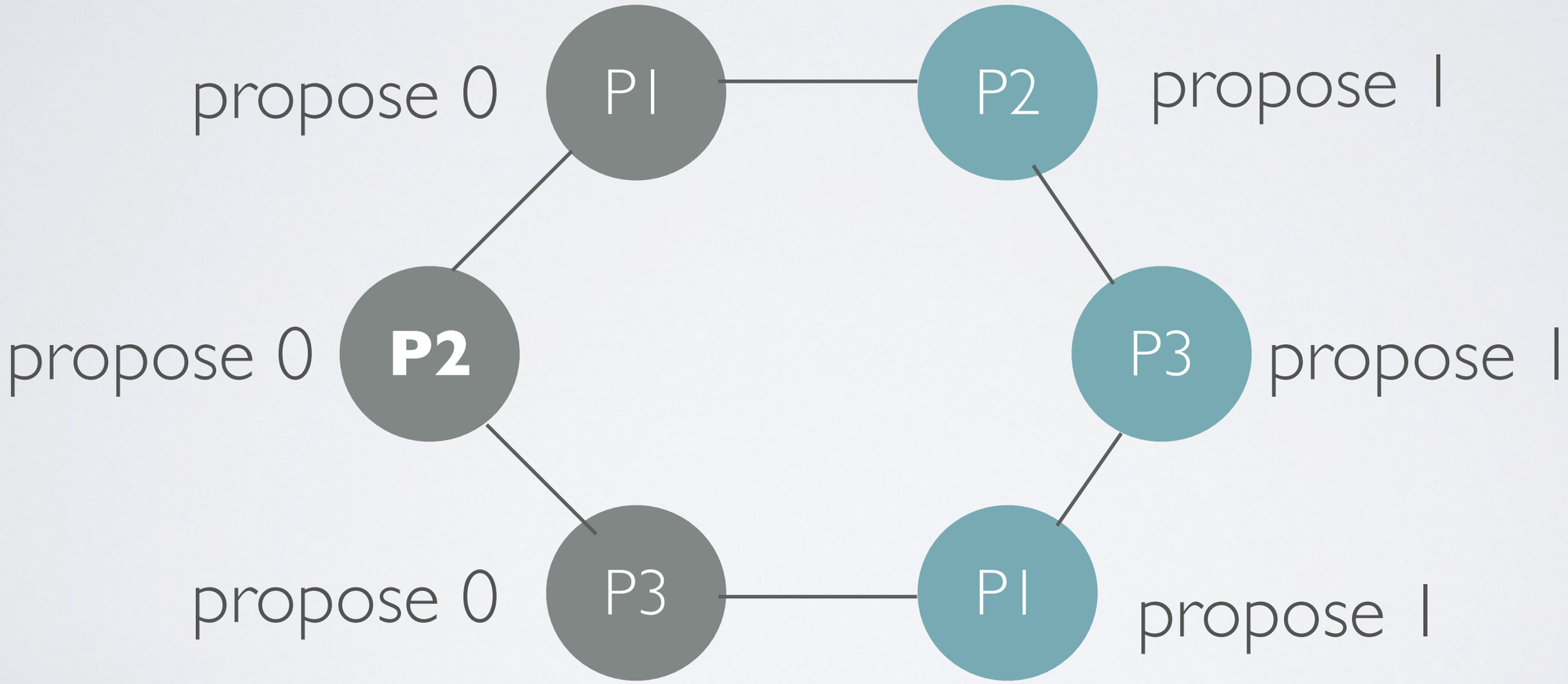
SCENARIO - 6 PROCESSES

Step 1: Since A has not been designed to run on S its behaviour can be arbitrary, so a violation of a property on **S does not directly imply that A is not correct**.



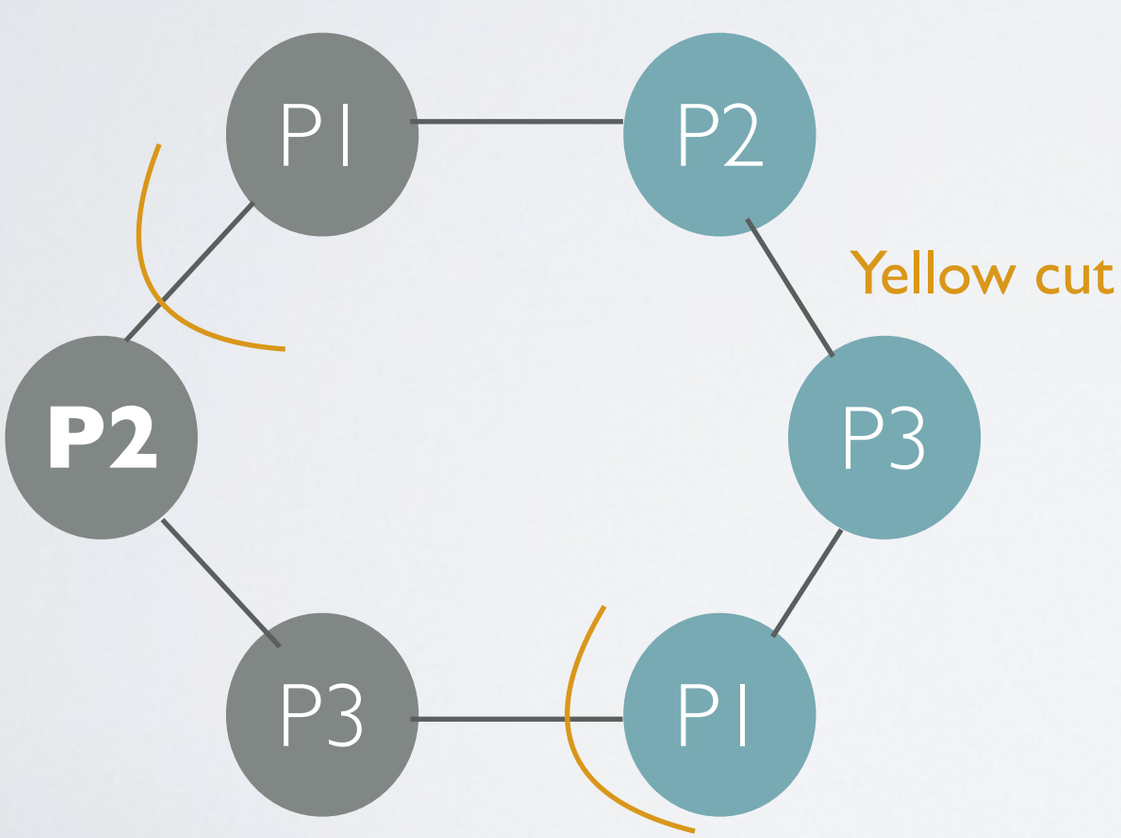
SCENARIO - 6 PROCESSES

Step 1: We have first to show that on S every process terminates. This is not granted! Suppose w.l.o.g. that grey P2 does not terminate.



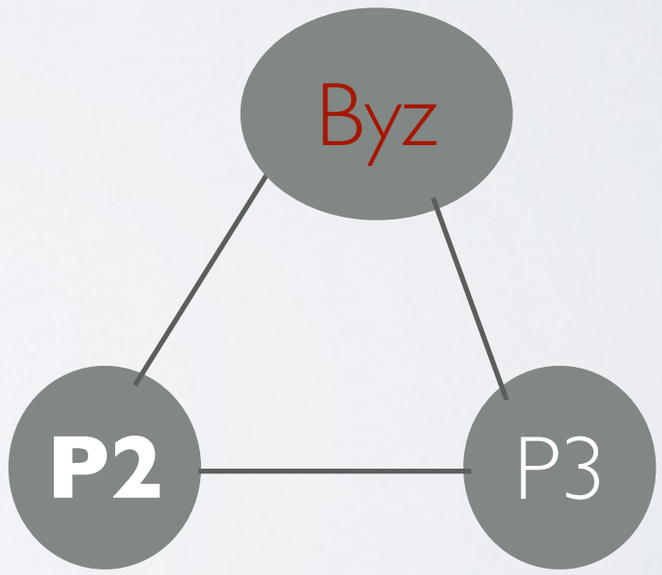
SCENARIO - 6 PROCESSES

Step 1: Then P2 does not terminate also in a reduced system B that is composed by 3 processes with one byzantine that simulates the remaining part of S. That is Byz B in S acts as it including in itself processes P1,P2,P3,P1 in S.



System S

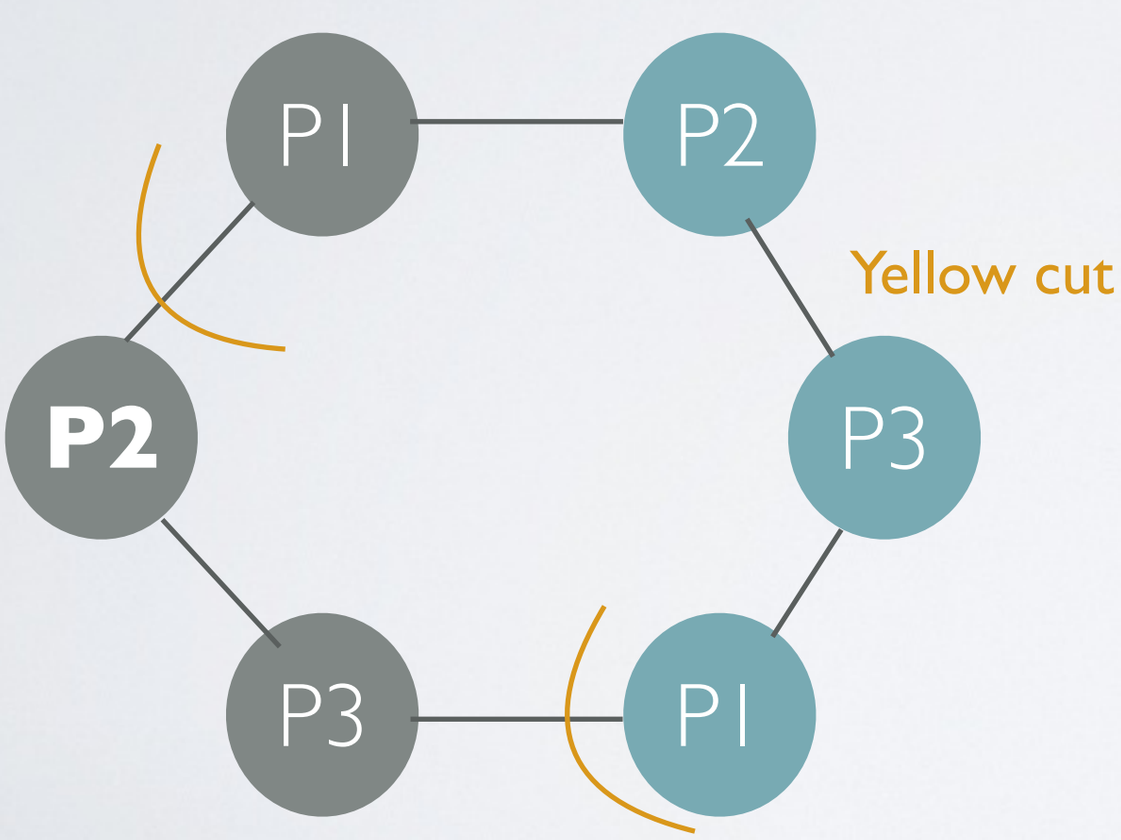
Process Byz simulates in B
all the processes that in S are in the yellow cut



System B

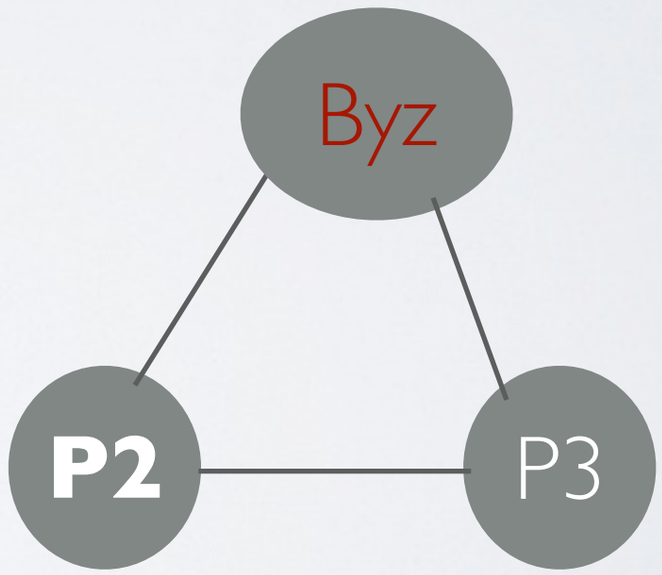
SCENARIO - 6 PROCESSES

Notice that the behaviour of Byz on B is well defined. if bluish P1 on S sends a message m to P3 at time t, Byz on B will send message m to P3 at time t, if grey P1 sends to p a message m' at time t' Byz will send m' to P2 in B at time t' and so on.



System S

Process Byz simulates in B
all the processes that in S are in the yellow cut

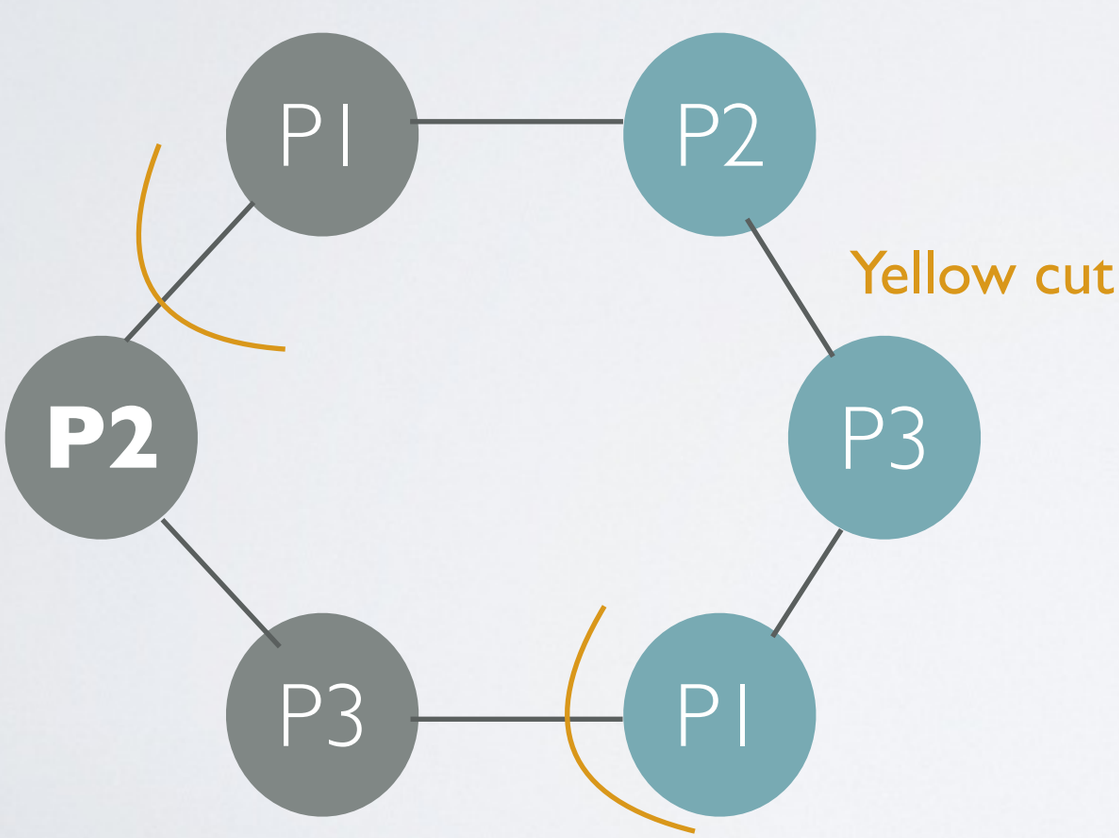


System B

SCENARIO - 6 PROCESSES

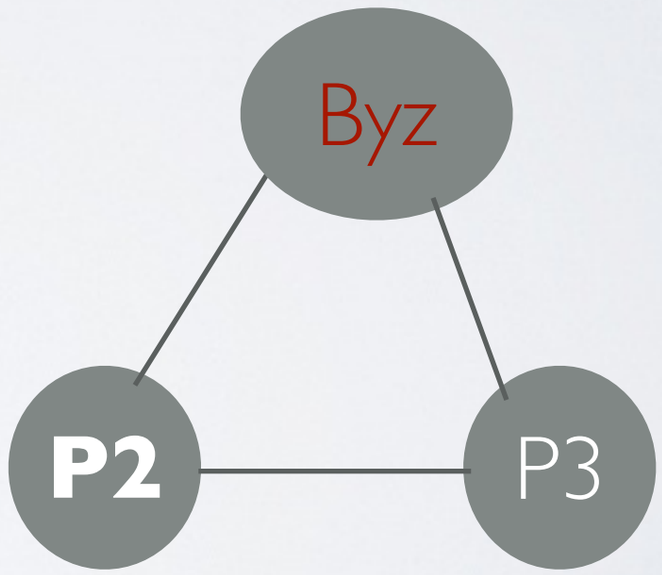
In System B process P2 has to terminate. We can replicate this algorithm for each process in S. Thus:

Fact 1: In system S each process terminates and outputs a decision value.



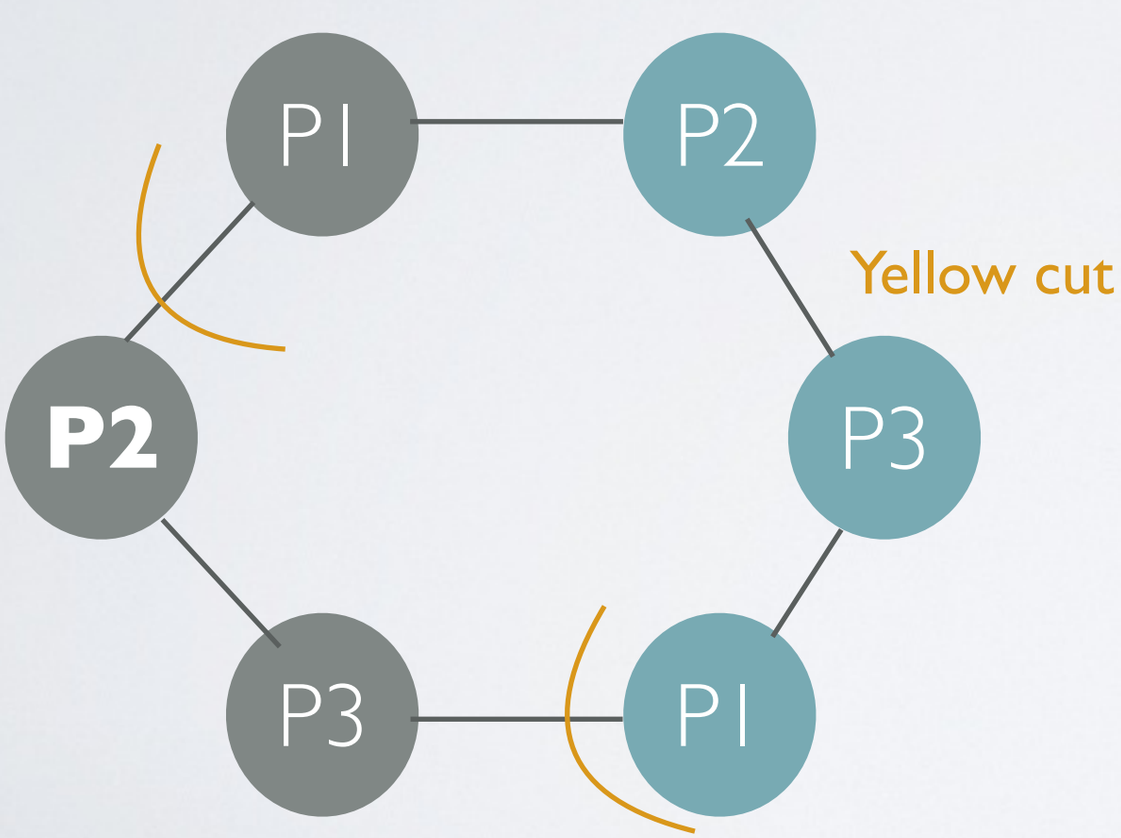
System S

Process Byz simulates in B
all the processes that in S are in the yellow cut



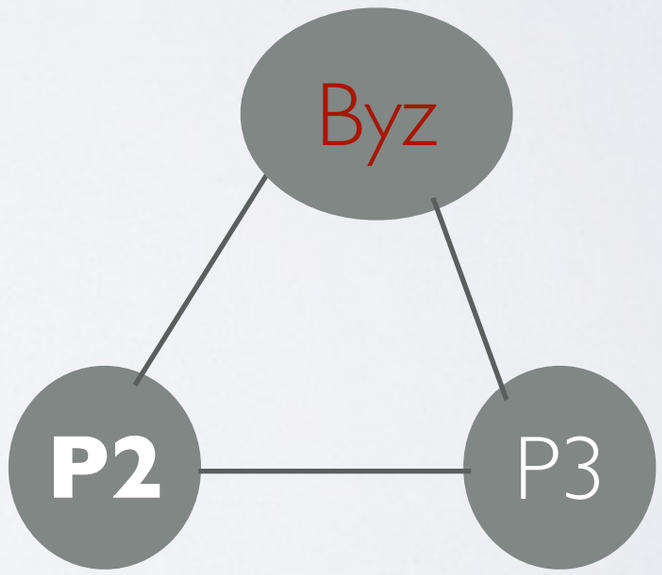
System B

Let us examine the grey process P2 in S. From its point of view S is undistinguishable from B. Process P2 in B has to decide 0 for the agreement and validity property (in B P2 and P3 are correct and both propose 0). This means that grey P2 has to output 0 in System S. The same holds for process P3. Therefore…



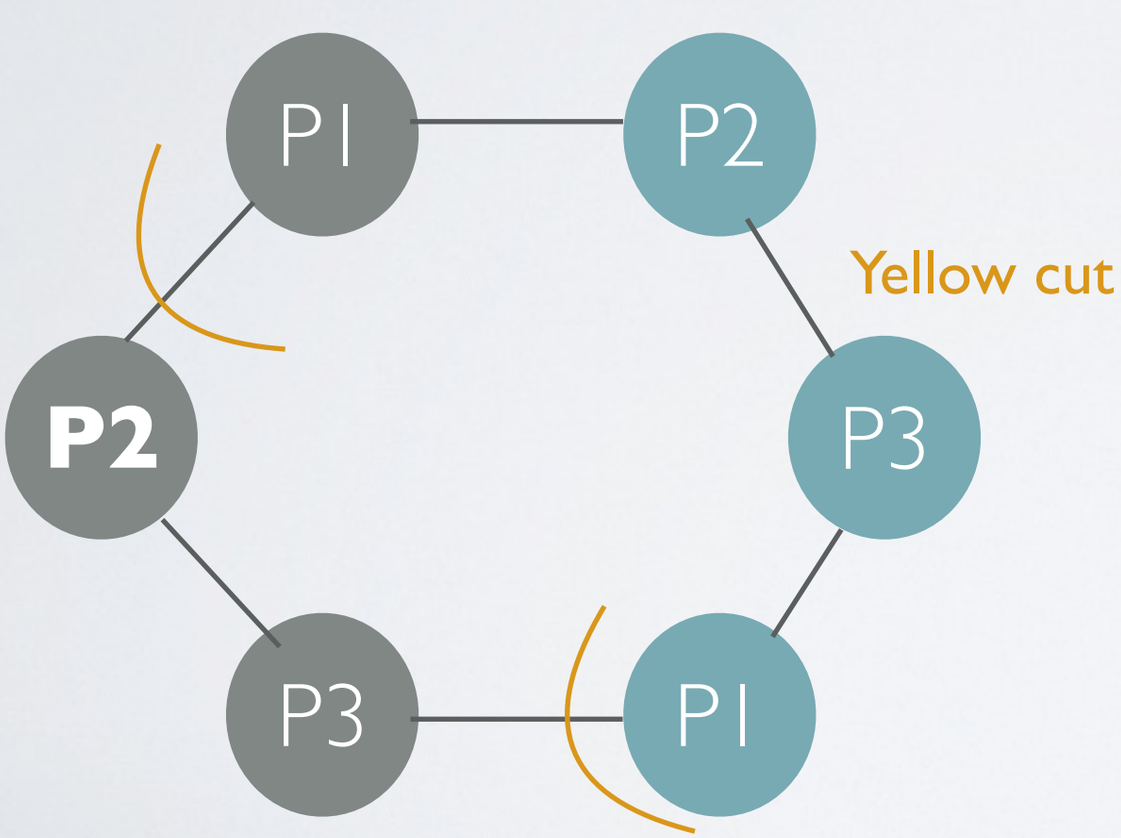
System S

Process Byz simulates in B
all the processes that in S are in the yellow cut



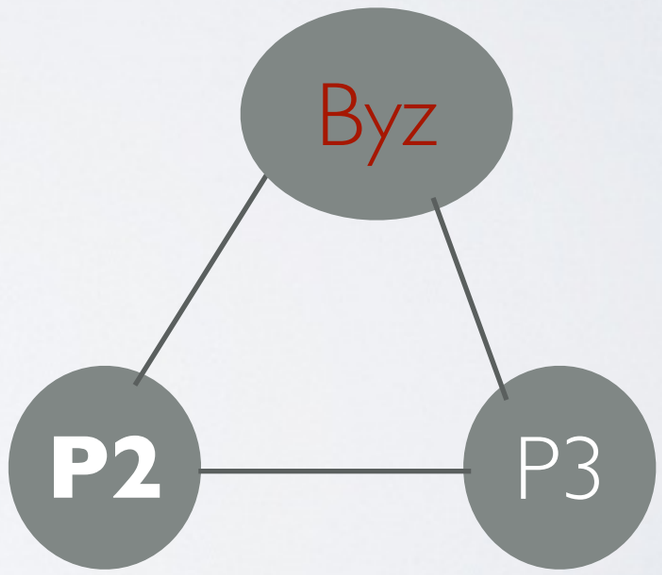
System B

FACT 2: P2 and P3 output the decision value 0 in S.



System S

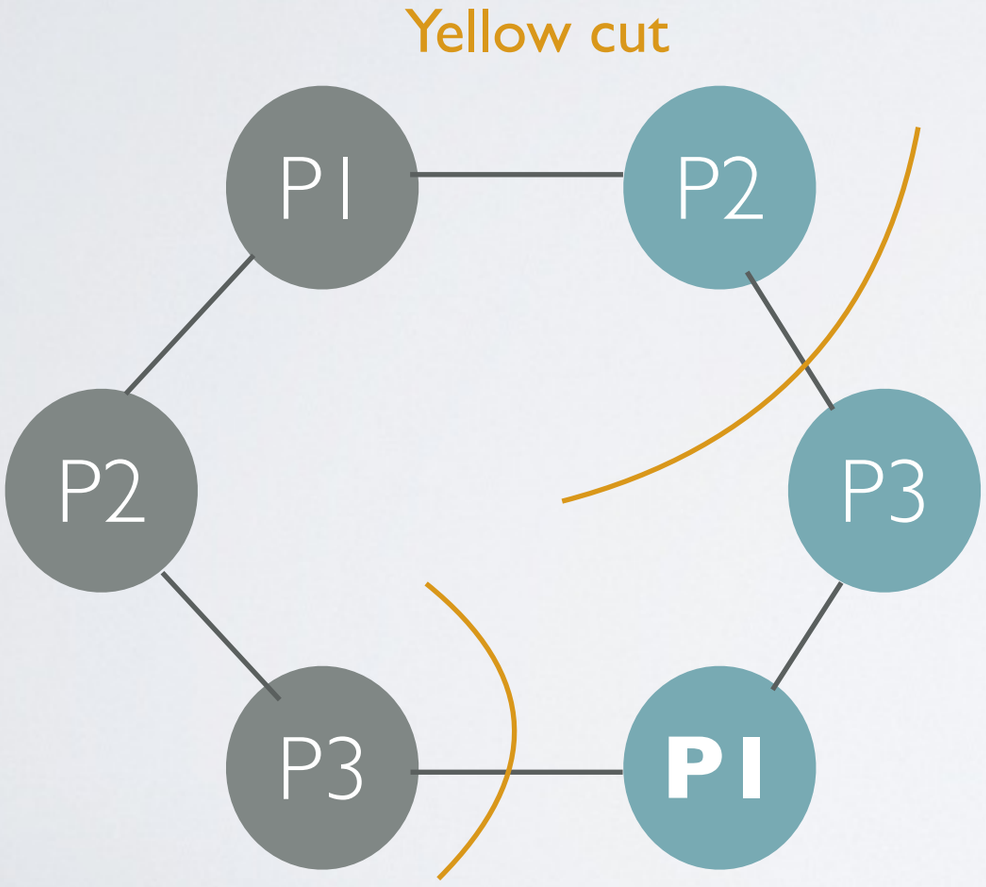
Process Byz simulates in B
all the processes that in S are in the yellow cut



System B

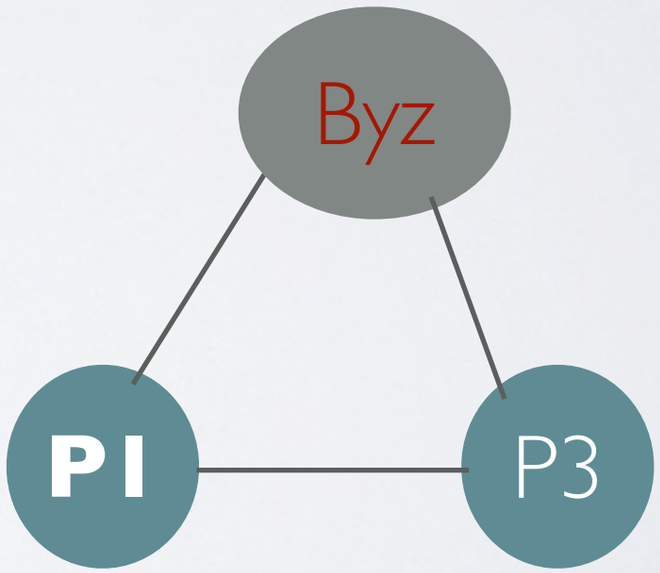
Consider now the bluish P1 and P3 processes in S. We can create a reduced system B' and repeat the reasoning of the previous slide. This shows that bluish P1 and P3 in S have to decide 1 as they cannot distinguish this system from B.

Fact 3: Bluish P1 and P3 have to decide 1 on S.



System S

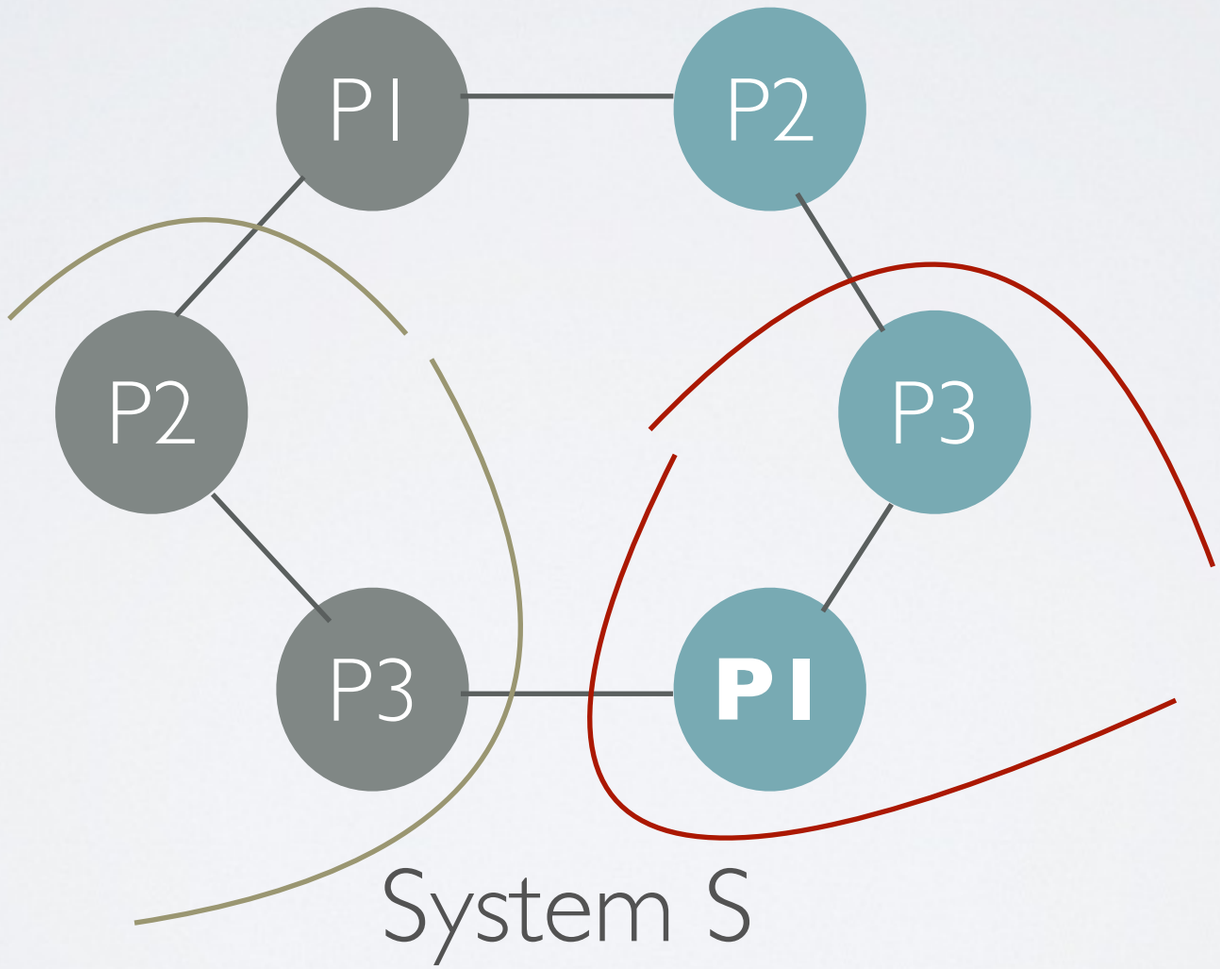
The bz process simulates the new yellow cut.



System B'

Therefore on System S we can identify two cuts:

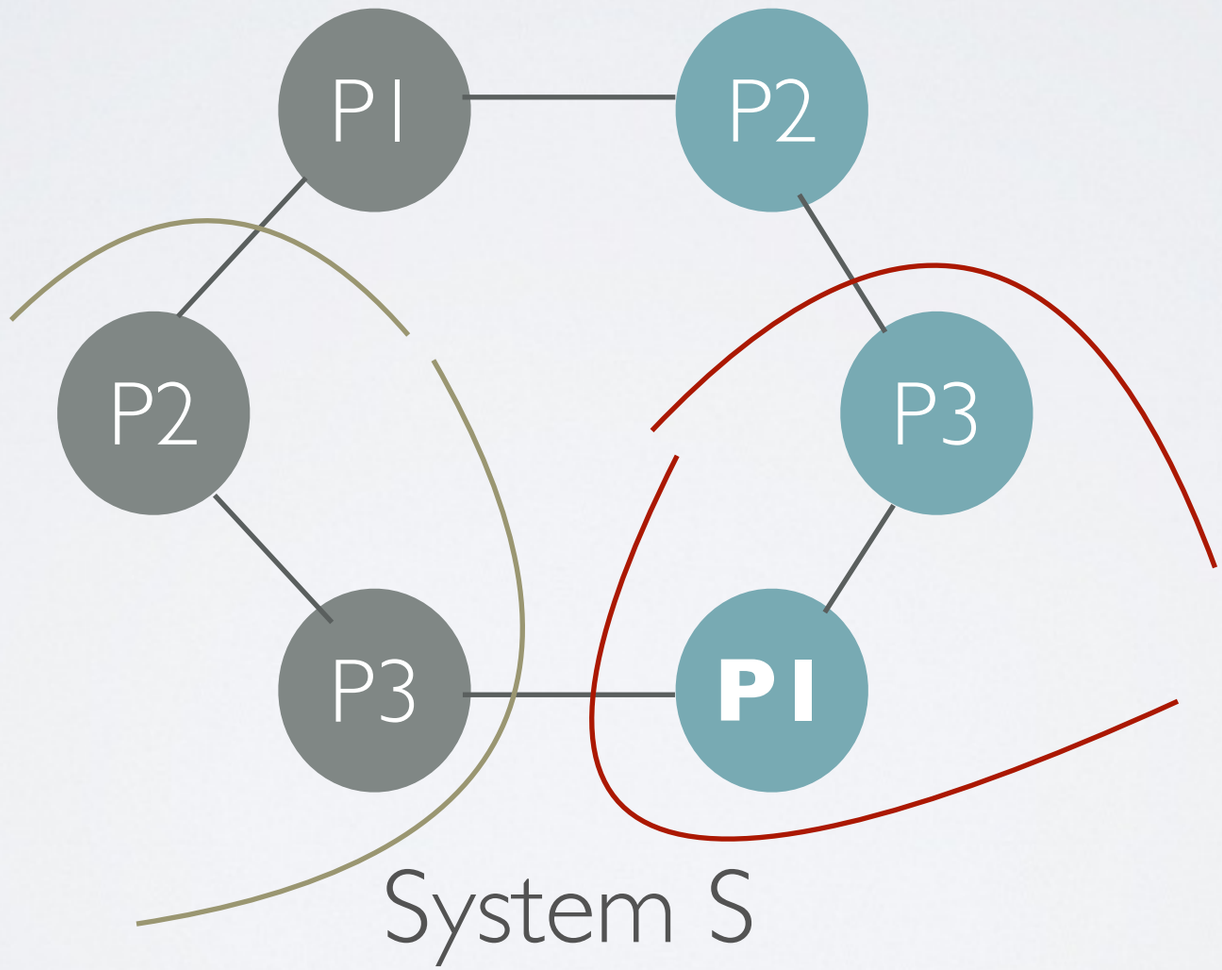
The green cut where grey P2 and P3 have to output 0 and the red cut where bluish P1 and P3 have to output 1. But this leads to a contradiction on the correctness of A, can you see why?



System S

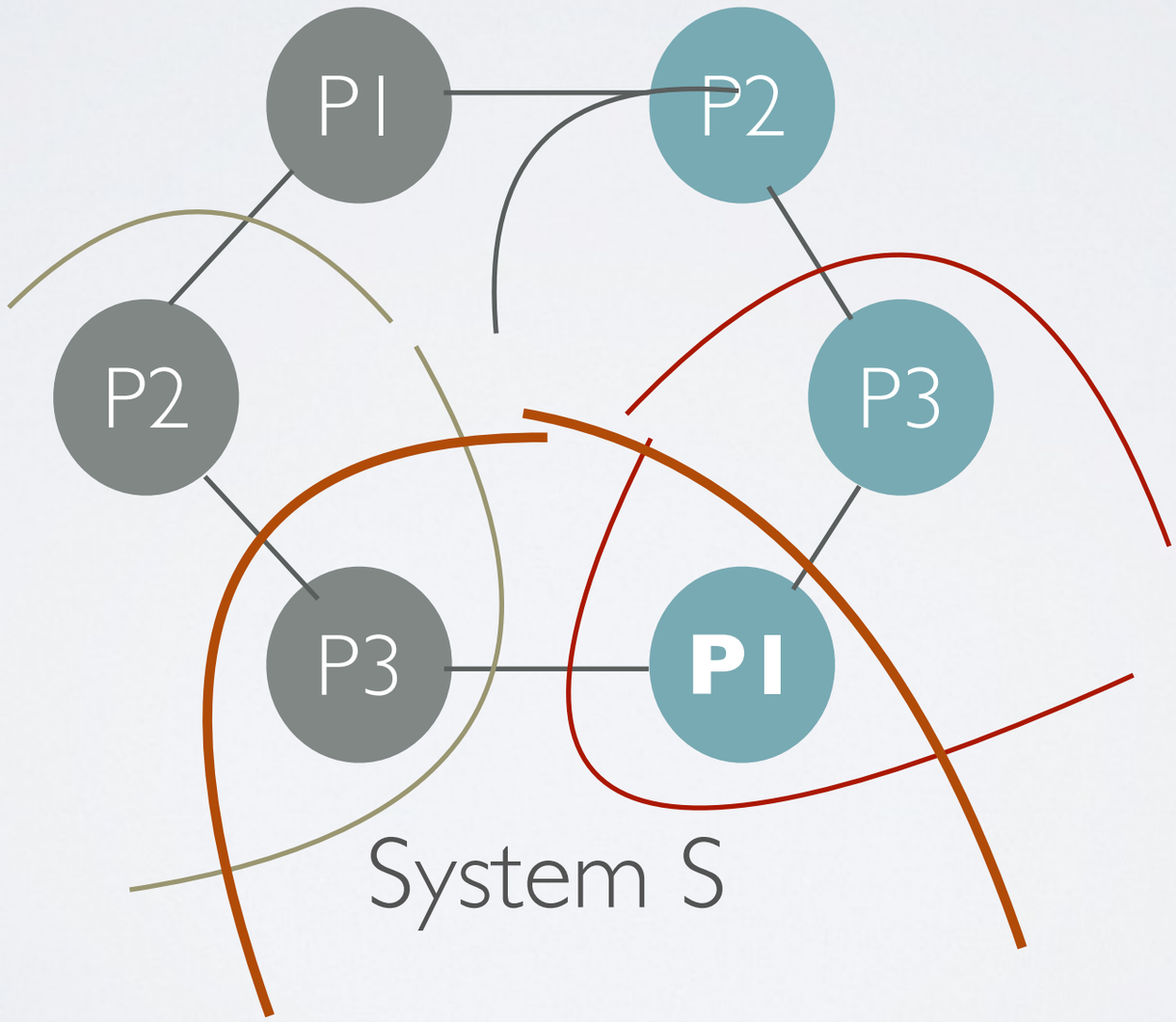
Therefore on System S we can identify two cuts:

The green cut where grey P2 and P3 have to output 0 and the red cut where bluish P1 and P3 have to output 1. But this leads to a contradiction on the correctness of A, **can you see why?**

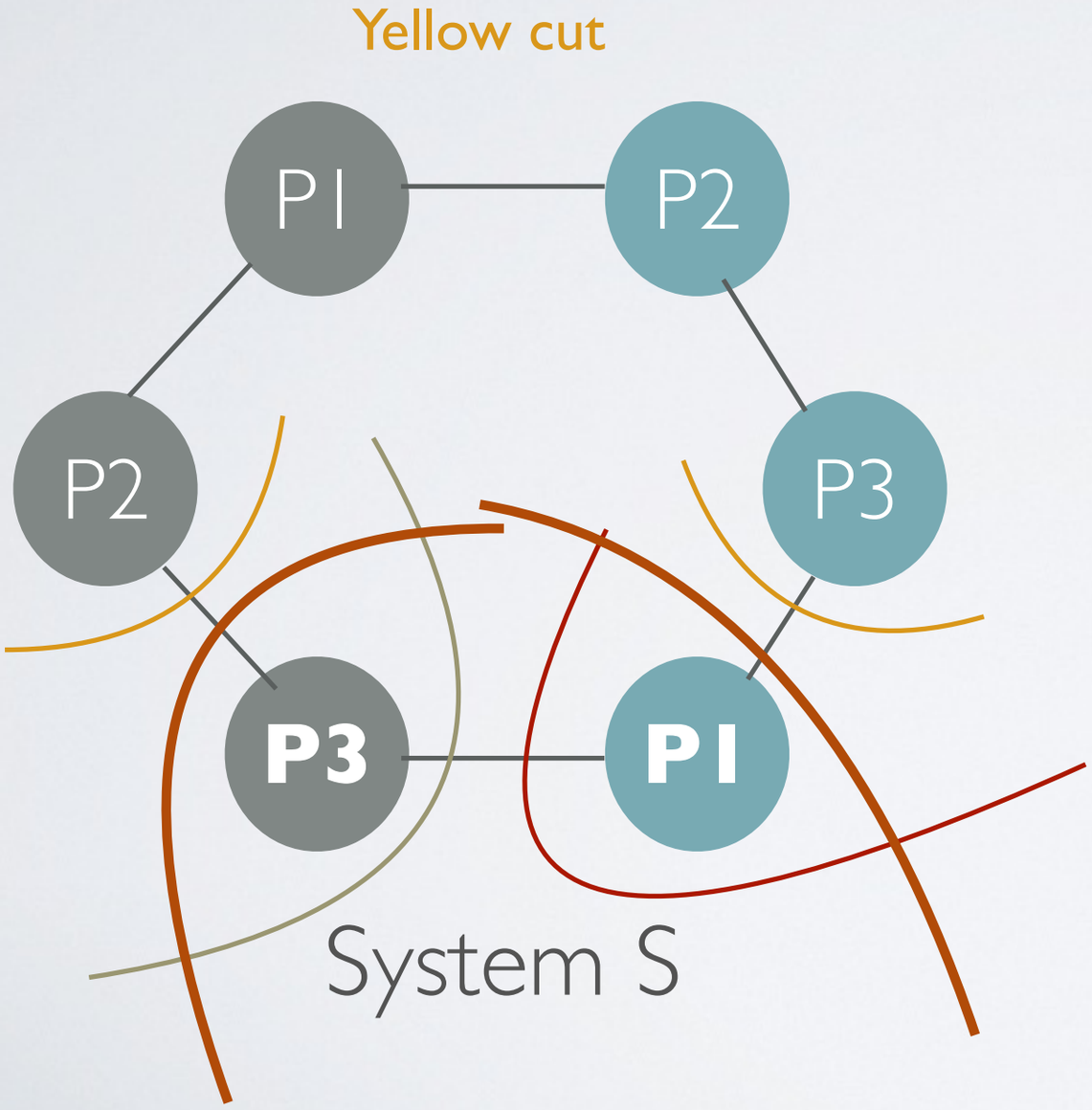


System S

can you see why? Consider the marked orange cut….

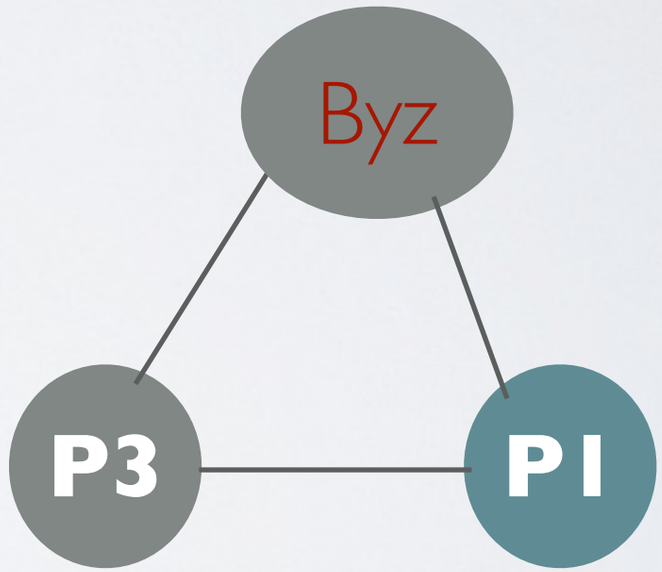


can you see why? Consider the marked orange cut…. We can build a proper system B'' on which **A must be correct.** On system B'' the grey P3 and the bluish P1 will output the same output they do on S. B'' cannot be distinguished from S. This implies that grey P3 output 0 and bluish P1 output 1.



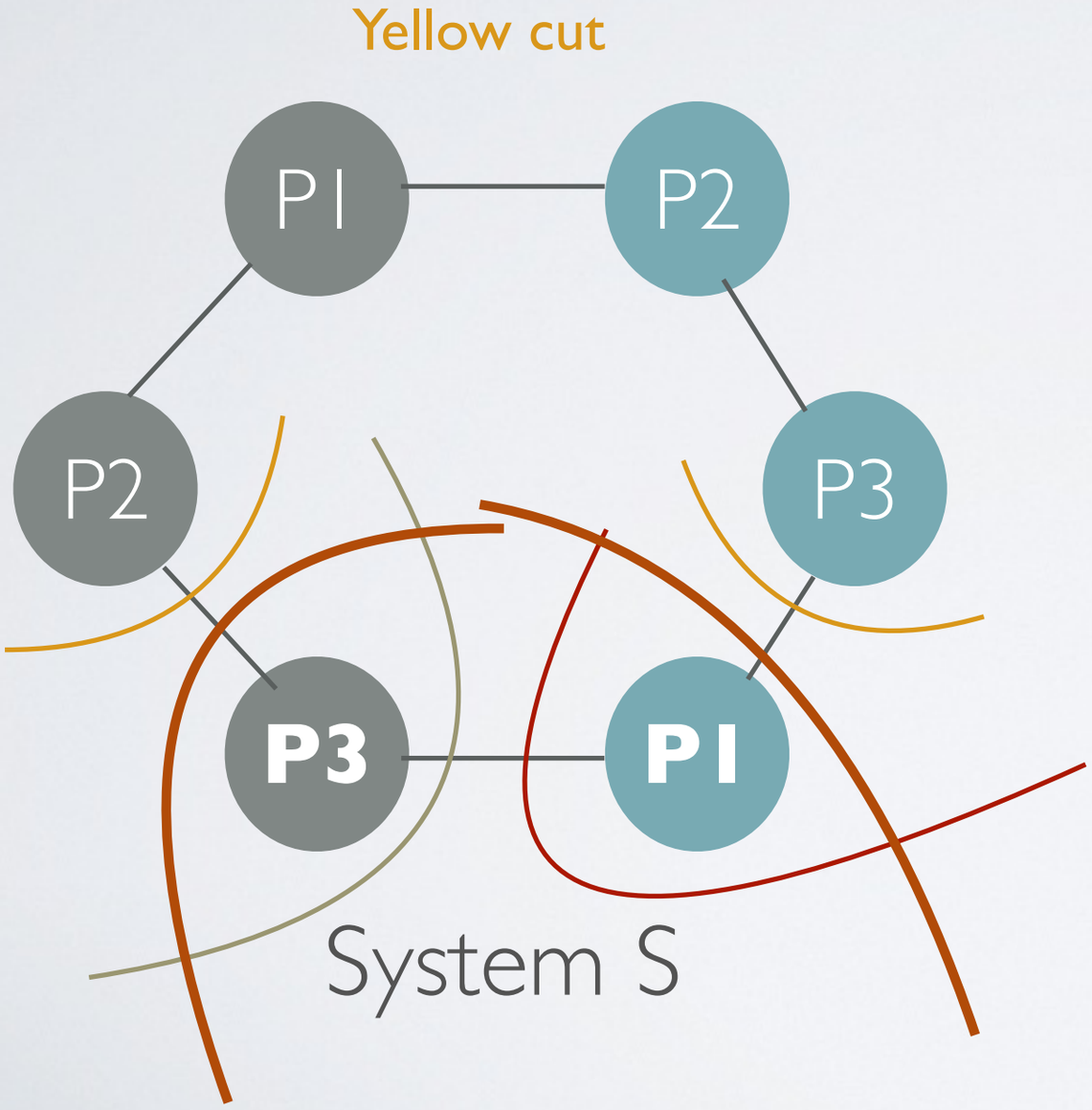
System S

The bz process simulates the last yellow cut.

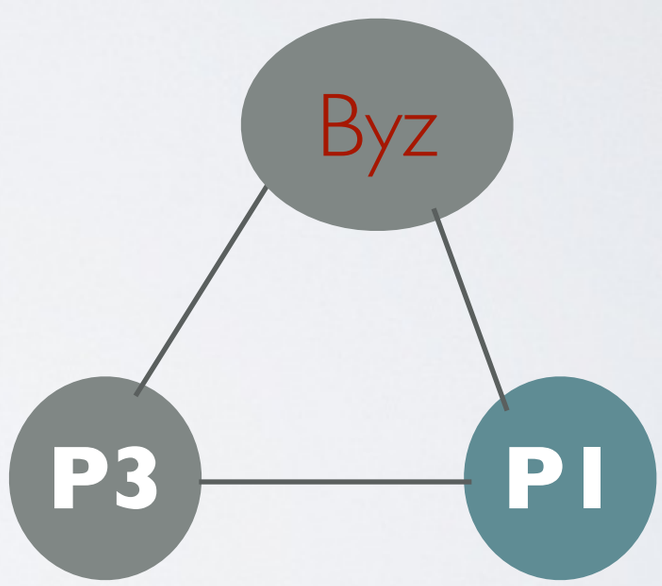


System B''

Contradiction: This implies that grey P3 output 0 and bluish P1 output 1. But by doing so they violate the agreement property on system B''. Therefore A is not correct on B''. Therefore, A is not a correct consensus algorithm.



The bz process simulates the last yellow cut.



WE CAN ITERATE THIS FOR ANY ROUND R

No consensus algorithm with all-same validity exists for 3 processes if one is byz.

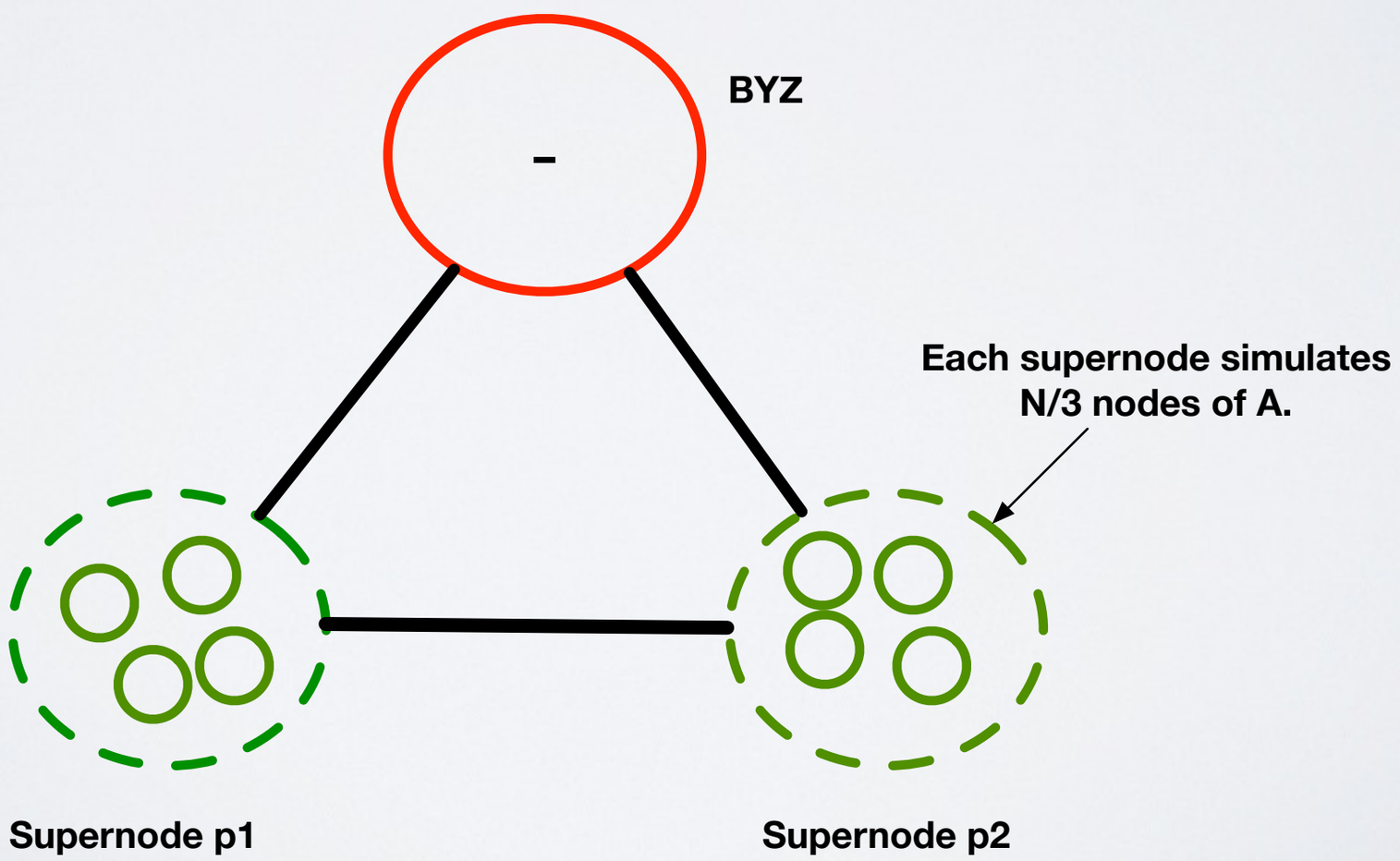
What about n=3f? It is still impossible. Can you see why?

WE CAN ITERATE THIS FOR ANY ROUND R

No consensus algorithm with all-same validity exists for 3 processes if one is byz.

What about n=3f? It is still impossible. Can you see why?

Intuitively: Suppose exists algorithm A that can with n=3f, then you can use A to solve the case of n=3



THEOREM

Theorem. Even **in a synchronous setting,** there is **no algorithm solving consensus** with any-input validity (and thus all the same validity) **on authenticated channels** if one third or more of the processes are byzantine. That is **f<N/3** is a **necessary requirement**.

REMEMBER: With crash failures we
Were able to tolerate **any amount** of failures
In Sync. System (using P)!

KING ALGORITHM

POSSIBILITY IF N>3F

If the number of correct is at least 3f+1 is possible to solve byzantine agreement. We will present the **King algorithm.**

This algorithm can be seen as an adaptation of Hierarchical consensus for byzantine failures.

The king algorithm works in synchronous, when f < n/3 and when no signatures are available. Authenticated channels - MAC are needed.

KING ALGORITHM

The king algorithm runs for f+1 phases.

- In each phase j there is a king. One way is for the king of phase j to be the node with id=j.
- Each node keep a variable x=to its proposal at the beginning.
- Variable x is updated during the algorithm.
- At the end of phase f+1 each correct decides variable x.

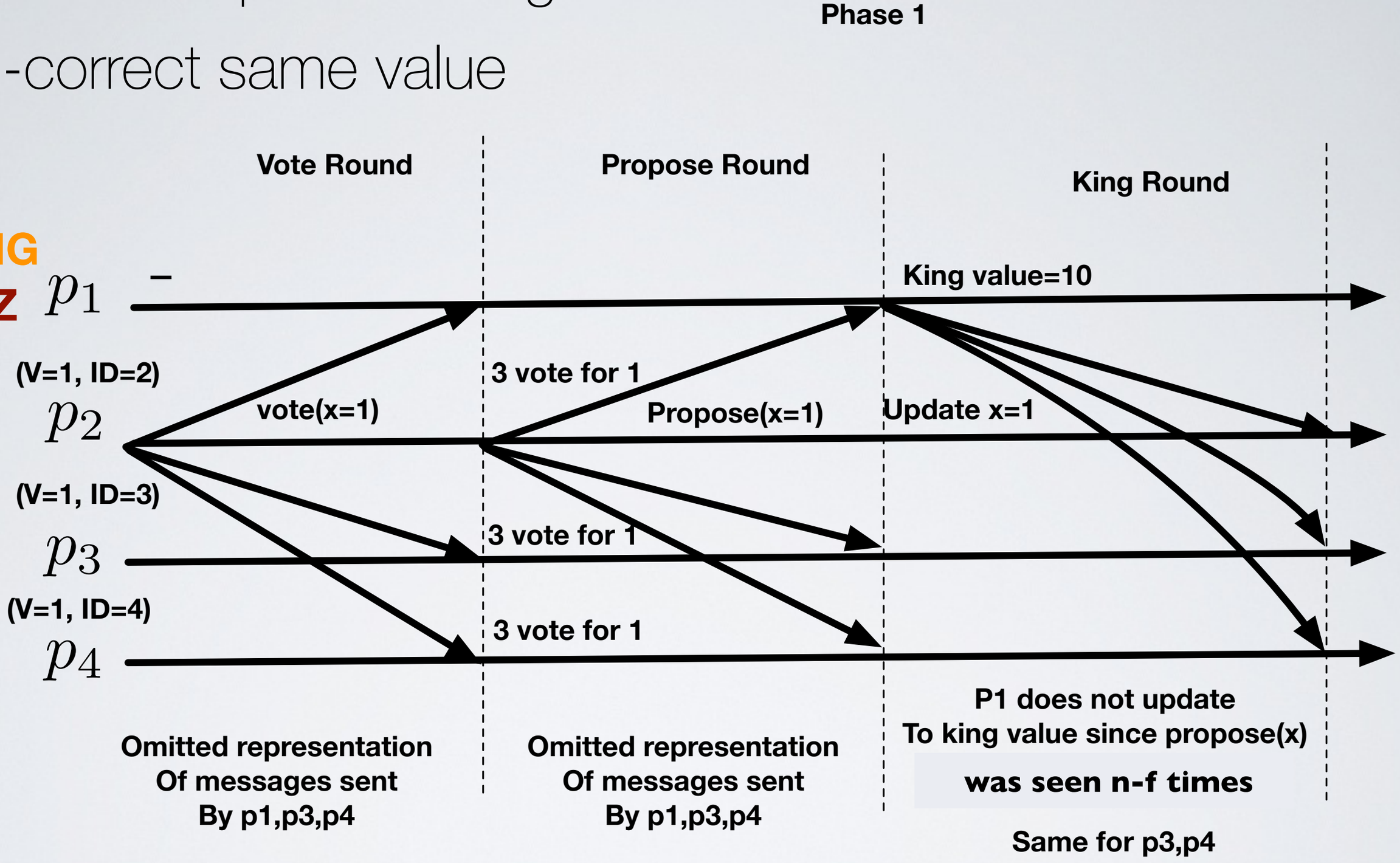
KING ALGORITHM

Each phase is divided in the following 3 rounds:

- **Vote round:** Each correct broadcasts x.
- **Propose round:** Each correct broadcasts a value y if it was received at least n-f times in vote round.
    - At the end of propose round a correct updates its value x to z if it sees z proposed at least f+1 times.
- **King round:** The king is the only one that broadcasts its value x.
    - At the end of the king round a node sets its value to king's value if during the propose round no value was seen n-f times.

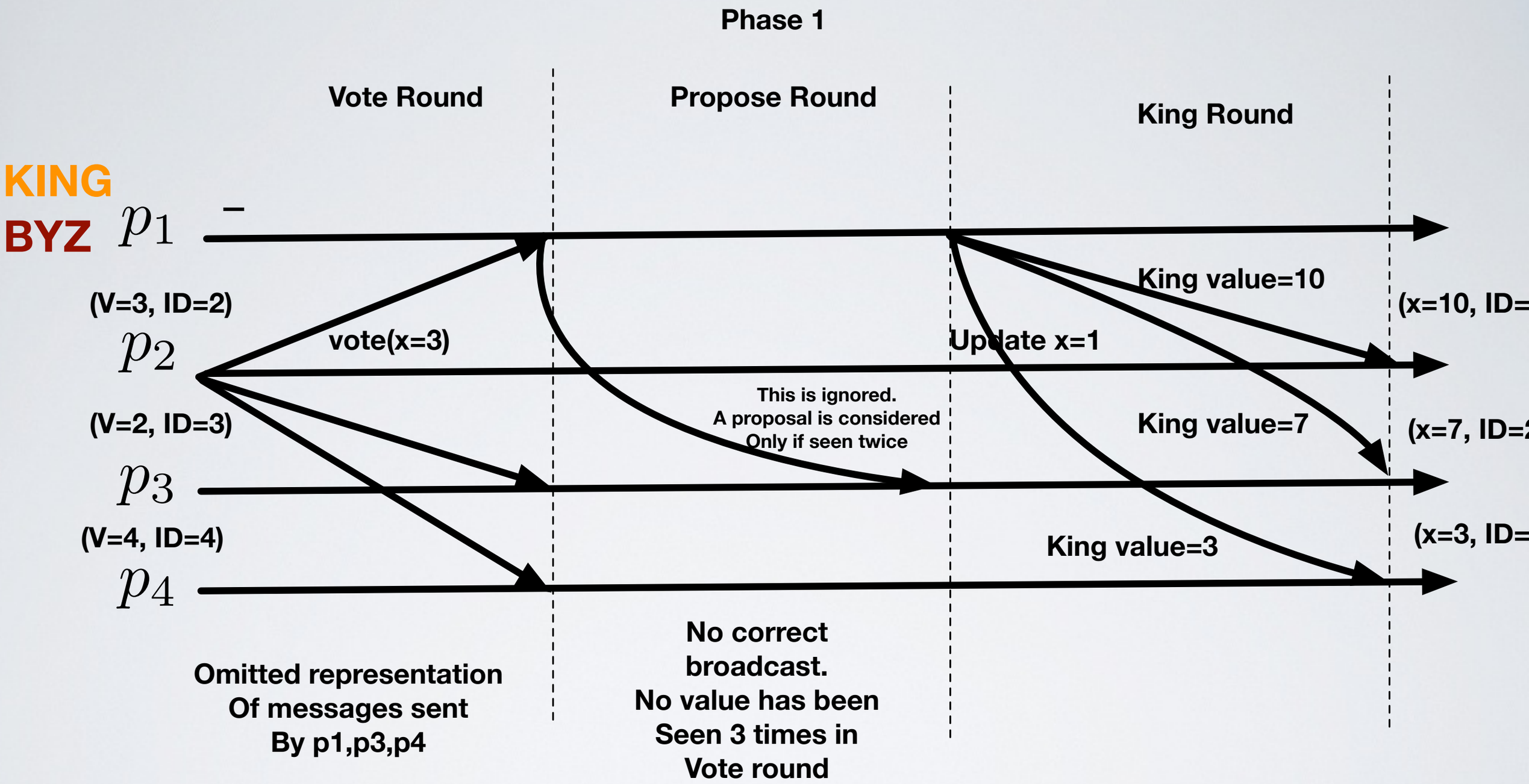
Phase example- Biz King

-All-correct same value

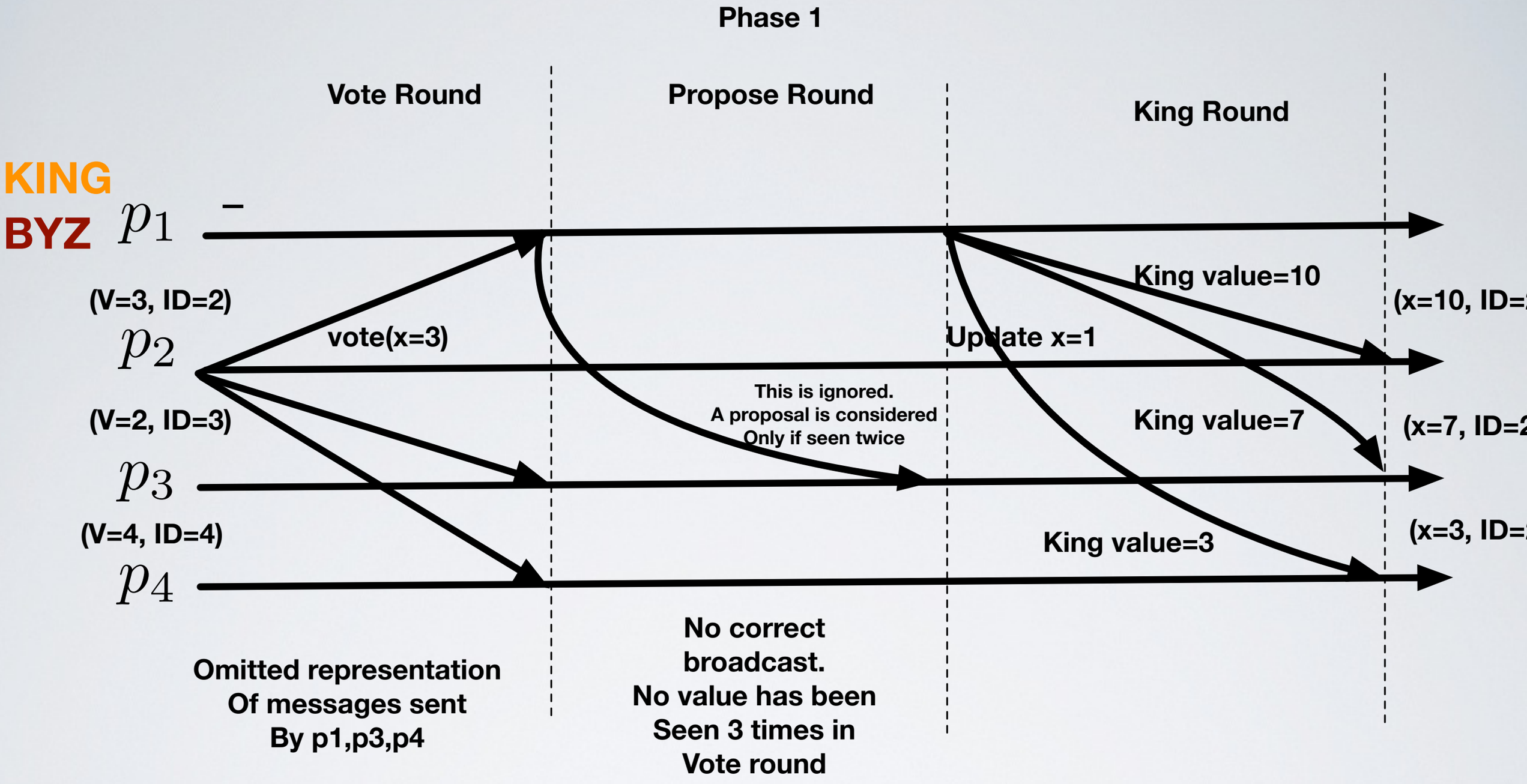


N=4, f=1, N-f=3, f+1=2

Phase example Biz King - not all correct same values



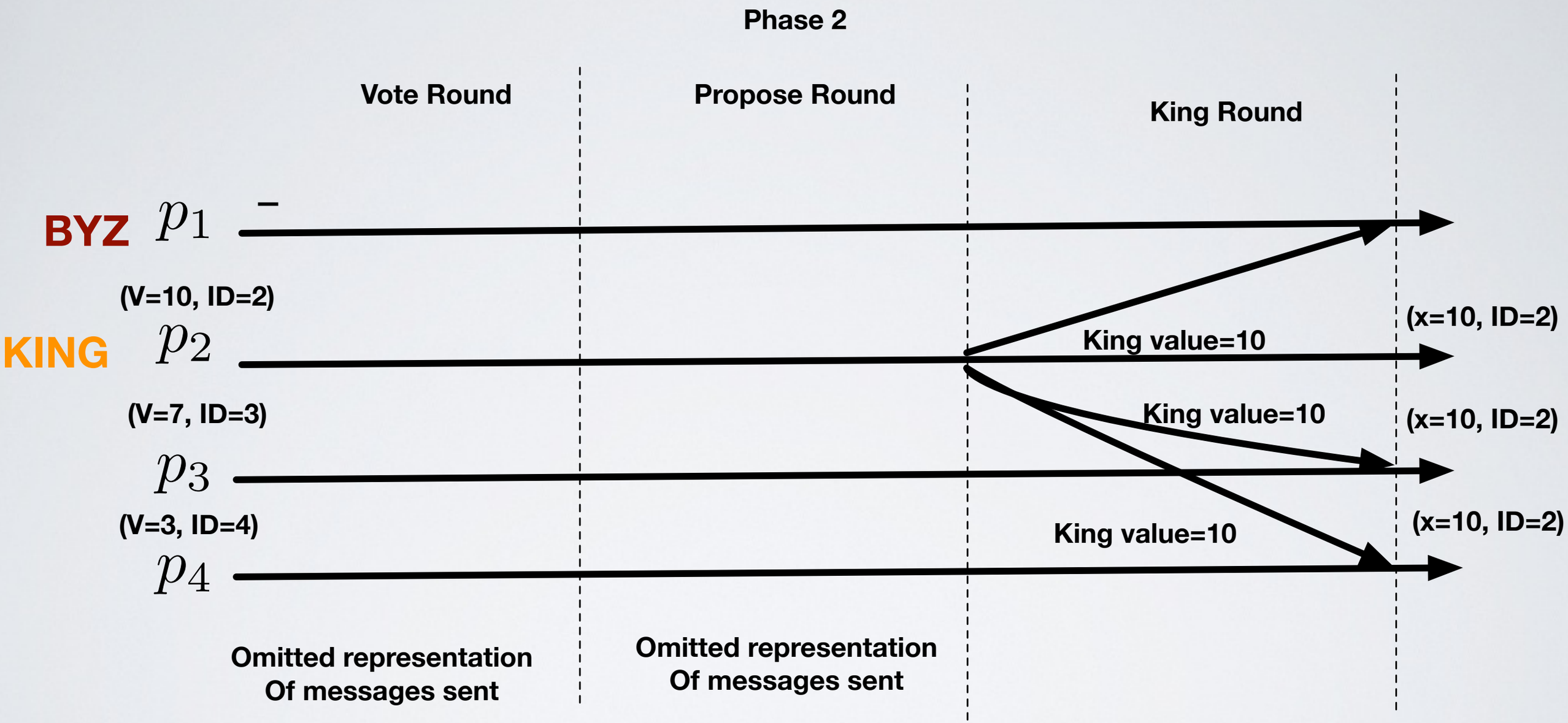
N=4, f=1, N-f=3, f+1=2



If corrects do not have all the same value, then bizKing Forces disagreement. Can you see which mechanism to fix it?

KING ALGORITHM

Phase example- not all correct same val, but good King



If a king is honest, it forces agreement (if not already reached).

KING ALGORITHM

Intuitively the king algorithm has the following mechanisms inside:

- Mechanism 1 - All same no change of idea (Implemented in Vote-Propose rounds): If all correct processes have the same value for x, they detect this during Vote-Propose round and they do not get influenced by a byzking and other byzantines.
- Mechanism 2 - Not All Same: Correct King imposes value (implemented in the king round) -> if corrects disagree the correct king imposes agreement.

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

1: $x =$ my input value
2: **for** phase $= 1$ to $f + 1$ **do**

Vote

3: Broadcast `value`(x)

Propose

4: **if** some `value`(y) received at least $n - f$ times **then**
5: Broadcast `propose`(y)
6: **end if**
7: **if** some `propose`(z) received more than f times **then**
8: $x = z$
9: **end if**

King

10: Let node v_i be the predefined king of this phase i
11: The king v_i broadcasts its current value w
12: **if** received strictly less than $n - f$ `propose`(y) **then**
13: $x = w$
14: **end if**
15: **end for**

At the end of phase f+1 each correct decides value x.

BIZANTINE AGREEMENT PROBLEM

Events:

- Propose(V)
- Decide=V

Properties:

- Termination: Every correct eventually decides
- All-same validity (Weak validity): If all correct processes propose v, the only decision is v.
- Integrity: No correct decides twice.
- Agreement: All correct decides the same value

PROOF

Termination is immediate from the structure of the algorithm, each correct decides at round f+1 and terminates.

Integrity is immediate from the fact that there exist a single line where a correct decides and this line can be executed just once.

We have to show:

- Validity -> if all correct starts with v they have to decide v.
- Agreement -> no two correct process decide differently.

PROOF

Mechanism 1

Th 1. If all corrects start with the same value v, then v is decided.

Proof:

Each correct broadcast v at the vote round.

Each correct sees v voted n-f times (note that a value voted only by the Byz has at most f votes).

Each correct proposes v at the propose round.

Each correct sees v proposed n-f times (note that a value proposed by the Byz has at most f proposes).

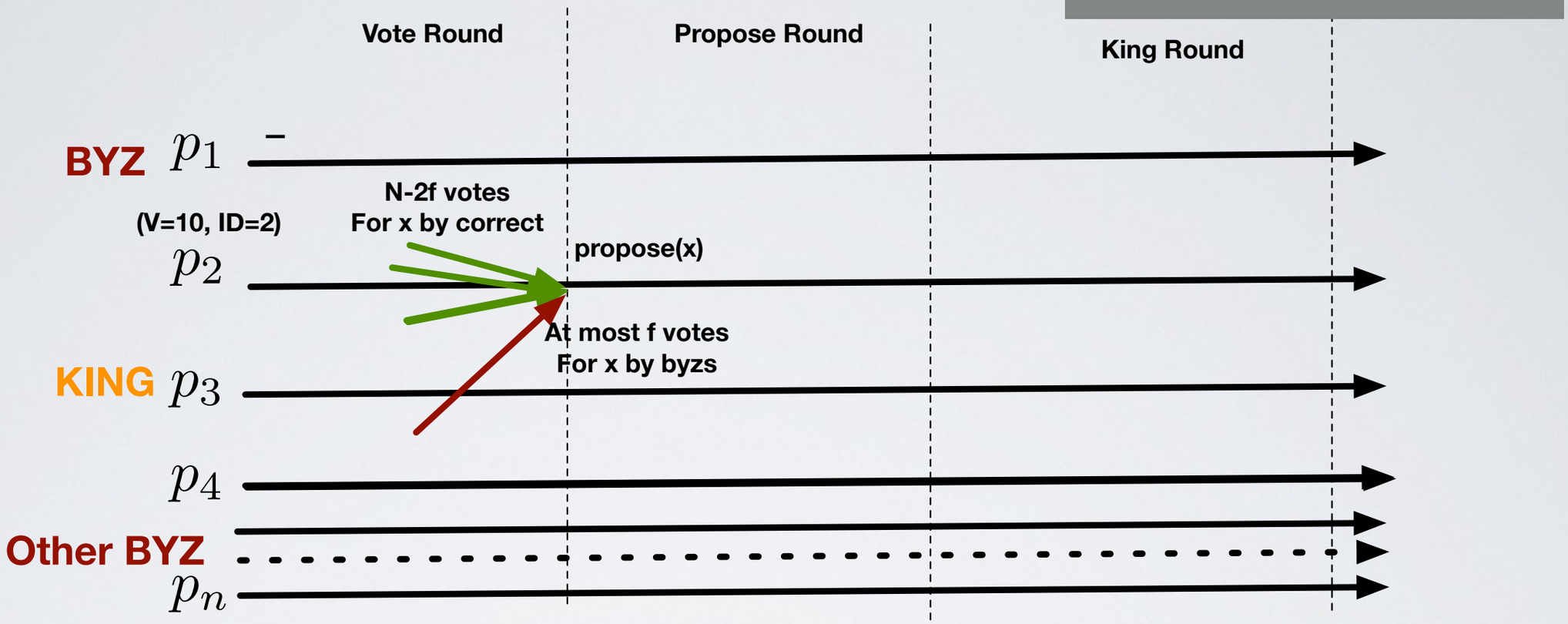
Each correct ignores the king round and keeps its value to v.

At phase f+1 each correct decides v.

Th. one shows that the algorithm respect validity.

PROOF

Mechanism 2



Lemma 1. If a correct process p proposes a value x in a phase, no other correct p' proposes a value y different than x.

Proof: If p proposes x, then N-2f votes seen are from correct.

If p' proposes y, then N-2f votes seen are from correct.

This implies that N>=2(N-2f)+f, N>=2N-3f (but **3f < N**), so N>=N+epsilon. Contradiction.

Alternatively, you may observe that, when N>3f, we have N-f>(N+f)/2

PROOF

Lemma 2. There is a phase with a correct king.

Proof: There f failures and f+1 phases, in each phase we pick a different king any set of f+1 processes contains at least a correct process.

Obs 1. Only a single value can be proposed more than f+1 times.

By Lemma 2 we have that all correct process propose the same value. Suppose there are two values v and v' each proposed f+1 times. Then there is a correct that proposes v and a correct that propose v'. By Lemma 2 we have v=v'.

PROOF

Lemma 3. After a phase with a correct king, all correct processes have the same value.

Proof. The correct king sends the same value to each process. All correct that change their proposal to the one of the king get the same value. It remains to show what happen to a correct process p' that do not accept the value of the king (let p be the king). P' does not accept the value if it has seen a value v proposed n-f times in the propose phase. This means that v has been proposed by n-2f correct processes (at least) n-2f>f+1 (recall n>3f). This implies that v has been seen at least f+1 time by each correct in the system. This implies that at the end of the propose round all correct have set their value to v (also the king p) (see Obs. 1). Thus p', even if does not accept the value of the king, has the same value v of the king (that he sets at line 8).

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

1: $x =$ my input value
2: **for** phase $= 1$ to $f + 1$ **do**

Vote

3: Broadcast `value`(x)

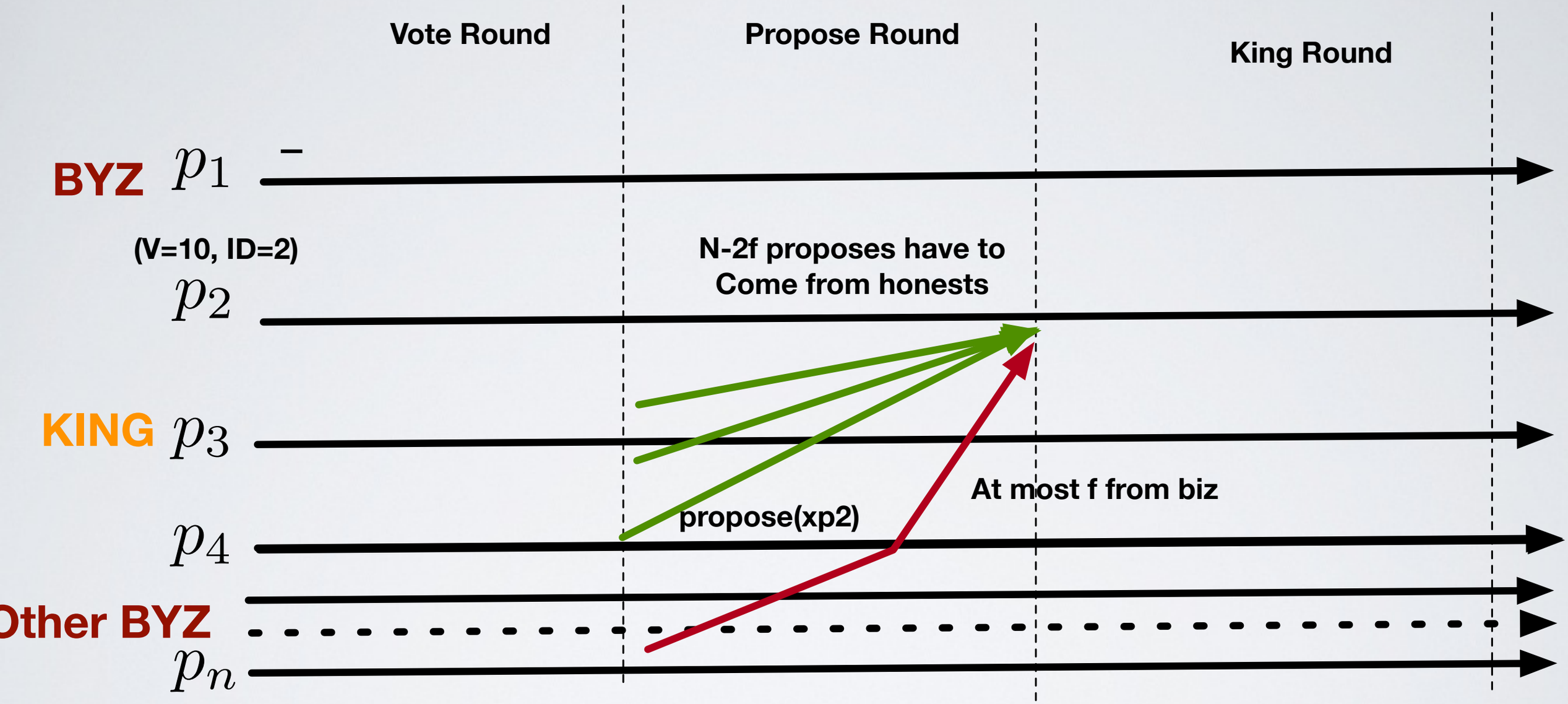
Propose

4: **if** some `value`(y) received at least $n - f$ times **then**
5: Broadcast `propose`(y)
6: **end if**
7: **if** some `propose`(z) received more than f times **then**
8: $x = z$
9: **end if**

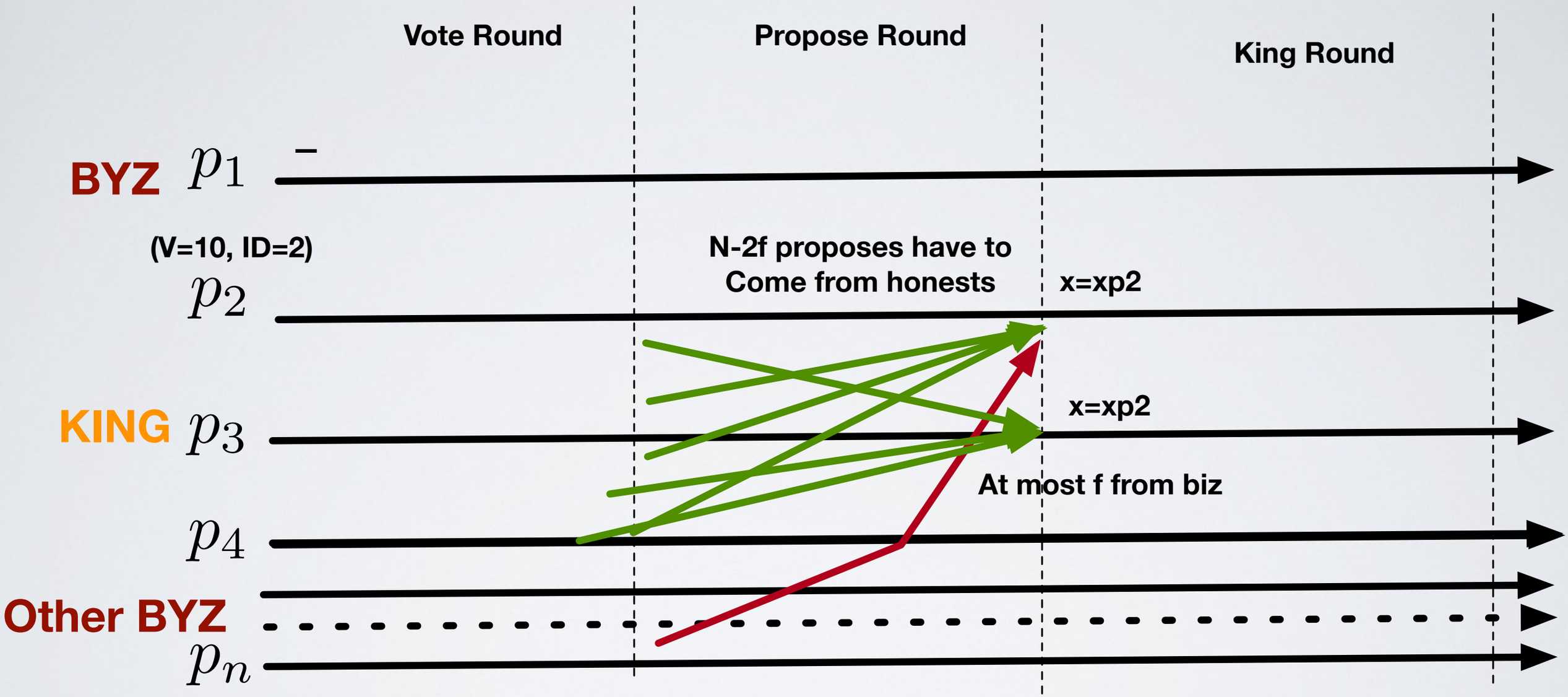
King

10: Let node v_i be the predefined king of this phase i
11: The king v_i broadcasts its current value w
12: **if** received strictly less than $n - f$ `propose`(y) **then**
13: $x = w$
14: **end if**
15: **end for**

KING ALGORITHM

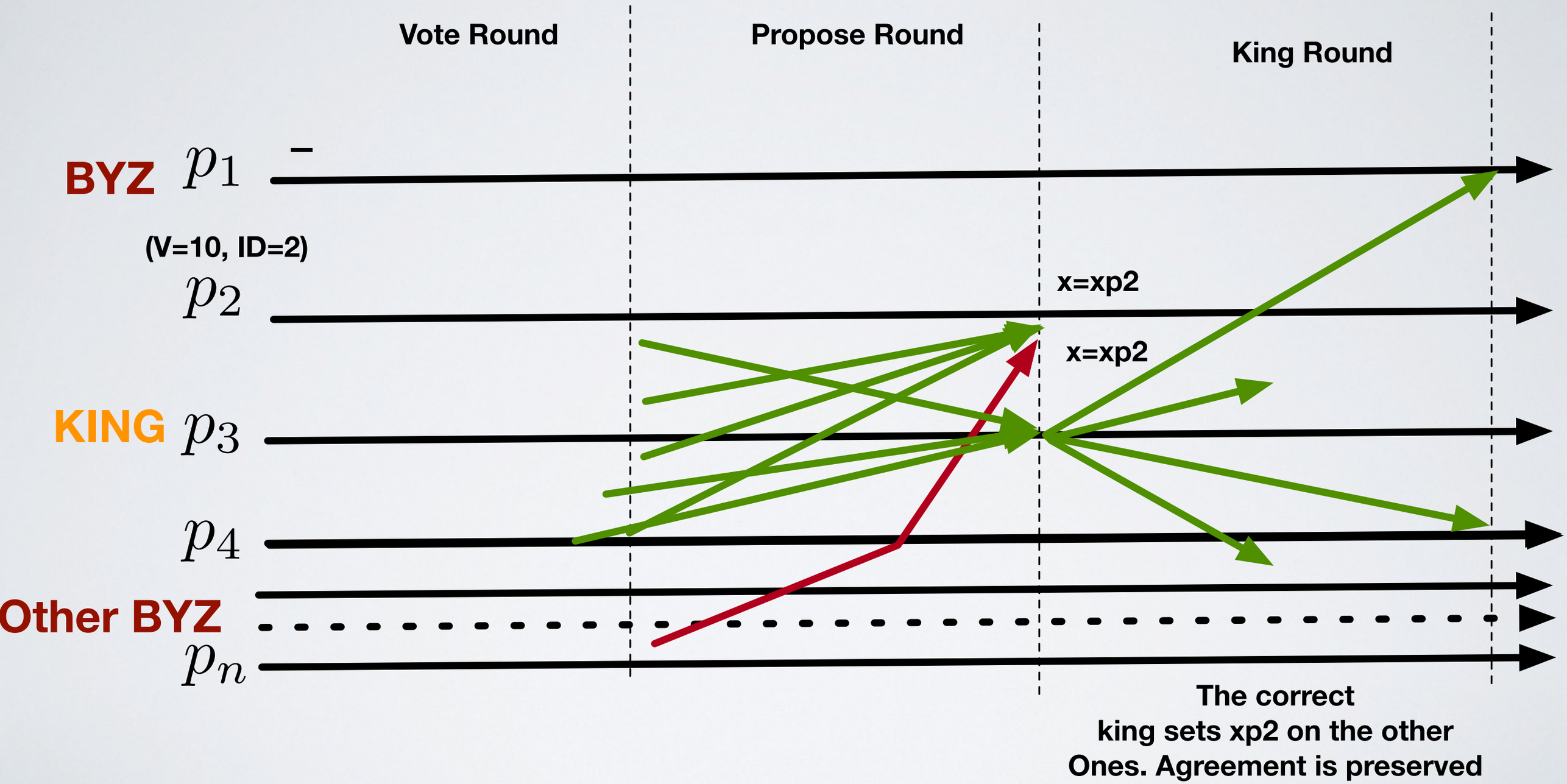


KING ALGORITHM



This implies that every honest has seen at least f+1 proposal
For Xp2 setting x to xp2, this includes the correct king

KING ALGORITHM



PROOF

Theorem 2. The king algorithm respects the agreement property.

Proof. From Lemma 3 and Lemma 2.

LANDSCAPE OF DISTRIBUTED COMPUTATION WITH BYZANTINE FAILURES

Primitive	System	Without Signatures	With Signatures
Reliable Broadcast	Synchronous	f<n/3	f<n
	Asynchronous	f<n/3	f<n
Consensus (all same validity)	Synchronous	f<n/3	f<n/2
	Asynchrounous	Impossible (FLP)	Impossible (FLP)