Consensus in Synchronous System

Sethanant Pipatpakorn

f: number of process the crashed
n: number of process

The algorithm will do for f + 1 round

Each process has three variable to keep track

- Values of previous round
- Values of this round (Initially as empty set {})
- Values of the next round

Round 0

At round 0 every process has values of empty set

Round 0 Values = {}

P1

P2

Round 0 Values = {}

P3

Round 0 Values = {}

P4

Round 0 Values = {}

Beginning of Round 1

Every process propose the value to itself

Round 1 Values = {10}

Round 2 Values = {10}

P1

P2

Round 1 Values = {3}

Round 2 Values = {3}

P3

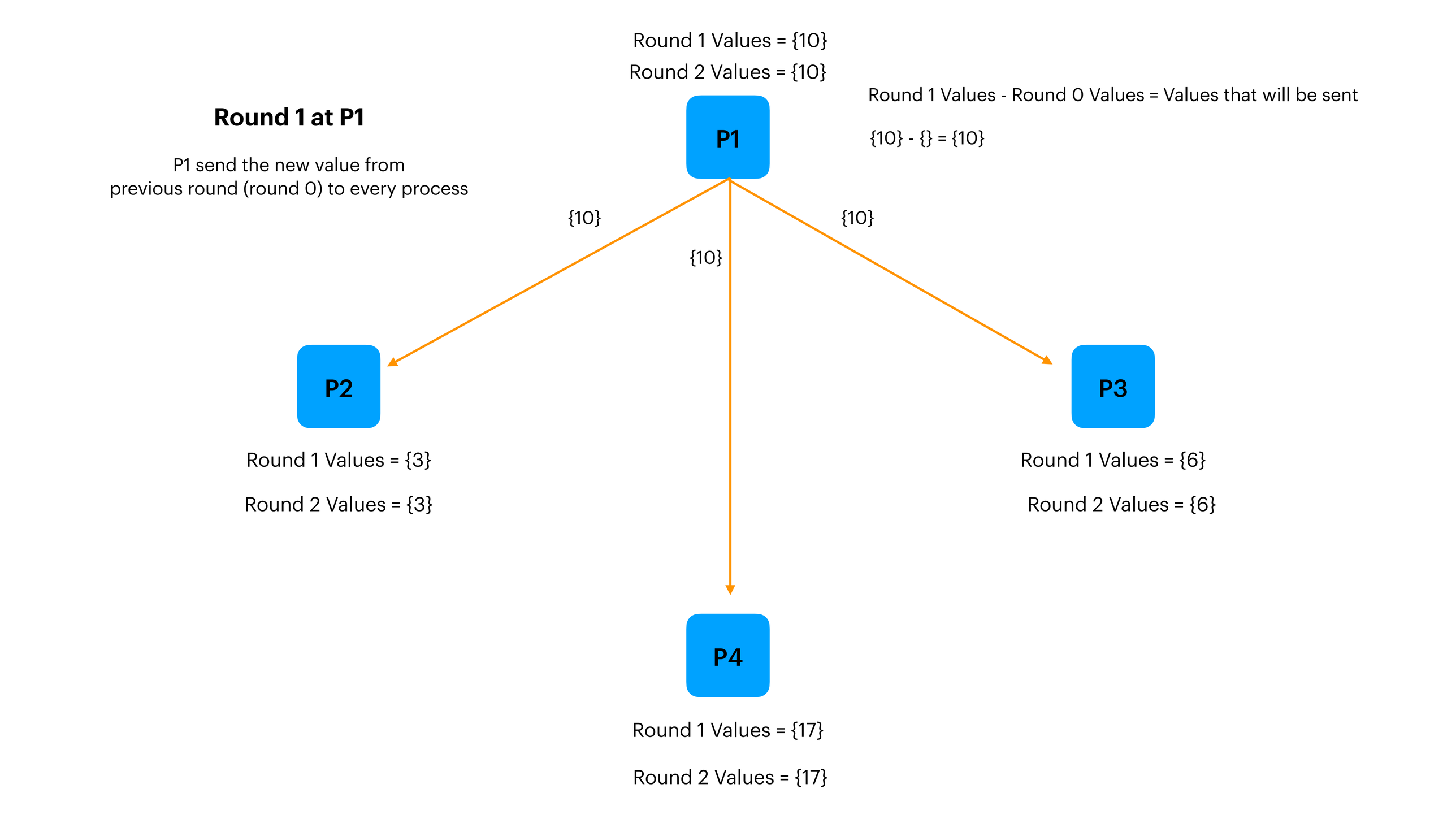
Round 1 Values = {6}

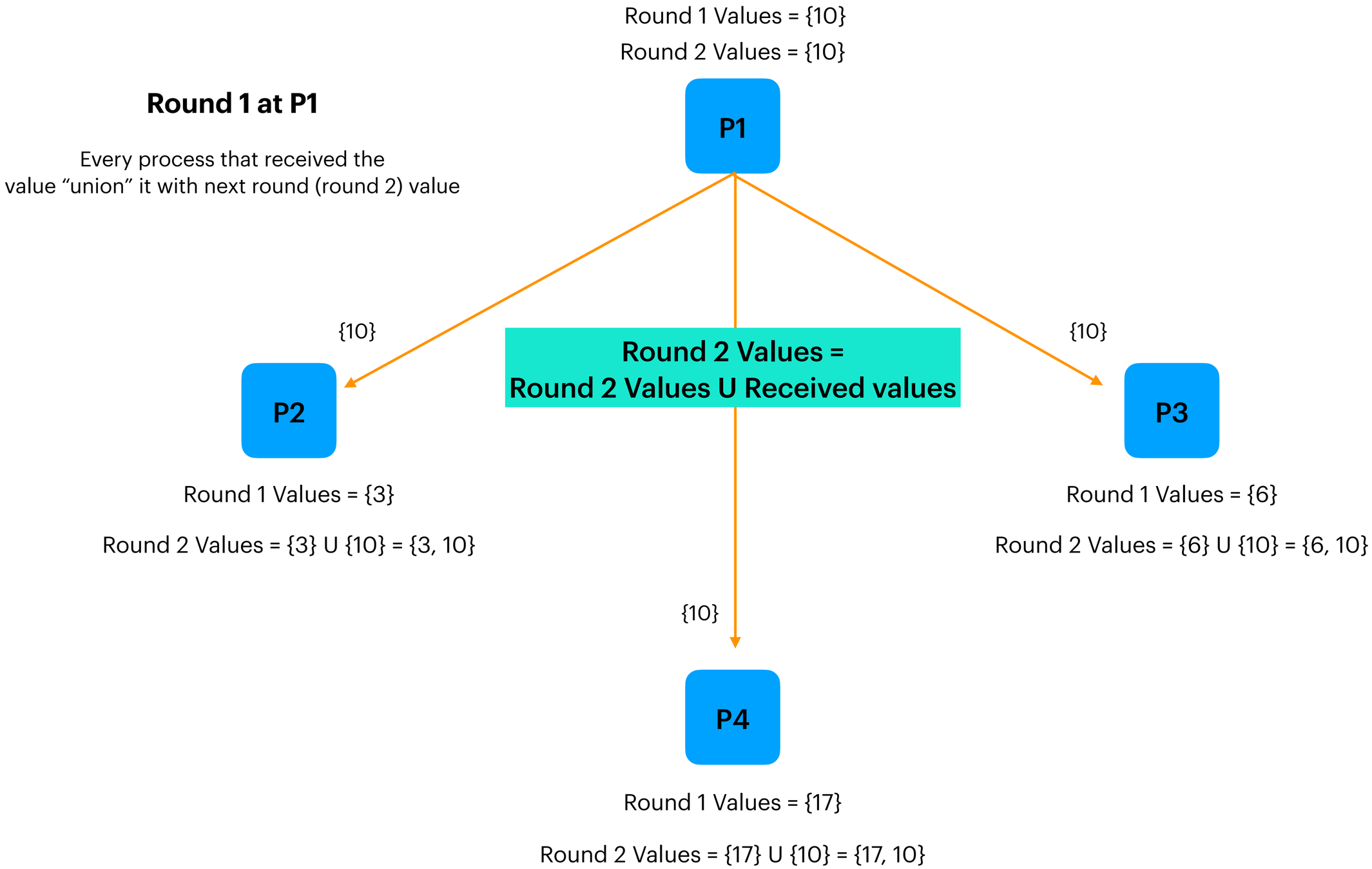
Round 2 Values = {6}

P4

Round 1 Values = {17}

Round 2 Values = {17}


Round 1 Values = {10}
Round 2 Values = {10}
Round 1 Values - Round 0 Values = Values that will be sent
{10} - {} = {10}
Round 1 at P1
P1 send the new value from
previous round (round 0) to every process
P1
{10}
{10}
{10}
P2
P3
Round 1 Values = {3}
Round 2 Values = {3}
Round 1 Values = {6}
Round 2 Values = {6}
P4
Round 1 Values = {17}
Round 2 Values = {17}


Round 1 Values = {10}
Round 2 Values = {10}
Round 1 at P1
Every process that received the value “union” it with next round (round 2) value
P1
{10}
Round 2 Values = Round 2 Values U Received values
{10}
P2
P3
Round 1 Values = {3}
Round 2 Values = {3} U {10} = {3, 10}
Round 1 Values = {6}
Round 2 Values = {6} U {10} = {6, 10}
{10}
P4
Round 1 Values = {17}
Round 2 Values = {17} U {10} = {17, 10}

Ending of Round 1 at P1

Round 1 Values = {10}

Round 2 Values = {10}

P1

P2

Round 1 Values = {3}

Round 2 Values = {3, 10}

P3

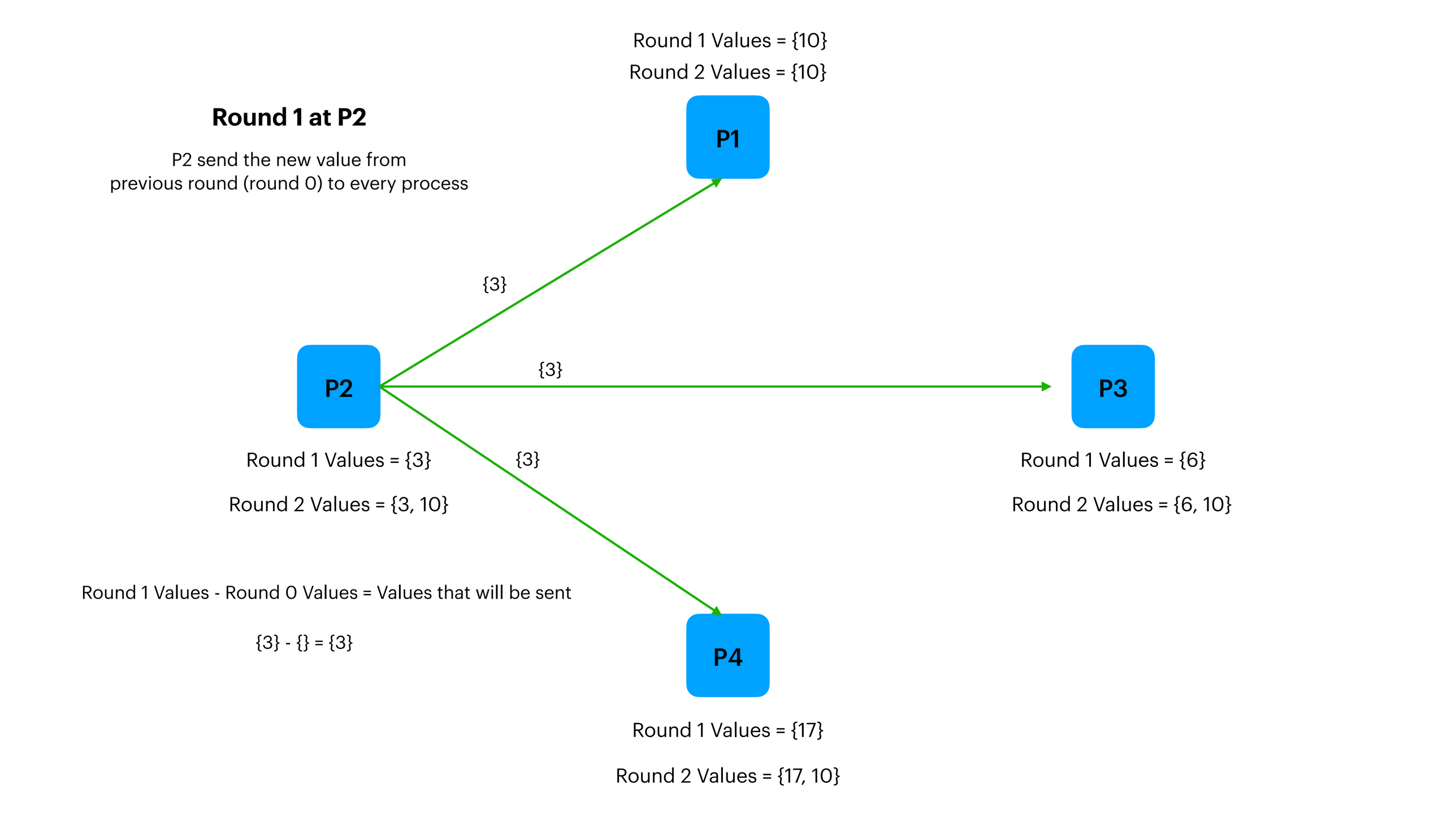
Round 1 Values = {6}

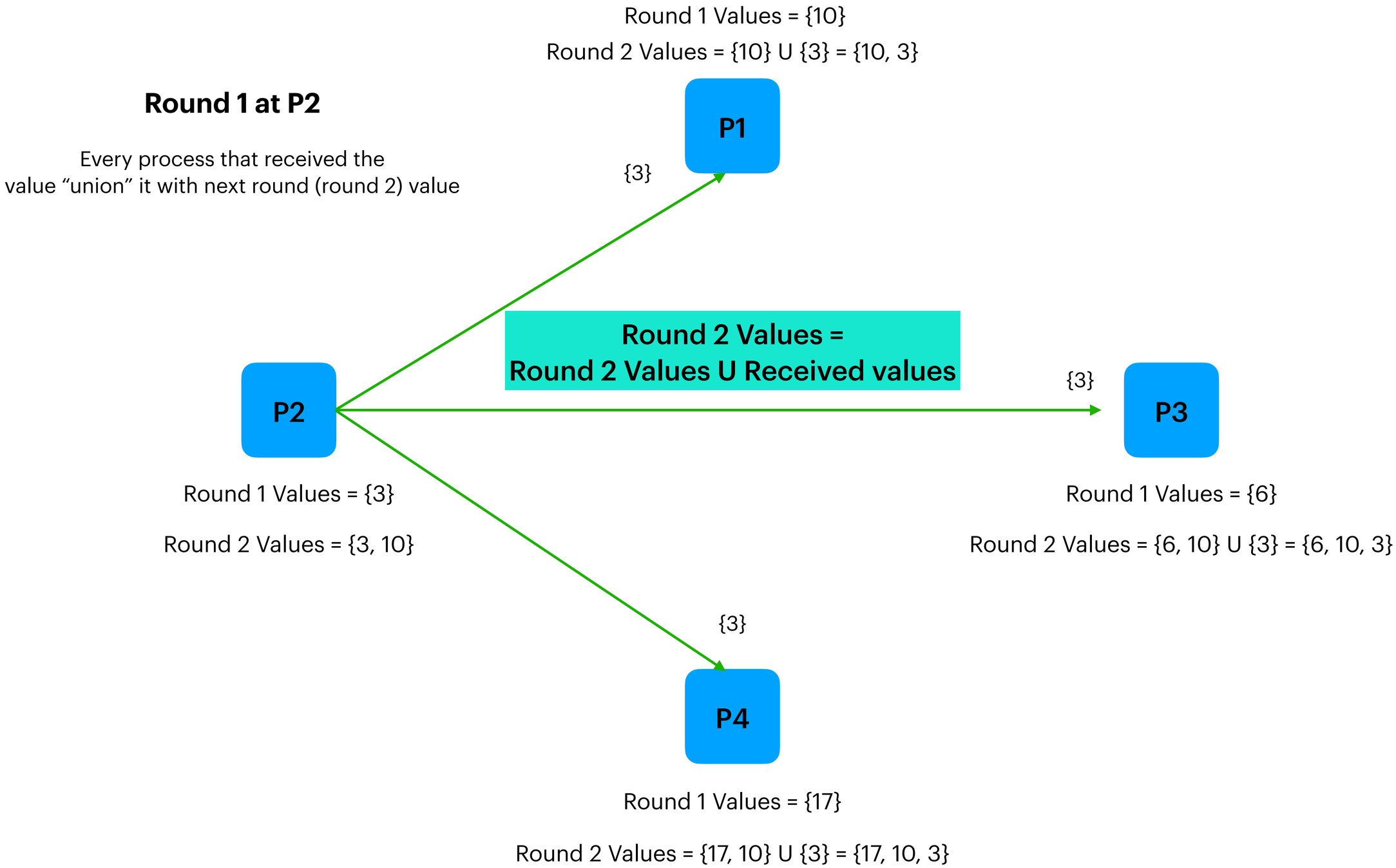
Round 2 Values = {6, 10}

P4

Round 1 Values = {17}

Round 2 Values = {17, 10}


Round 1 Values = {10}
Round 2 Values = {10}
Round 1 at P2
P2 send the new value from previous round (round 0) to every process
P1
{3}
P2
{3}
P3
Round 1 Values = {3}
Round 2 Values = {3, 10}
{3}
Round 1 Values = {6}
Round 2 Values = {6, 10}
Round 1 Values - Round 0 Values = Values that will be sent
{3} - {} = {3}
P4
Round 1 Values = {17}
Round 2 Values = {17, 10}


Round 1 Values = {10}
Round 2 Values = {10} U {3} = {10, 3}
Round 1 at P2
Every process that received the value “union” it with next round (round 2) value
P1
{3}
Round 2 Values = Round 2 Values U Received values
P2
{3}
P3
Round 1 Values = {3}
Round 2 Values = {3, 10}
Round 1 Values = {6}
Round 2 Values = {6, 10} U {3} = {6, 10, 3}
{3}
P4
Round 1 Values = {17}
Round 2 Values = {17, 10} U {3} = {17, 10, 3}

Ending of Round 1 at P2

Round 1 Values = {10}

Round 2 Values = {10, 3}

P1

P2

Round 1 Values = {3}

Round 2 Values = {3, 10}

P3

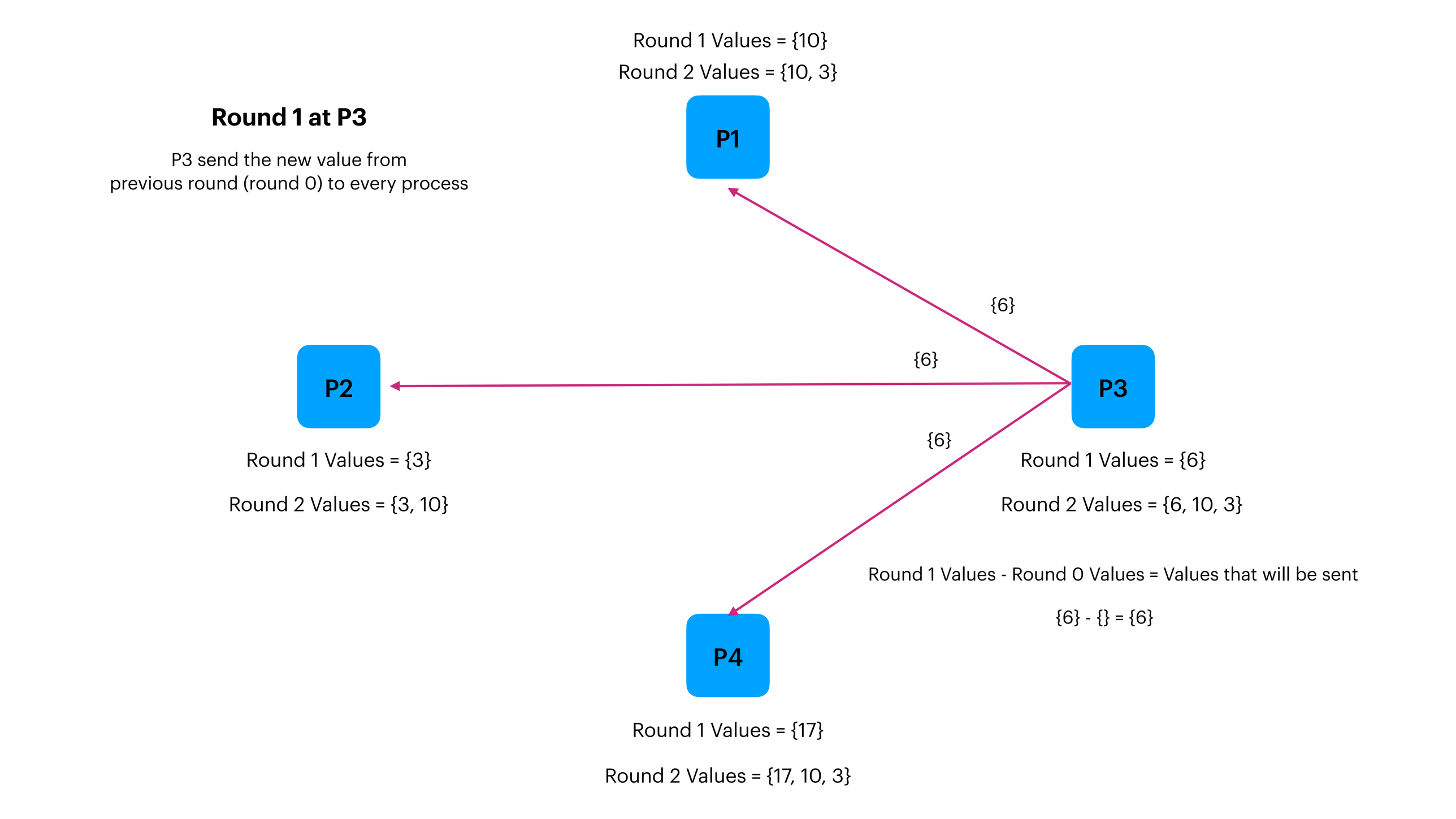
Round 1 Values = {6}

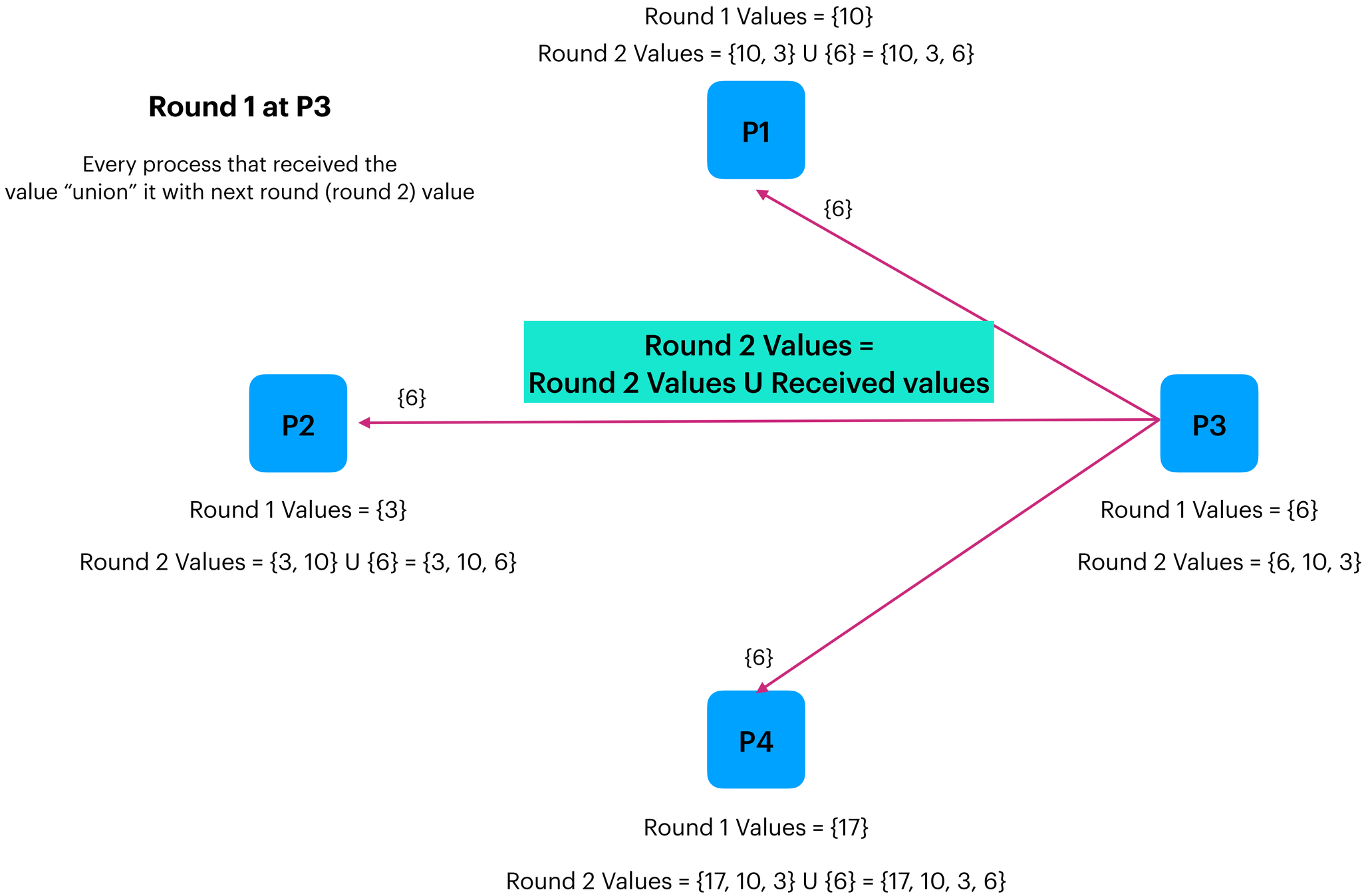
Round 2 Values = {6, 10, 3}

P4

Round 1 Values = {17}

Round 2 Values = {17, 10, 3}


Round 1 Values = {10}
Round 2 Values = {10, 3}
Round 1 at P3
P3 send the new value from previous round (round 0) to every process
P1
{6}
{6}
P2
P3
{6}
Round 1 Values = {3}
Round 2 Values = {3, 10}
Round 1 Values = {6}
Round 2 Values = {6, 10, 3}
Round 1 Values - Round 0 Values = Values that will be sent
{6} - {} = {6}
P4
Round 1 Values = {17}
Round 2 Values = {17, 10, 3}


Round 1 Values = {10}
Round 2 Values = {10, 3} U {6} = {10, 3, 6}
Round 1 at P3
Every process that received the value “union” it with next round (round 2) value
P1
{6}
Round 2 Values = Round 2 Values U Received values
P2
{6}
P3
Round 1 Values = {3}
Round 2 Values = {3, 10} U {6} = {3, 10, 6}
Round 1 Values = {6}
Round 2 Values = {6, 10, 3}
{6}
P4
Round 1 Values = {17}
Round 2 Values = {17, 10, 3} U {6} = {17, 10, 3, 6}

Ending of Round 1 at P3

Round 1 Values = {10}

Round 2 Values = {10, 3, 6}

P1

P2

Round 1 Values = {3}

Round 2 Values = {3, 10, 6}

P3

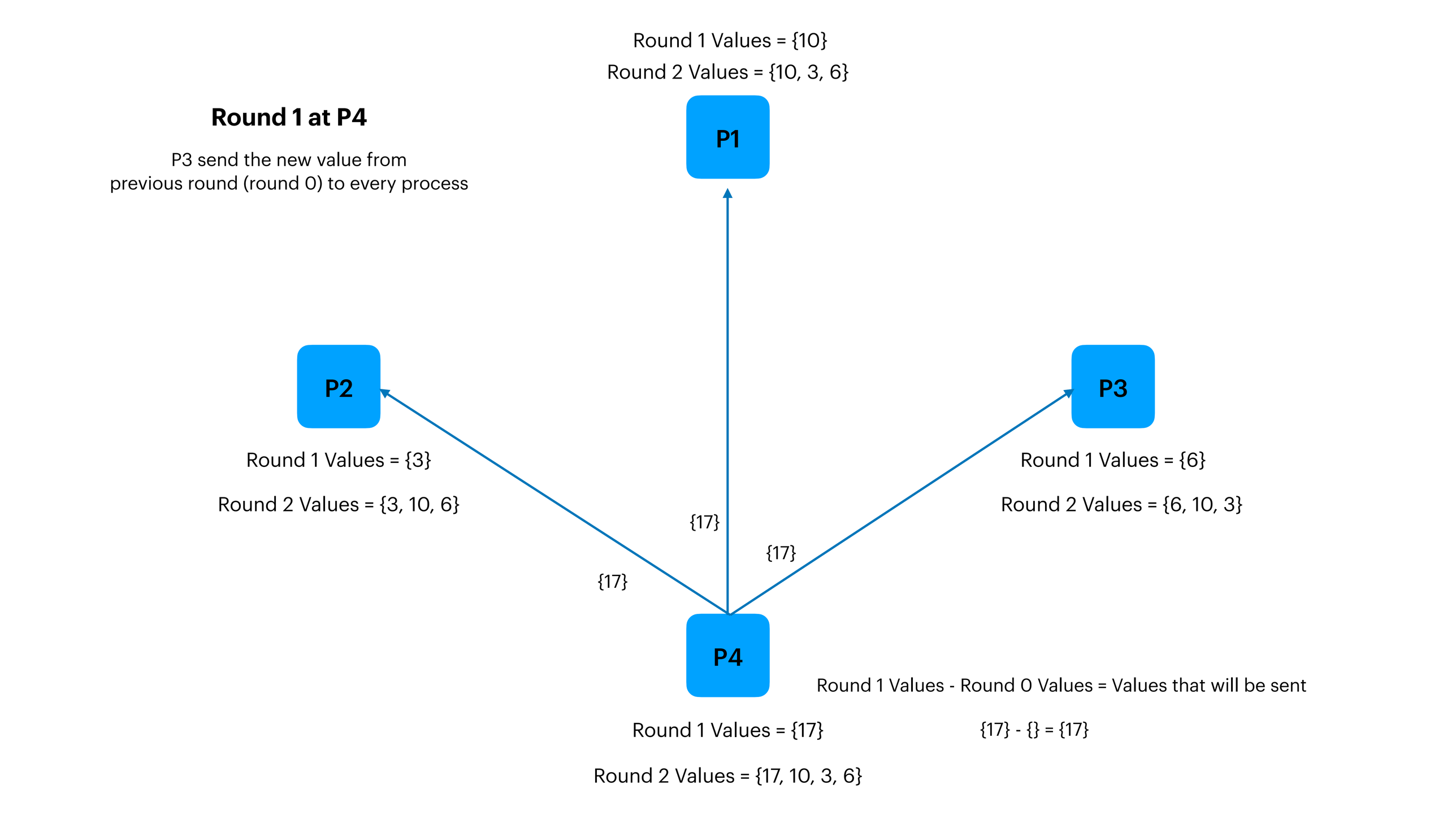
Round 1 Values = {6}

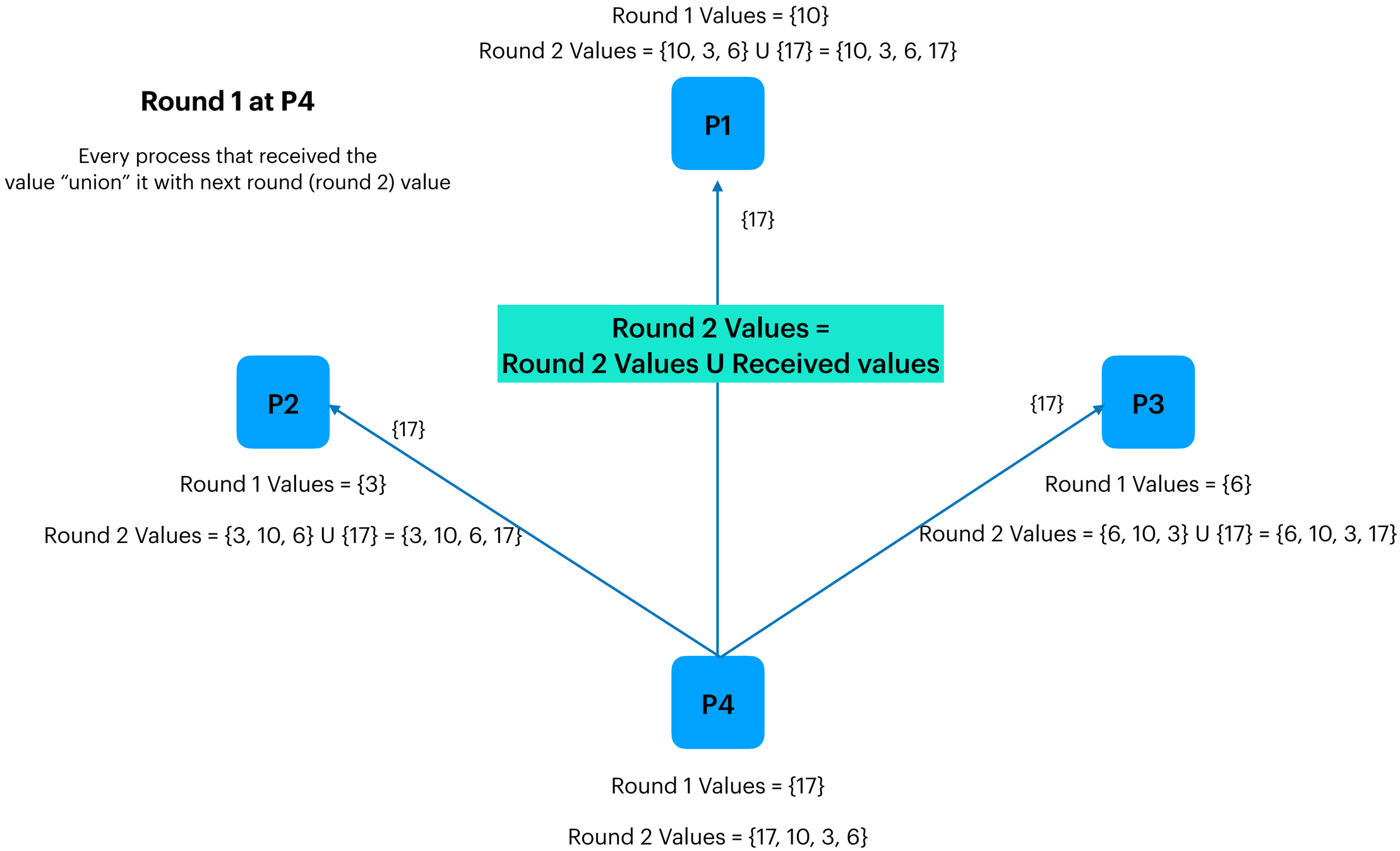
Round 2 Values = {6, 10, 3}

P4

Round 1 Values = {17}

Round 2 Values = {17, 10, 3, 6}


Round 1 Values = {10}
Round 2 Values = {10, 3, 6}
Round 1 at P4
P3 send the new value from previous round (round 0) to every process
P1
P2
P3
P4
Round 1 Values = {3}
Round 2 Values = {3, 10, 6}
Round 1 Values = {6}
Round 2 Values = {6, 10, 3}
{17}
{17}
{17}
Round 1 Values - Round 0 Values = Values that will be sent
Round 1 Values = {17}
{17} - {} = {17}
Round 2 Values = {17, 10, 3, 6}


Round 1 Values = {10}
Round 2 Values = {10, 3, 6} U {17} = {10, 3, 6, 17}
Round 1 at P4
Every process that received the value “union” it with next round (round 2) value
P1
{17}
Round 2 Values = Round 2 Values U Received values
P2
{17}
{17}
P3
Round 1 Values = {3}
Round 2 Values = {3, 10, 6} U {17} = {3, 10, 6, 17}
Round 1 Values = {6}
Round 2 Values = {6, 10, 3} U {17} = {6, 10, 3, 17}
P4
Round 1 Values = {17}
Round 2 Values = {17, 10, 3, 6}

Ending of Round 1 at P4

Round 1 Values = {10}

Round 2 Values = {10, 3, 6, 17}

P1

P2

Round 1 Values = {3}

Round 2 Values = {3, 10, 6, 17}

{17}

P3

Round 1 Values = {6}

Round 2 Values = {6, 10, 3, 17}

P4

Round 1 Values = {17}

Round 2 Values = {17, 10, 3, 6}

Round 1 Values = {10}

Round 2 Values = **{10, 3, 6, 17}**

Finish the f+1 round

P1

P2

All the processes got the same sets of values

{17}

P3

Round 1 Values = {3}

Round 2 Values = **{3, 10, 6, 17}**

Round 1 Values = {6}

Round 2 Values = **{6, 10, 3, 17}**

P4

Round 1 Values = {17}

Round 2 Values = **{17, 10, 3, 6}**

Finish the f+1 round

Round 1 Values = {10}

Round 2 Values = {10, 3, 6, 17}

Decided value = 3

P1

P2

Round 1 Values = {3}

Round 2 Values = {3, 10, 6, 17}

Decided value = 3

Get the min of next round value to be decided value

The consensus is reached!!

{17}

P3

Round 1 Values = {6}

Round 2 Values = {6, 10, 3, 17}

Decided value = 3

P4

Round 1 Values = {17}

Round 2 Values = {17, 10, 3, 6}

Decided value = 3

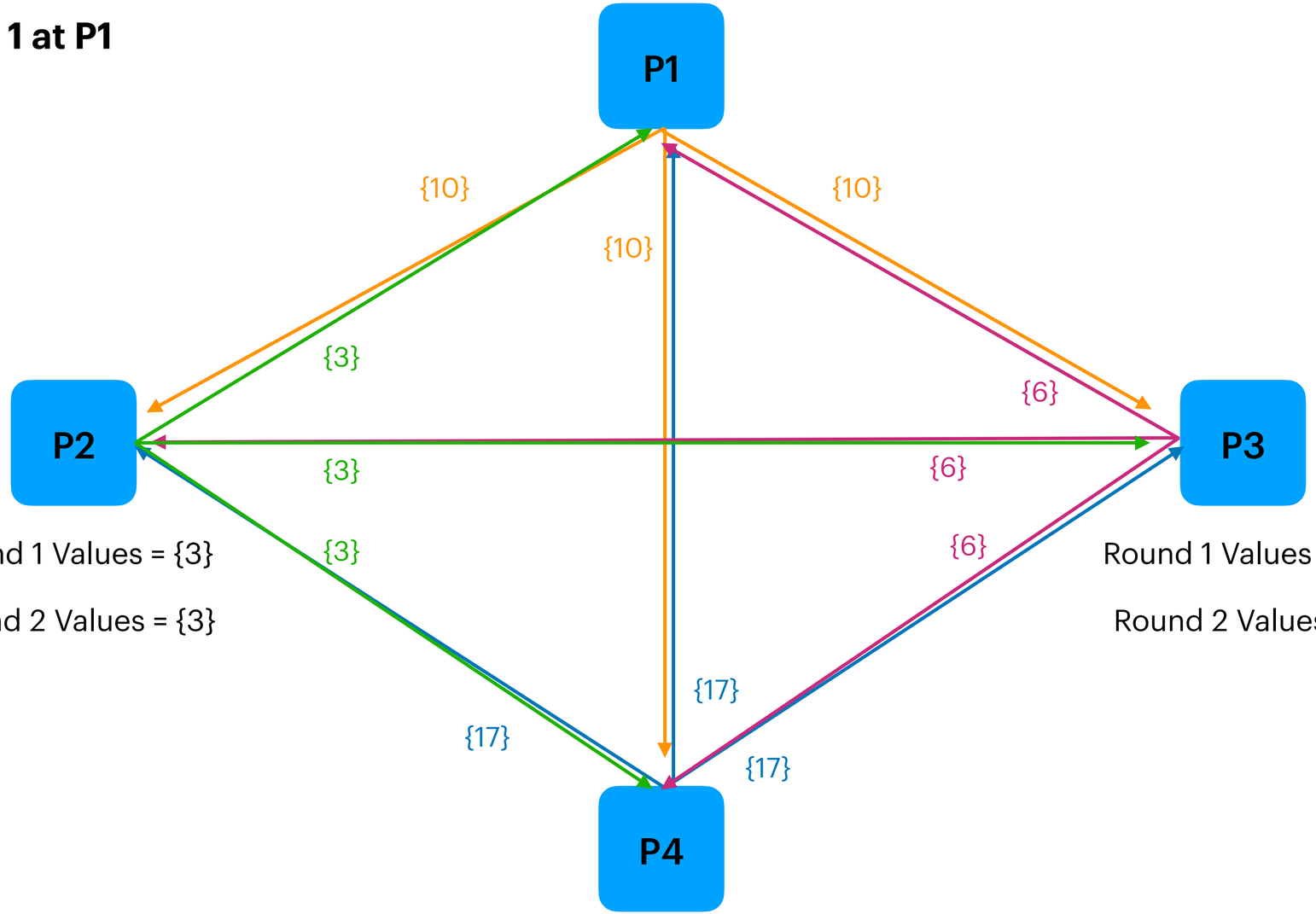
But everything in the round happened in the same time

Round 1 Values = {10}

Round 2 Values = {10}

{10} - {} = {10}

Round 1 at P1



Round 1 Values = {3}

Round 2 Values = {3}

Round 1 Values = {6}

Round 2 Values = {6}

Round 1 Values = {17}

Round 2 Values = {17}

Round 1 Values = {10}

Round 2 Values = {10, 3, 6, 17}

Decided value = 3

Round 1 at P1

P1

P2

Round 1 Values = {3}

Round 2 Values = {3, 10, 6, 17}

Decided value = 3

{6}

P3

Round 1 Values = {6}

Round 2 Values = {6, 10, 3, 17}

Decided value = 3

P4

Round 1 Values = {17}

Round 2 Values = {17, 10, 3, 6}

Decided value = 3

PAXOS

Sethanant Pipatpakorn

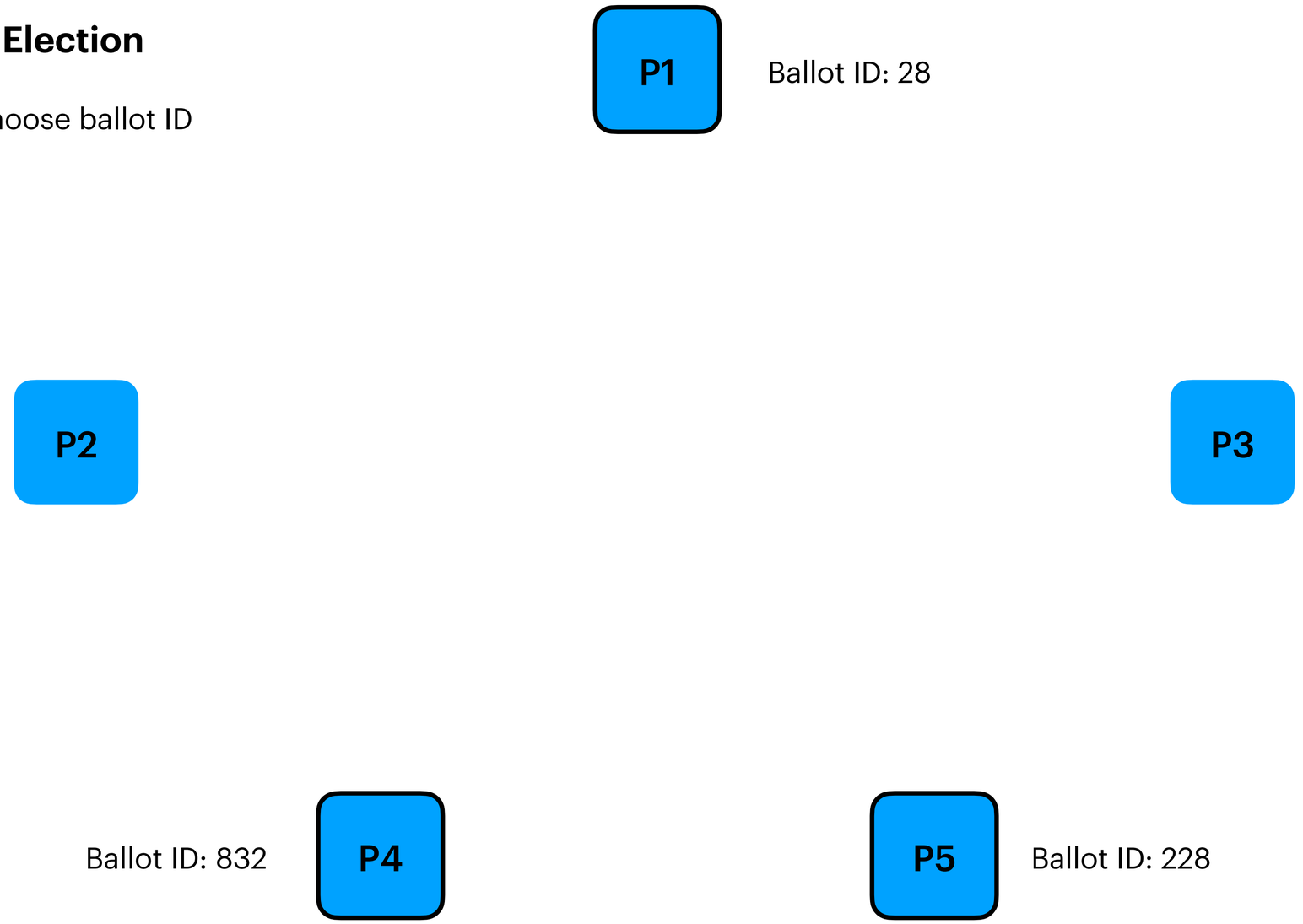
Quick overview

Paxos works in “rounds”. Each round has 3 phase

- **Phase 1: Election** -> Choose the leader
- **Phase 2: Bill** -> Leader propose the value
- **Phase 3: Law** -> Leader multicast the final value

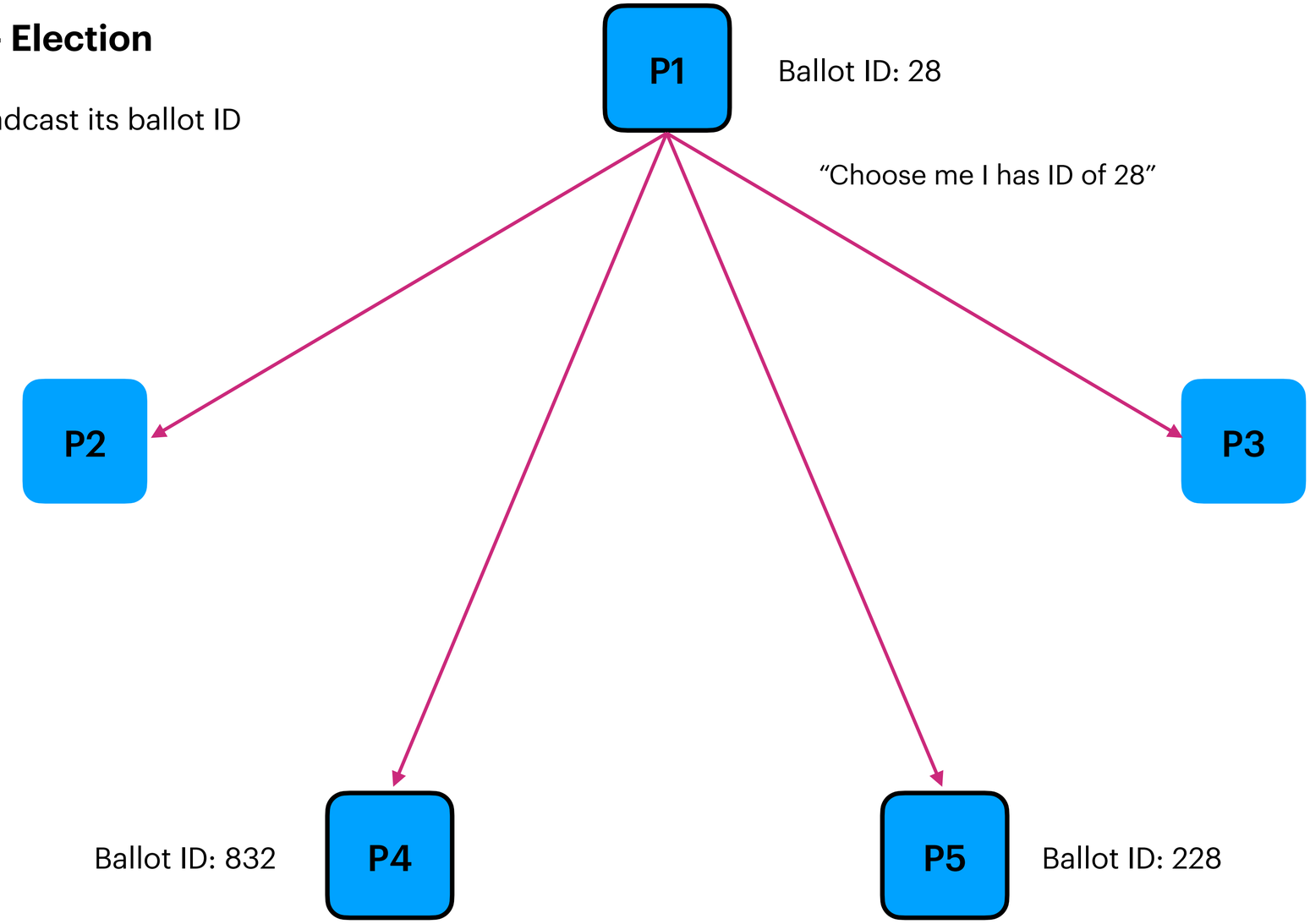
Phase 1 - Election

Potential leader choose ballot ID



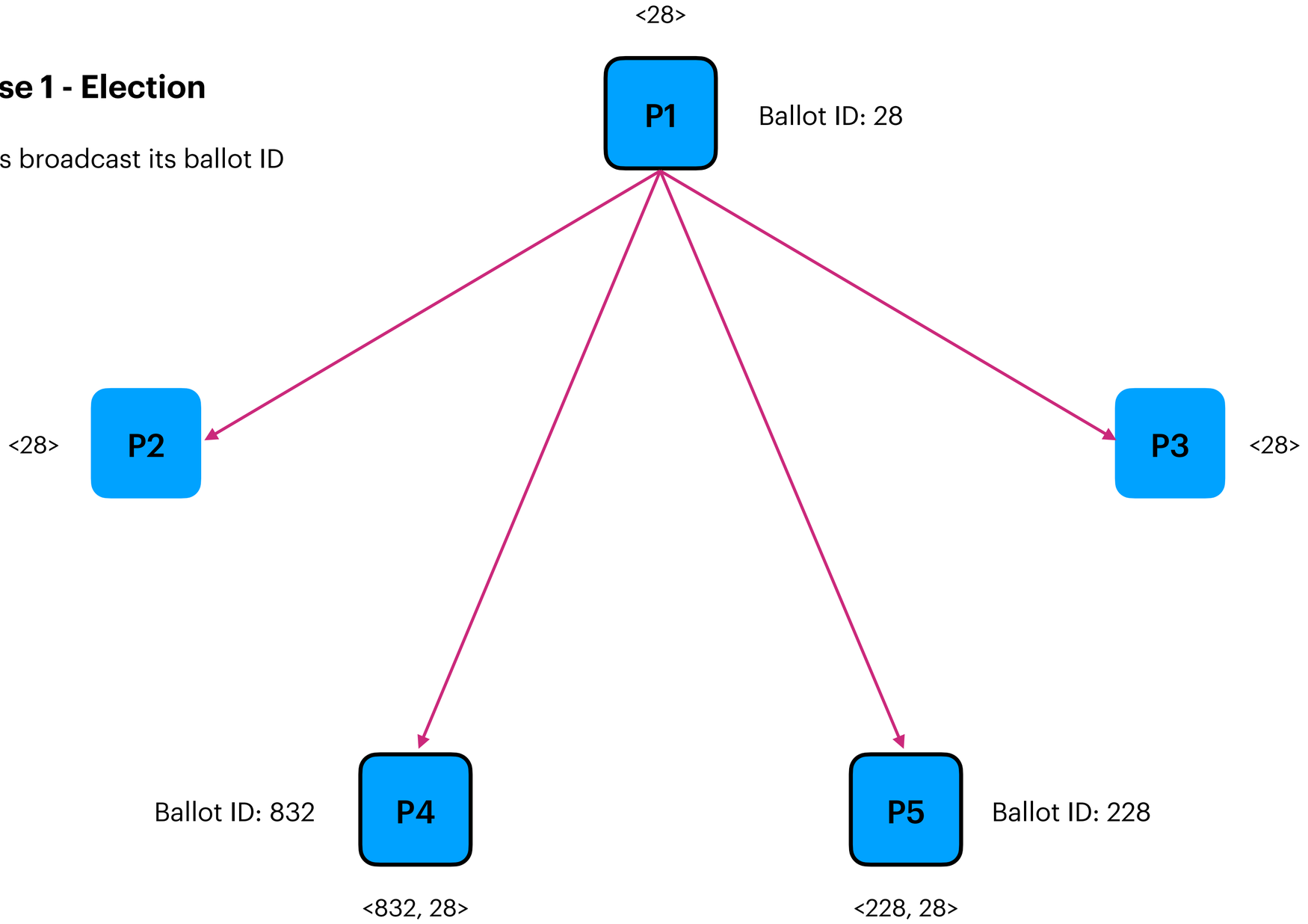
Phase 1 - Election

Potential leaders broadcast its ballot ID



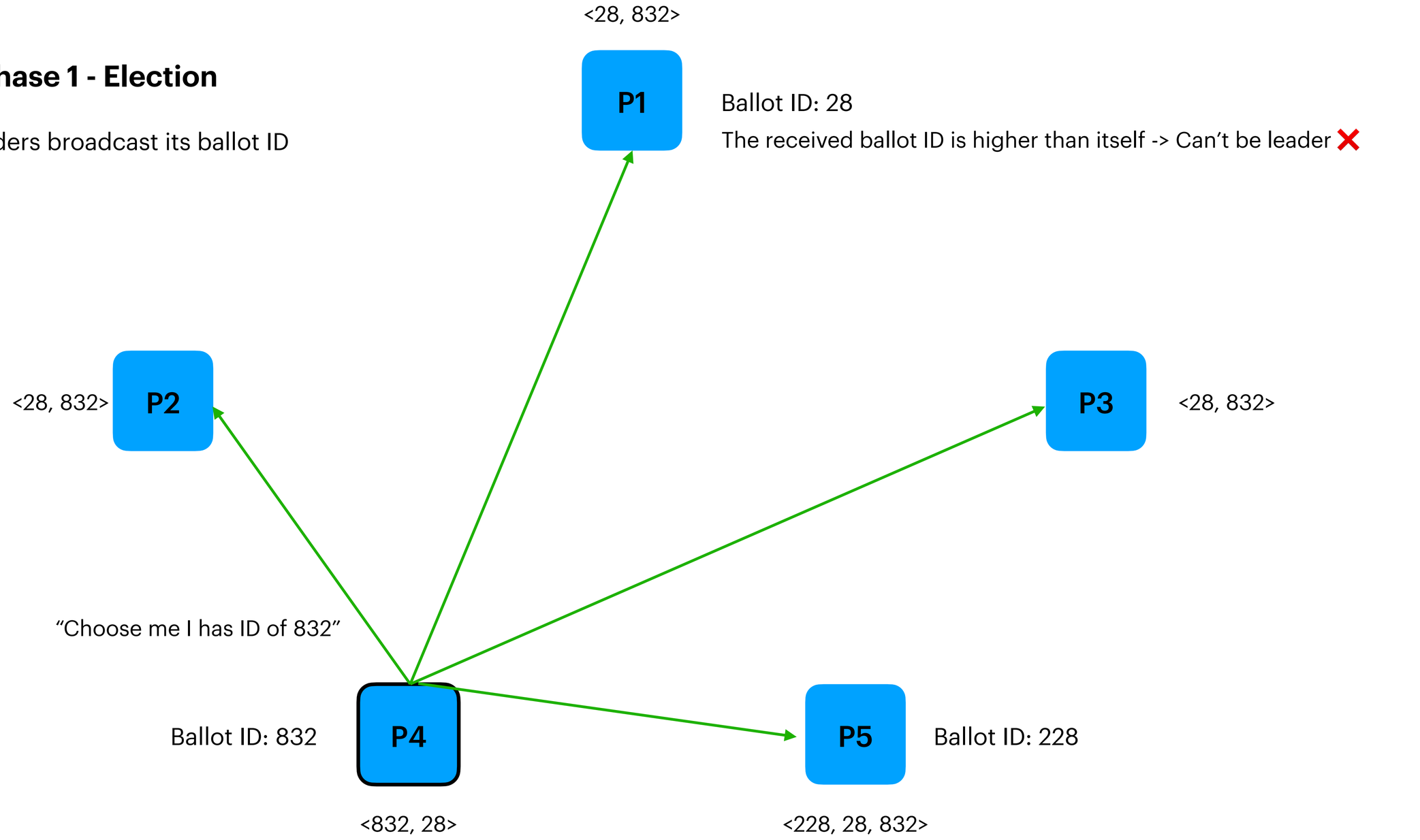
Phase 1 - Election

Potential leaders broadcast its ballot ID



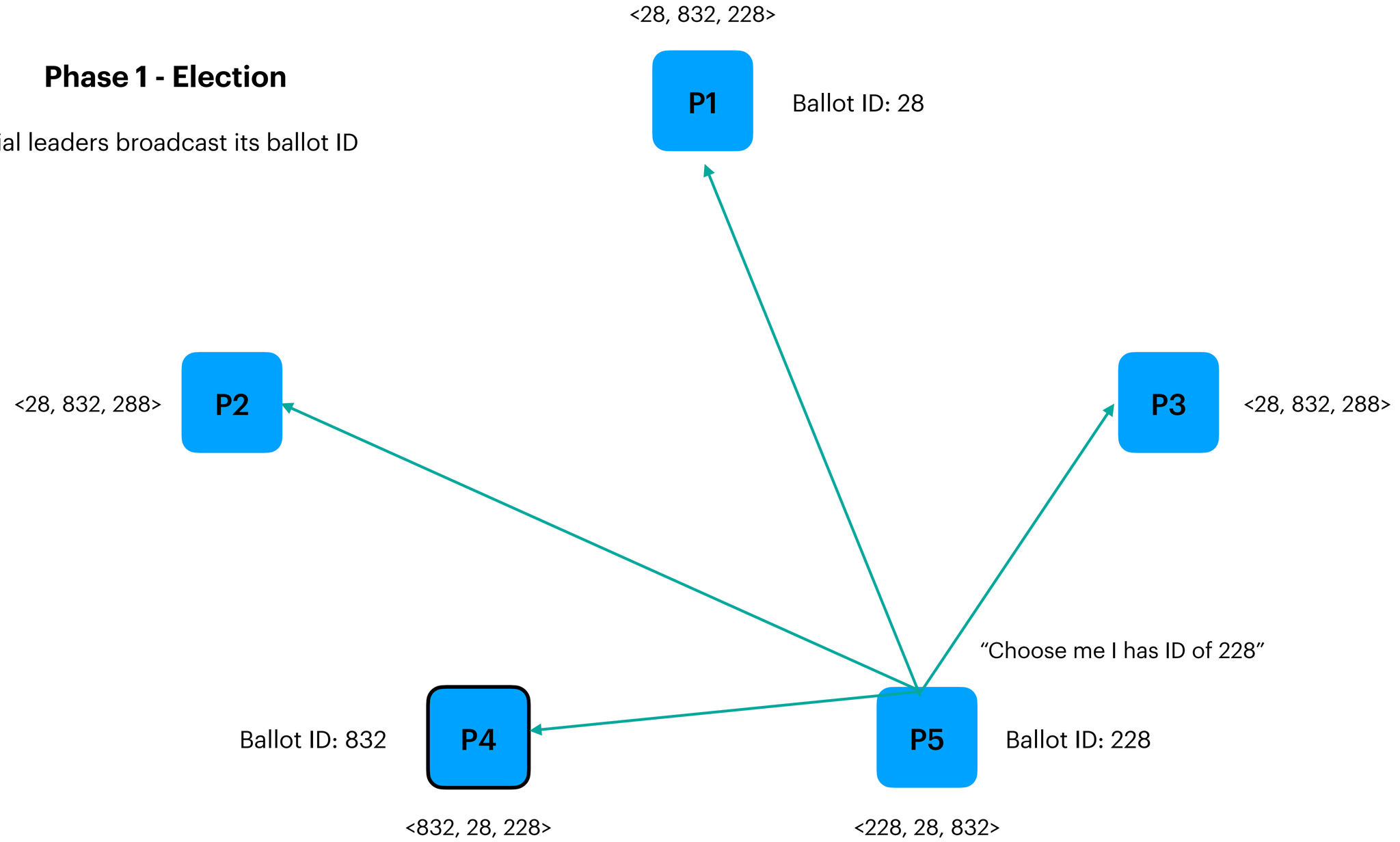
Phase 1 - Election

Potential leaders broadcast its ballot ID



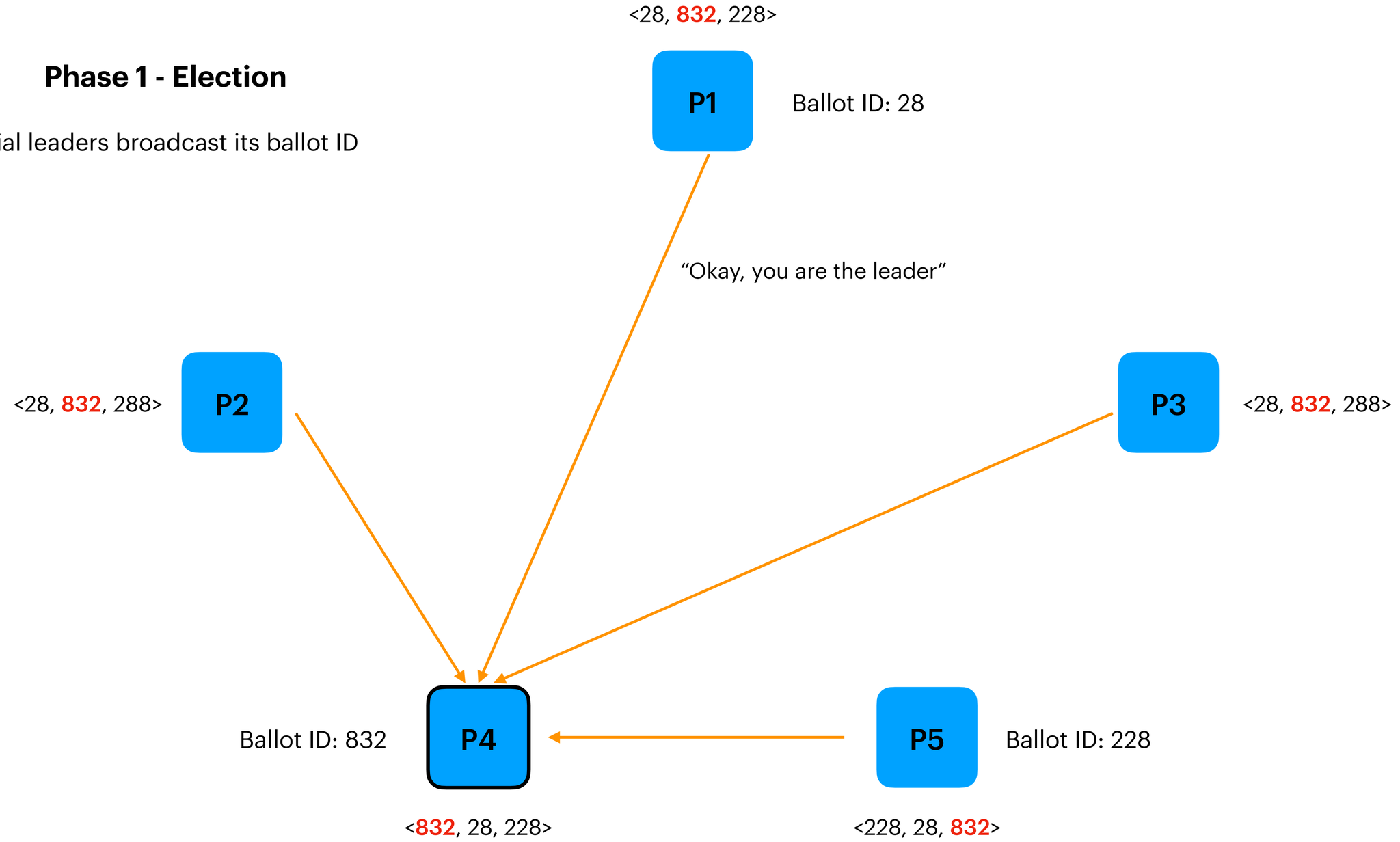
Phase 1 - Election

Potential leaders broadcast its ballot ID



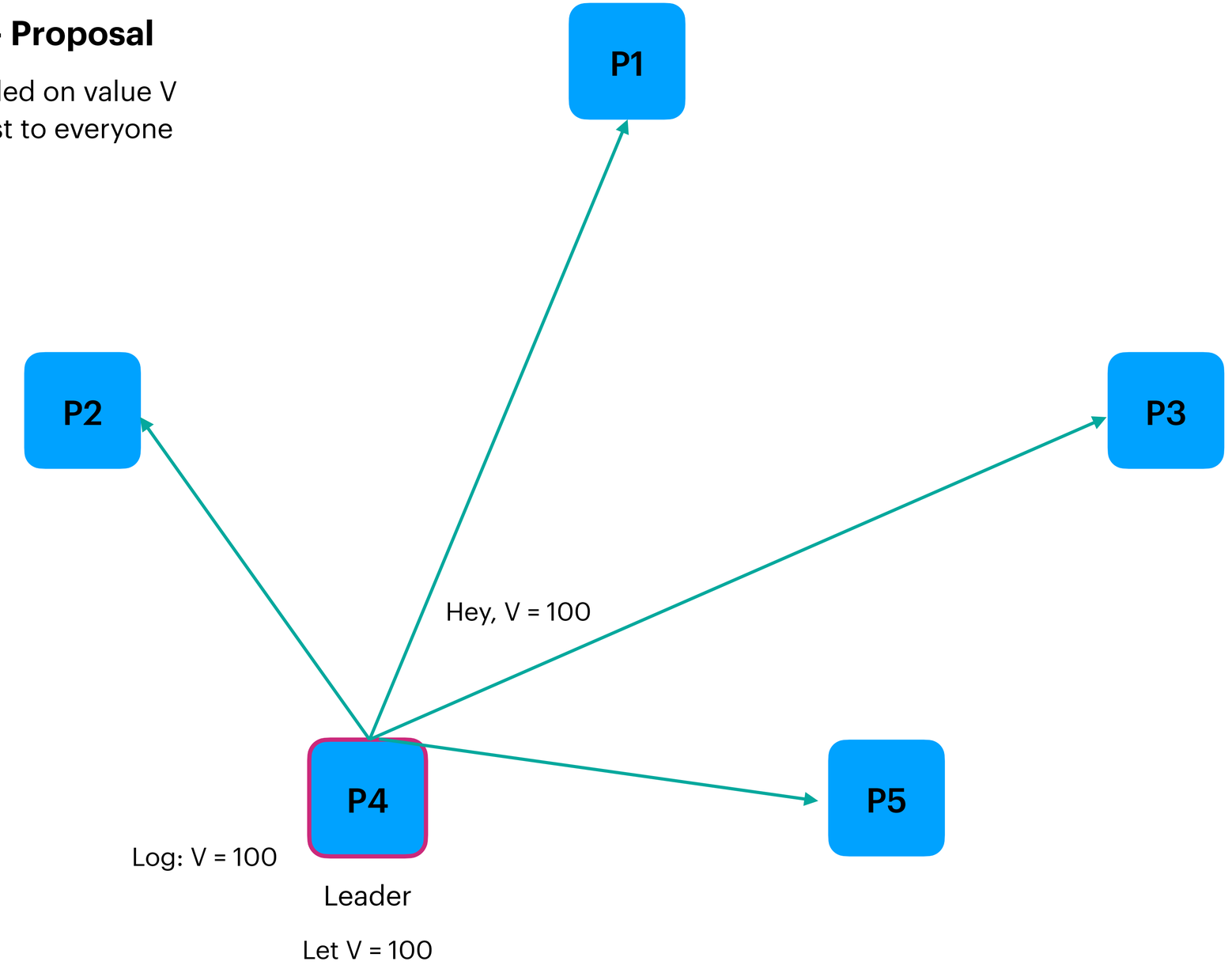
Phase 1 - Election

Potential leaders broadcast its ballot ID



Phase 2 - Proposal

Leader decided on value V
and multicast to everyone

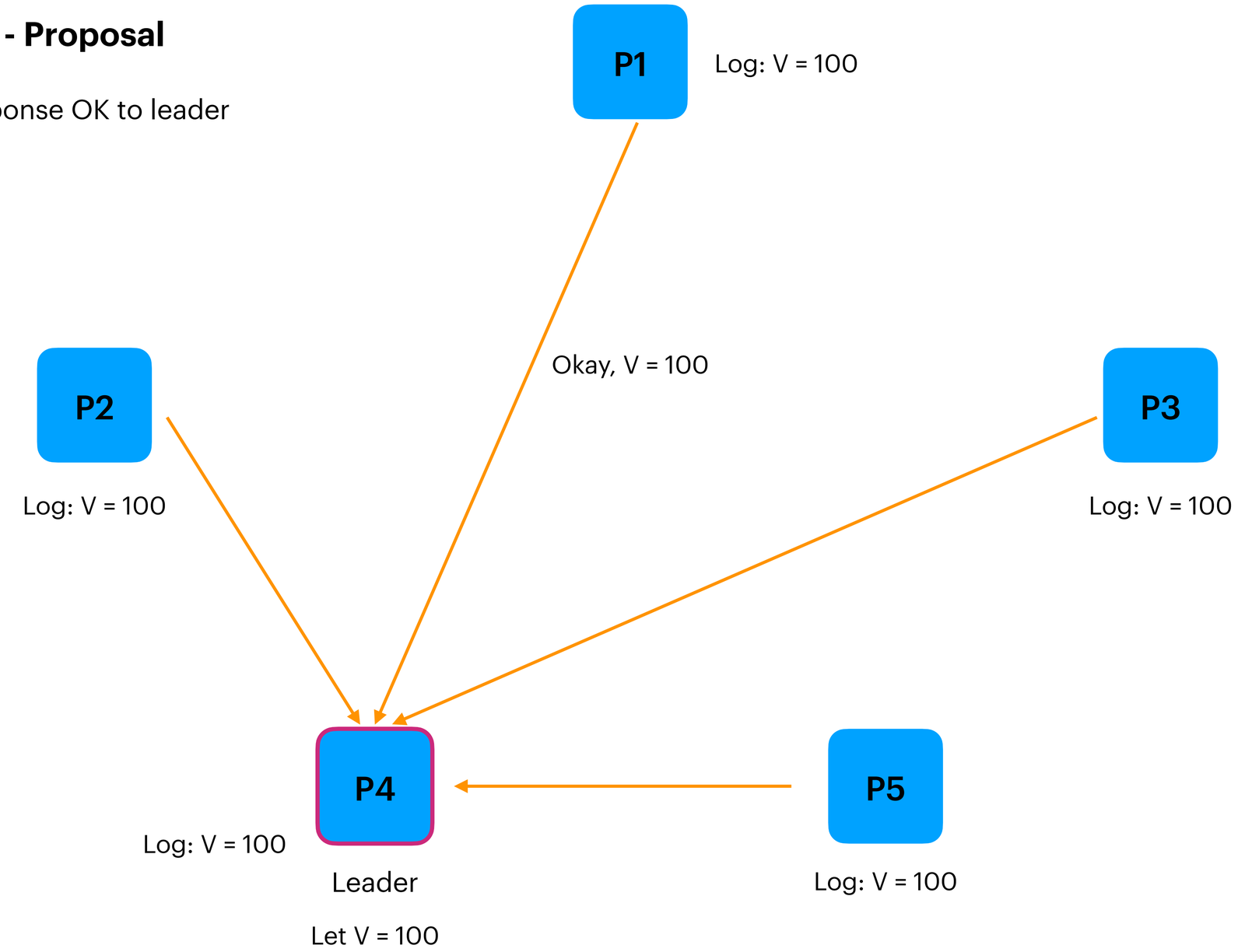


Phase 2 - Proposal

Recipients log the value to disk

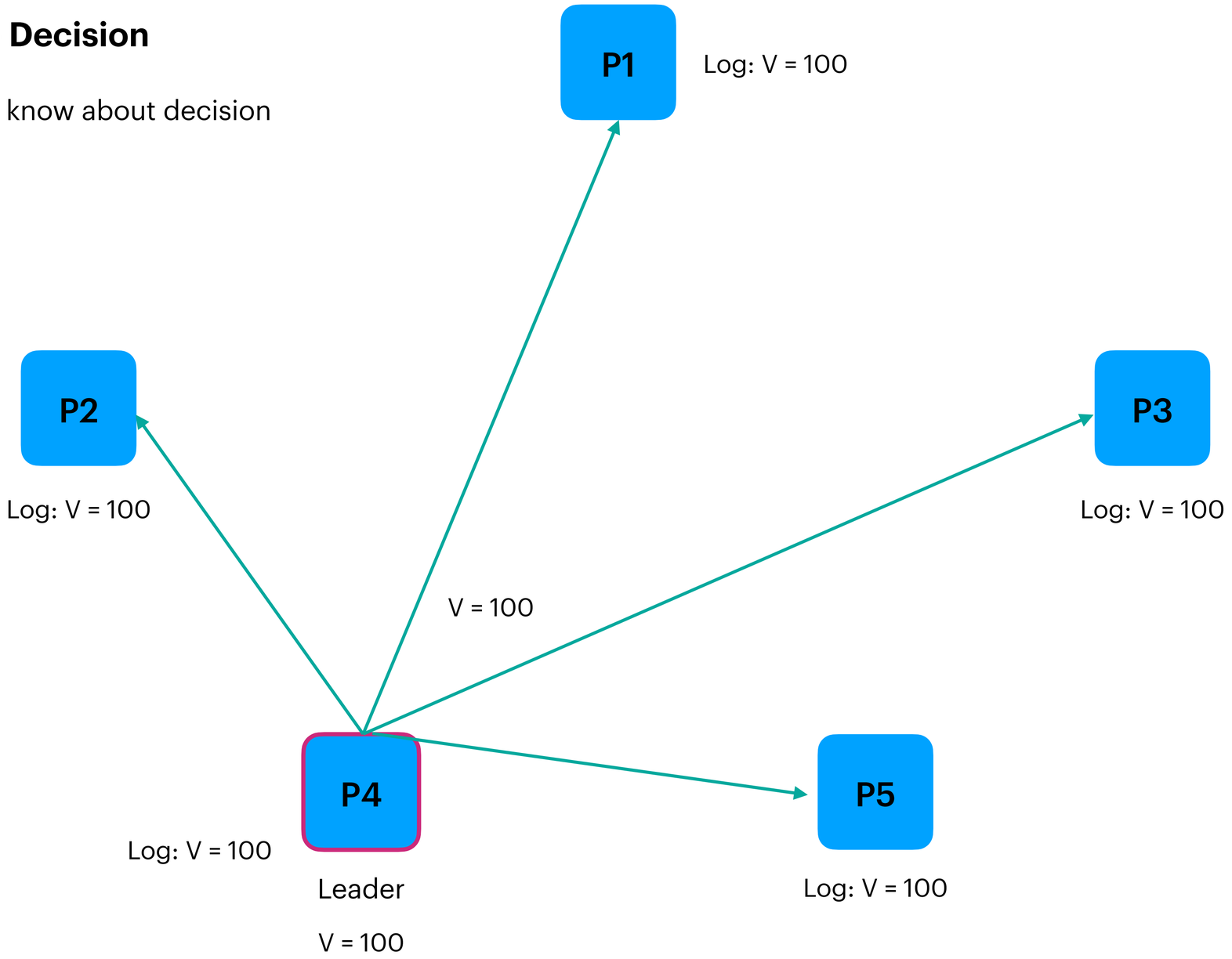
Phase 2 - Proposal

Recipients response OK to leader

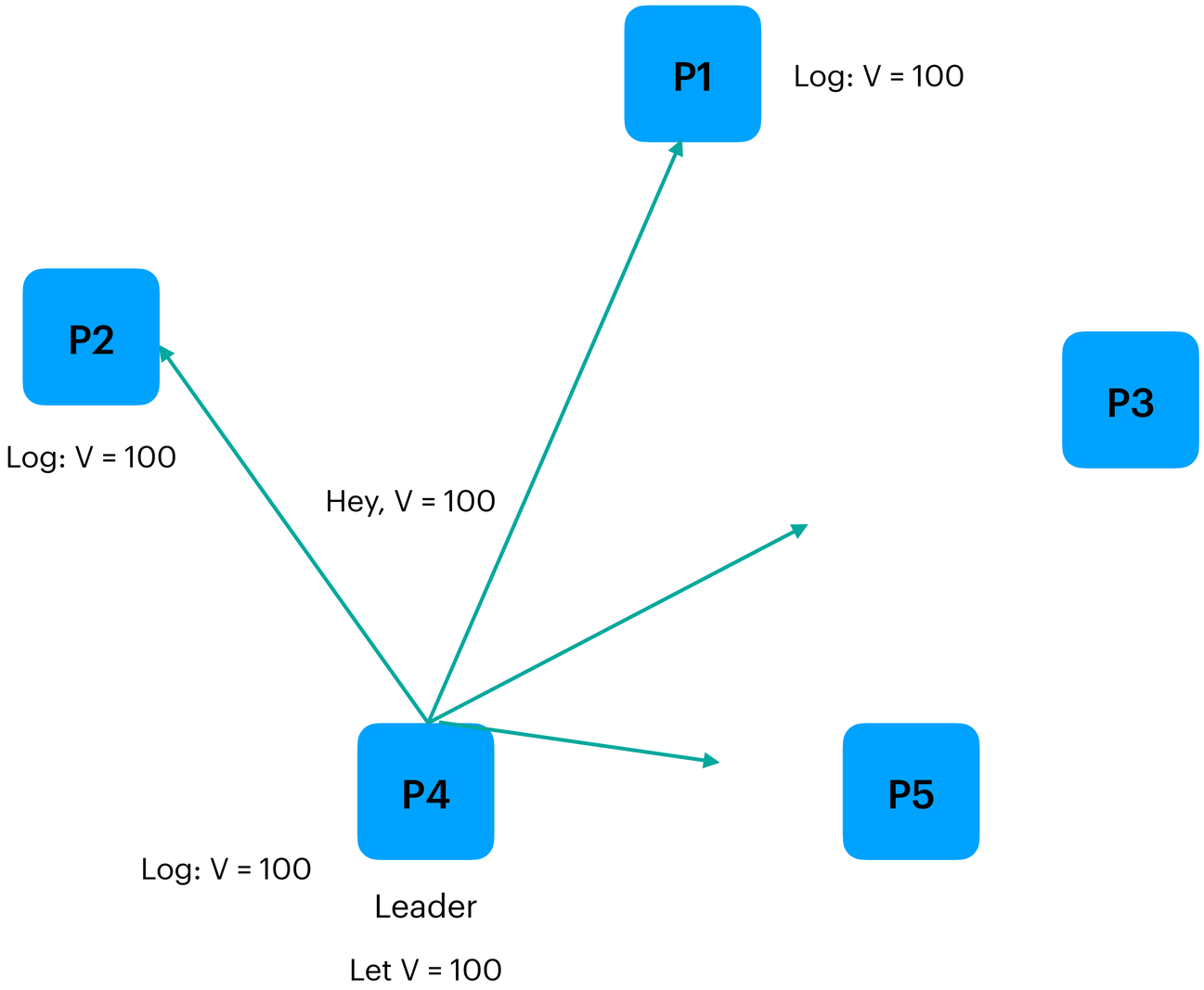


Phase 3 - Decision

Leader lets everyone know about decision



Point of No Return



When the majority has received and log proposed value to the log

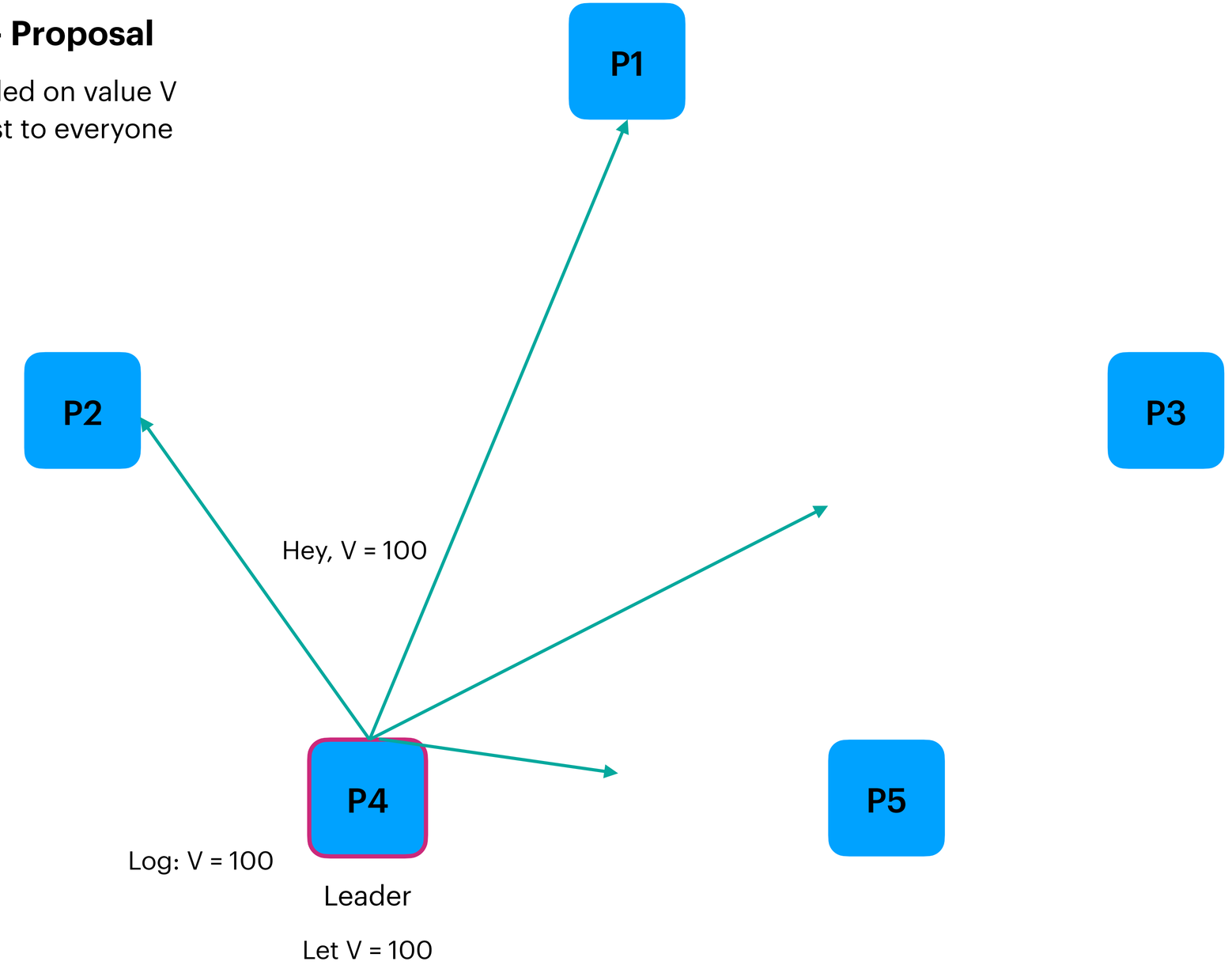
When it reach this point, the decided value stay.
It cannot be changed anymore

PAXOS with Failure

Sethanant Pipatpakorn

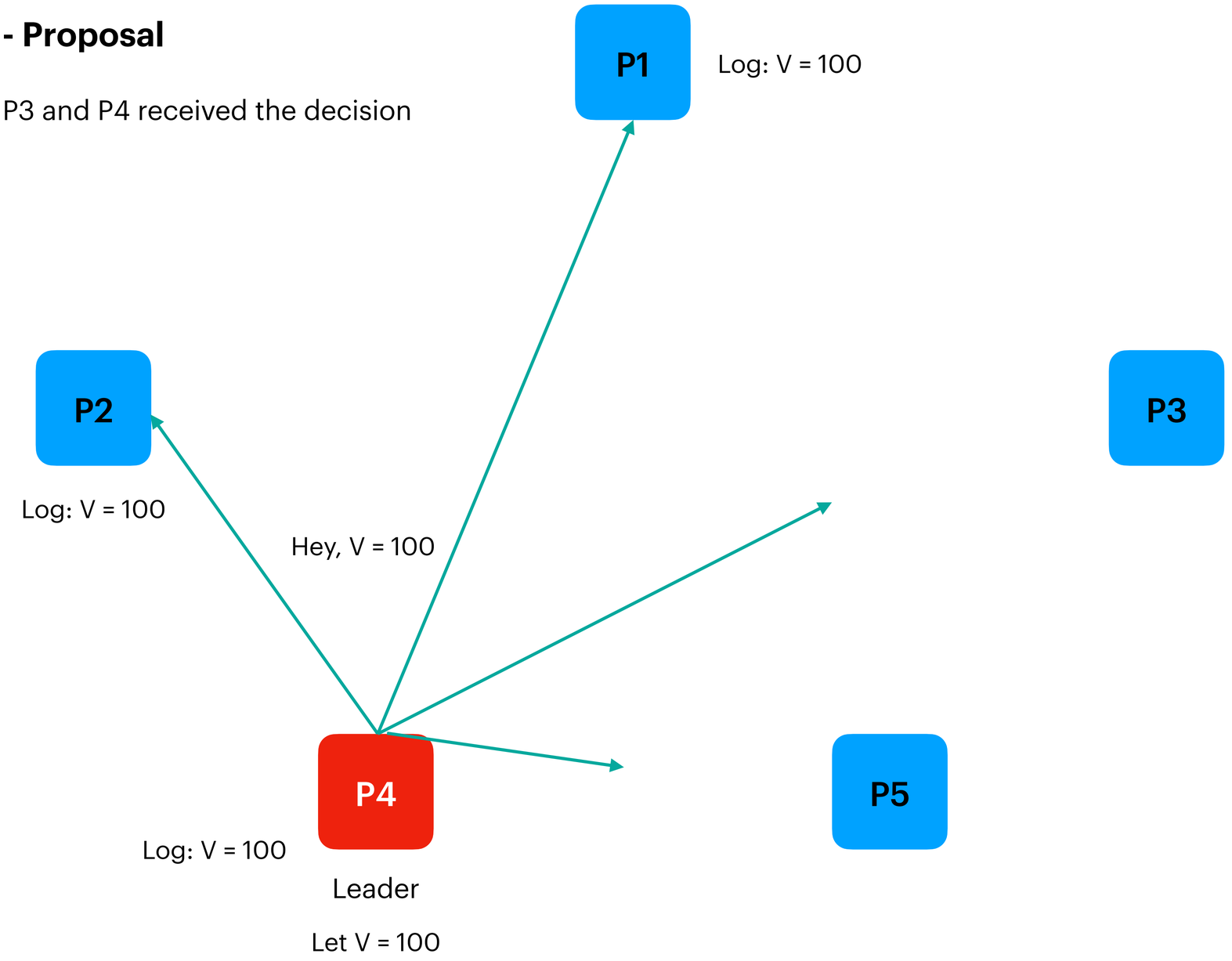
Phase 2 - Proposal

Leader decided on value V
and multicast to everyone

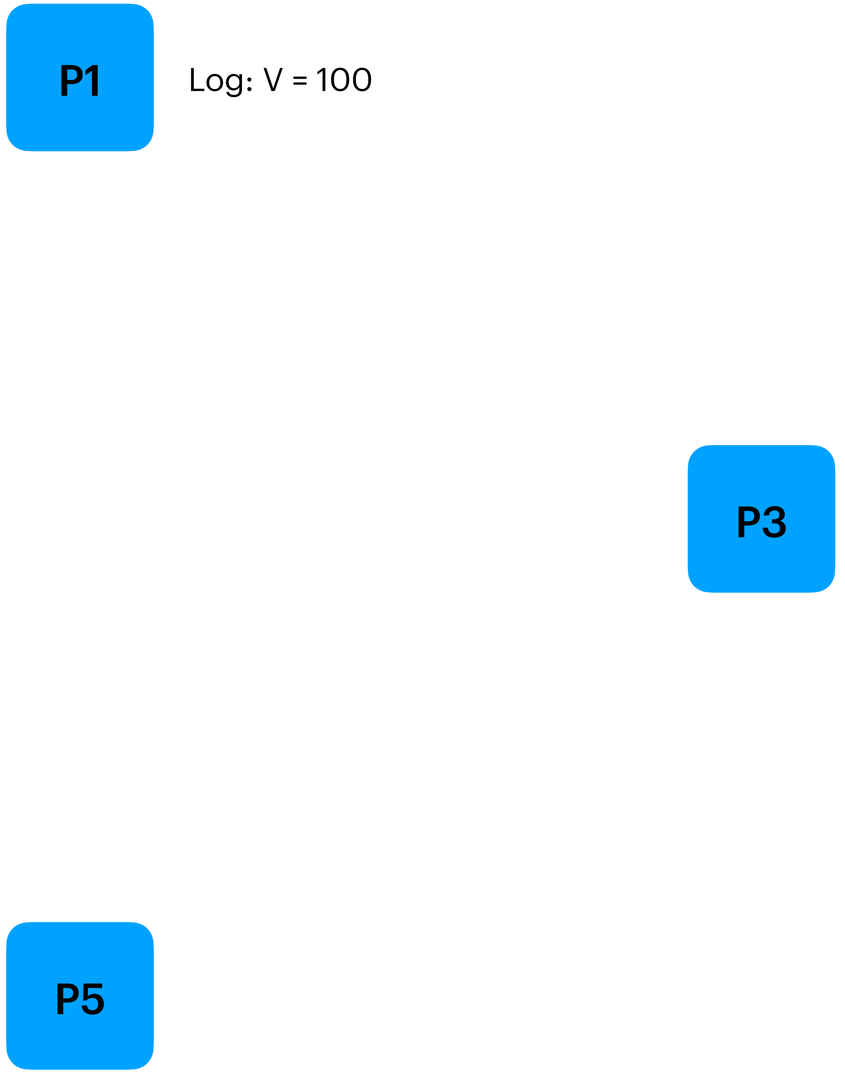


Phase 2 - Proposal

Leader is dead before P3 and P4 received the decision

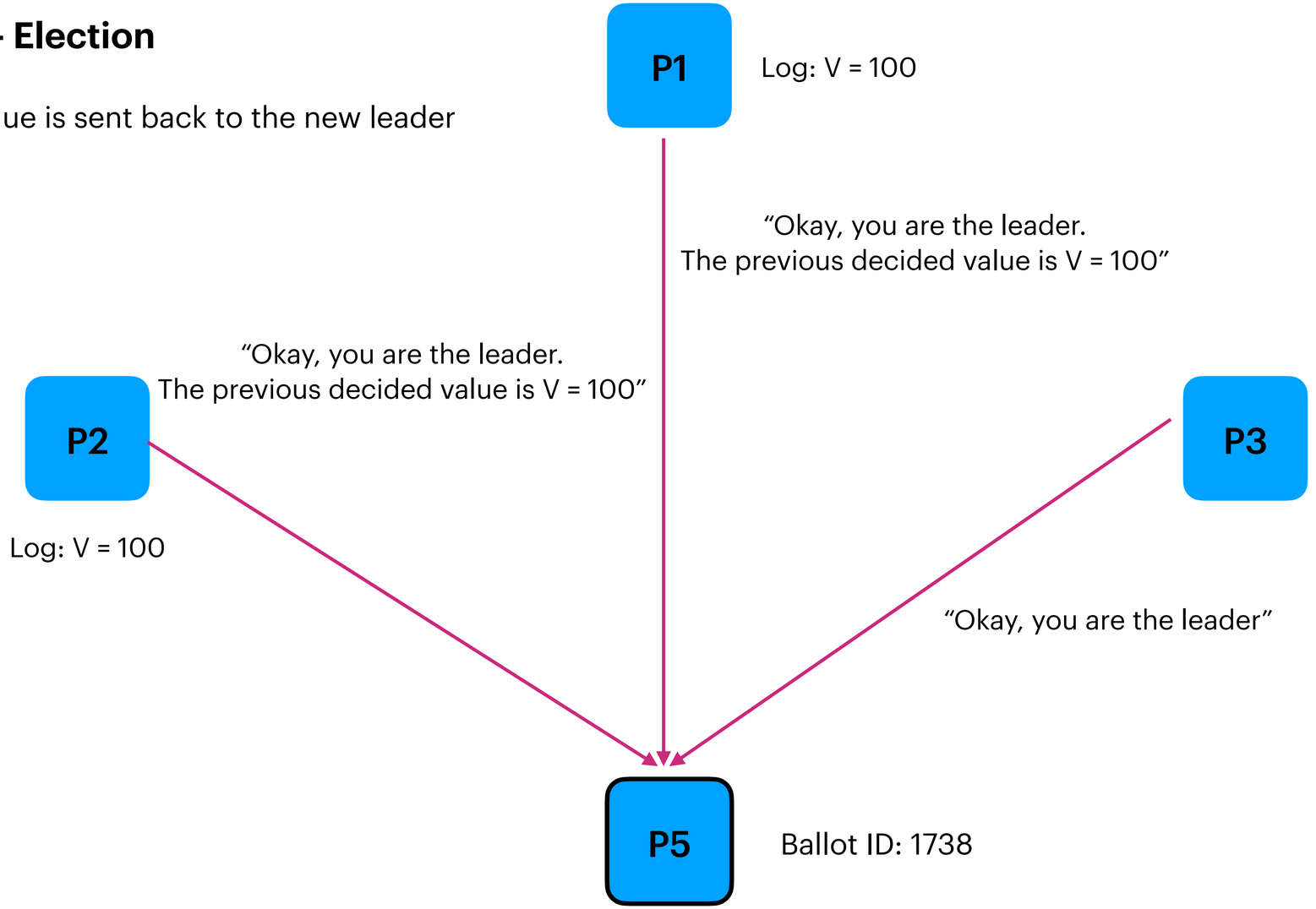


The process is stop and start again



Phase 1 - Election

The previous decided value is sent back to the new leader



Phase 2 - Proposal

The leader use the received value as decided value for this round

Log: V = 100

Log: V = 100

“They tells me that the decide value is V = 100. I will use that value”

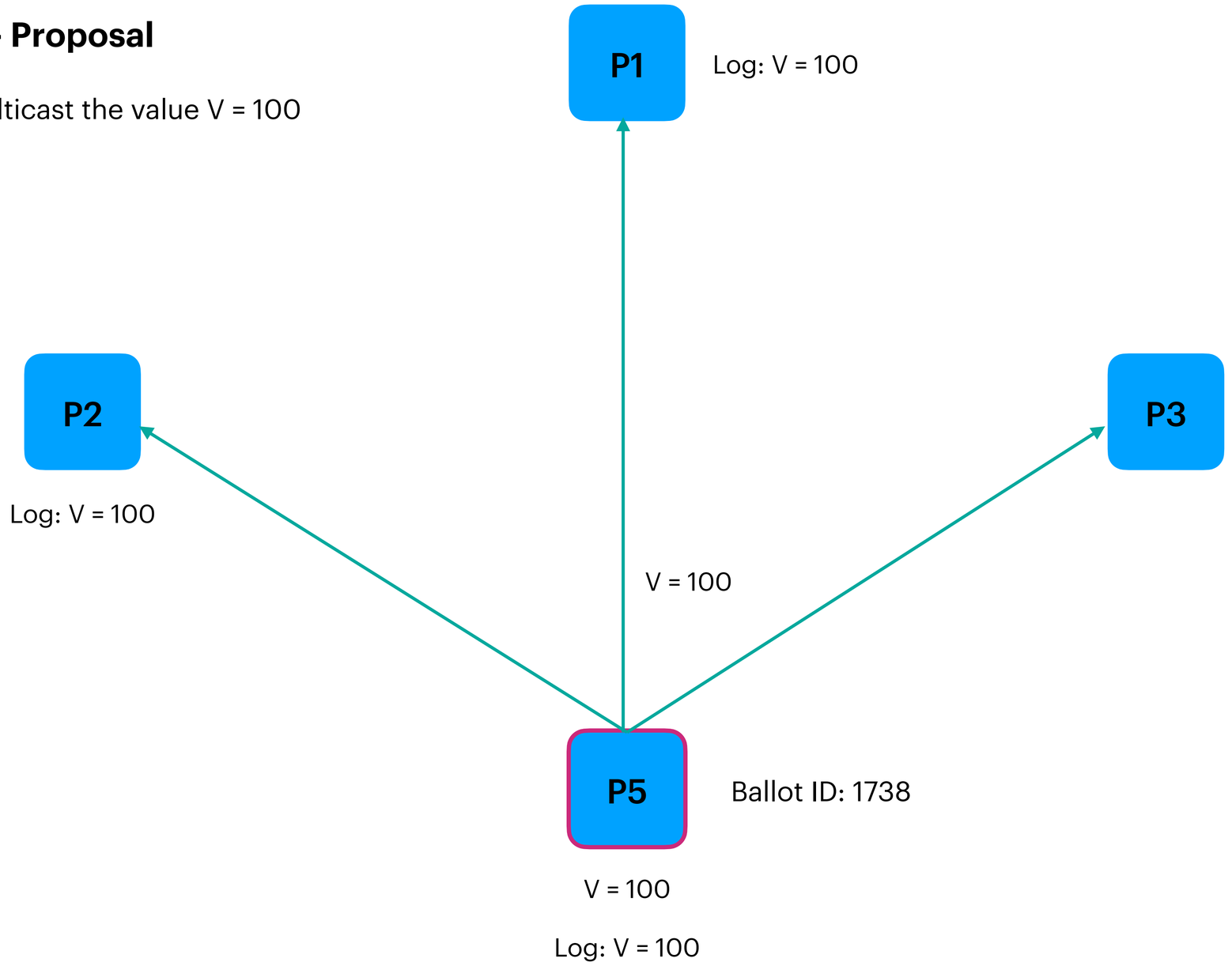
Ballot ID: 1738

V = 100

Log: V = 100

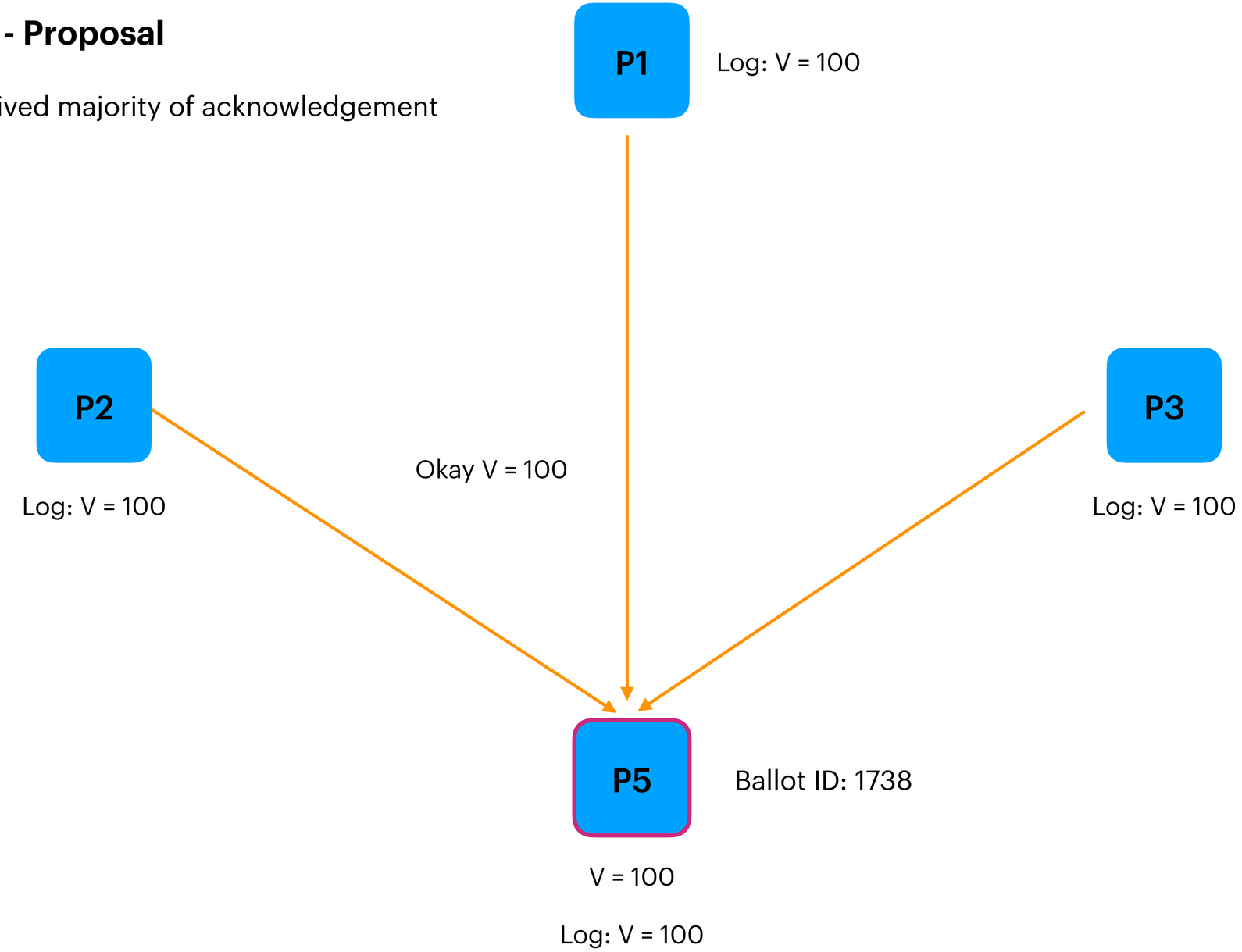
Phase 2 - Proposal

The leader multicast the value V = 100



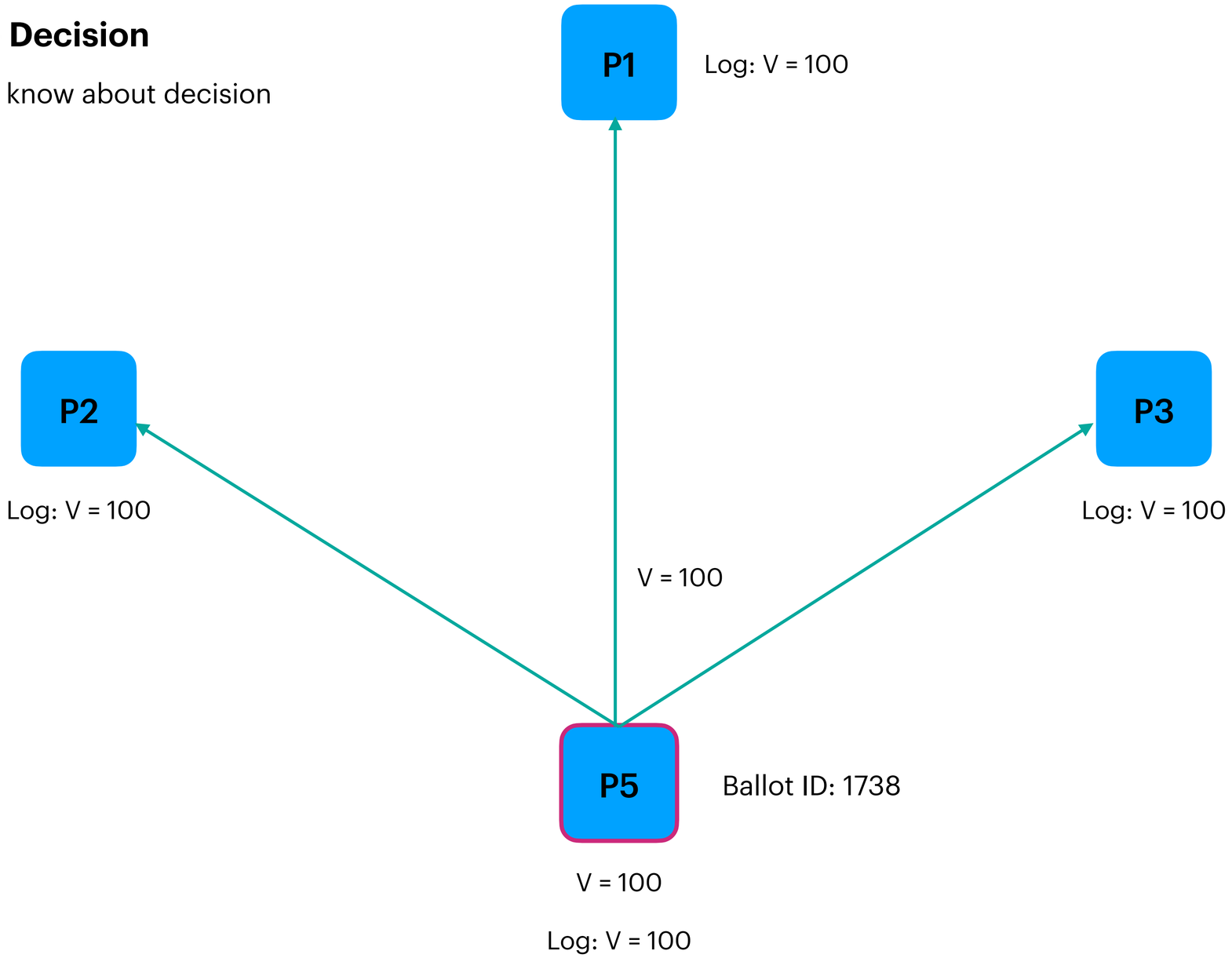
Phase 2 - Proposal

Leader wait until it received majority of acknowledgement



Phase 3 - Decision

Leader lets everyone know about decision

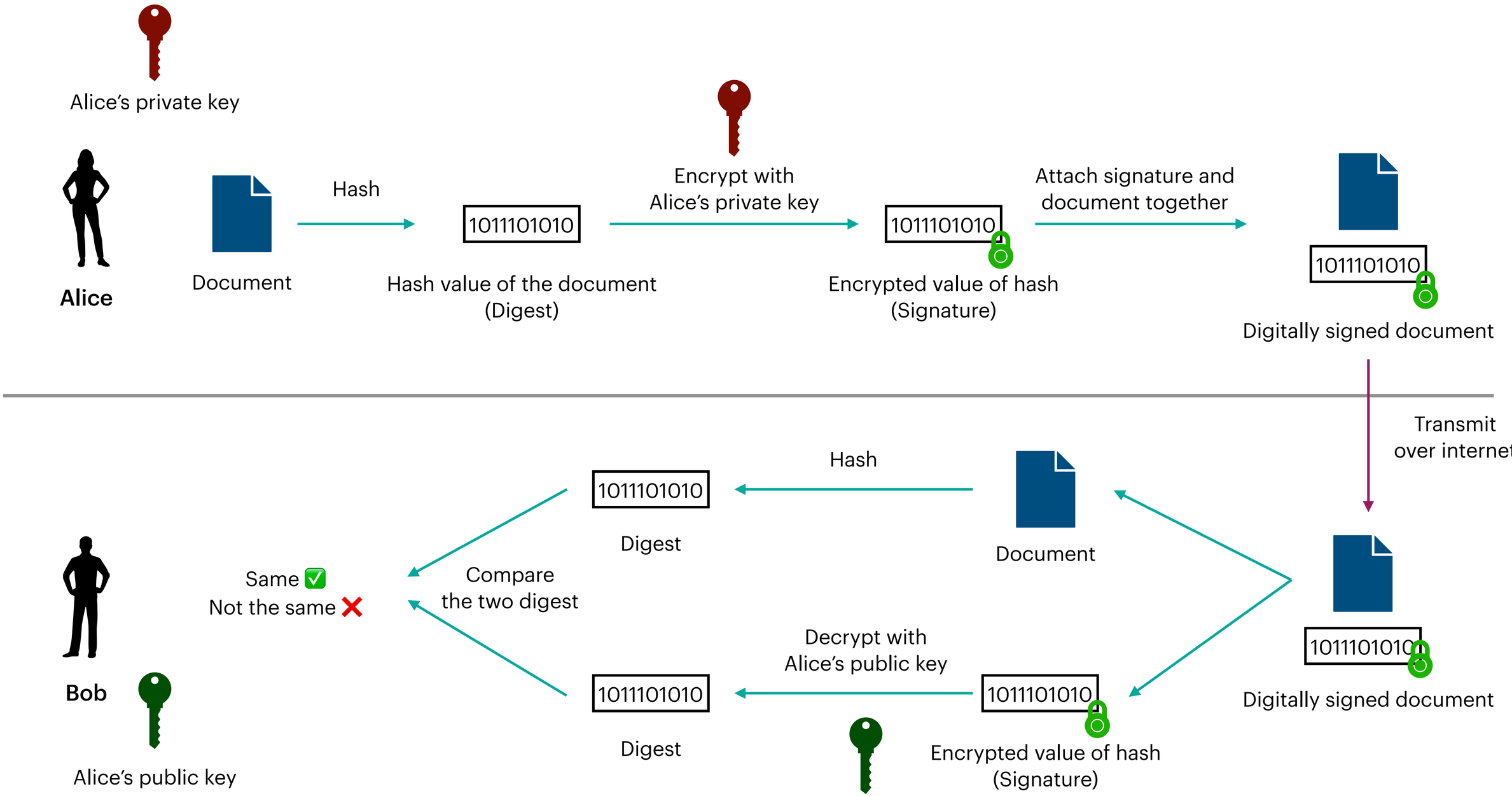


Digital Signature

Sethanant Pipatpakorn

Digital Signature is a way to ensure that this data come from this guy

Alice sent document with digital signature to Bob



Bitcoin mining (PoW)

Sethanant Pipatpakorn

SHA-256

- Hashing algorithm that output as 256 bits
- Total number of 2^{256} combination
- Usually represented by 64 character of hexadecimal

Ex: 7e34be935448089d52216e7b53bcae3fefe084683697496e72a98e7ee6c618f4

We want to get this value hashing with SHA-267. How to get it?

Target Value: 63e56408dbd35c8fdedad11ea151a8c82ae6df8da77751bbd117c5cb5958723b

B r u t h F o r c e ! !

Probability = 1 of 2^{256}

* stands for any value

If we say that, we want hash value that leading with one 0

Target Value: 0**********

Probability = 2^{252} of 2^{256}

Easy to find based on probability

* stands for any value

What about two leading 0

Target Value: 00**

Probability = 2^{248} of 2^{256}

Still, easy to find!

* stands for any value

How about 10 leading zero

Target Value: 0000000000**

Probability = 2^{216} of 2^{256}

Harder, I guess

* stands for any value

How about 20!!

Target Value: 00000000000000000000**********

Probability = 2^{176} of 2^{256}

It’s getting harder

* stands for any value

Push the limit

Target Value: 0000000000000000000000000000000000000000000000000000000000000000

Probability = 1 of 2^{256}

ชาติหน้าก็หาไม่ได้

The more number of leading zero required, the harder to get the satisfy hash value

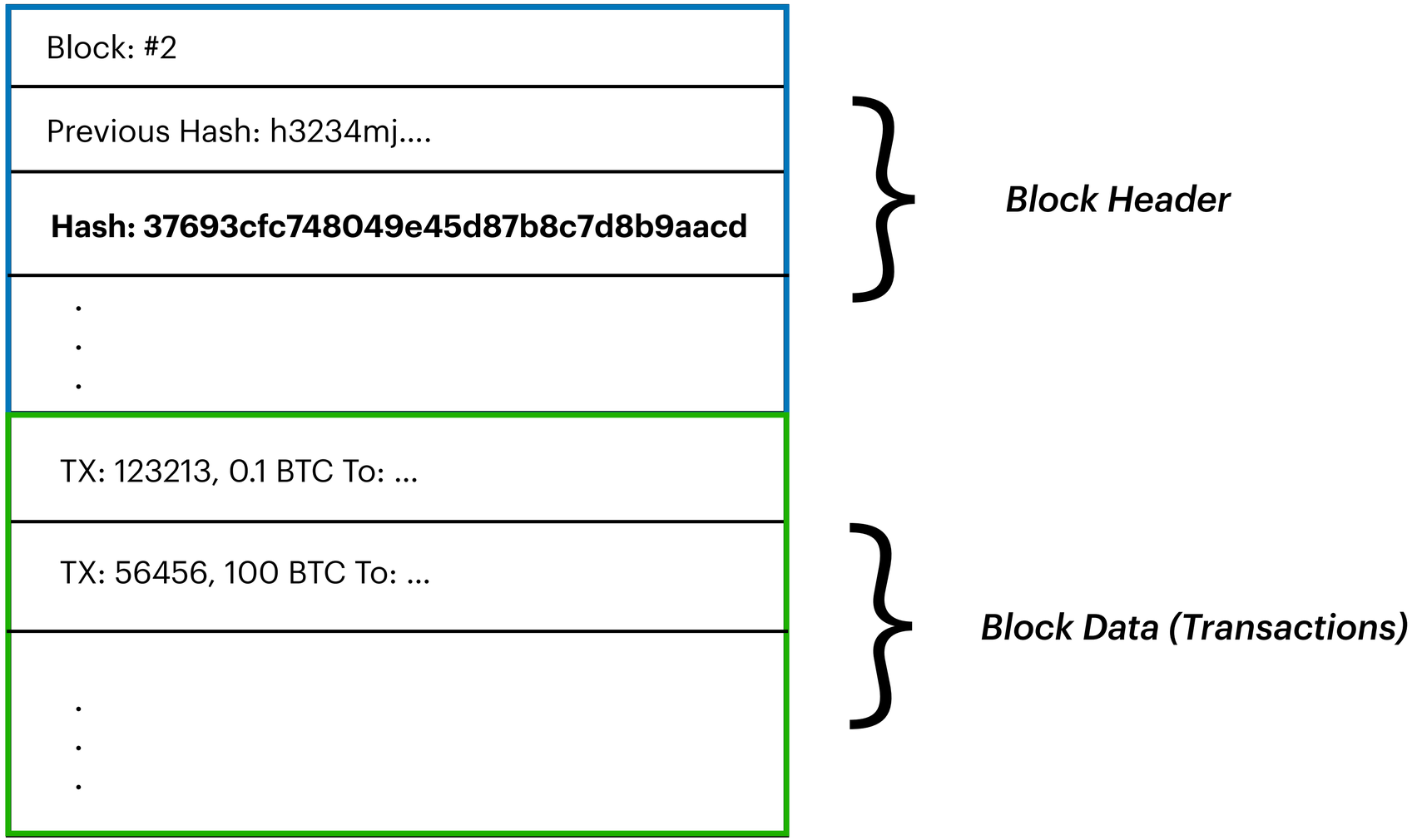
Proof of Work (PoW)

- Given the desire number of leading 0, compute the hash that satisfy the condition
- The one who find the hash, get to write a new block (and get coin as incentive)
- The difficulty can be adjust by adjusting the number of leading zero in output hash value

For example, if we set difficulty to 4 leading zero

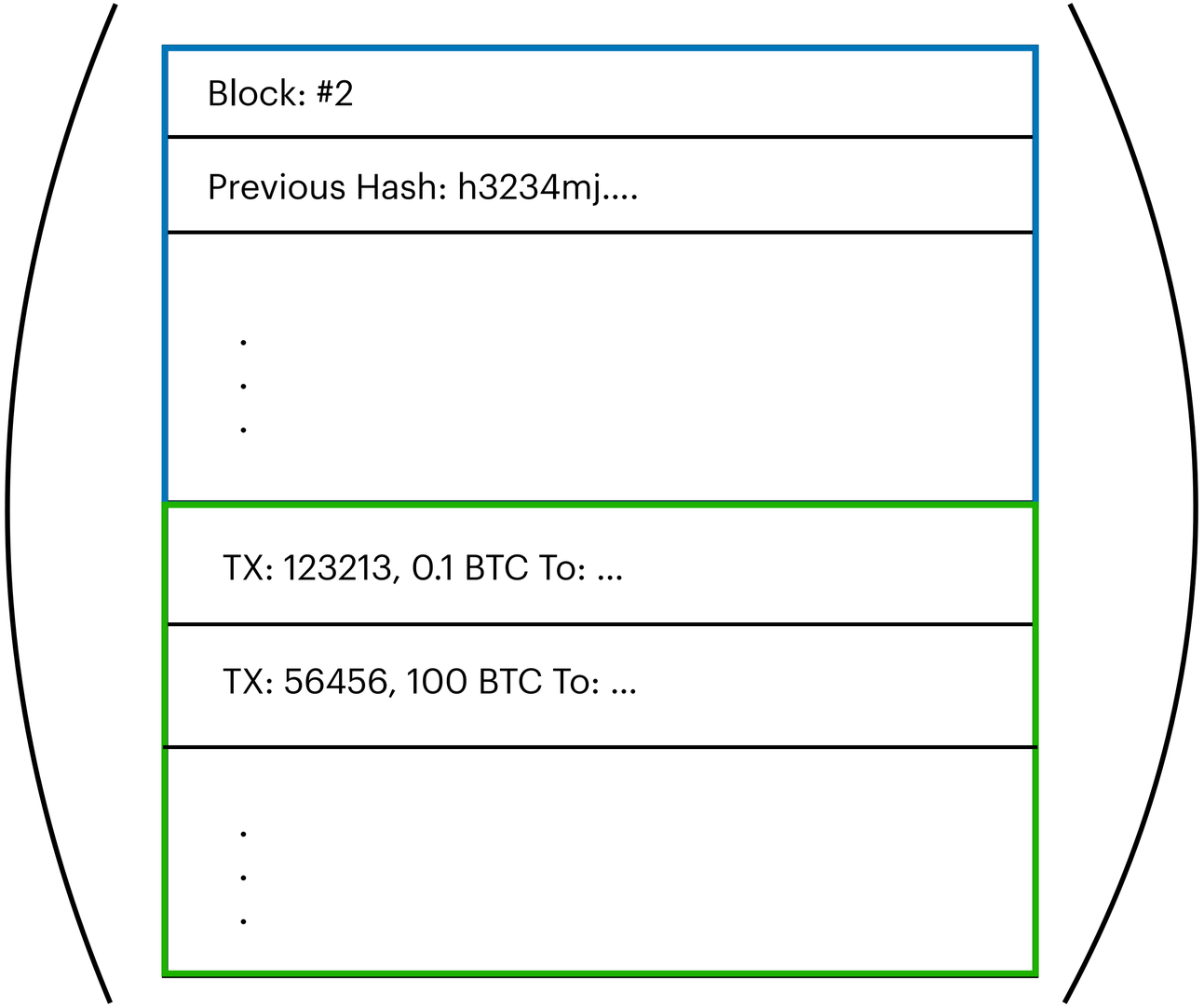
Target: 4 leading zero

Target: 4 leading zero



Hash of the block

SHA256

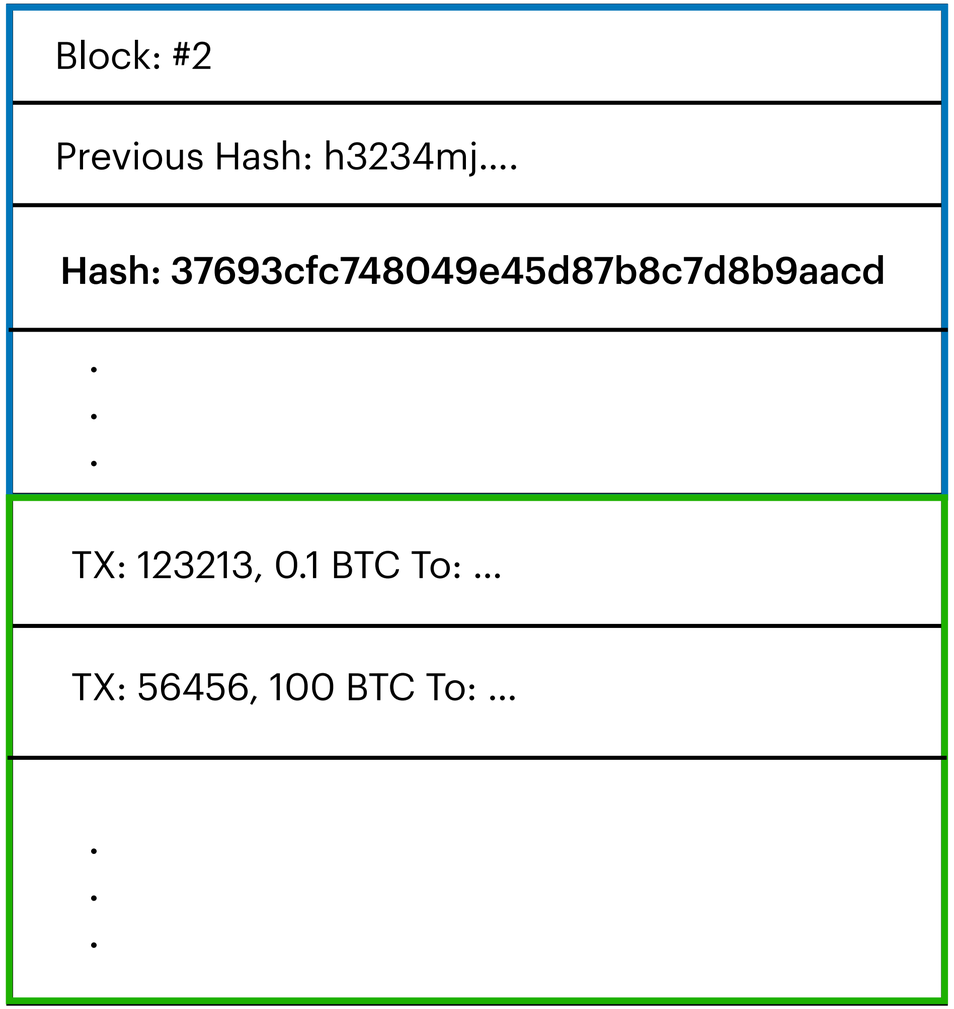


= hash value of this block

Changing anything in the block -> Hash value is changed -> Very hard to change the historical records

How can we get the desire number of leading zero

Target: 4 leading zero



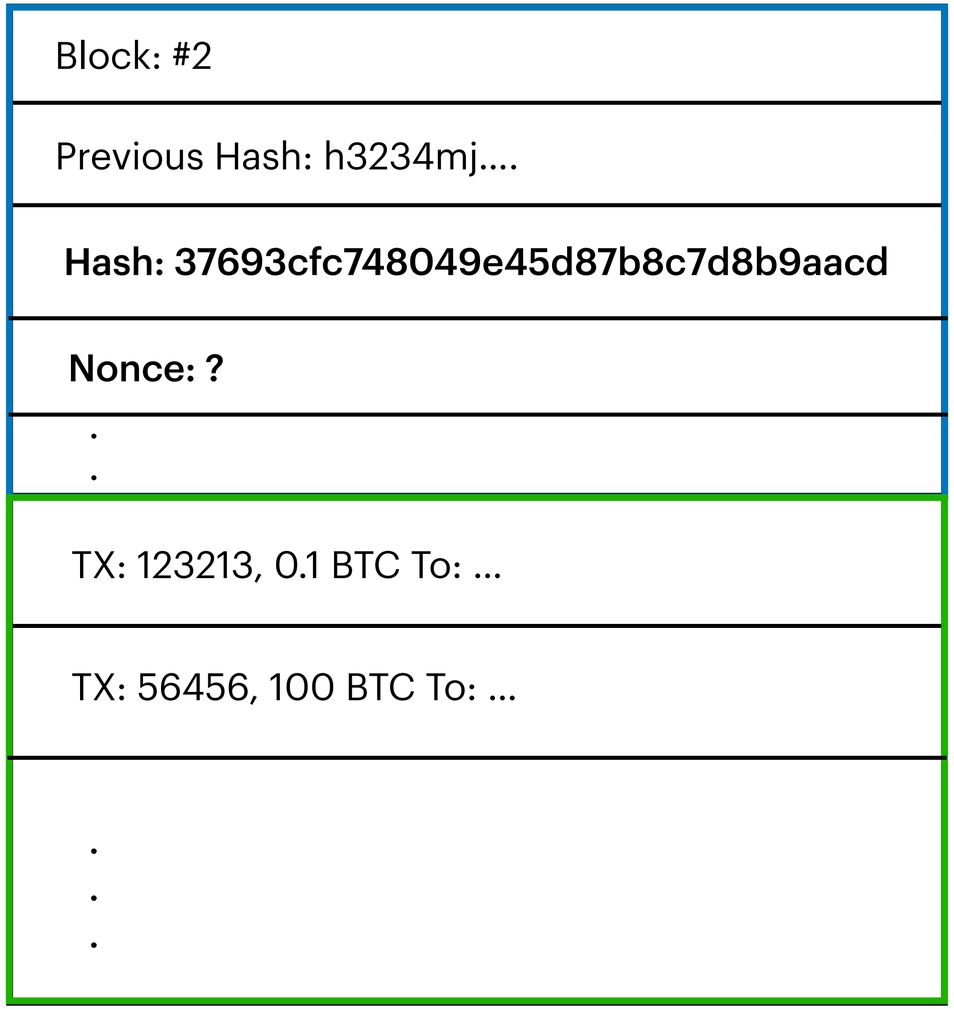
Suppose that hash Value of this block is 37693cfc748049e45d87b8c7d8b9aacd

How can we get the desire number of leading zero

Target: 4 leading zero

Added a new value in
block header called “Nonce”

Nonce is a variable that can be changed
to get satisfy hash value

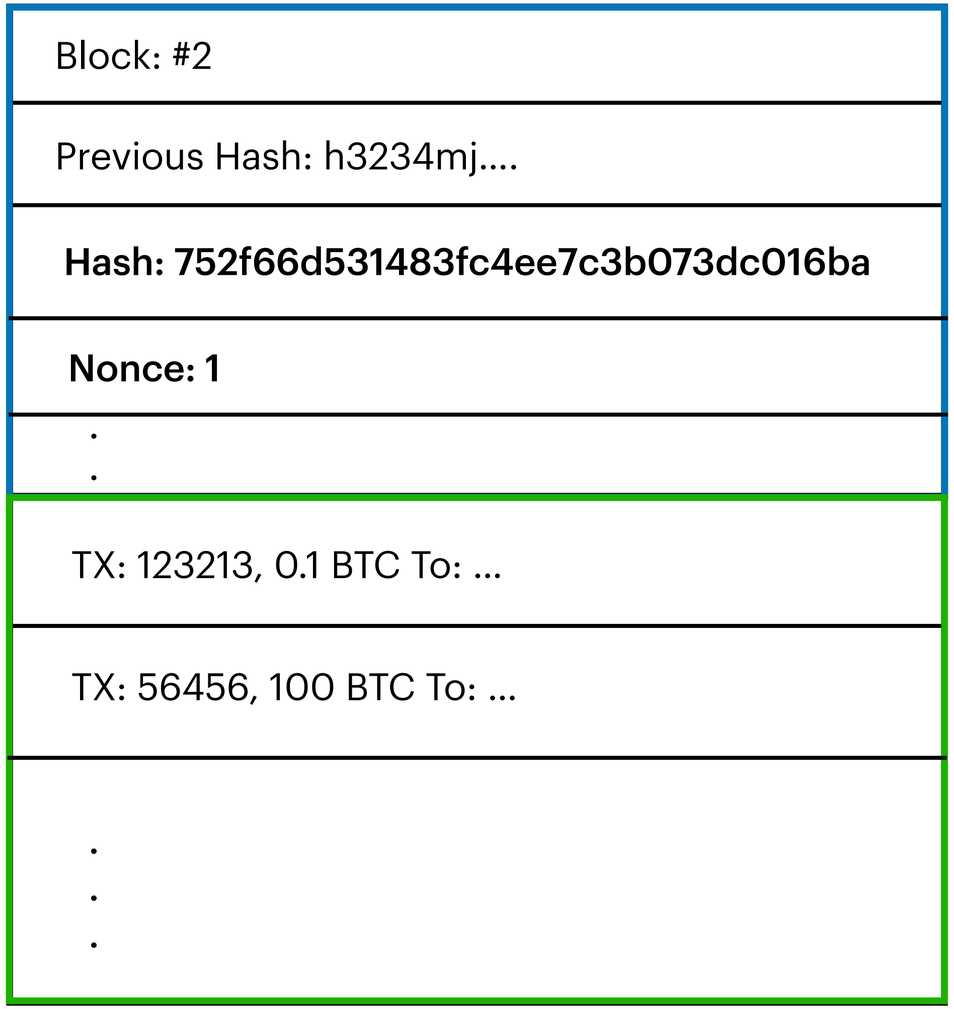


Suppose that hash Value of this block is 37693cfc748049e45d87b8c7d8b9aacd

How can we get the desire number of leading zero

Target: 4 leading zero

Nonce cause hash value to change dramatically due to characteristic of hash function



How can we get the desire number of leading zero

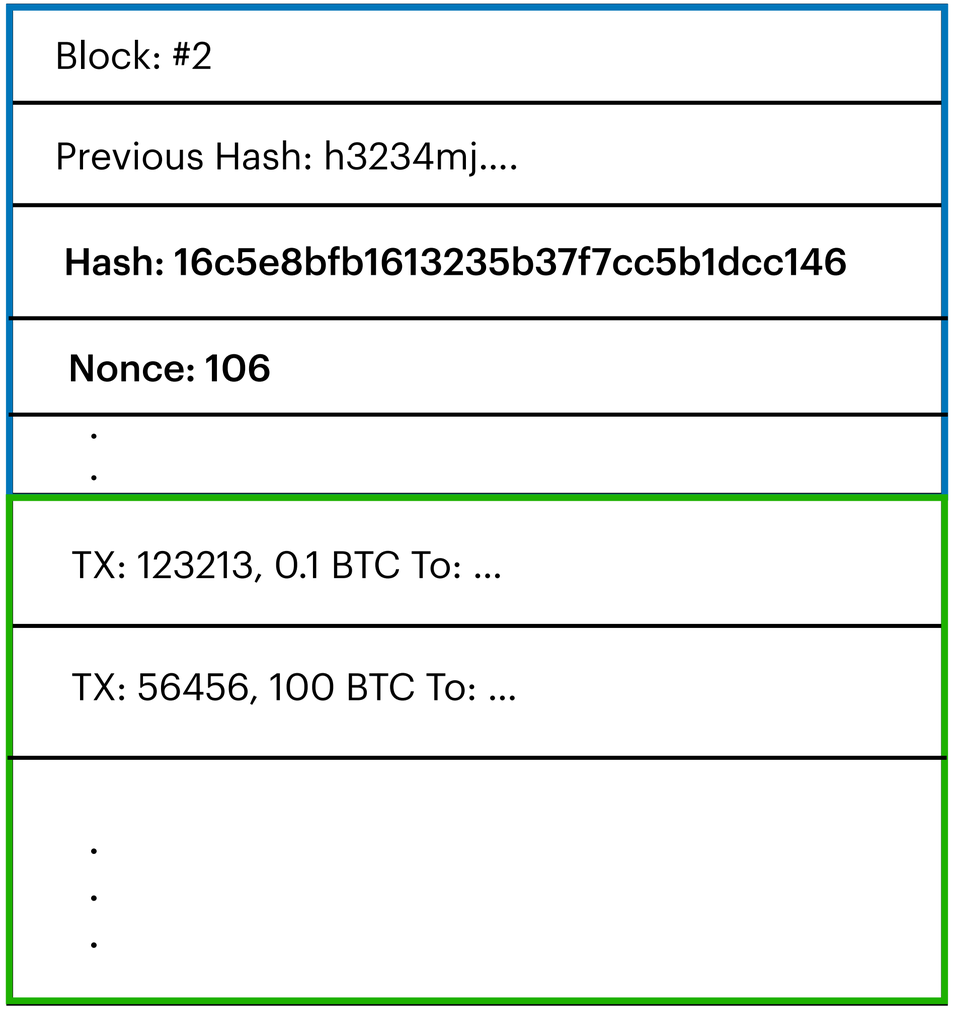
Target: 4 leading zero

Nonce cause hash value to change dramatically due to characteristic of hash function

How can we get the desire number of leading zero

Target: 4 leading zero

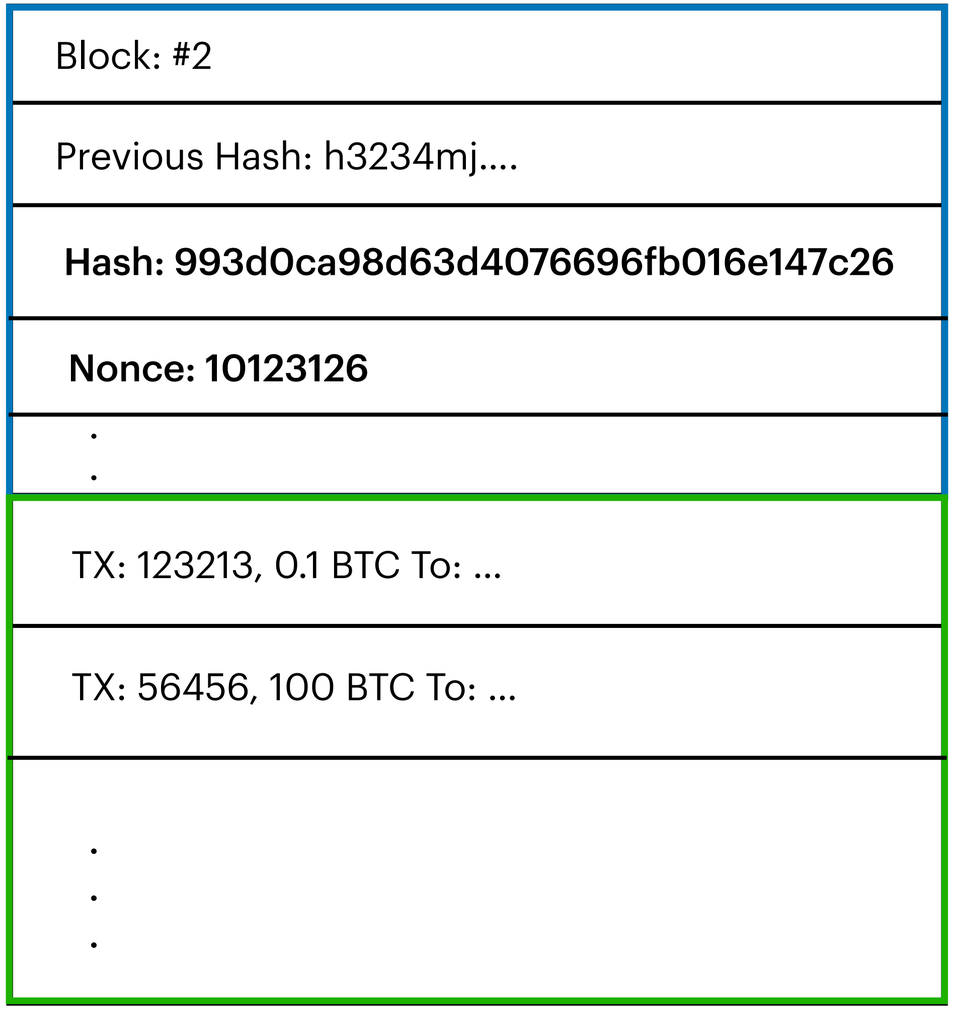
Nonce cause hash value to change dramatically due to characteristic of hash function



How can we get the desire number of leading zero

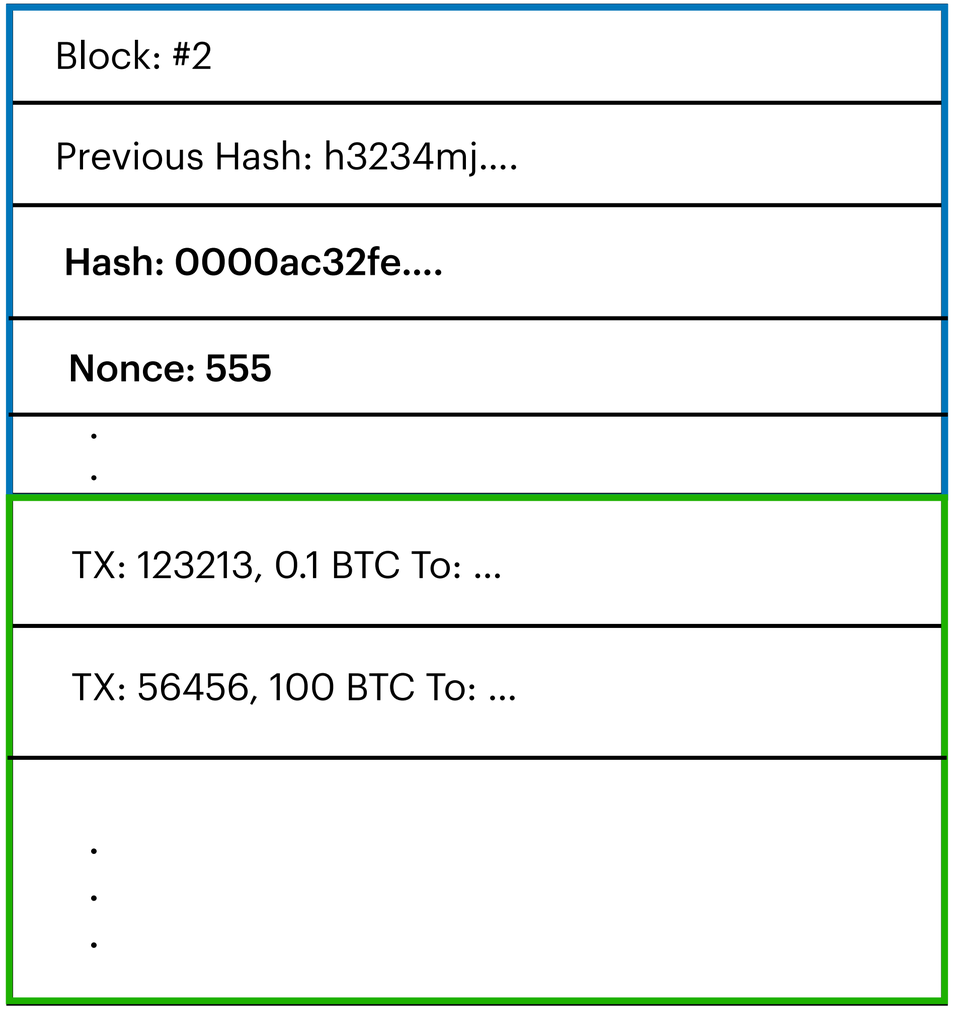
Target: 4 leading zero

Nonce cause hash value to change dramatically due to characteristic of hash function



How can we get the desire number of leading zero

Target: 4 leading zero



The satisfy nonce is found!!

- The miner received coin for his work
- This block can be broadcast to other node in the network

Conclusion

- Number of leading zero in block hash defined the difficulty of the problem
- “Nonce” is used as a variable to change hash value of the block
- Miner bruthforce to find the satisfy nonce