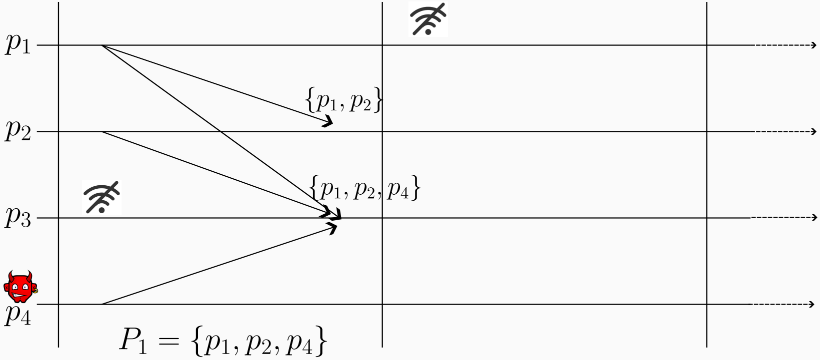
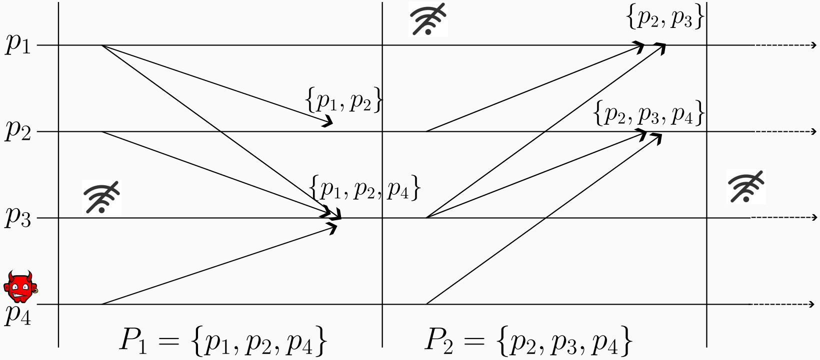
Consensus under dynamic participation with a minority of failures

Giuliano Losa, Stellar Development Foundation
Eli Gafni, UCLA

We have players in a synchronous system with dynamic participation and a static, always-online, minority adversary



We have players in a synchronous system with dynamic participation and a static, always-online, minority adversary



We want to solve consensus with constant expected latency

Each player is given an external input and must produce an output such that:

Agreement

No two well-behaved players output differently

Validity

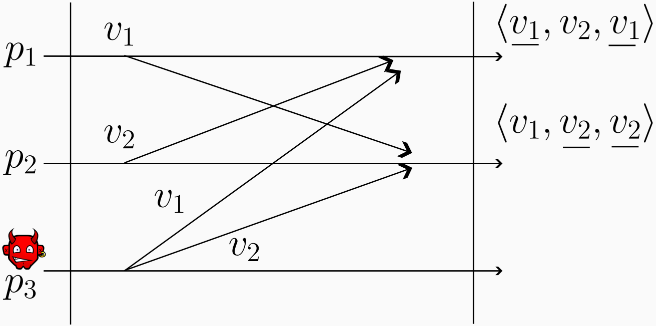
If all well-behaved players have the same input v, then no players outputs $v' \neq v$

Termination

There is a constant[1] N such that, in expectation, in every round $r \geq N$, every online, well-behaved player outputs.

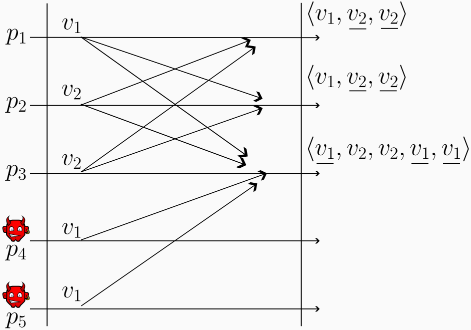
[1]Constant means independent of the number of players and level of the participation.

Difficulty: we cannot expect to rely on more than a majority; yet players may witness conflicting majorities



p_1 gets a majority for v_1; p_2 gets a majority for v_2

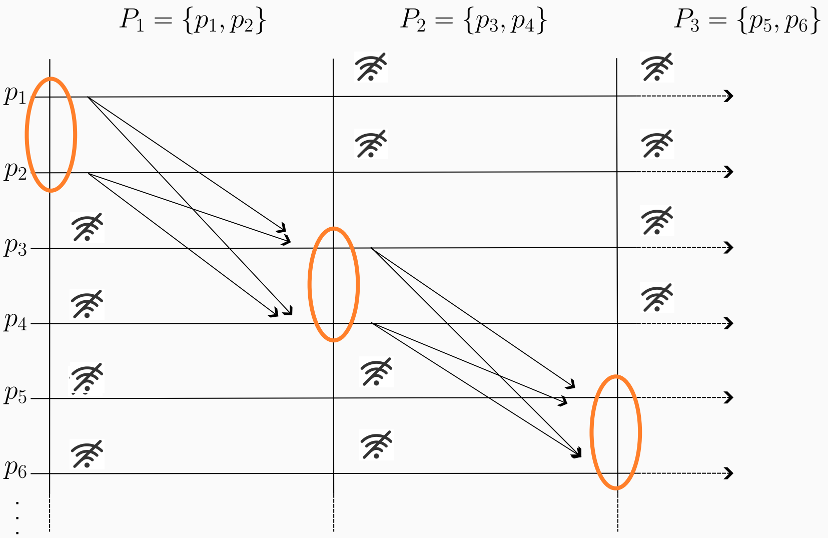
Difficulty: players may witness conflicting majorities even without equivocation



Players p_1 and p_2 get a majority for v_2; p_3 gets a majority for v_1.

For p_1 and p_2, this is indistinguishable from p_4 and p_5 being offline.

Difficulty: local state is useless because no well-behaved player may participate more than once



What are the properties we can rely on?

Each round:

- All online players received all the messages from the well-behaved players of the previous round
- If a player *p* receives *v* from a strict majority, then at least one well-behaved player sent *v*

What are the properties we can rely on?

Each round:

- All online players received all the messages from the well-behaved players of the previous round
- If a player p receives v from a strict majority, then at least one well-behaved player sent v

Plan of attack:

1. Simulate a model that forbids equivocation and selective disclosure of participation. This rules out conflicting majorities.
2. Solve consensus in this new model.

The no-equivocation model prevents conflicting majorities

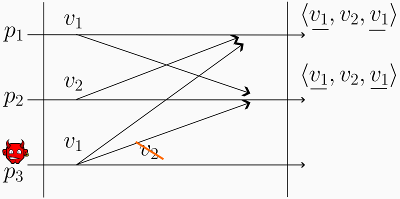
The no-equivocation model with failure-notification λ

1. Players cannot equivocate:
   if p' receives $v \neq \lambda$ from p and p'' receives v' from p,
   then $v' = v$ or $v' = \lambda$.
2. Players cannot selectively send messages:
   if p' receives v from p and p'' does not,
   then p'' receives the failure notification λ from p.

The no-equivocation model prevents conflicting majorities

The no-equivocation model with failure-notification λ

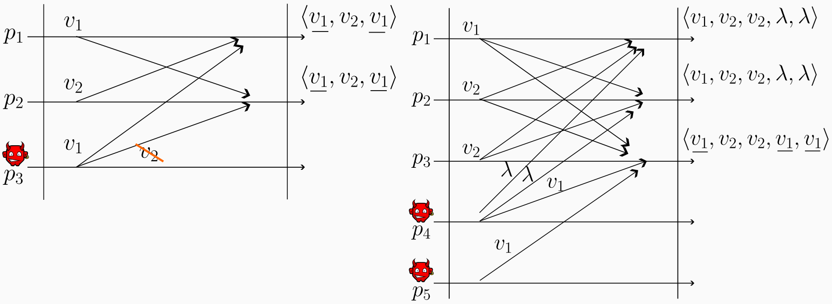
1. Players cannot equivocate:
   if p' receives $v \neq \lambda$ from p and p'' receives v' from p,
   then $v' = v$ or $v' = \lambda$.
2. Players cannot selectively send messages:
   if p' receives v from p and p'' does not,
   then p'' receives the failure notification λ from p.



The no-equivocation model prevents conflicting majorities

The no-equivocation model with failure-notification λ

1. Players cannot equivocate:
   if p' receives $v \neq \lambda$ from p and p'' receives v' from p,
   then $v' = v$ or $v' = \lambda$.
2. Players cannot selectively send messages:
   if p' receives v from p and p'' does not,
   then p'' receives the failure notification λ from p.



We implement a no-equivocation round in 2 base rounds using message authentication

We implement a no-equivocation round in 2 base rounds using message authentication

Round 1

Each online player signs and broadcasts the message to simulate

We implement a no-equivocation round in 2 base rounds using message authentication

Round 1

Each online player signs and broadcasts the message to simulate

Round 2

Each online player re-broadcasts all the signed messages it received

We implement a no-equivocation round in 2 base rounds using message authentication

Round 1

Each online player signs and broadcasts the message to simulate

Round 2

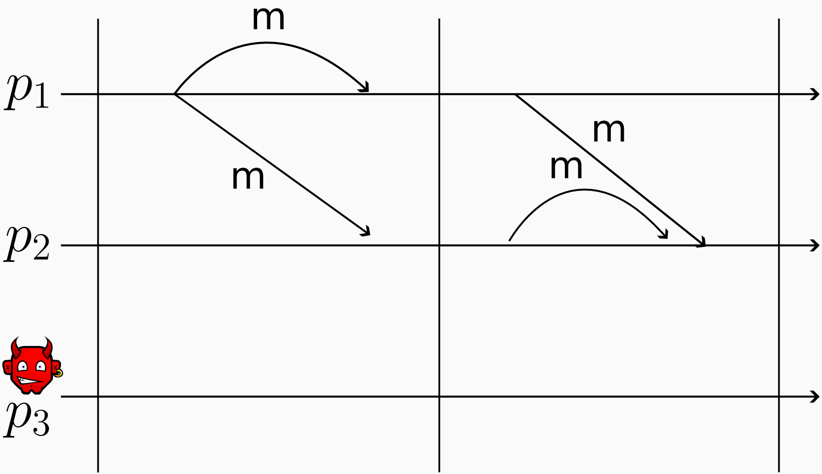
Each online player re-broadcasts all the signed messages it received

Output

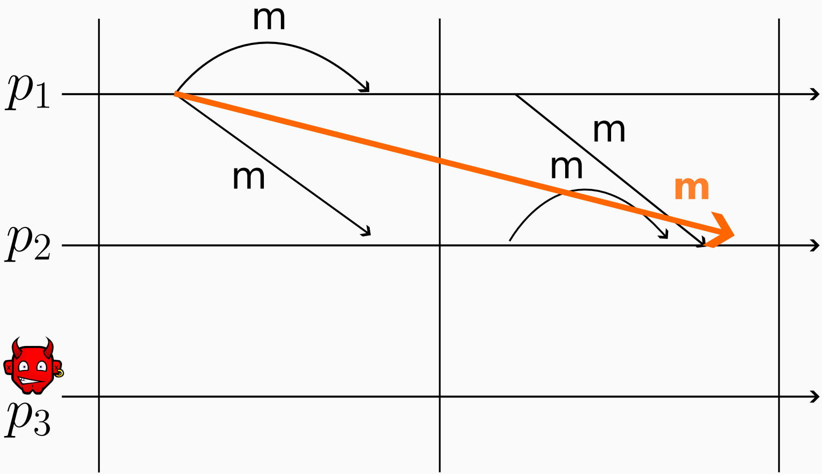
For each online player p: for each player p' that p hears of:

1. If p hears that p' sent two different messages in round 1, then simulate receiving λ from p'.
2. Else, if a strict majority forwarded a message m from p', then simulate receiving m for p'.
3. Else, simulate receiving λ from p'

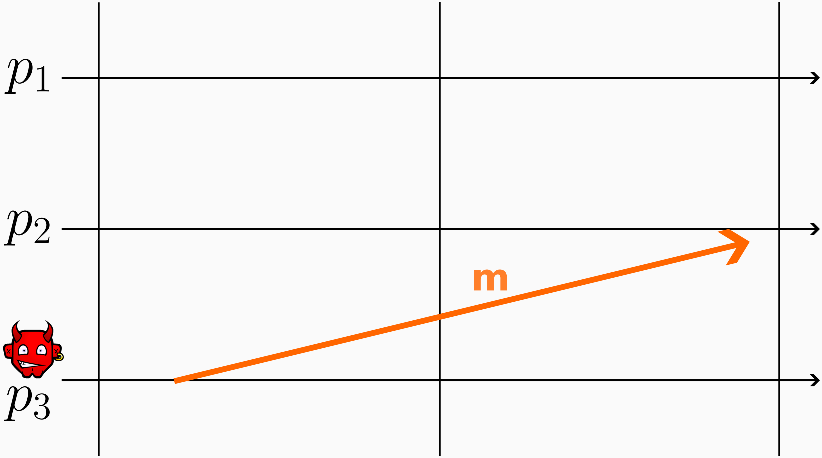
A well-behaved player always gets its message simulated



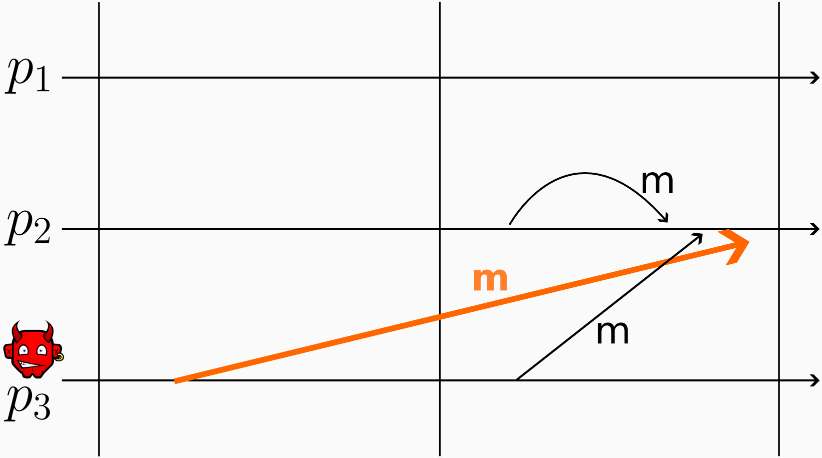
A well-behaved player always gets its message simulated



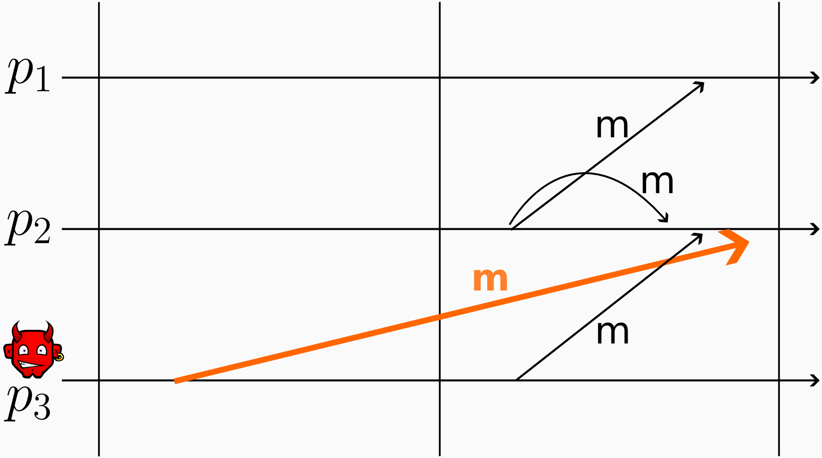
An ill-behaved player cannot cheat



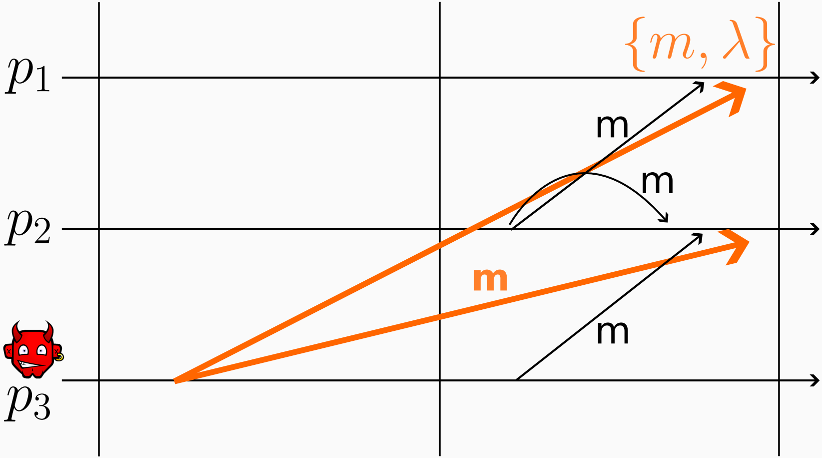
An ill-behaved player cannot cheat



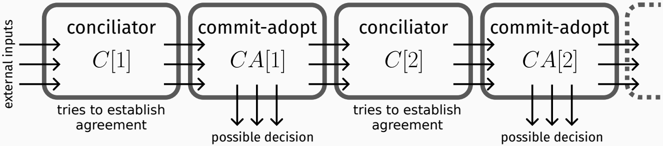
An ill-behaved player cannot cheat



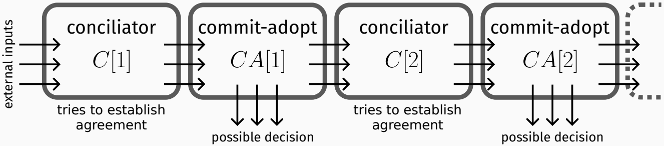
An ill-behaved player cannot cheat



We solve consensus with an alternating sequence of conciliator and adopt-commit phases



We solve consensus with an alternating sequence of conciliator and adopt-commit phases



Commit-Adopt

Each player outputs either commit(*v*) or adopt(*v*) for some *v*, subject to:

- Validity: If all well-behaved have input *v*, then all well-behaved commit *v*.
- Agreement: If a well-behaved player commits a value *v*, then all well-behaved players either commit or adopt *v*.

Conciliator

Each player outputs a value, subject to:

- Validity: If all well-behaved have input *v*, then all well-behaved commit *v*.
- Agreement: With probability 1/2, all players output the same value *v*.

We implement commit-adopt in 2 no-equivocation rounds

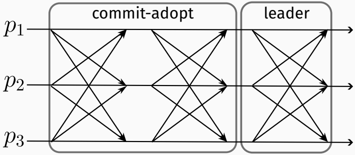
Round 1

Broadcast input

Round 2

1. If received a value v from a strict majority
   then broadcast v else broadcast $\perp$
2. At the end of the round:
   a) If received v from a strict majority, commit v.
   b) Else, if received v more often than any other value, adopt v.
   c) Else, adopt any value.

The conciliator takes 3 no-equivocation rounds



Rounds 1 and 2:

Do commit-adopt.

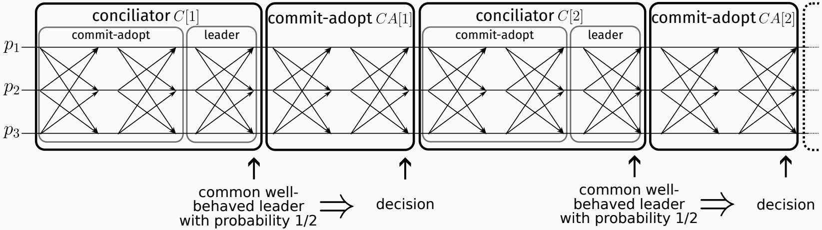
Round 3:

- Broadcast commit-adopt output and VRF evaluation
- End of round:
  - If received a value *v* from a majority, output *v*.
  - Else, output the value of the player with largest VRF output.

Question

Why do we need the two commit-adopt rounds before leader-election?

Consensus in $N = 10$ no-equivocation rounds in expectation



Why is the commit-adopt algorithm correct?

Round 1

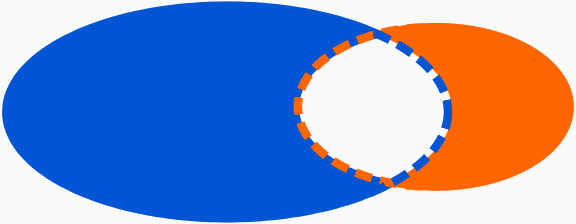
Broadcast input

Round 2

1. If received a value v from a strict majority
   then broadcast v else broadcast $\perp$
2. At the end of the round:
   a) If received v from a strict majority, commit v.
   b) Else, if received v more often than any other value, adopt v.
   c) Else, adopt any value.

The commit-adopt algorithm relies on a simple property of sets: if $|X| > |Y|$, then $|X \setminus Y| > |Y \setminus X|$

$$|X \setminus Y| > |Y \setminus X|$$



Note: no two well-behaved players broadcast different values in round 2

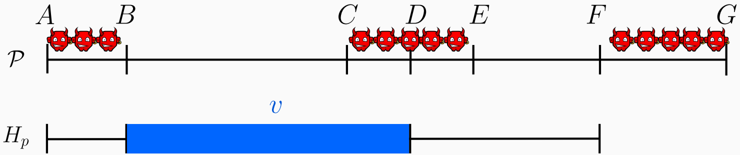
Round 1

Broadcast input

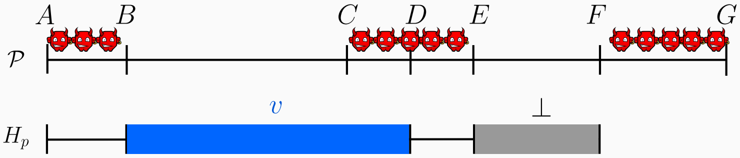
Round 2

1. If received a value v from a strict majority
   then broadcast v else broadcast $\perp$
2. At the end of the round:
   a) If received v from a strict majority, commit v.
   b) Else, if received v more often than any other value, adopt v.
   c) Else, adopt any value.

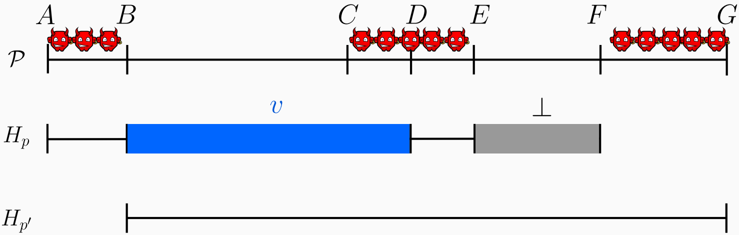
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



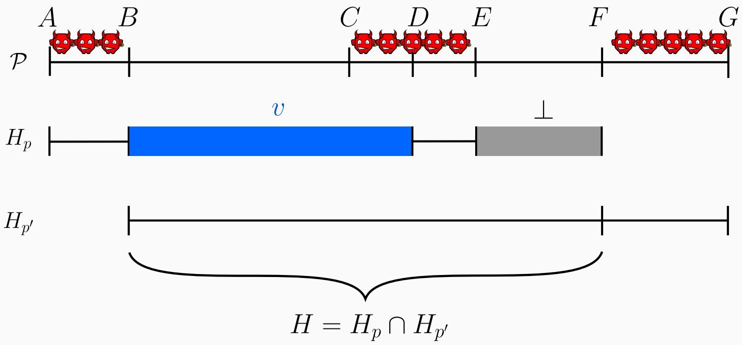
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



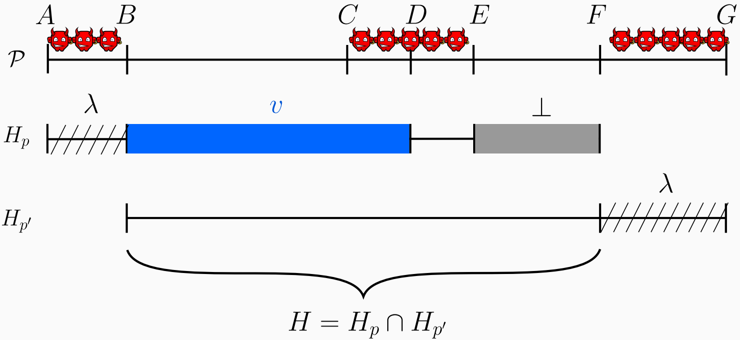
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



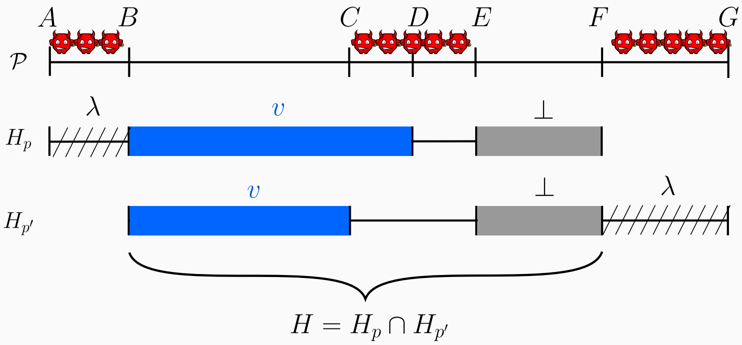
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



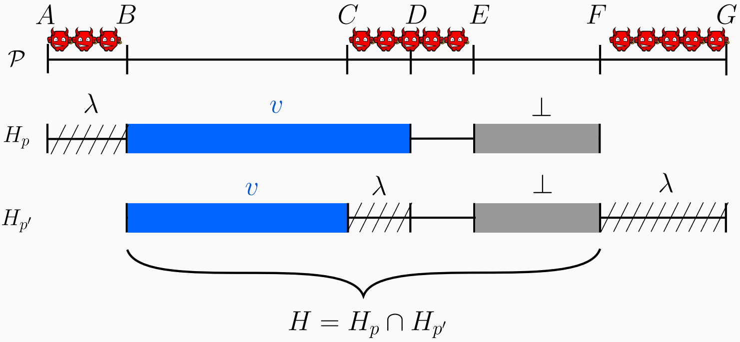
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



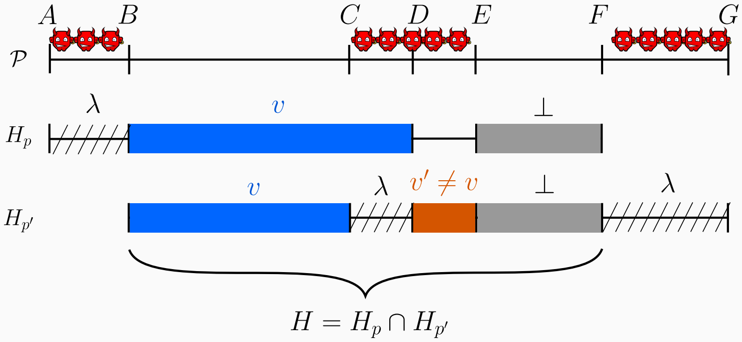
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



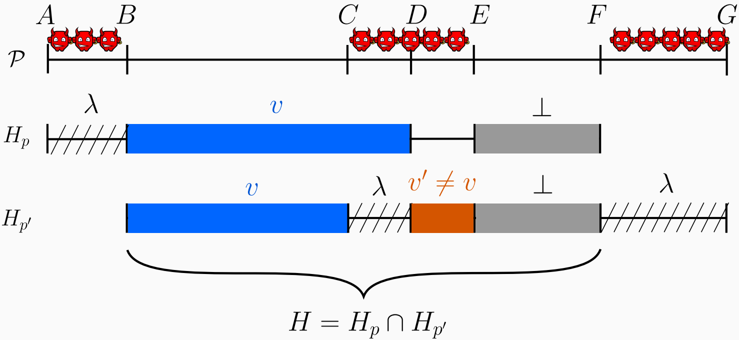
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$

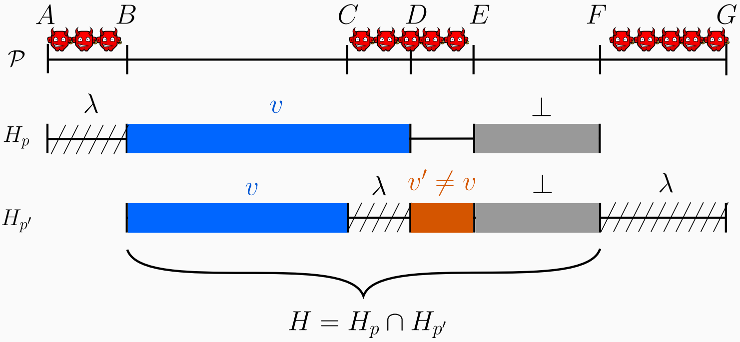


Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



$\#_{p'}(v) = [B, D] - [C, E]$

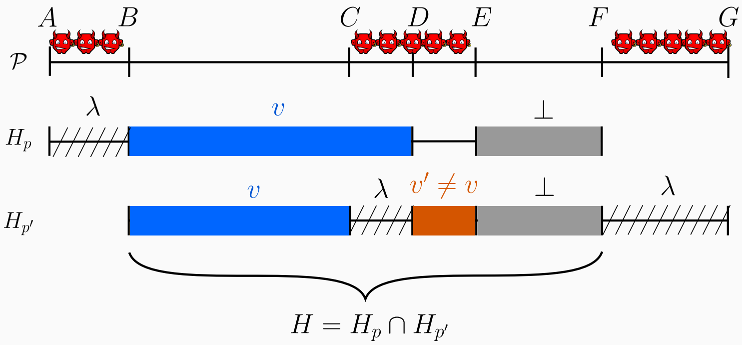
Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



$\#_{p'}(v) = [B, D] - [C, E]$

$\#_{p'}(v') = [C, E] - [B, D]$

Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$

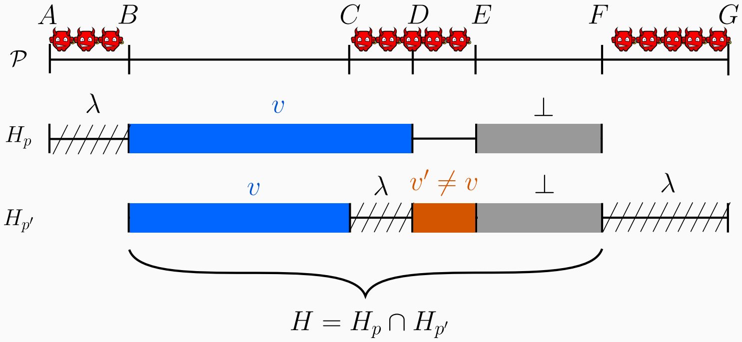


$\#_{p'}(v) = [B, D] - [C, E]$

$\#_{p'}(v') = [C, E] - [B, D]$

$[C, E]$ is a minority among H and $[B, D]$ a majority, so $[B, D] > [C, E]$

Assume p commits v; assume by contradiction that p' commits or adopts $v' \neq v$



$\#_{p'}(v) = [B, D] - [C, E]$

$\#_{p'}(v') = [C, E] - [B, D]$

$[C, E]$ is a minority among H and $[B, D]$ a majority, so $[B, D] > [C, E]$

By the property of sets: $\#_{p'}(v) > \#_{p'}(v')$

Q.E.D.

Paper and supplemental material

To be published at DISC 2023 (as a brief announcement), and available at `https://www.losa.fr`.

Supplemental material available at `https://github.com/nano-o/dynamic-participation-supplemental`

- TLA+ specifications of the algorithms.
- Mechanized proof of the commit-adopt algorithm in Isabelle/HOL.

Related work

- In a blog post, Malkhi, Momose, and Ren propose a different algorithm solving the same problem
- Pu et al. propose the Gorilla algorithm (DISC 2023), which solves consensus with deterministic safety using VDF proof of work.

Future work

It seems easy to implement the no-equivocation model in the resource-constrained VDF model of Gorilla.

This yields the first fully permissionless consensus algorithm with unconstrained participation (Bitcoin needs a known upper bound on participation)