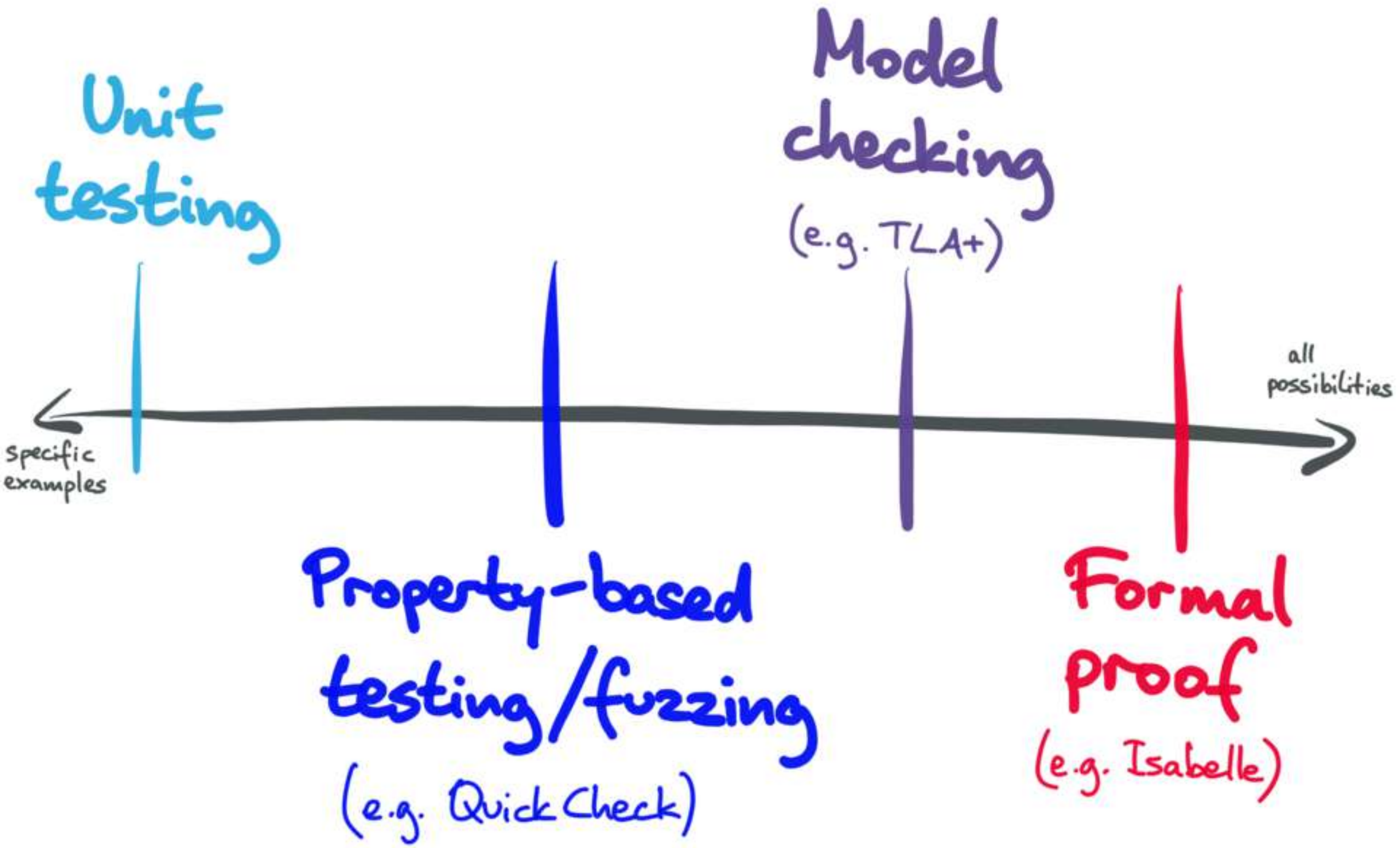
Correctness proofs of distributed systems with Isabelle/HOL

Martin Kleppmann • University of Cambridge, UK

martin@kleppmann.com • @martinkl


Unit testing
Property-based testing/fuzzing
(e.g. QuickCheck)
Model checking
(e.g. TLA+)
Formal proof
(e.g. Isabelle)
specific examples
all possibilities

WHY BOTHER?

- Subtle algorithms
  (correctness not obvious)
- Complicated state space
  (e.g. distributed systems, concurrency)
- For better human understanding
  (forcing yourself to be thorough & precise)

WHY BOTHER?

"Isabelle is the world's most complicated video game"

— Dominic Mulligan

DISTRIBUTED SYSTEMS

Approach: model the system using Isabelle/HOL data structures (lists, sets, ...)

DISTRIBUTED SYSTEMS

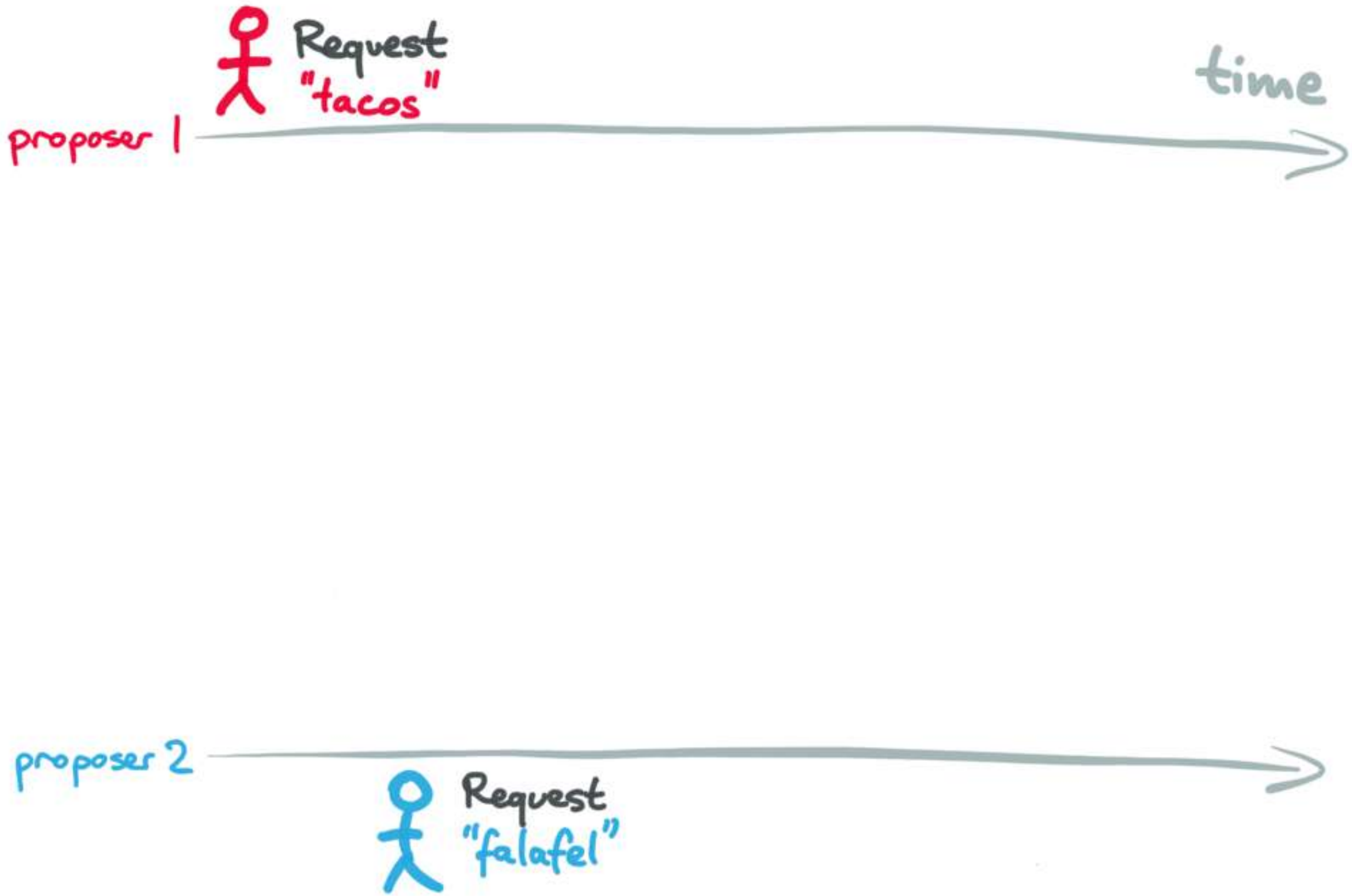
Approach: model the system using Isabelle/HOL data structures (lists, sets, ...)

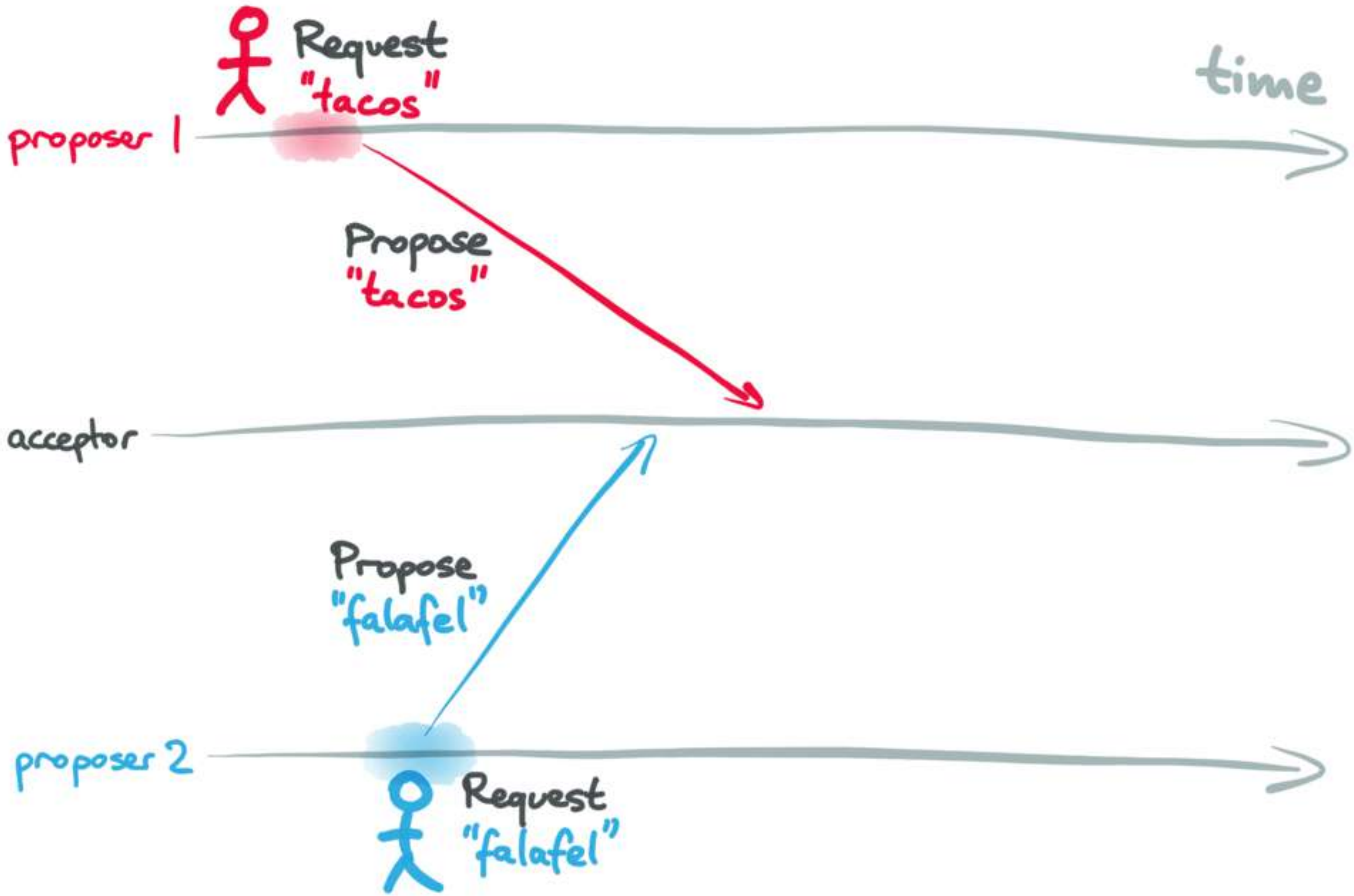
Example problem: consensus

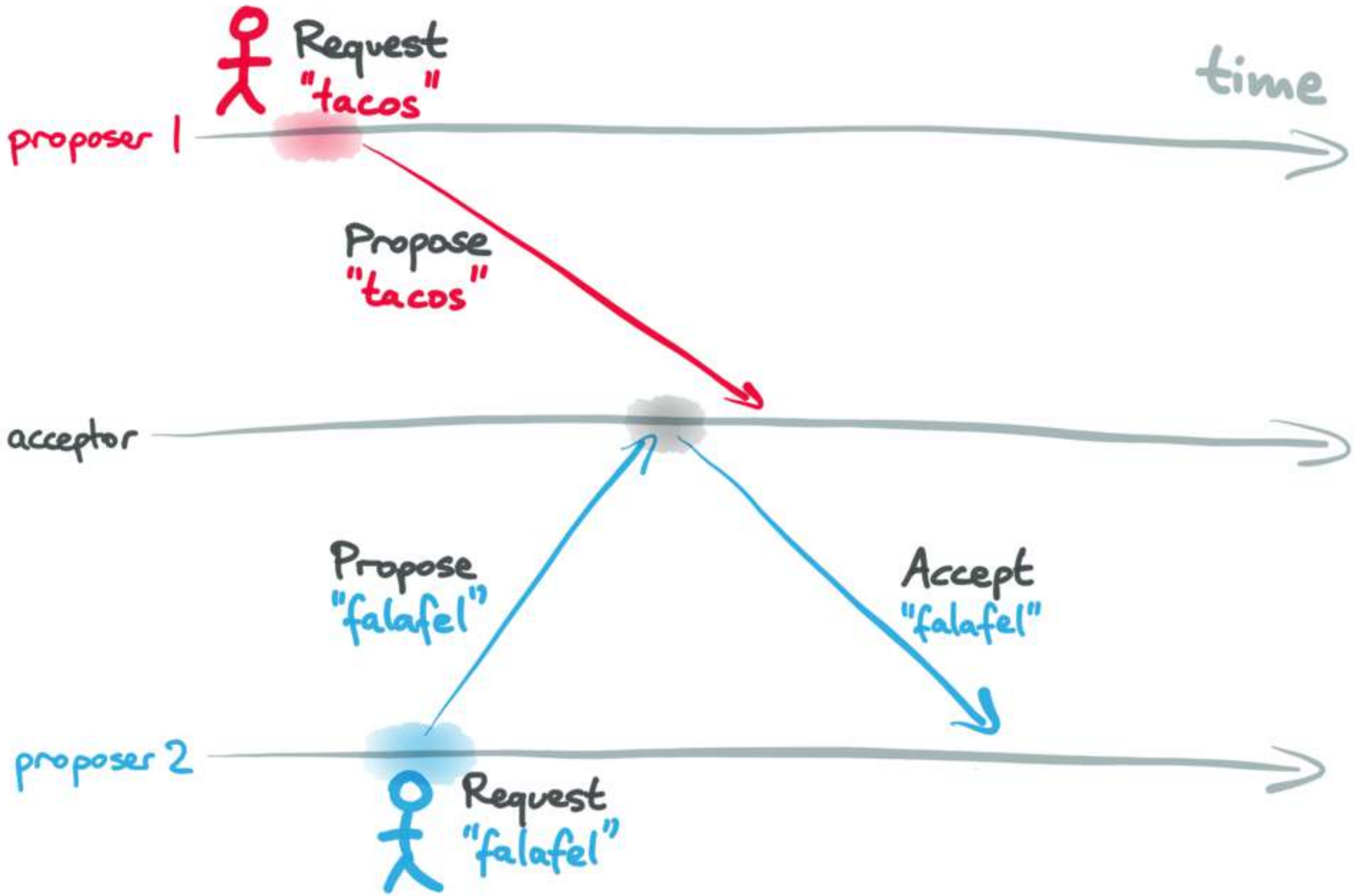
(a simple, non-fault-tolerant algorithm – can't fit Paxos/Raft in this talk)

THE AGREEMENT PROPERTY

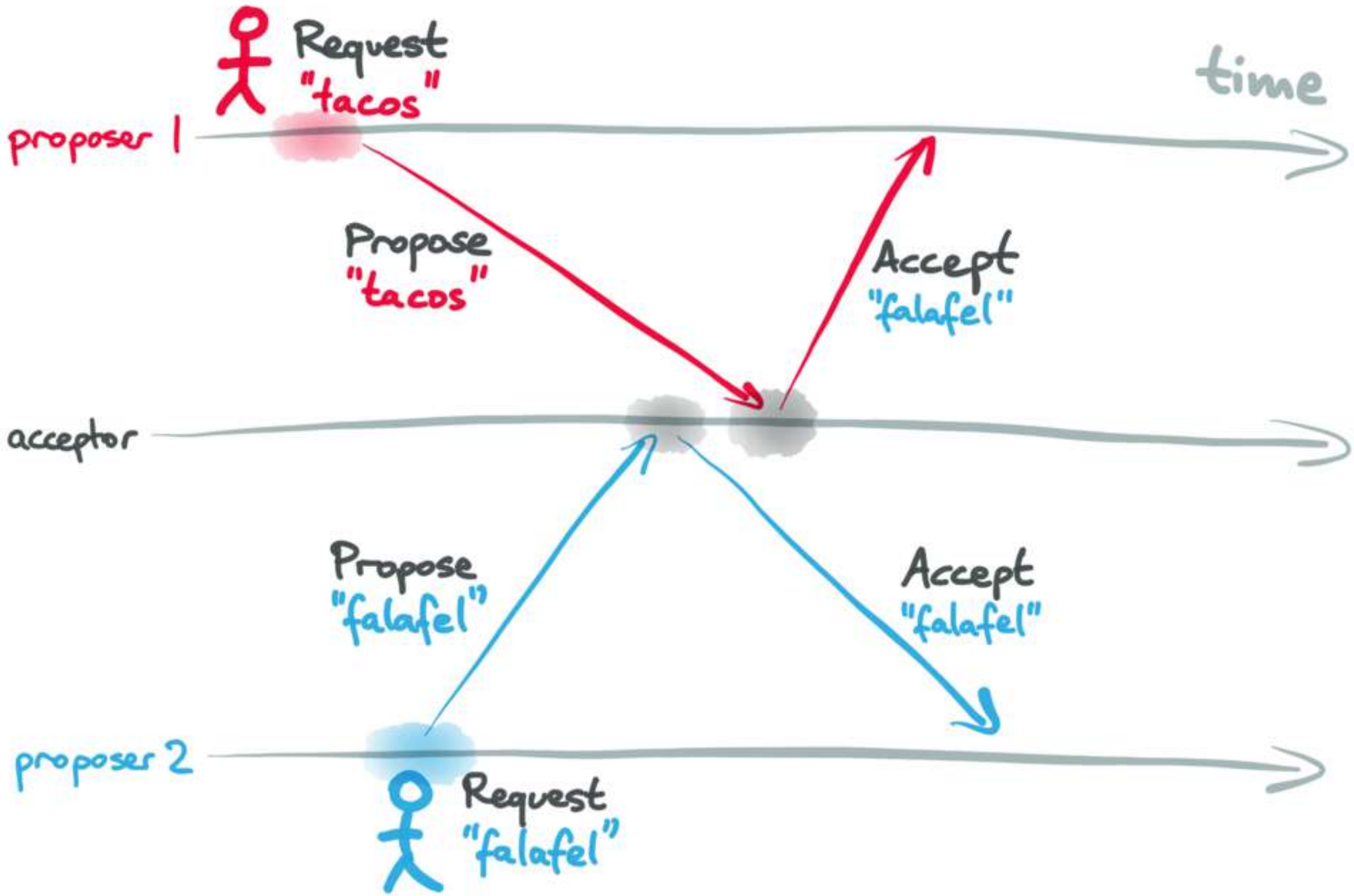
If any two processes learn decided values, those values are the same.

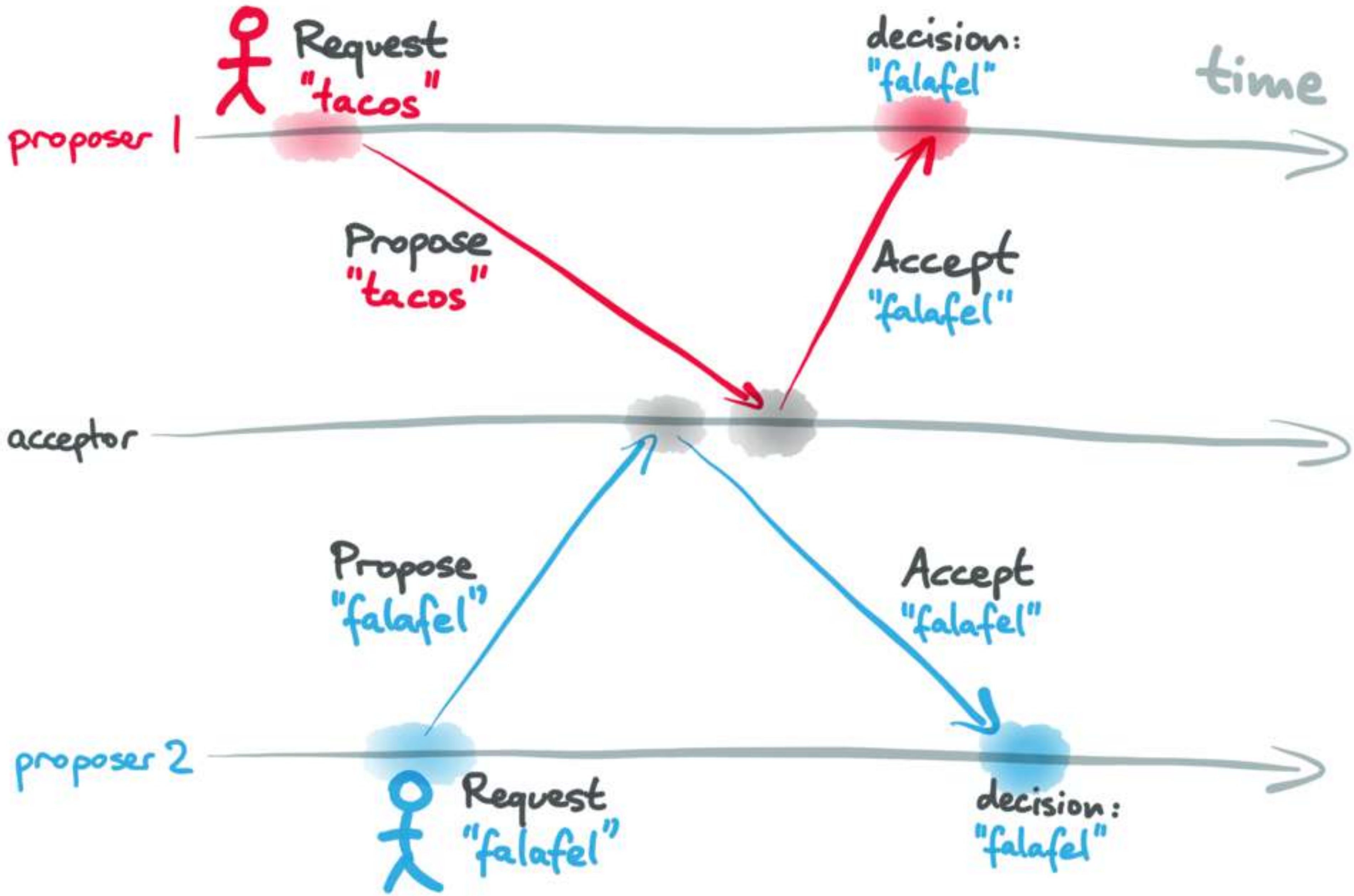

Request
"tacos"
time
proposer 1
proposer 2
Request
"falafel"


Request
"tacos"
time
proposer 1
Propose
"tacos"
acceptor
Propose
"falafel"
proposer 2
Request
"falafel"

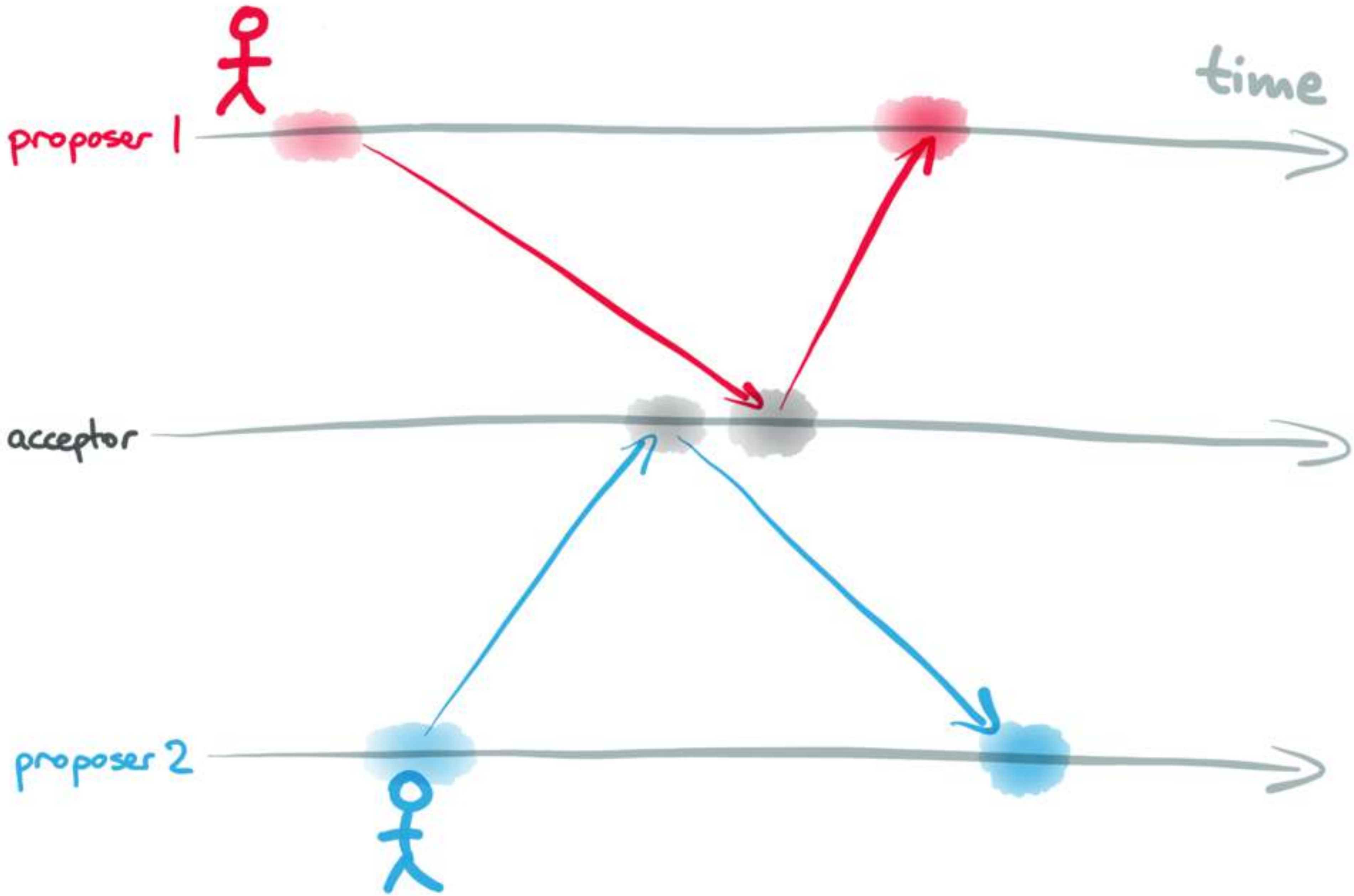


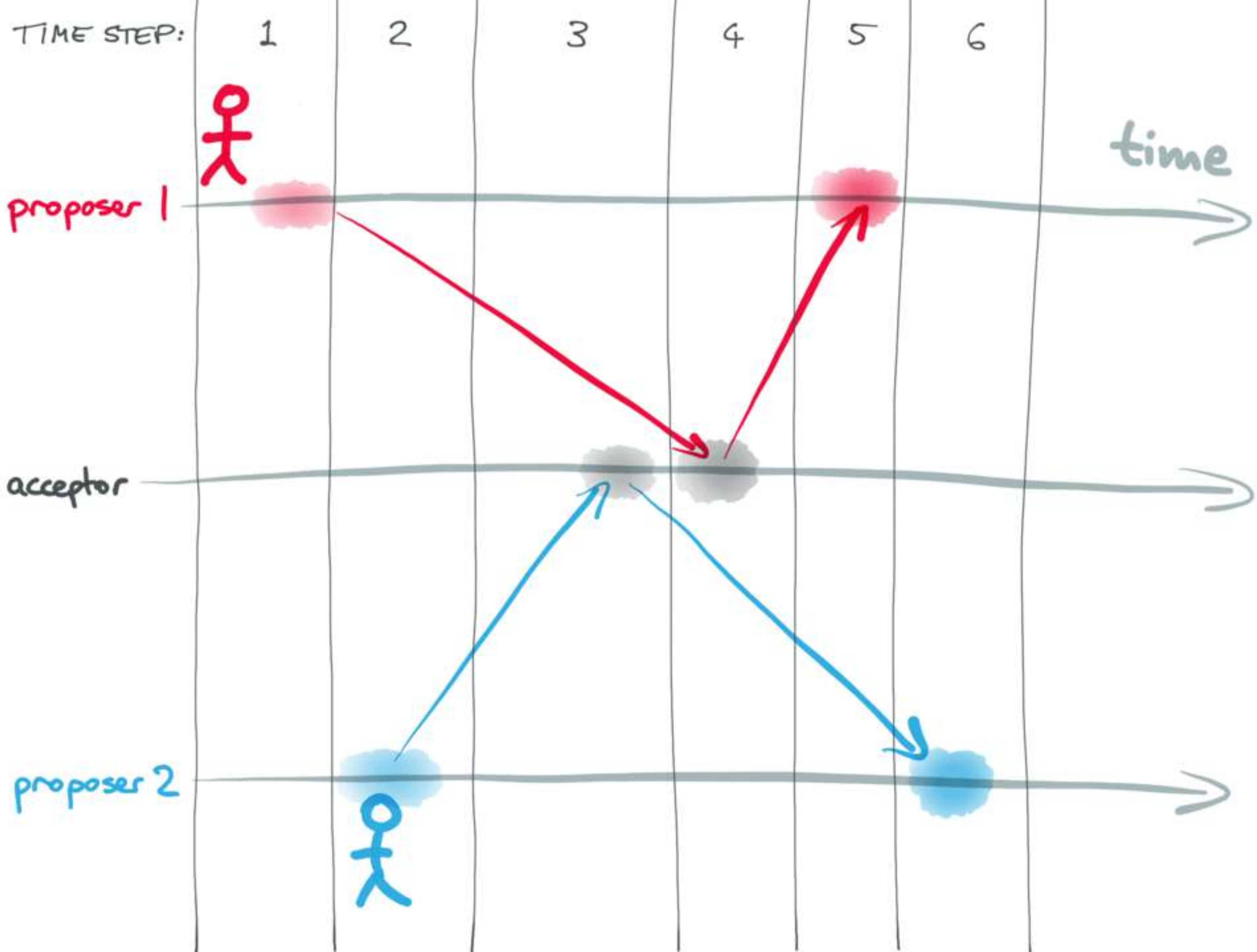
Request
"tacos"
time
proposer 1
Propose
"tacos"
acceptor
Propose
"falafel"
Accept
"falafel"
proposer 2
Request
"falafel"

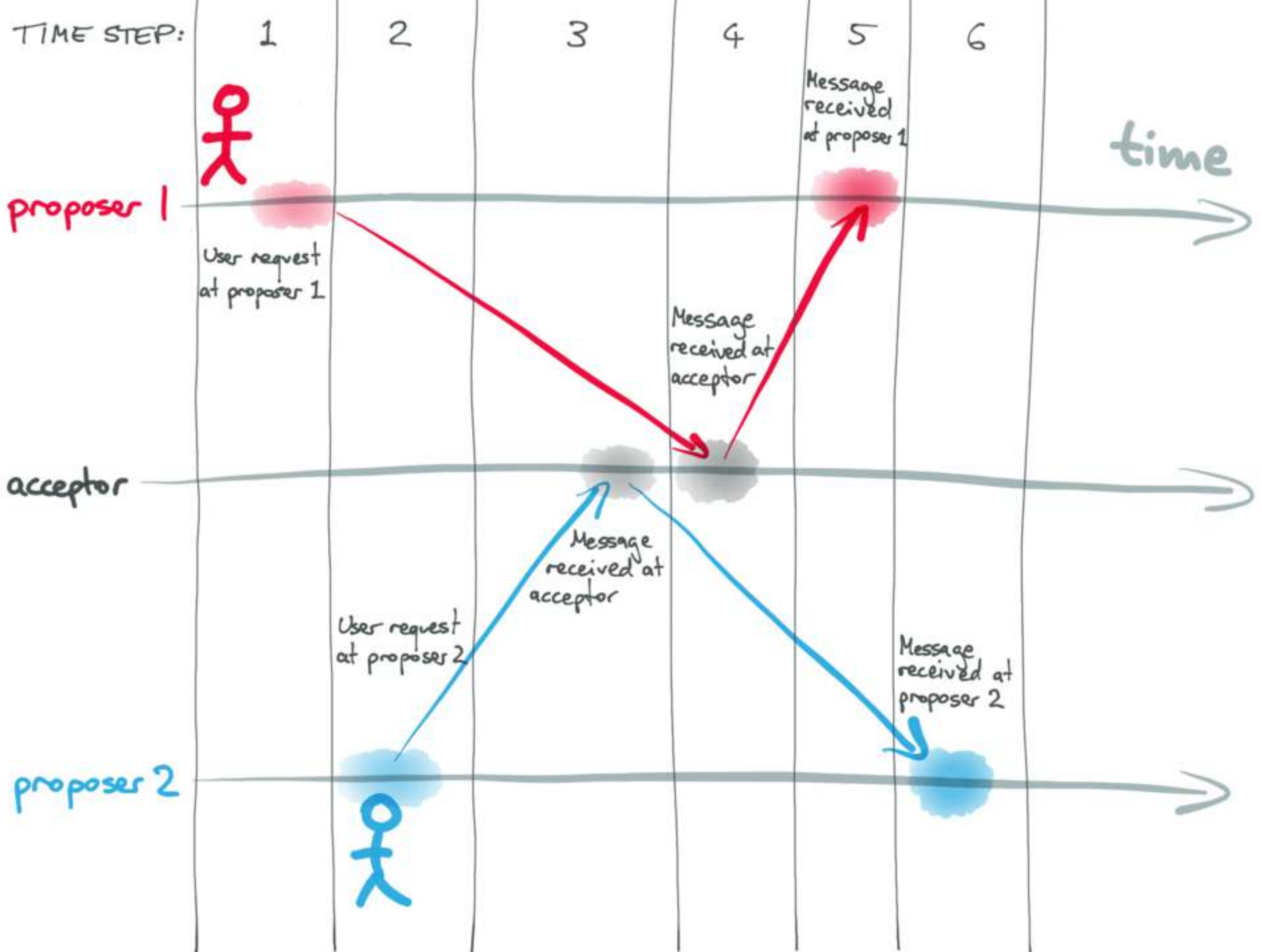

Request
"tacos"
time
proposer 1
Propose
"tacos"
Accept
"falafel"
acceptor
Propose
"falafel"
Accept
"falafel"
proposer 2
Request
"falafel"



Request
"tacos"
decision:
"falafel"
time
proposer 1
Propose
"tacos"
Accept
"falafel"
acceptor
Propose
"falafel"
Accept
"falafel"
proposer 2
Request
"falafel"
decision:
"falafel"


time
proposer 1
acceptor
proposer 2


TIME STEP:
1
2
3
4
5
6
time
proposer 1
acceptor
proposer 2



TIME STEP:
1
2
3
4
5
6
time
proposer 1
acceptor
proposer 2
User request at proposer 1
Message received at acceptor
Message received at proposer 1
User request at proposer 2
Message received at acceptor
Message received at proposer 2

SYSTEM MODEL IN ISABELLE

- Linear sequence of time steps
  (like TLA+)

SYSTEM MODEL IN ISABELLE

- Linear sequence of time steps

  (like TLA+)

- Each step: one process handles event

  (event types: user request, message received, timeout)

SYSTEM MODEL IN ISABELLE

- Linear sequence of time steps
  (Like TLA+)
- Each step: one process handles event
  (event types: user request, message received, timeout)
- Step function type signature:

$$\underbrace{\text{processID}}_{\text{who is executing?}} \Rightarrow \underbrace{\text{state}}_{\text{current local state}} \Rightarrow \underbrace{\text{event}}_{\text{what happened?}} \Rightarrow (\underbrace{\text{state}}_{\text{new local state}} \times \underbrace{\text{msg set}}_{\text{messages to send}})$$

PYTHON	ISABELLE/HOL

`def identity(x):`
`    return x`
	`fun identity where`
	`  ‹identity x = x›`
	OR
	`definition identity where`
	`  ‹identity x ≡ x›`

PYTHON	ISABELLE/HOL

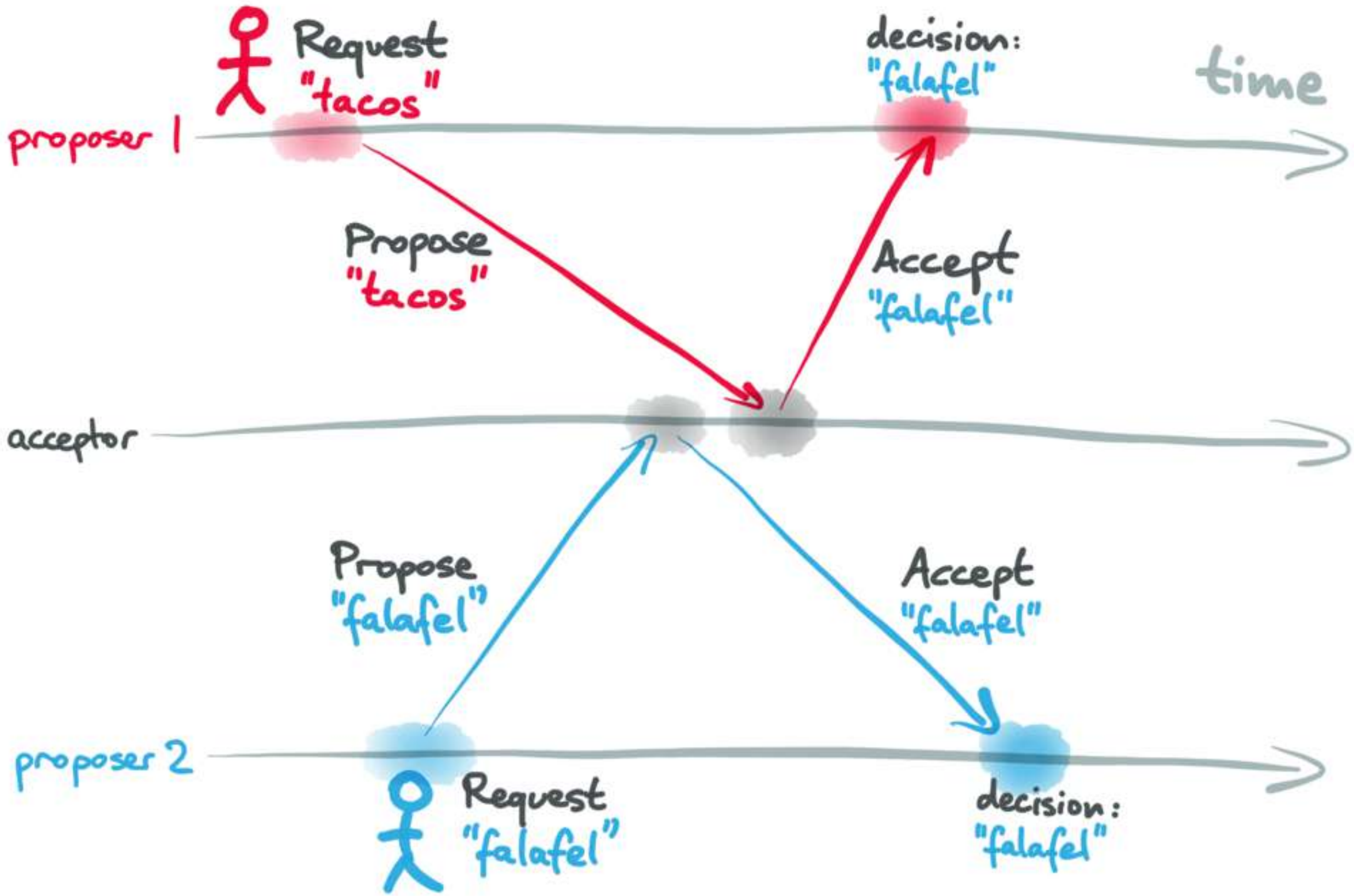
def identity(x):
    return x
	fun identity where
	  ‹identity x = x›
	OR
	definition identity where
	  ‹identity x ≡ x›

lambda x: x	λx. x

PYTHON	ISABELLE/HOL

`def identity(x):`
`    return x`
	`fun identity where`
	`  ‹identity x = x›`
	OR
	`definition identity where`
	`  ‹identity x ≡ x›`

`lambda x: x`	`λx. x`
`identity(3)`	`identity 3`



Request
"tacos"
decision:
"falafel"
time
proposer 1
Propose
"tacos"
Accept
"falafel"
acceptor
Propose
"falafel"
Accept
"falafel"
proposer 2
Request
"falafel"
decision:
"falafel"

STEP FUNCTIONS

PROPOSER:

ON user request:
  send proposed value to ACCEPTOR
ON response from ACCEPTOR:
  learn decided value

ACCEPTOR:

ON proposal received from PROPOSER:
  IF value has been previously decided:
    send value to PROPOSER
  ELSE:
    decide proposed value
    send it to PROPOSER

PROOF ESSENTIALS

Logical implication:

$$\underbrace{P_1 \wedge P_2 \wedge \ldots \wedge P_n}_{\text{assumptions}} \Rightarrow \underbrace{Q}_{\text{consequent}}$$

PROOF ESSENTIALS

Logical implication:

$$\underbrace{P_1 \wedge P_2 \wedge \ldots \wedge P_n}_{\text{assumptions}} \Longrightarrow \underbrace{Q}_{\text{consequent}}$$

... also written as:

$$P_1 \Longrightarrow P_2 \Longrightarrow \ldots \Longrightarrow P_n \Longrightarrow Q$$

THE AGREEMENT PROPERTY

If any two processes learn decided values, those values are the same.

THE AGREEMENT PROPERTY

If any two processes learn decided values, those values are the same.

THEOREM.

Assuming states are the states of all processes after executing any number of steps of the consensus algorithm

and states proc1 = Some val1
and states proc2 = Some val2

for any proc1, proc2

then we prove that val1 = val2.

INVARIANT 1:

For any proposer p, if p's state is Some val, then there exists a process a that has sent a message Accept val to p.

INVARIANT 1:

For any proposer p, if p's state is Some val, then there exists a process a that has sent a message Accept val to p.

INVARIANT 2:

If a message Accept val has been sent, then the acceptor is in the state Some val.

PROOF ESSENTIALS

$\forall x.\ P(x)$

for all values of x, the statement $P(x)$ is true

PROOF ESSENTIALS

$\forall x.\ P(x)$ — for all values of x, the statement $P(x)$ is true

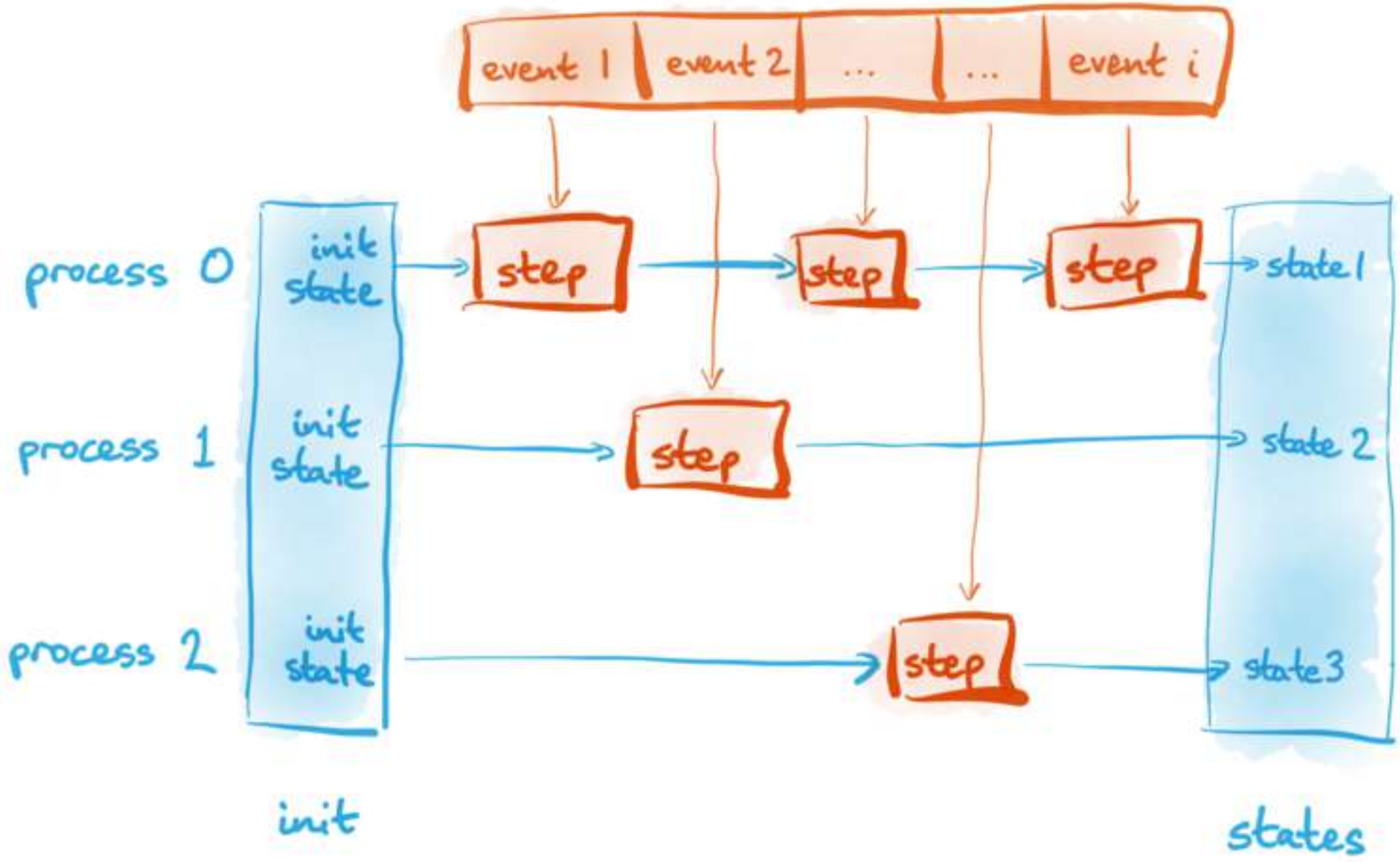
$\exists x.\ P(x)$ — there exists some value x for which the statement $P(x)$ is true

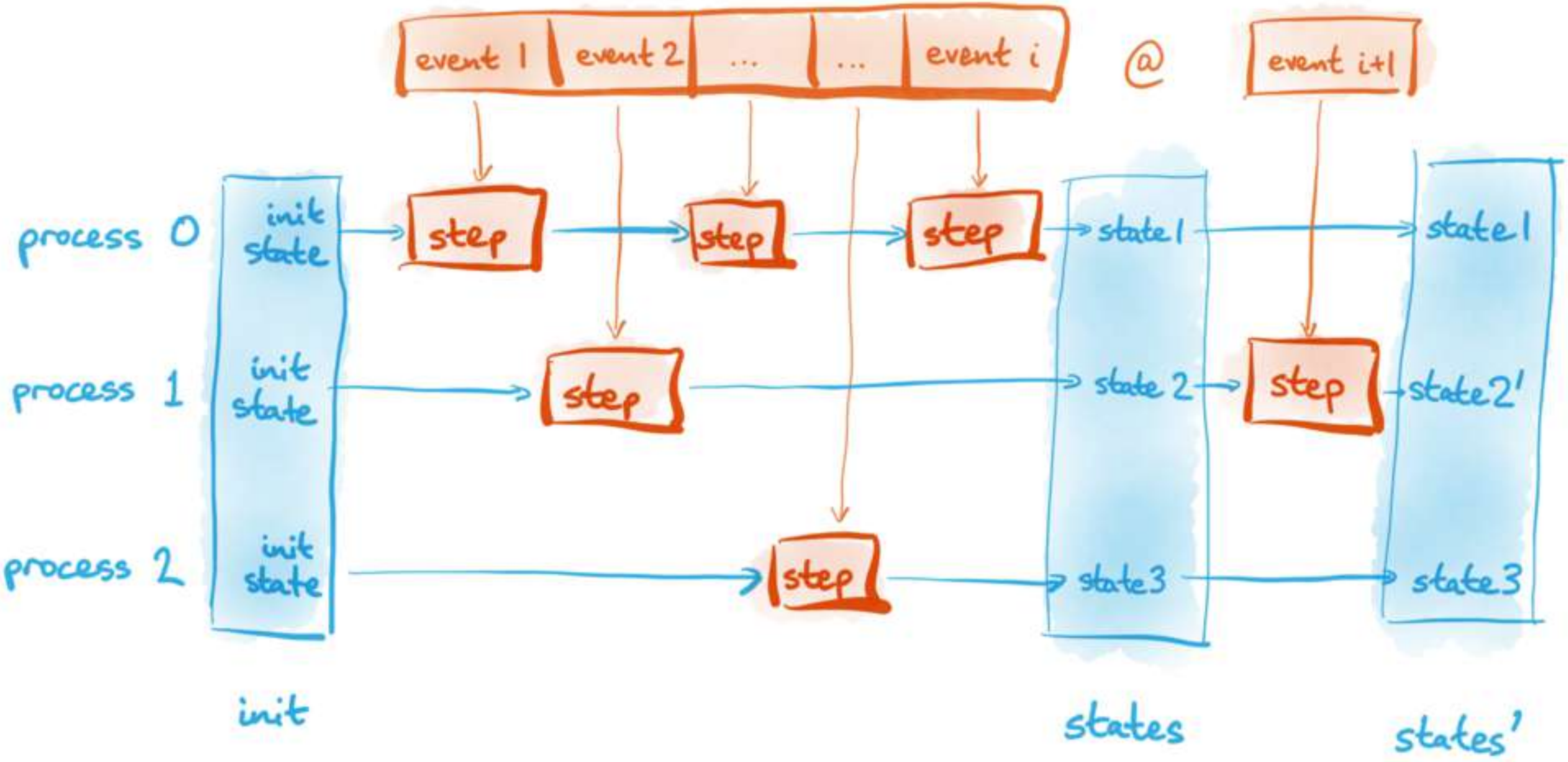
INVARIANT 1:

For any proposer p, if p's state is Some val, then there exists a process a that has sent a message Accept val to p.

INVARIANT 2:

If a message Accept val has been sent, then the acceptor is in the state Some val.


event 1
event 2
...
...
event i
process 0
process 1
process 2
init state
init state
init state
step
step
step
step
step
state 1
state 2
state 3
init
states



event 1
event 2
...
...
event i
@
event i+1
process 0
process 1
process 2
init state
init state
init state
step
step
step
step
step
step
state 1
state 2
state 3
state 1
state 2'
state 3
init
states
states'

PROOF TECHNIQUE

Induction on lists!

If we have $P([\,])$

and also $P(xs) \Rightarrow P(xs @ [x])$

then $P(xs)$ for all lists xs

PROOF TECHNIQUE

Induction on lists!

BASE CASE

If we have $P([\,])$

INDUCTIVE STEP

and also $P(xs) \Rightarrow P(xs @ [x])$

then $P(xs)$ for all lists xs

PROOF TECHNIQUE

Induction on lists!

BASE CASE

If we have $P([])$

INDUCTIVE STEP

and also $P(xs) \Rightarrow P(xs @ [x])$

then $P(xs)$ for all lists xs

Finite amount of proof effort, even though set of lists is infinite!

Full proof at:

https://martinkl.com/agree

Thanks!

Martin Kleppmann

@martinkl