Testing Eventual Consistency in

Ulf Norell, John Hughes
Quviq AB
(with Scott Lystig Fritchie, Jon Meredith
Dave Smith)

- Distributed
- Scalable
- Replicated
- Fault-tolerant
- High availability
- Low latency
- No SQL—just keys and values

How can we be sure Riak is correct?

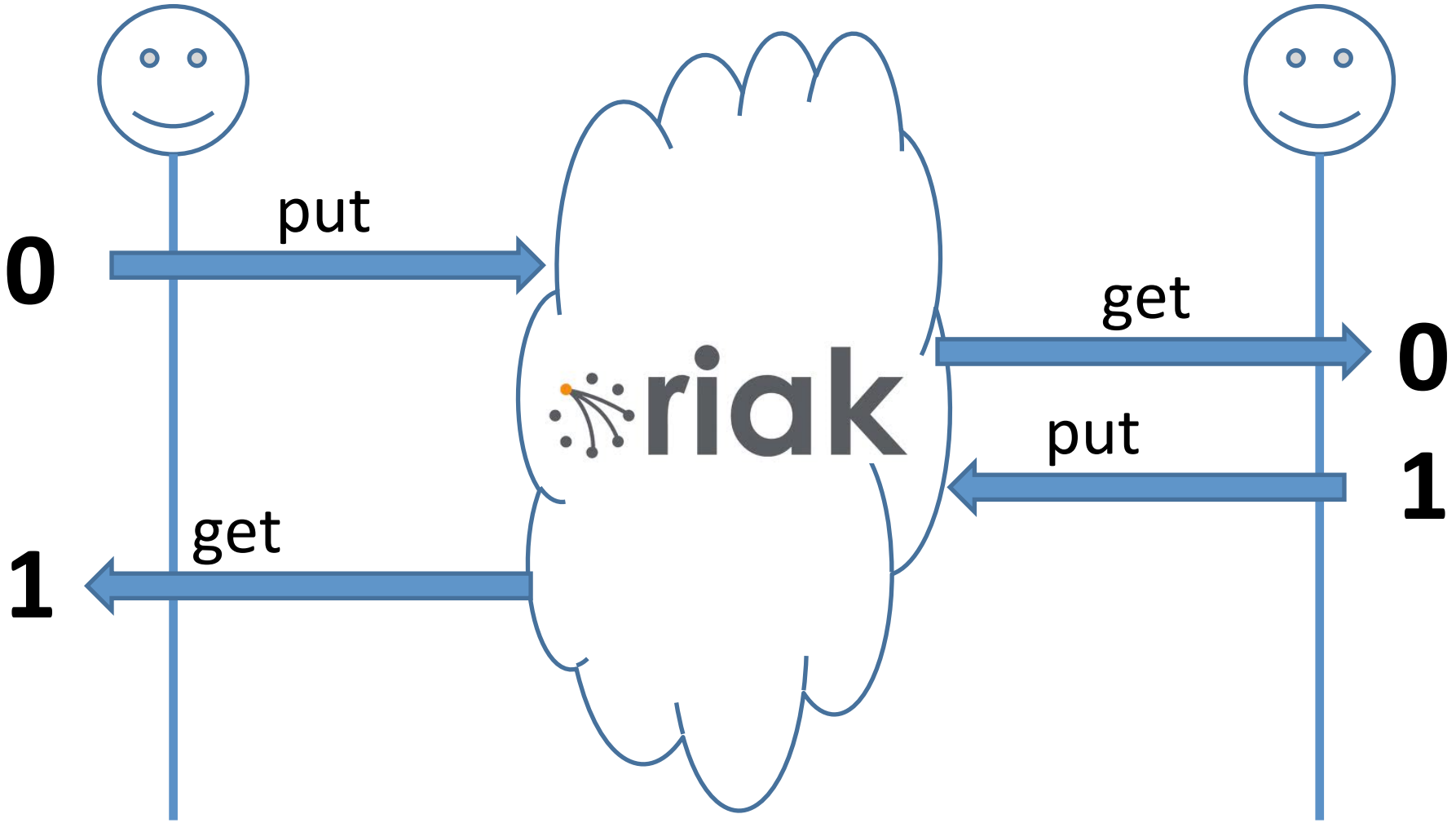
What does that even mean?

QuickCheck!

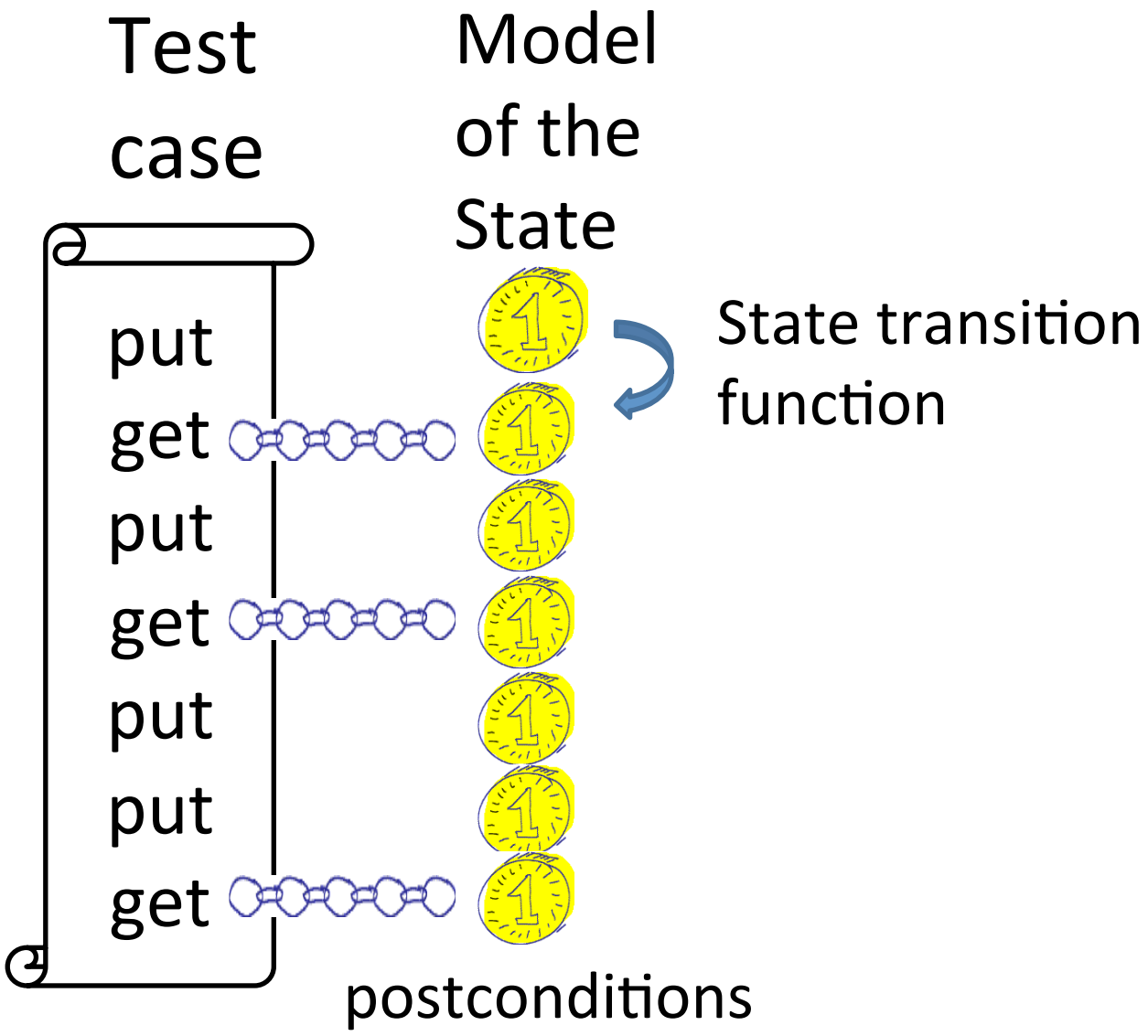
Keep It Simple!

one  one 

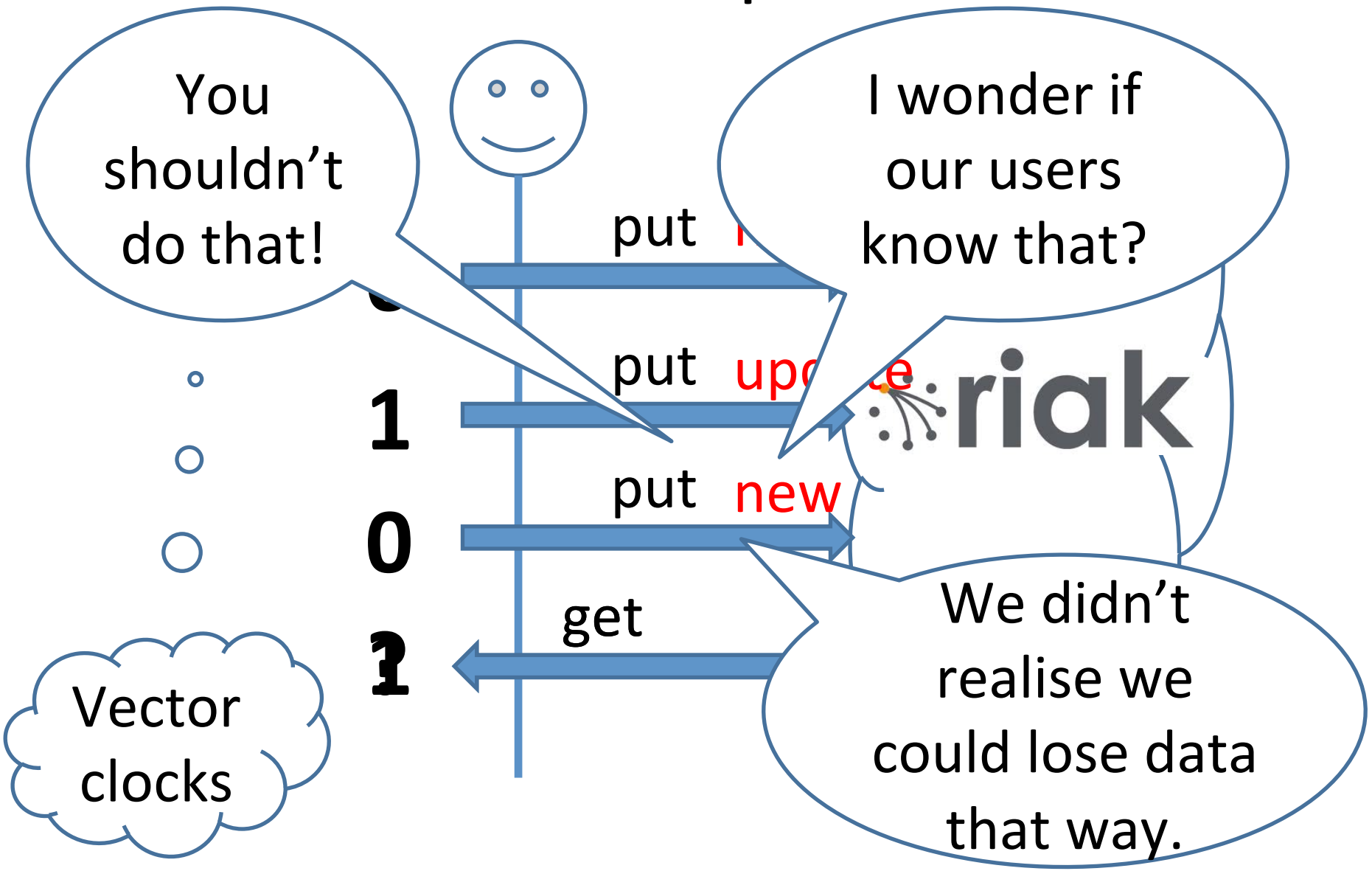
Put and Get



Quick Check


Test case
Model of the State
put
get
put
get
put
put
get
State transition function
postconditions

Example


You shouldn't do that!
I wonder if our users know that?
put
put
riak
1
put new
0
get
1
We didn't realise we could lose data that way.
Vector clocks

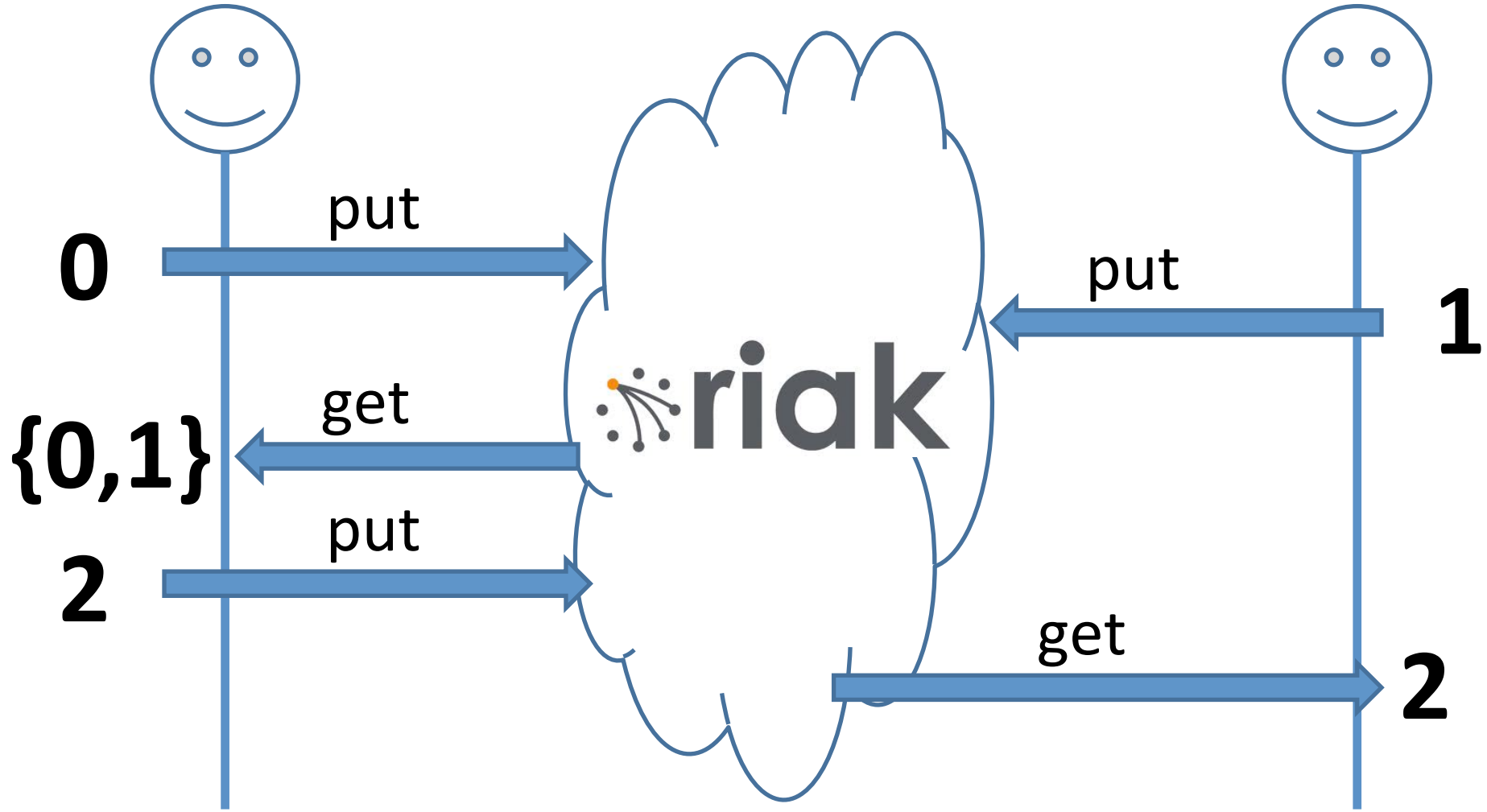
Should we put vector clocks into the model?

NO!!

Formalise ”You shouldn’t do that!”

- Add to the model:
  - Client’s last view of the value (result of get)
- Add a *precondition:*
  - Every put must update the client’s view (if present)
- QuickCheck generates tests respecting the precondition

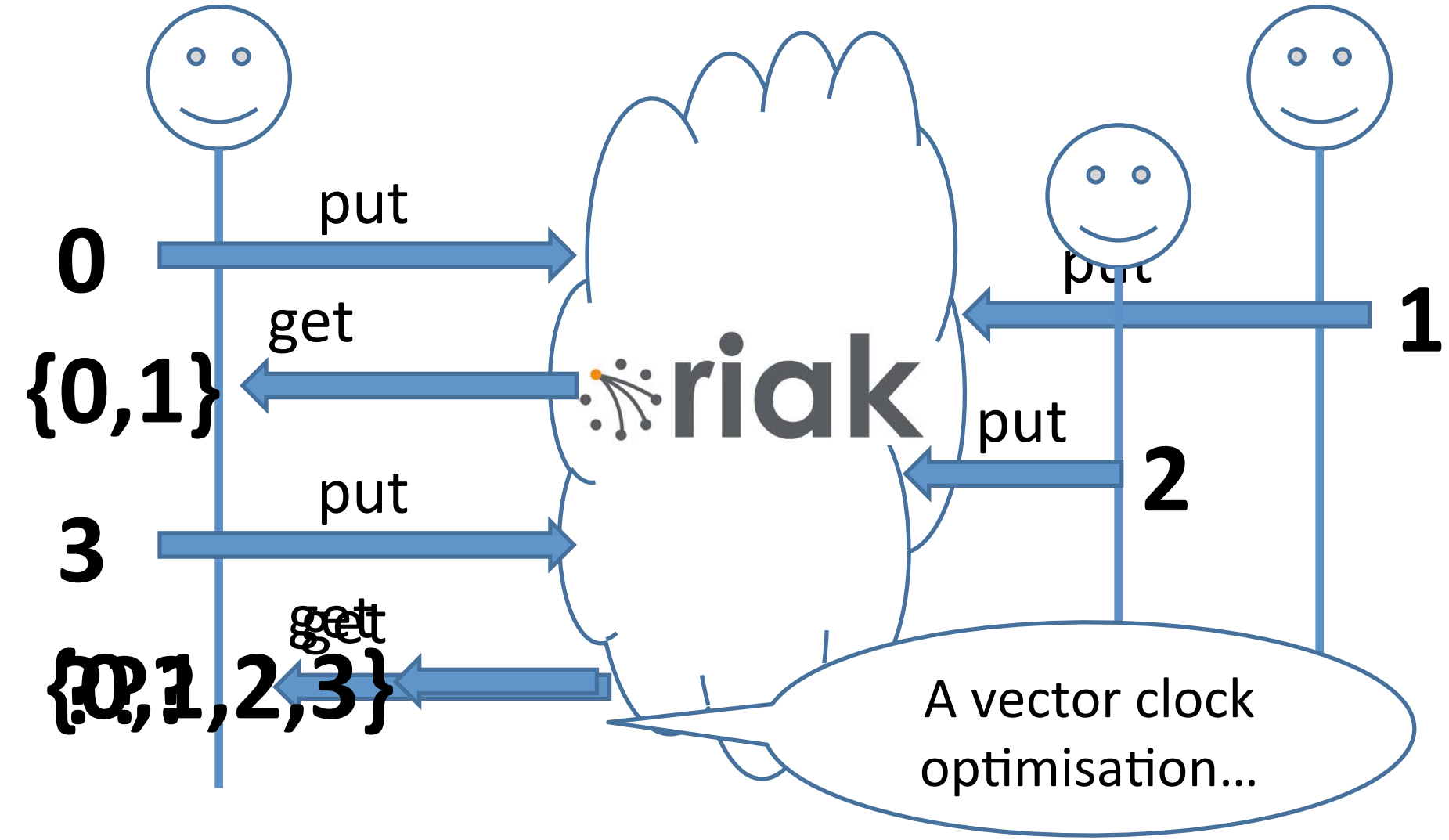
Conflicts



Modelling Conflicts

- The state is a *list* (actually bag) of values
- The client's view is a list of values
- put replaces those values in the state

Example


put
0
get
{0,1}
riak
put
1
put
2
put
3
get
{0,1,2,3}
A vector clock optimisation…

Modelling Conflicts

- The state is a *list* (actually bag) of values

  ”fresh” or ”stale”

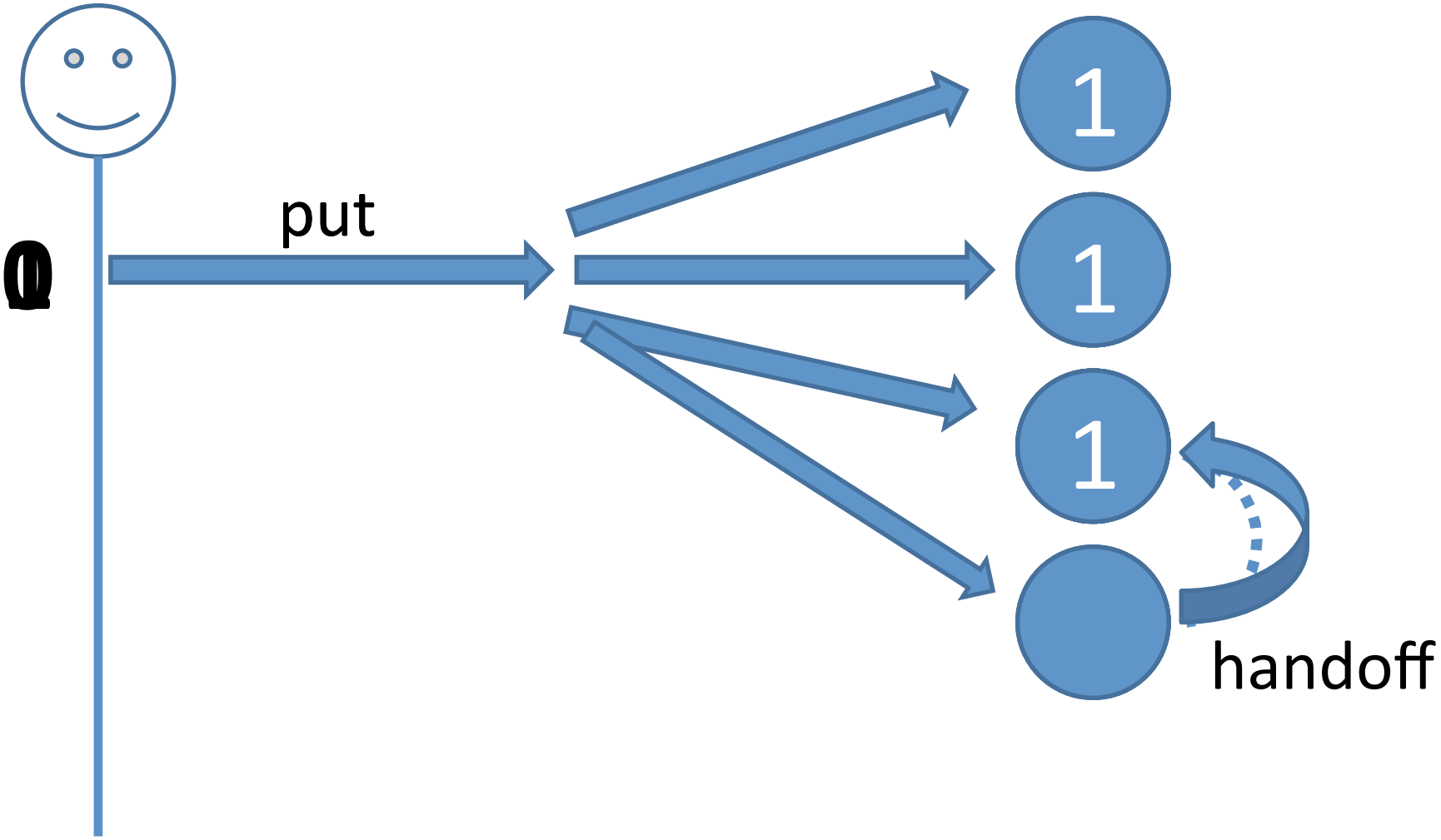
- The client’s view is a list of values

- put replaces those values in the state

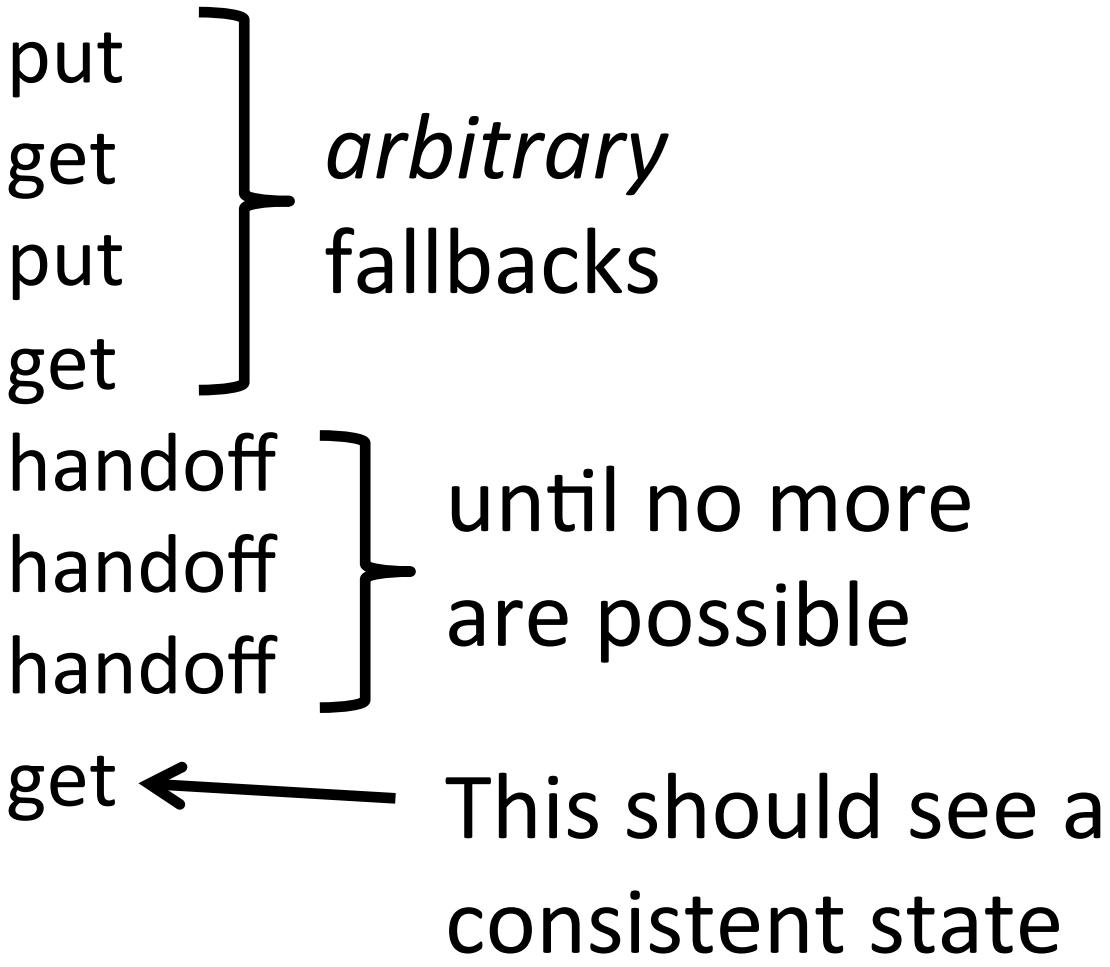
  the state if the client’s view was fresh

  adds a conflict if the client’s view was stale

Redundancy and Fault Tolerance



Testing Eventual Consistency

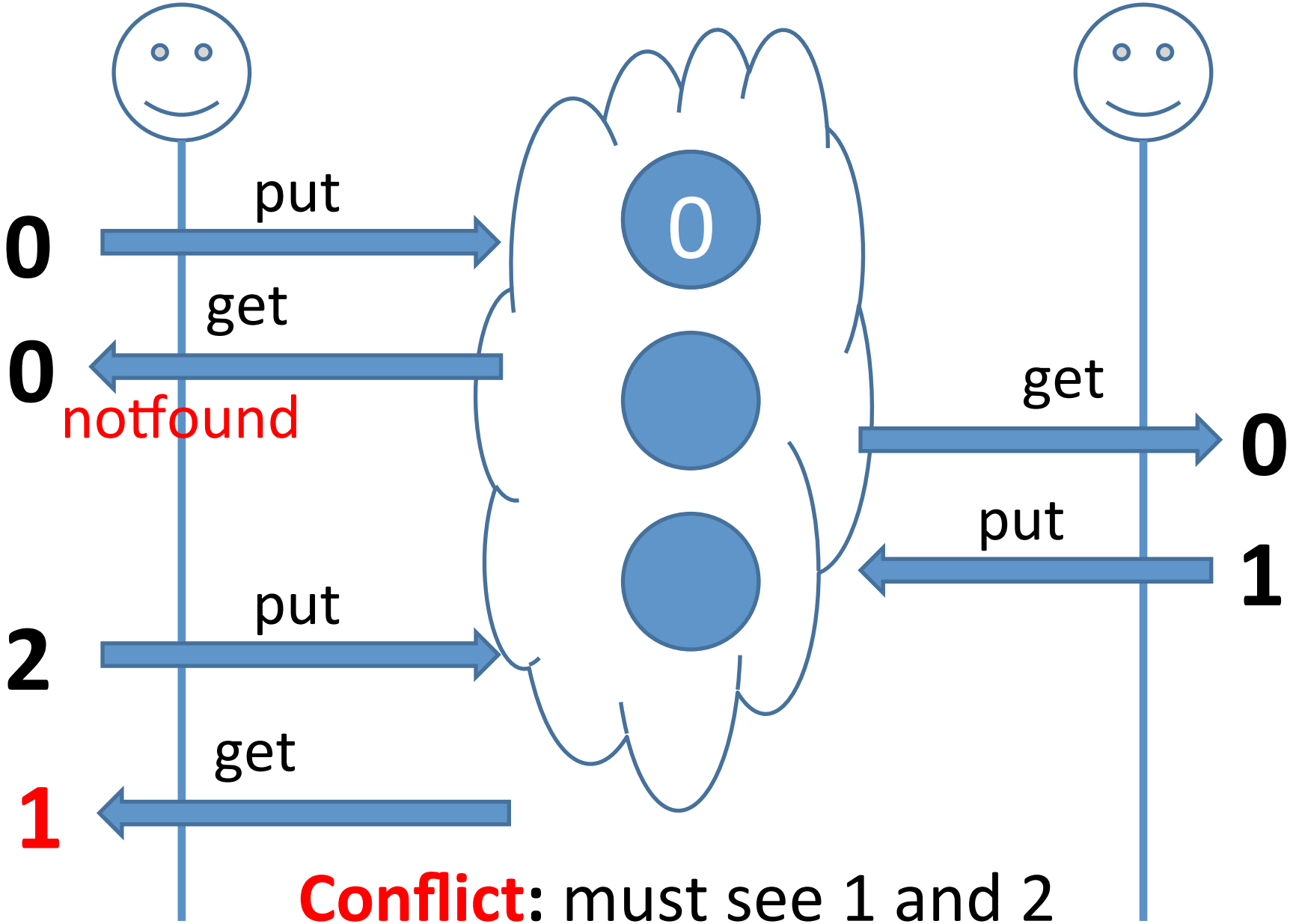


Should we model the *locations* of values?

NO!!

Modelling Eventual Consistency

- A value ***may*** appear in the final get, if it was ever put
- A value ***must*** appear in the final get, if it was put, and never replaced

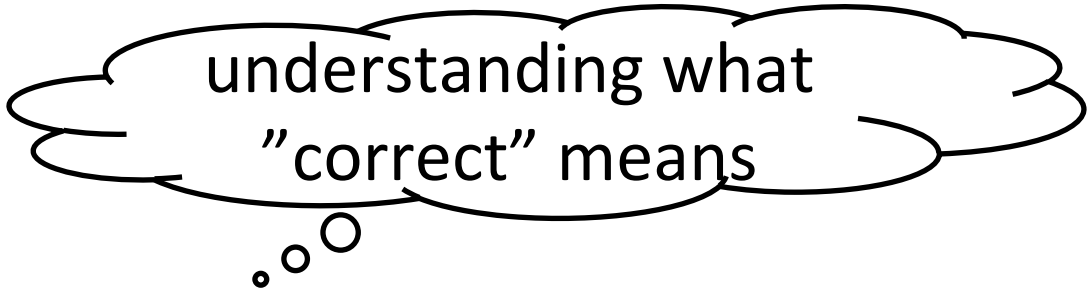

put
0
get
0
notfound
0
get
0
put
1
put
2
get
1
Conflict: must see 1 and 2

SHOCK!!

HORROR!

Riak ~~is~~ was not eventually consistent!

QuickCheck…



- …let us gradually develop a specification, validated against the implementation
- …revealed potentially serious bugs
- …guided the development of alternative solutions that fix the problems

GOOD NEWS!!

Riak *IS* eventually consistent!

—thanks to QuickCheck ☺