Joint work with Hernan Melgratti & Christian Roldan

On the semantics of replicated data types

Fabio Gadducci
University of Pisa

The quickest background, I

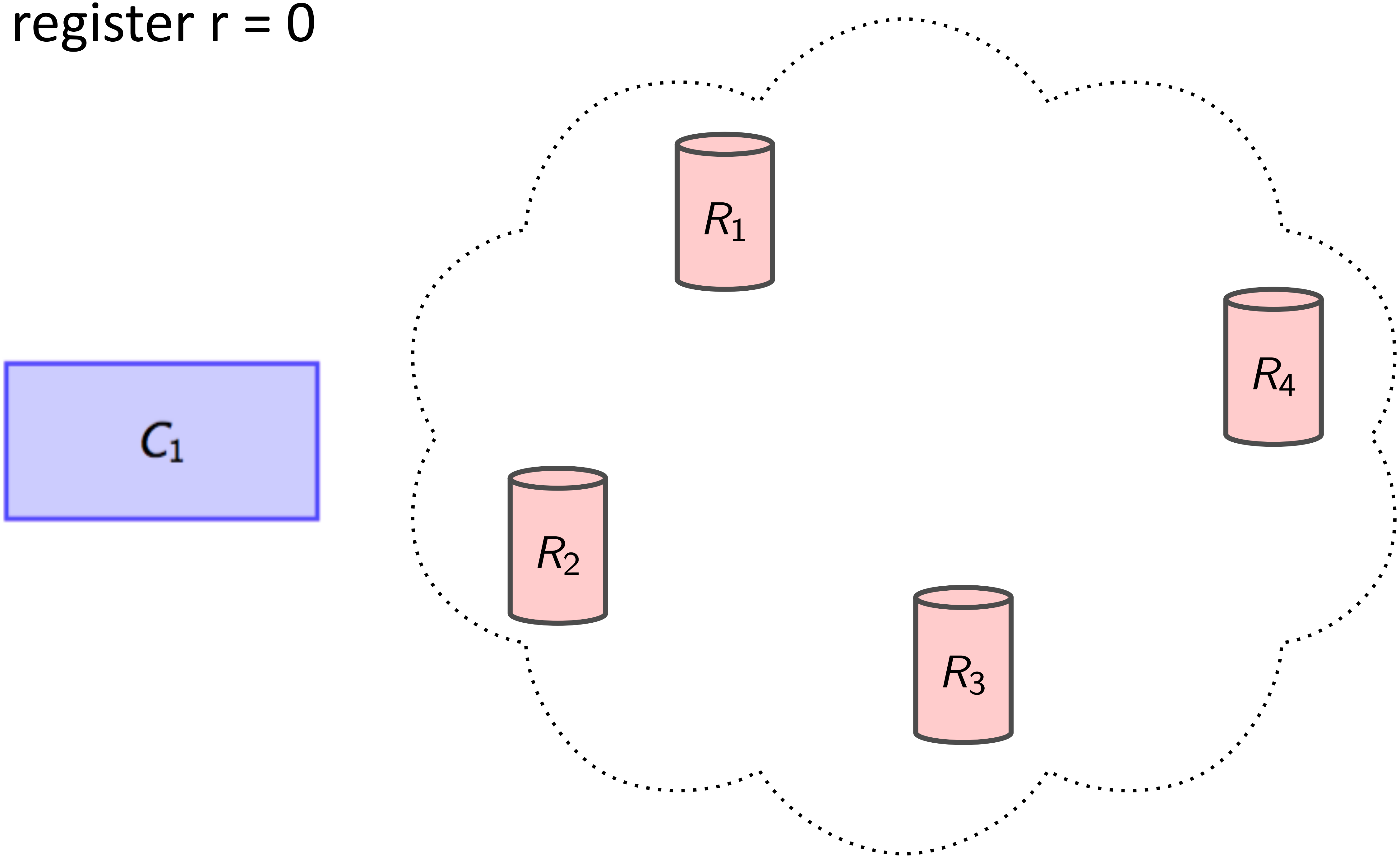
- Distributed systems replicate their state over different nodes in order to satisfy non-functional requirements.
- *Strong consistency* (every request receives the most recent update) of replicated data is in conflict with *availability* (every request is eventually executed) and tolerance to network *partitions* (the system operates even in the presence of failures that temporarily prevent communication among components).
- CAP theorem: it is impossible to simultaneously achieve strong Consistency, Availability and Partition tolerance [GL2002].

The quickest background, II

- *Weak* consistency: replicas may (temporarily) exhibit discrepancies (every request receives a correct update).
- How are the data specified? States, state transitions and returned values should account for the different views that a data item may simultaneously have.
- In the end, consistency has to be *eventually* guaranteed (if no new updates are made to a data item, eventually all accesses to that item will return the most recent update).

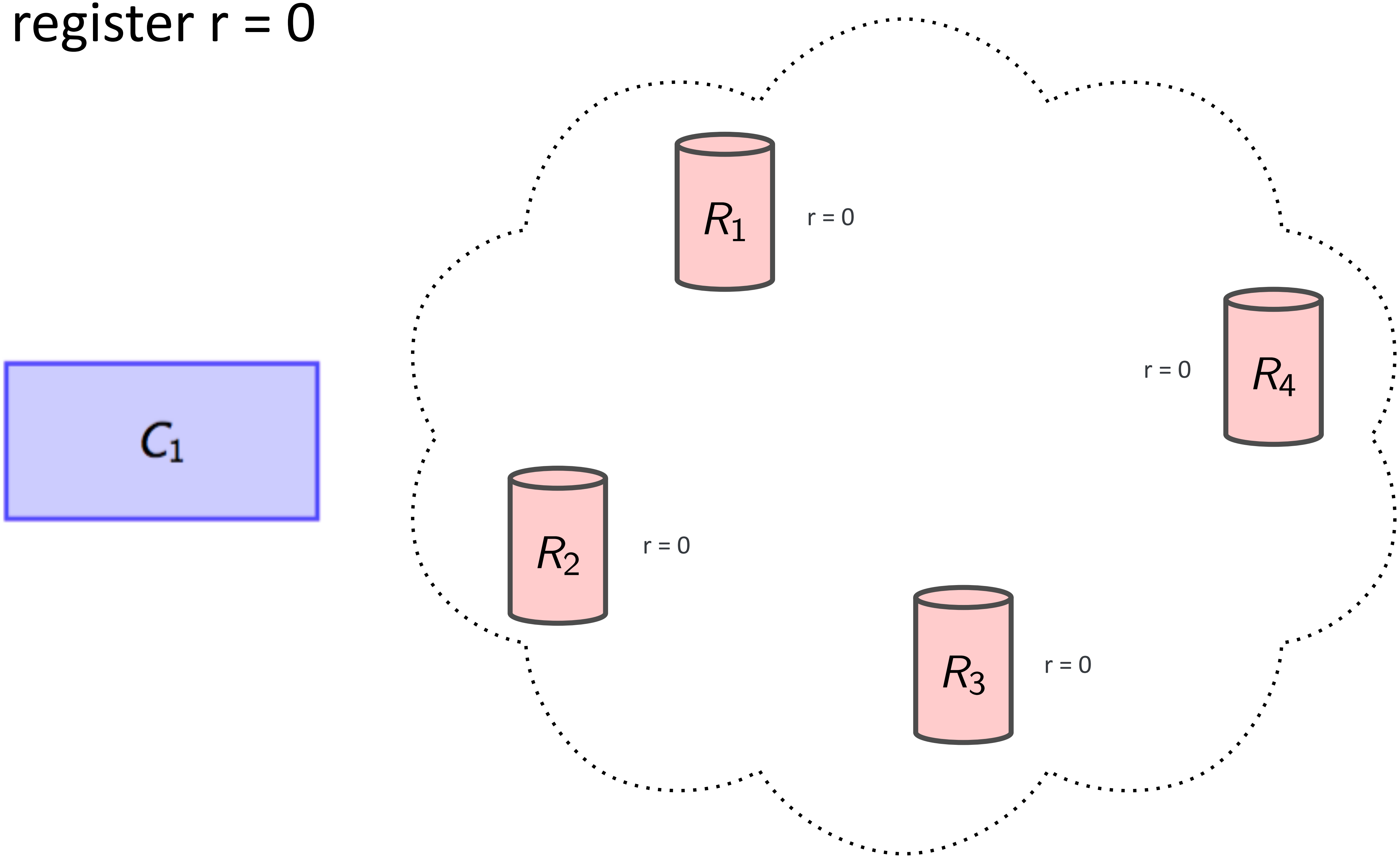
A register

register r = 0



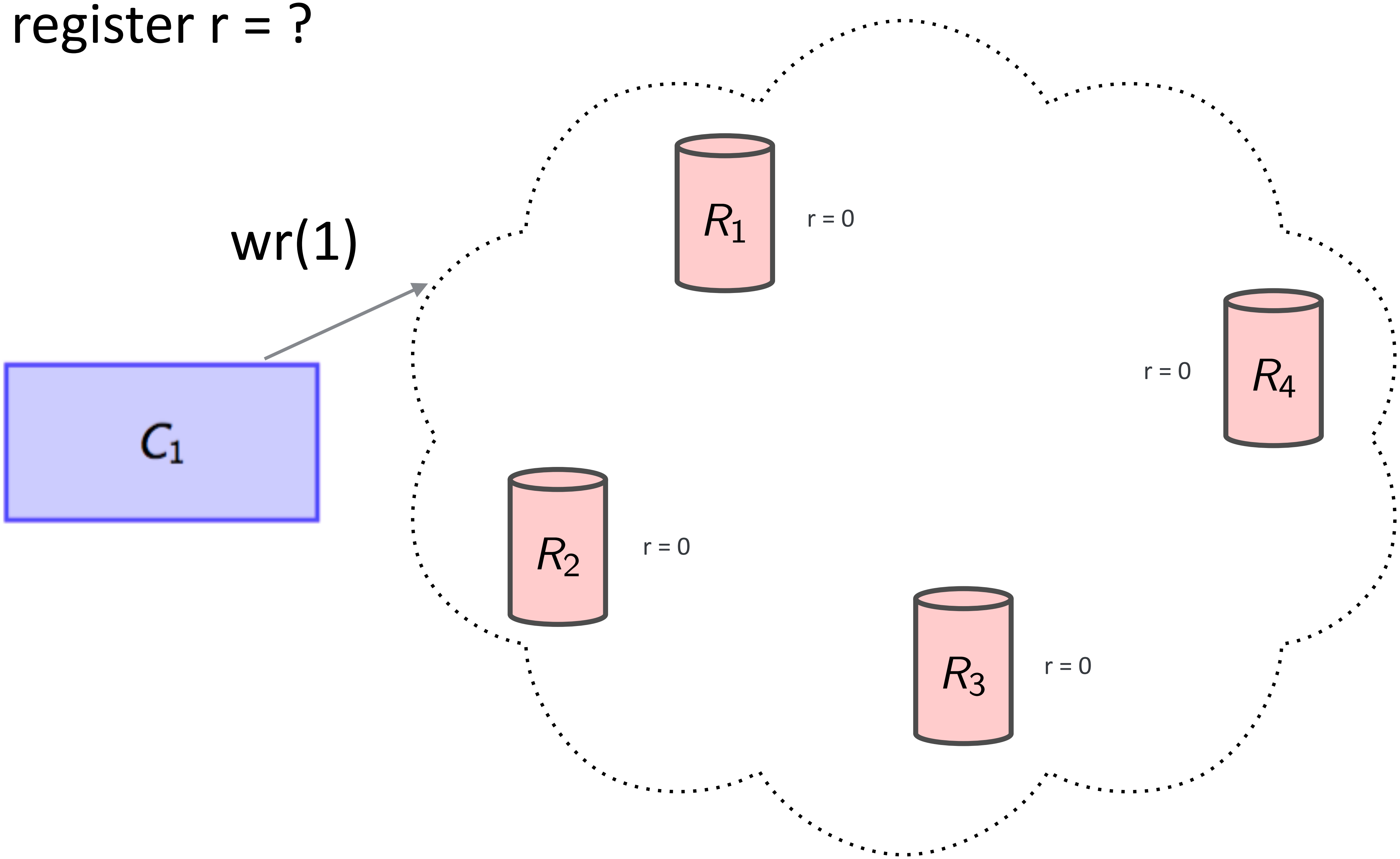
A register

register r = 0

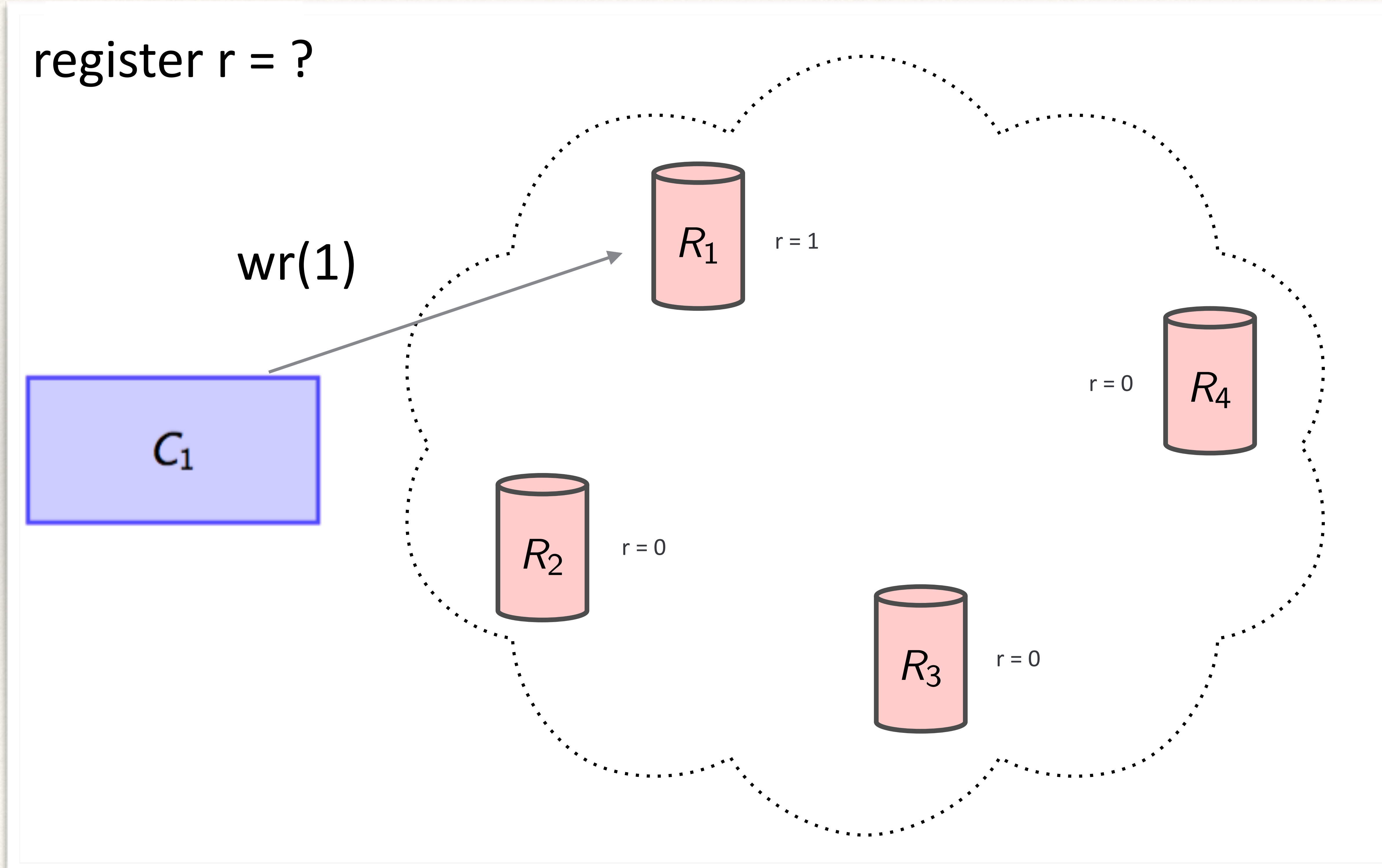


A register

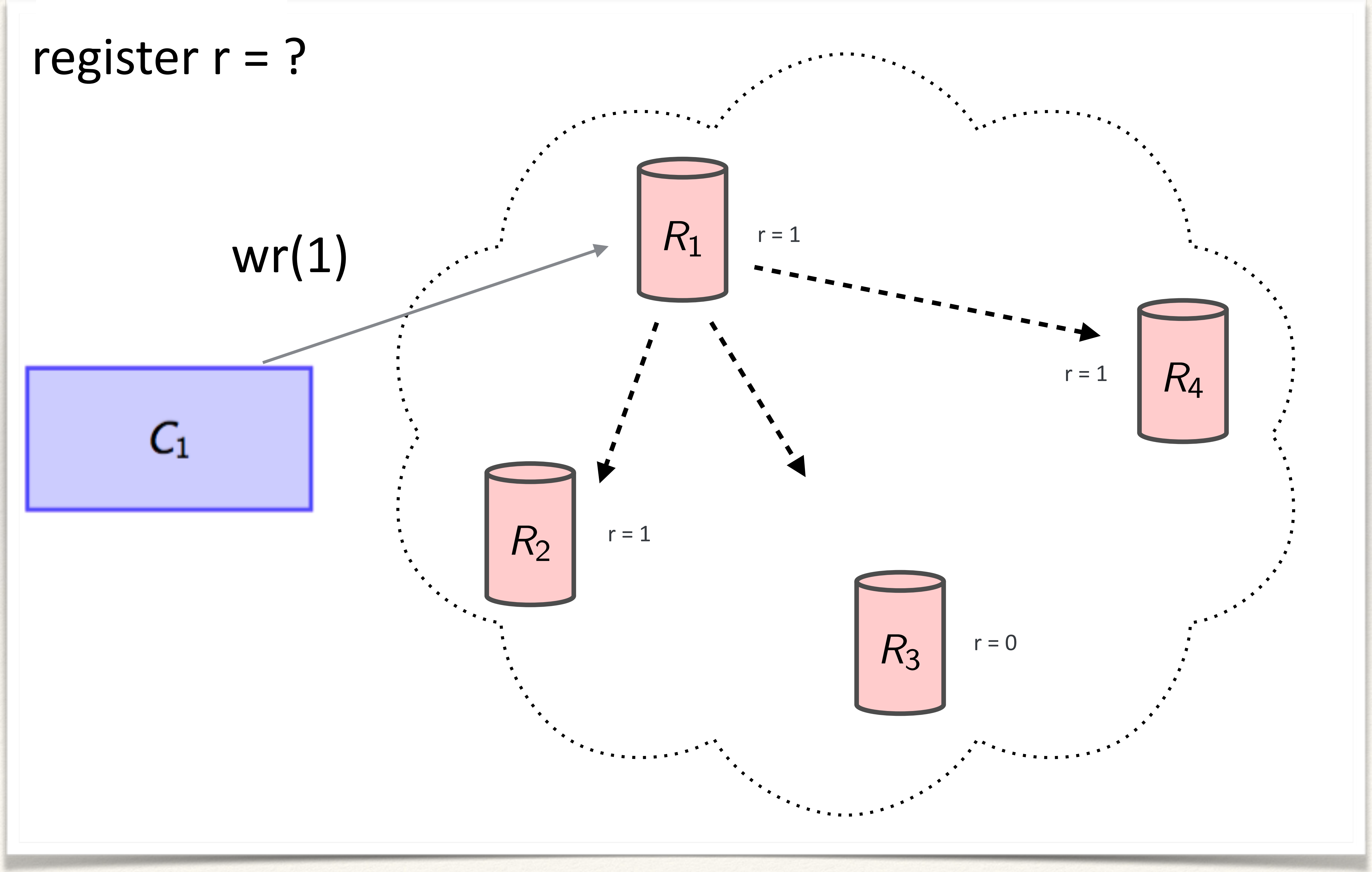
register r = ?



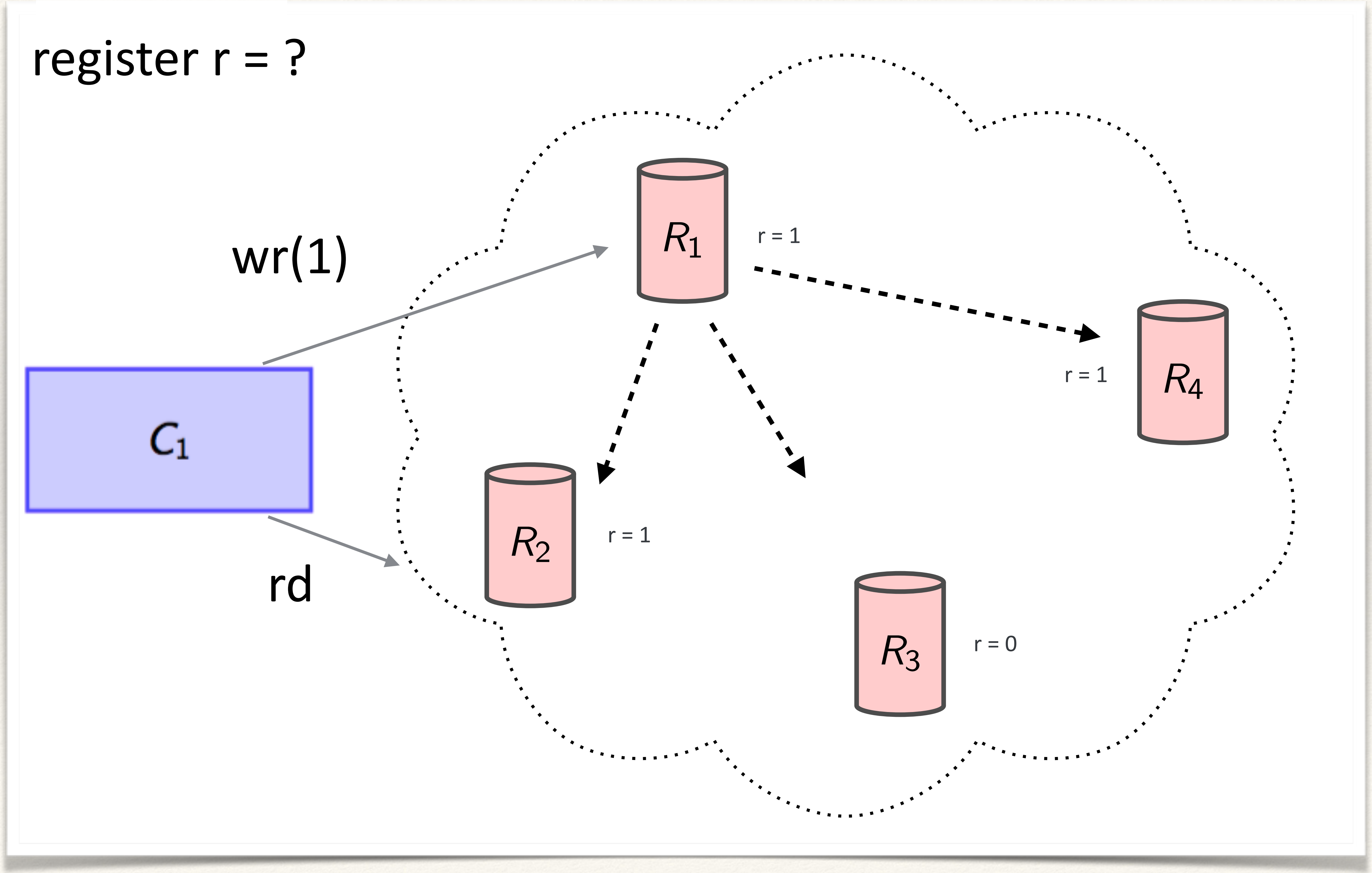
A register



A register

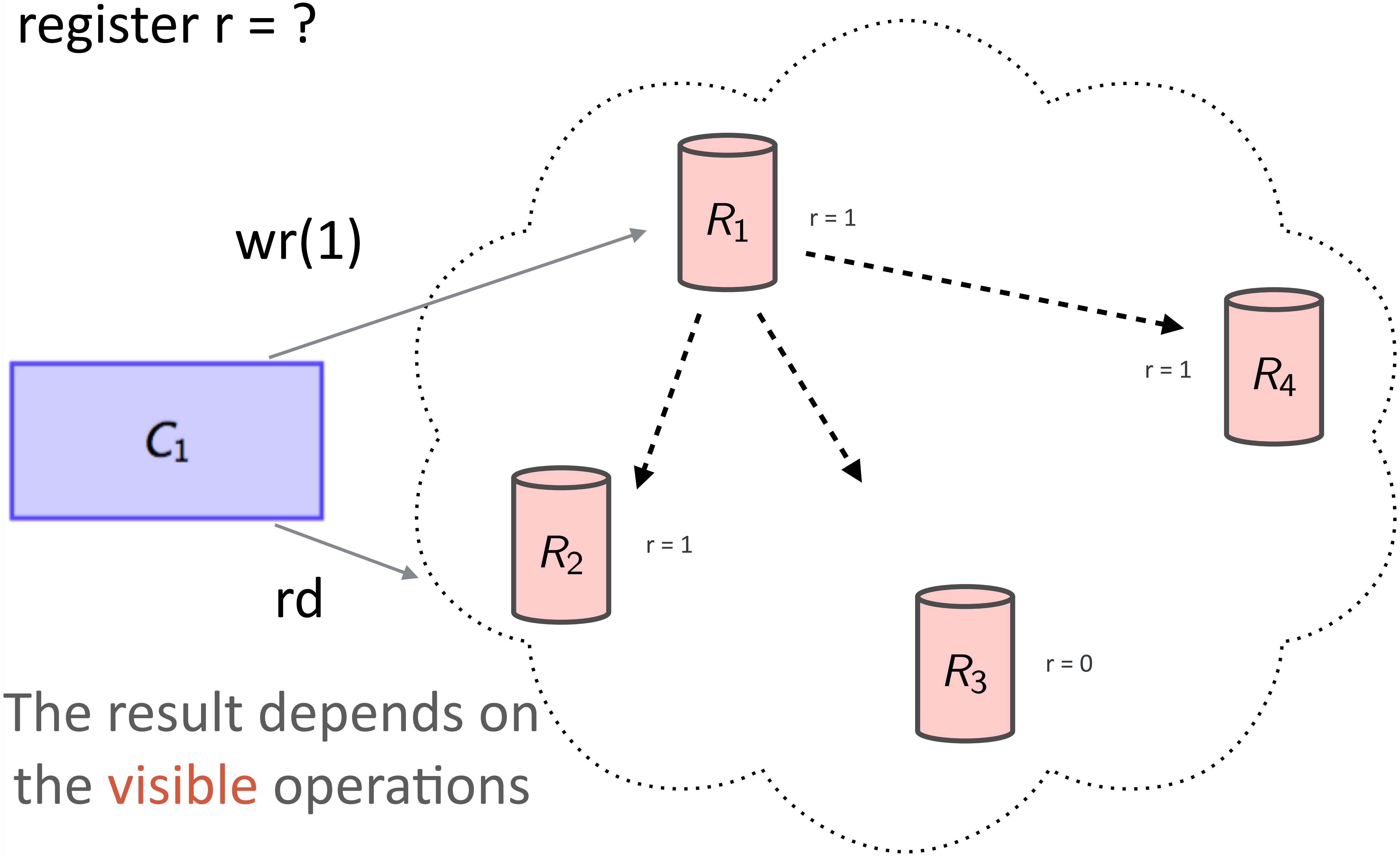


A register



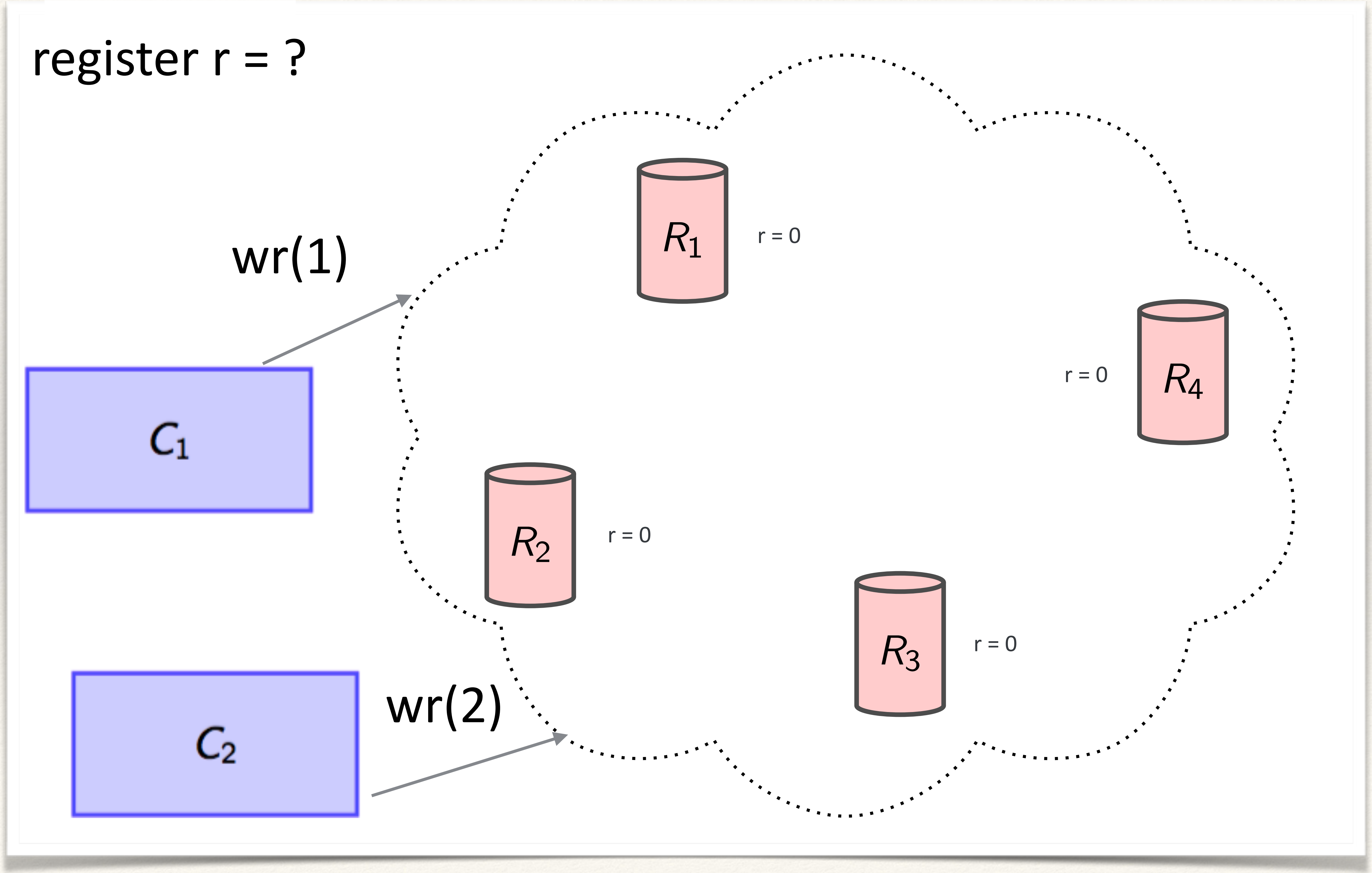
A register

register r = ?



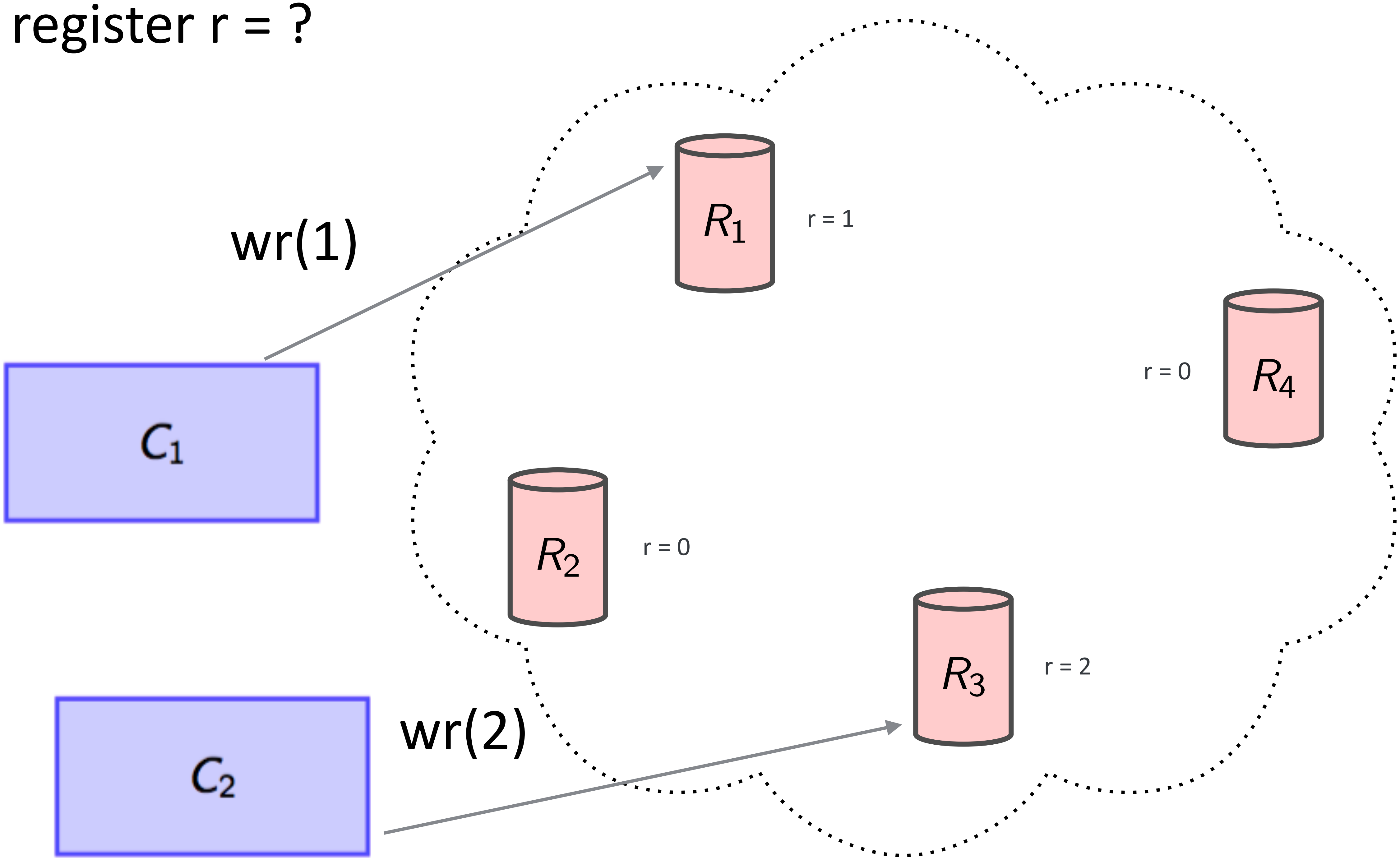
The result depends on the visible operations

A register

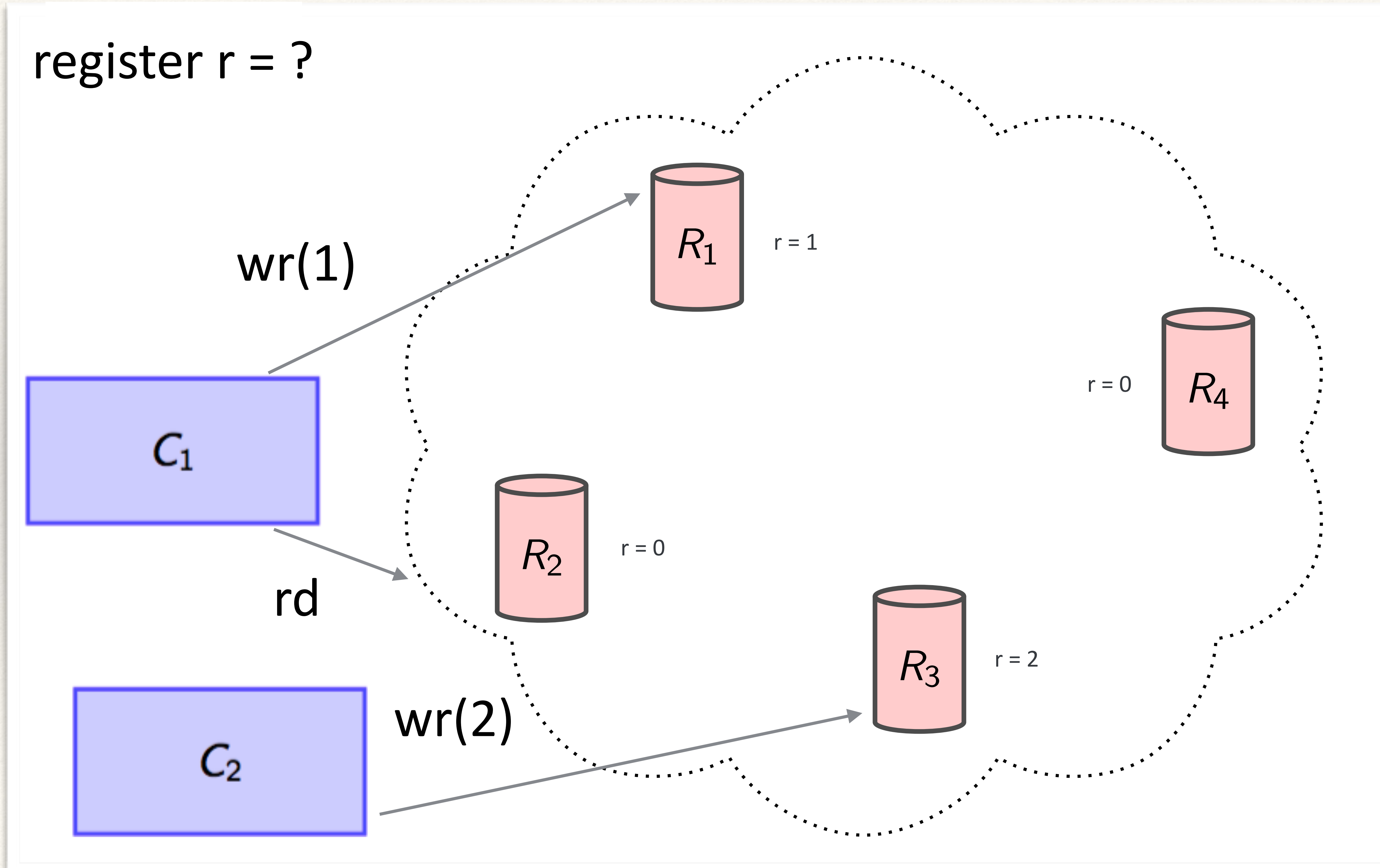


A register

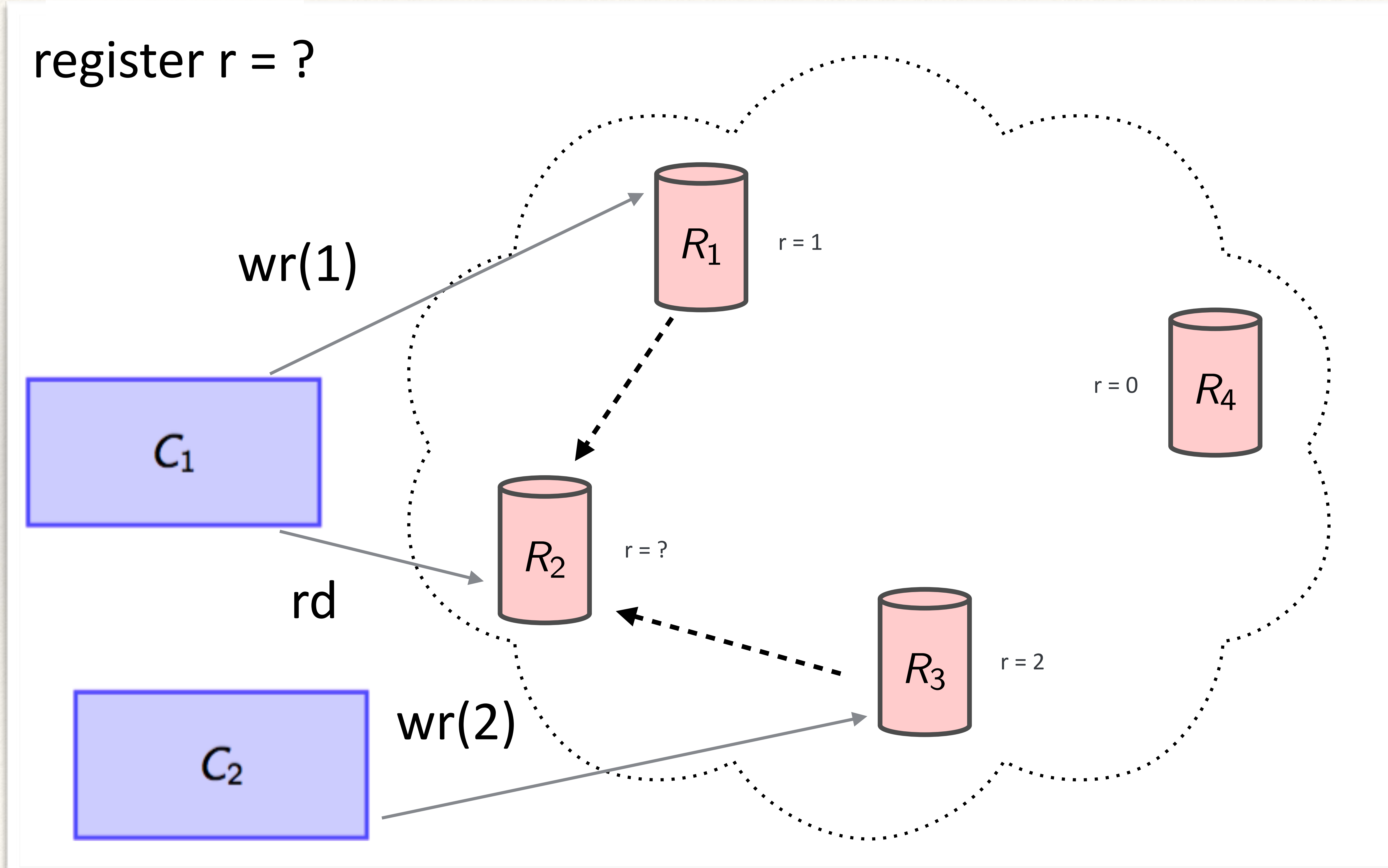
register r = ?



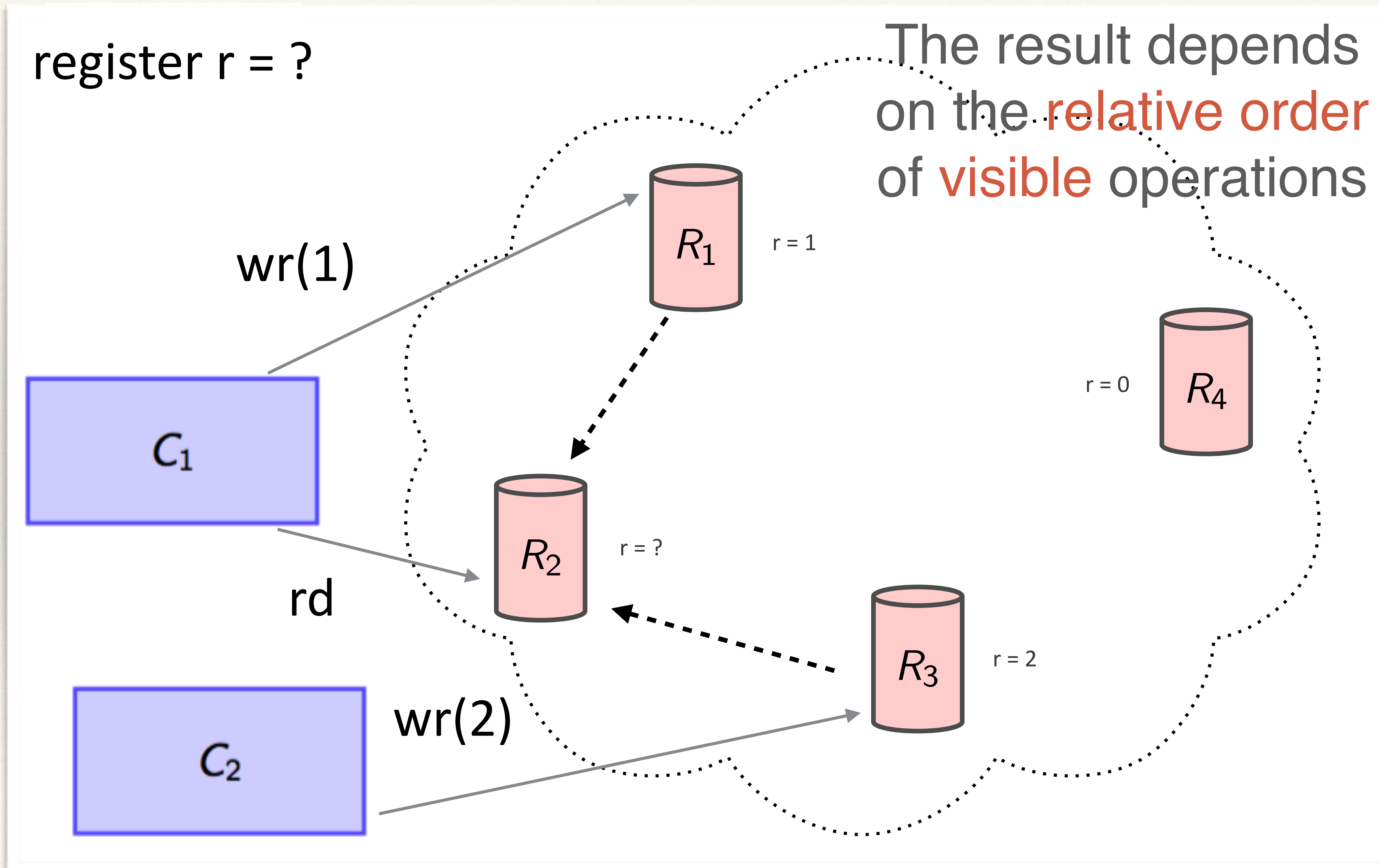
A register



A register



A register


register r = ?
The result depends on the relative order of visible operations
wr(1)
rd
wr(2)
C_1
C_2
R_1
r = 1
R_2
r = ?
R_3
r = 2
R_4
r = 0

Replicated Data Types

op : **VIS × ARB → RVAL**

- **VIS**ibility: A partial order of operations over a replica
- **ARB**itration: A total order of such operations
- **R**eturn **VAL**ue: The value returned by the last operation

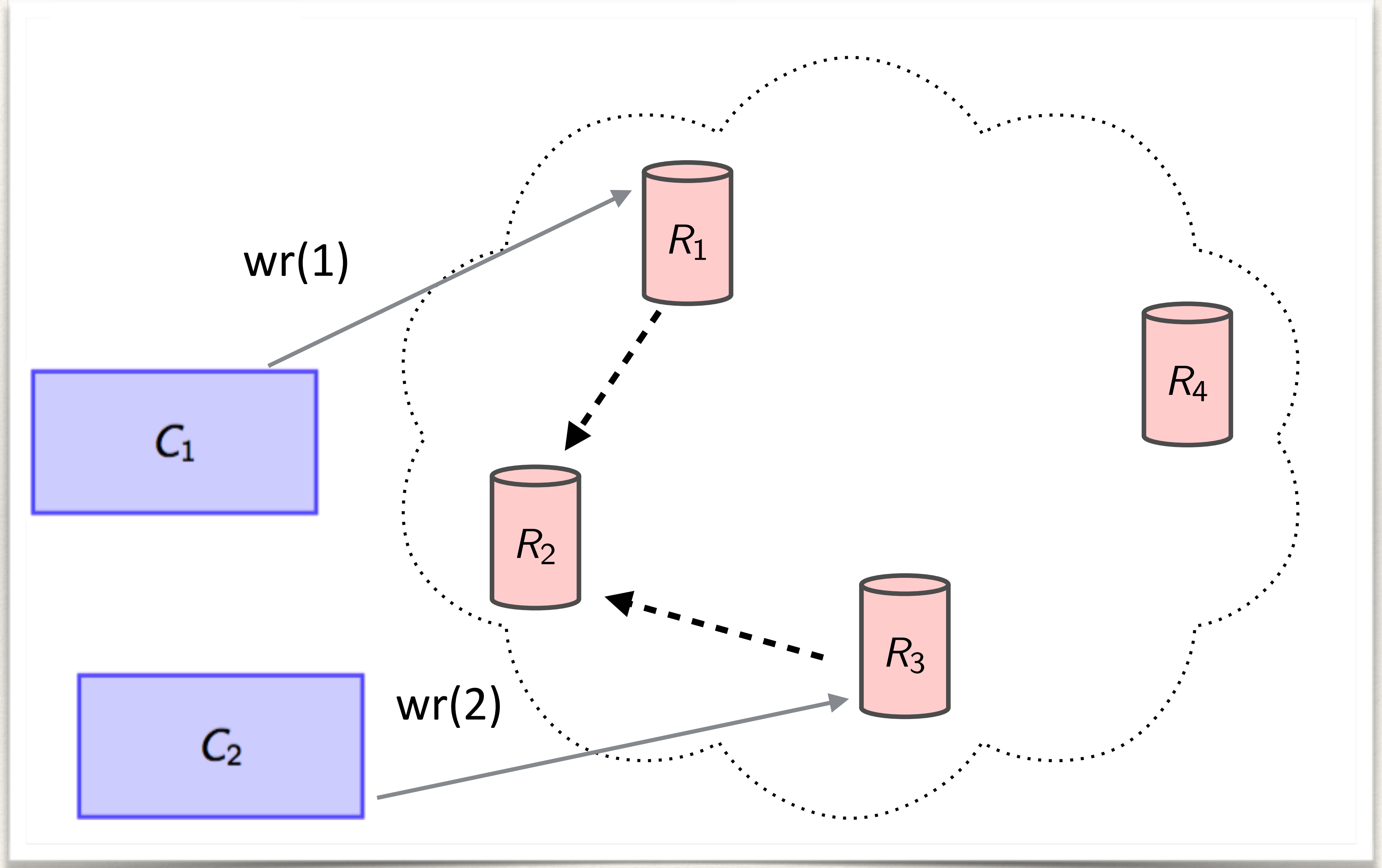
[BURCKHARDT, GOTSMAN,YANG, ZAWIRSKI 2015]

A register

- Two operations
  - rd(_,_) = ?
  - wr(k)(_,_) = ok

A register

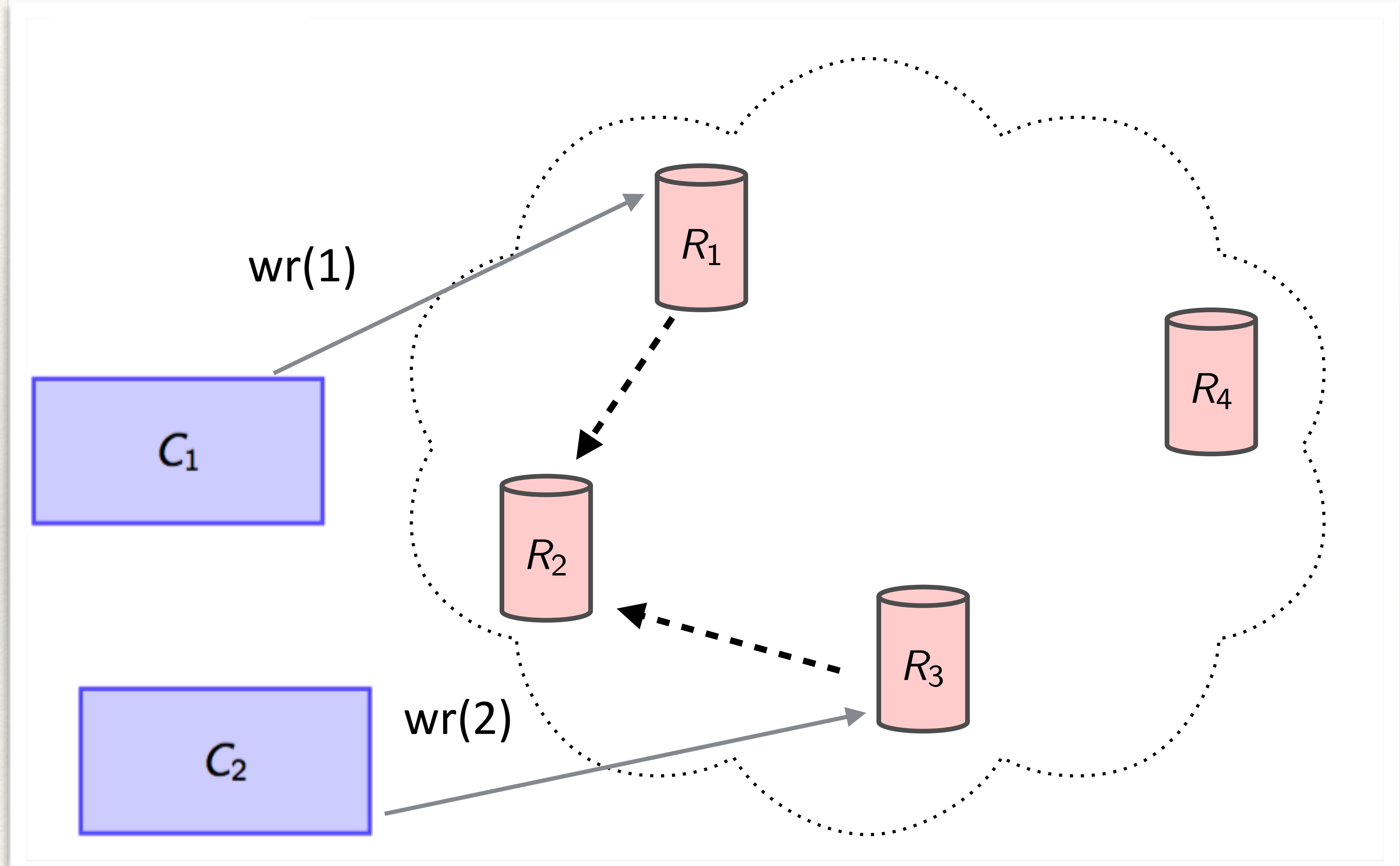
- Two operations
  - rd(_,_) = ?
  - wr(k)(_,_) = ok



A register

$\mathtt{wr}(1)$ $\mathtt{wr}(2)$

VISibility



A register

`wr(1)` `wr(2)`

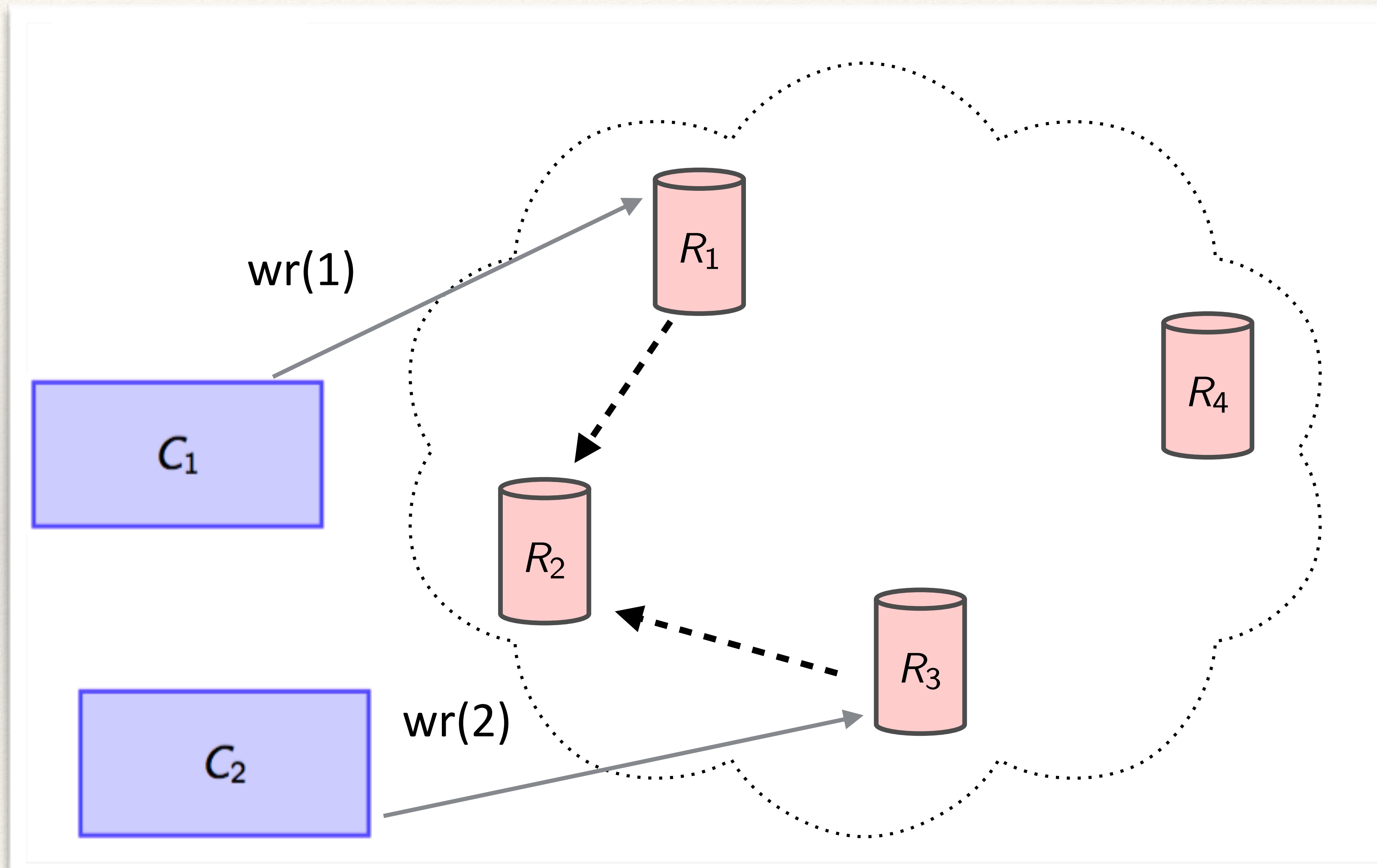
VISibility

`wr(1)`

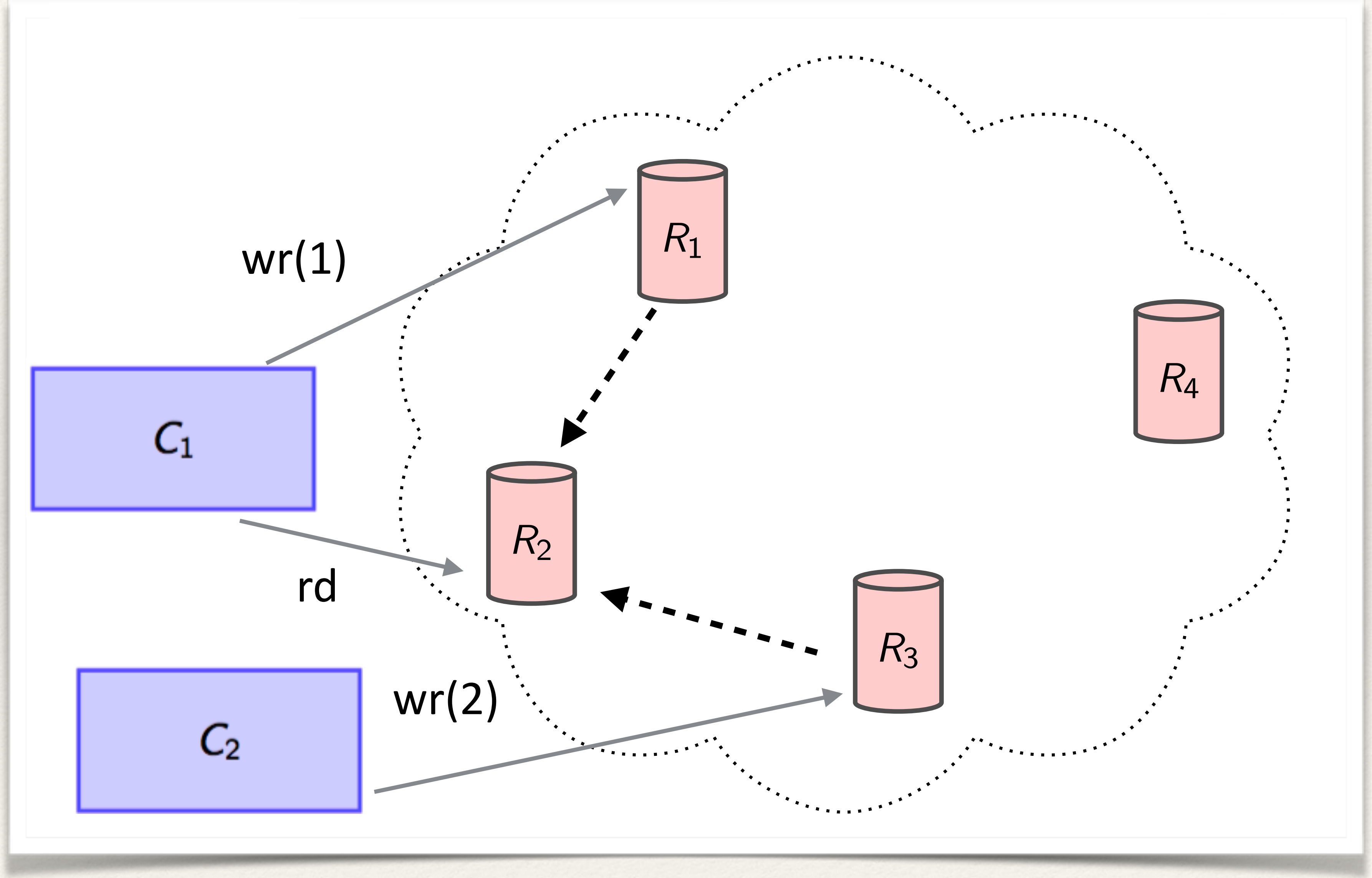
|

`wr(2)`

ARBitration

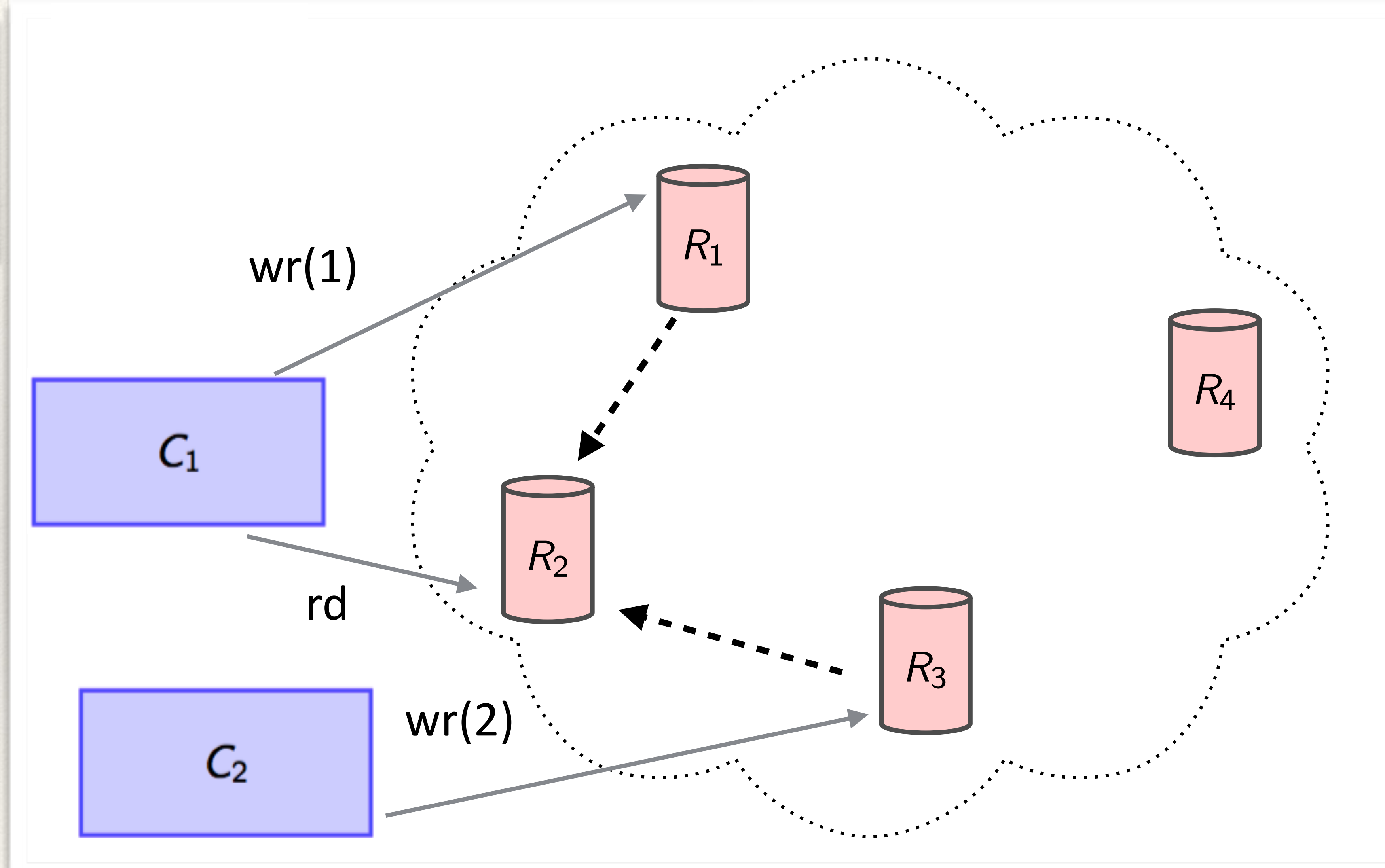


A register



A register

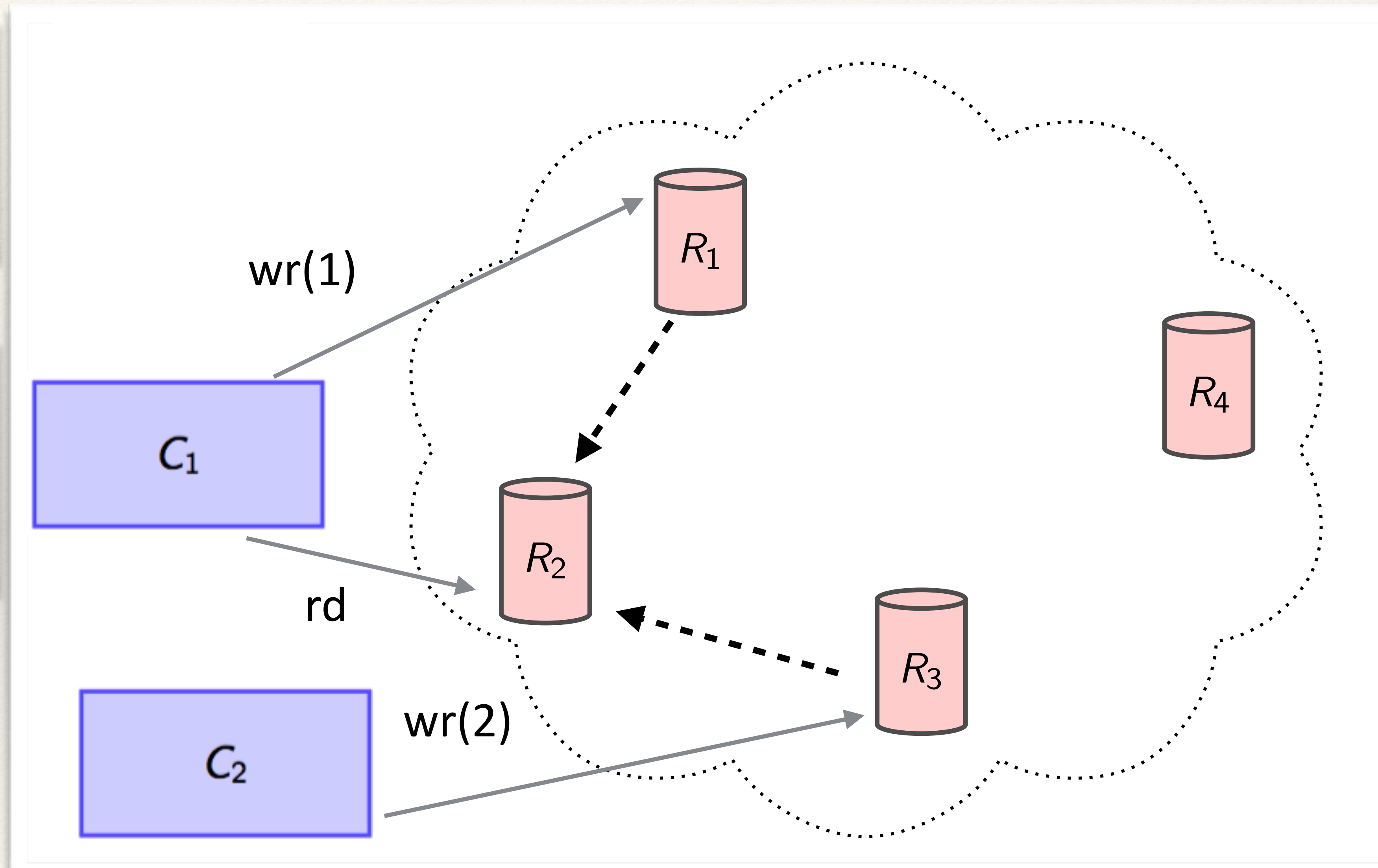
$$\mathrm{rd}\left(\begin{array}{ccc} \texttt{wr}(1) & \texttt{wr}(2) & \begin{array}{c}\texttt{wr}(1)\\ | \\ \texttt{wr}(2)\end{array}\end{array}\right) = 2$$



A register

$$\mathrm{rd}\left(\texttt{wr}(1) \qquad \texttt{wr}(2) \qquad \begin{array}{c}\texttt{wr}(1)\\ | \\ \texttt{wr}(2)\end{array}\right) = 2$$

$$\mathrm{rd}\left(\texttt{wr}(1) \qquad \texttt{wr}(2) \qquad \begin{array}{c}\texttt{wr}(2)\\ | \\ \texttt{wr}(1)\end{array}\right) = 1$$

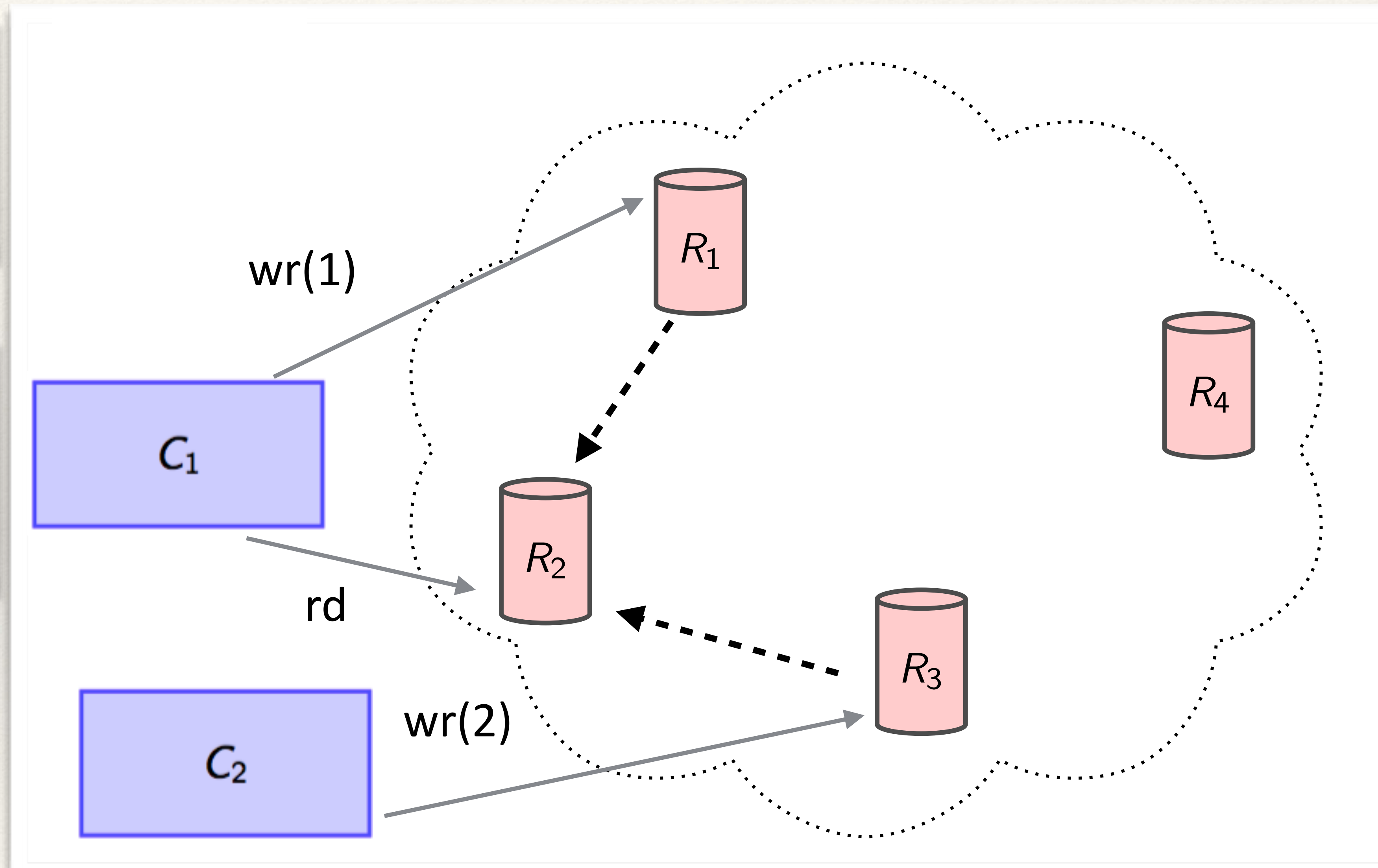


A register

$$\mathrm{rd}\left(\begin{array}{ccc} & & \mathtt{wr}(1)\\ \mathtt{wr}(1) & \mathtt{wr}(2) & \mid \\ & & \mathtt{wr}(2)\end{array}\right) = 2$$

$$\mathrm{rd}\left(\begin{array}{ccc} & & \mathtt{wr}(2)\\ \mathtt{wr}(1) & \mathtt{wr}(2) & \mid \\ & & \mathtt{wr}(1)\end{array}\right) = 1$$

Last-write-wins



Understanding RDTs

- Implementing RTDs means to provide a communication mechanism among replicas, to ensure its compatibility wrt. the behaviour of the operations and to guarantee that some global properties (e.g. eventual convergence of replicas) are preserved.
- But first…
    - Is it possible to get a traditional presentation of RTDs?
    - Is there any implicit assumption on the arbitrations?
    - Are RDTs compositional? I.e., are arbitrations of larger visibility orders explained in terms of smaller ones?

Internalising values

$$\langle \mathtt{wr}(1), ok \rangle \quad \langle \mathtt{wr}(2), ok \rangle$$

,

Internalising values

$$\langle \mathtt{wr}(1), \textit{ok}\rangle \qquad \langle \mathtt{wr}(2), \textit{ok}\rangle$$

,

$$\mathrm{rd}\left(\begin{array}{ccc} & & \mathtt{wr}(1) \\ \mathtt{wr}(1) & \mathtt{wr}(2) & \Big| \\ & & \mathtt{wr}(2) \end{array}\right) = 2$$

Internalising values

$$\langle \mathtt{wr}(1), \mathit{ok} \rangle \qquad \langle \mathtt{wr}(2), \mathit{ok} \rangle$$

,

$$\mathrm{rd}\left(\mathtt{wr}(1) \qquad \mathtt{wr}(2) \qquad \begin{array}{c} \mathtt{wr}(1) \\ | \\ \mathtt{wr}(2) \end{array} \right) = 2$$

$$\begin{tikzcd}
\langle \mathtt{wr}(1), \mathit{ok} \rangle \arrow[dr] & & \langle \mathtt{wr}(2), \mathit{ok} \rangle \arrow[dl] \\
& \langle \mathtt{rd}, 2 \rangle &
\end{tikzcd}$$

Internalising values

$$\mathsf{rd}\left(\ \mathtt{wr}(1)\qquad \mathtt{wr}(2)\qquad \begin{array}{c}\mathtt{wr}(1)\\ |\\ \mathtt{wr}(2)\end{array}\right) = 2$$

$$\mathcal{S}\left(\begin{array}{ccc}\langle \mathtt{wr}(1), ok\rangle & & \langle \mathtt{wr}(2), ok\rangle\\ & \searrow \quad \swarrow & \\ & \langle \mathtt{rd}, 2\rangle & \end{array}\right) = \left\{\begin{array}{c}\langle \mathtt{wr}(1), ok\rangle\\ |\\ \langle \mathtt{wr}(2), ok\rangle\\ |\\ \langle \mathtt{rd}, 2\rangle\end{array}\right\}$$

A specification goes
from configurations
to sets of arbitrations

Internalising values

$$\mathrm{rd}\left(\begin{array}{ccc} & & \texttt{wr}(1)\\ \texttt{wr}(1) & \texttt{wr}(2) & |\\ & & \texttt{wr}(2)\end{array}\right) = 2$$

$$\mathrm{rd}\left(\begin{array}{ccc} & & \texttt{wr}(2)\\ \texttt{wr}(1) & \texttt{wr}(2) & |\\ & & \texttt{wr}(1)\end{array}\right) = 1$$

$$\mathcal{S}\left(\begin{tikzcd} \langle \texttt{wr}(1), ok\rangle \arrow[dr] & & \langle \texttt{wr}(2), ok\rangle \arrow[dl] \\ & \langle \texttt{rd}, 0\rangle & \end{tikzcd}\right) = \left\{ \qquad \right\}$$

A specification goes
from configurations
to sets of arbitrations

Recovering RDTs: saturation

$$\mathrm{rd}\left(\begin{array}{ccc} & & \mathtt{wr}(1) \\ \mathtt{wr}(1) & \mathtt{wr}(2) & | \\ & & \mathtt{wr}(2) \end{array}\right) = 2$$

$$\mathcal{S}\left(\begin{array}{ccc} \langle \mathtt{wr}(1), ok \rangle & & \langle \mathtt{wr}(2), ok \rangle \\ & \searrow \quad \swarrow & \\ & \langle \mathtt{rd}, 2 \rangle & \end{array}\right) = \left\{\begin{array}{c} \langle \mathtt{wr}(1), ok \rangle \\ | \\ \langle \mathtt{wr}(2), ok \rangle \\ | \\ \langle \mathtt{rd}, 2 \rangle \end{array}\right\}$$

A specification goes
from configurations
to sets of arbitrations

Recovering RDTs: saturation

$$\mathrm{rd}\left(\begin{array}{ccc} & & \mathtt{wr}(1) \\ \mathtt{wr}(1) & \mathtt{wr}(2) & \big| \\ & & \mathtt{wr}(2) \end{array} \right) = 2$$

$$\mathcal{S}\left(\begin{tikzcd} \langle \mathtt{wr}(1), ok \rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok \rangle \arrow[dl] \\ & \langle \mathtt{rd}, 2 \rangle & \end{tikzcd} \right) = \left\{ \begin{array}{ccc} \langle \mathtt{wr}(1), ok \rangle & \langle \mathtt{wr}(1), ok \rangle & \langle \mathtt{rd}, 2 \rangle \\ \big| & \big| & \big| \\ \langle \mathtt{wr}(2), ok \rangle & \langle \mathtt{rd}, 2 \rangle & \langle \mathtt{wr}(1), ok \rangle \\ \big| & \big| & \big| \\ \langle \mathtt{rd}, 2 \rangle & \langle \mathtt{wr}(2), ok \rangle & \langle \mathtt{wr}(2), ok \rangle \end{array} \right\}$$

A specification goes
from configurations
to sets of arbitrations

Recovering RDTs: determinism

$$\mathcal{S}\left(\begin{tikzcd}[column sep=small] \langle \mathtt{wr}(1), ok\rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok\rangle \arrow[dl] \\ & \langle \mathtt{rd}, 2\rangle & \end{tikzcd}\right)$$

$$\mathcal{S}\left(\begin{tikzcd}[column sep=small] \langle \mathtt{wr}(1), ok\rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok\rangle \arrow[dl] \\ & \langle \mathtt{rd}, 1\rangle & \end{tikzcd}\right)$$

Recovering RDTs: determinism

$$\mathcal{S}\left(\begin{tikzcd}[column sep=tiny] \langle \mathtt{wr}(1), ok \rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok \rangle \arrow[dl] \\ & \langle \mathtt{rd}, 2 \rangle & \end{tikzcd}\right) \qquad \mathcal{S}\left(\begin{tikzcd}[column sep=tiny] \langle \mathtt{wr}(1), ok \rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok \rangle \arrow[dl] \\ & \langle \mathtt{rd}, 1 \rangle & \end{tikzcd}\right)$$

value-deterministic: empty intersection (after removing the last event)

Recovering RDTs: determinism

$$\mathcal{S}\left(\begin{tikzcd} \langle \mathtt{wr}(1), ok\rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok\rangle \arrow[dl] \\ & \langle \mathtt{rd}, 2\rangle & \end{tikzcd}\right)$$

$$\mathcal{S}\left(\begin{tikzcd} \langle \mathtt{wr}(1), ok\rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok\rangle \arrow[dl] \\ & \langle \mathtt{rd}, 1\rangle & \end{tikzcd}\right)$$

value-deterministic: empty intersection (after removing the last event)

deterministic: empty intersection *even* forgetting the value component

Recovering RDTs: determinism

$$\mathcal{S}\left(\begin{tikzcd} \langle \mathtt{wr}(1), ok\rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok\rangle \arrow[dl] \\ & \langle \mathtt{rd}, 2\rangle & \end{tikzcd}\right)$$

$$\mathcal{S}\left(\begin{tikzcd} \langle \mathtt{wr}(1), ok\rangle \arrow[dr] & & \langle \mathtt{wr}(2), ok\rangle \arrow[dl] \\ & \langle \mathtt{rd}, 1\rangle & \end{tikzcd}\right)$$

value-deterministic: empty intersection (after removing the last event)

deterministic: empty intersection *even* forgetting the value component

RTDs have chosen the second path, thus e.g. forbidding write failures

Recovering RDTs: coherence

$$
\begin{array}{c}
\langle \mathtt{wr}(1), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{wr}(2), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{rd}, 2 \rangle
\end{array}
\otimes
\begin{array}{c}
\langle \mathtt{wr}(2), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{wr}(3), \mathtt{ok} \rangle
\end{array}
=
\left\{
\begin{array}{c}
\langle \mathtt{wr}(1), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{wr}(2), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{rd}, 2 \rangle \\
| \\
\langle \mathtt{wr}(3), \mathtt{ok} \rangle
\end{array}
\qquad
\begin{array}{c}
\langle \mathtt{wr}(1), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{wr}(2), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{wr}(3), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{rd}, 2 \rangle
\end{array}
\right\}
$$

Recovering RDTs: coherence

$$
\begin{array}{c}
\langle \mathtt{wr}(1), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{wr}(2), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{rd}, 2 \rangle
\end{array}
\otimes
\begin{array}{c}
\langle \mathtt{wr}(2), \mathtt{ok} \rangle \\
| \\
\langle \mathtt{wr}(3), \mathtt{ok} \rangle
\end{array}
=
\left\{
\begin{array}{cc}
\langle \mathtt{wr}(1), \mathtt{ok} \rangle & \langle \mathtt{wr}(1), \mathtt{ok} \rangle \\
| & | \\
\langle \mathtt{wr}(2), \mathtt{ok} \rangle & \langle \mathtt{wr}(2), \mathtt{ok} \rangle \\
| & | \\
\langle \mathtt{rd}, 2 \rangle & \langle \mathtt{wr}(3), \mathtt{ok} \rangle \\
| & | \\
\langle \mathtt{wr}(3), \mathtt{ok} \rangle & \langle \mathtt{rd}, 2 \rangle
\end{array}
\right\}
$$

$$
\forall \mathtt{G}.\ \mathcal{S}(\mathtt{G}) = \bigotimes_{\mathtt{e} \in \mathcal{E}_{\mathtt{G}}} \mathcal{S}(\mathtt{G}|_{- \prec^{*} \mathtt{e}})
$$

Recovering RDTs: coherence

$$\begin{array}{c}\langle \mathtt{wr}(1),\mathtt{ok}\rangle\\ | \\ \langle \mathtt{wr}(2),\mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd},2\rangle\end{array} \otimes \begin{array}{c}\langle \mathtt{wr}(2),\mathtt{ok}\rangle\\ | \\ \langle \mathtt{wr}(3),\mathtt{ok}\rangle\end{array} = \left\{ \begin{array}{cc} \langle \mathtt{wr}(1),\mathtt{ok}\rangle & \langle \mathtt{wr}(1),\mathtt{ok}\rangle \\ | & | \\ \langle \mathtt{wr}(2),\mathtt{ok}\rangle & \langle \mathtt{wr}(2),\mathtt{ok}\rangle \\ | & | \\ \langle \mathtt{rd},2\rangle & \langle \mathtt{wr}(3),\mathtt{ok}\rangle \\ | & | \\ \langle \mathtt{wr}(3),\mathtt{ok}\rangle & \langle \mathtt{rd},2\rangle \end{array}\right\}$$

$$\forall \mathtt{G}.\ \mathcal{S}(\mathtt{G}) = \bigotimes_{\mathtt{e}\in\mathcal{E}_{\mathtt{G}}} \mathcal{S}(\mathtt{G}|_{-\prec^{*}\mathtt{e}})$$

Admissible arbitrations
never increases when
extending the visibility

Recovering RDTs: the theorem

- There is a one-to-one correspondence between RTDs and saturated, deterministic, and coherent specifications

Recovering RDTs: the theorem

- There is a one-to-one correspondence between RTDs and saturated, deterministic, and coherent specifications

….which is bad for RDTs

Recovering RDTs: the theorem

- There is a one-to-one correspondence between RTDs and saturated, deterministic, and coherent specifications

….which is bad for RDTs

both saturation and determinism are bad!!

Recovering RDTs: the theorem

- There is a one-to-one correspondence between RTDs and saturated, deterministic, and coherent specifications

....which is bad for RDTs

both saturation and determinism are bad!!

$$\mathcal{S}\left(\begin{array}{c}\langle \mathtt{inc}, \mathtt{ok}\rangle \\ \downarrow \\ \langle \mathtt{rd}, 1\rangle\end{array}\right) = \left\{\begin{array}{c}\langle \mathtt{inc}, \mathtt{ok}\rangle \\ | \\ \langle \mathtt{rd}, 1\rangle\end{array}\right\} \quad \mathcal{S}\left(\begin{array}{c}\langle \mathtt{inc}, \mathtt{fail}\rangle \\ \downarrow \\ \langle \mathtt{rd}, \bot\rangle\end{array}\right) = \left\{\begin{array}{c}\langle \mathtt{inc}, \mathtt{fail}\rangle \\ | \\ \langle \mathtt{rd}, \bot\rangle\end{array}\right\}$$

value-deterministic, yet not deterministic

From specifications to transition systems

$\langle \mathtt{G}, \mathtt{P} \rangle$ $\mathtt{P} \in \mathcal{S}(\mathtt{G})$

states

From specifications to transition systems

$\langle \mathtt{G}, \mathtt{P} \rangle$ $\mathtt{P} \in \mathcal{S}(\mathtt{G})$ states

$\langle \mathtt{G}, \mathtt{P} \rangle \xrightarrow{\ell} \langle \mathtt{G}', \mathtt{P}' \rangle$ transitions

$\mathtt{G}' = \mathtt{G}^{\ell}$ $\mathtt{P}'|_{\mathcal{E}_{\mathtt{G}}} = \mathtt{P}$

From specifications to transition systems

(COMP)

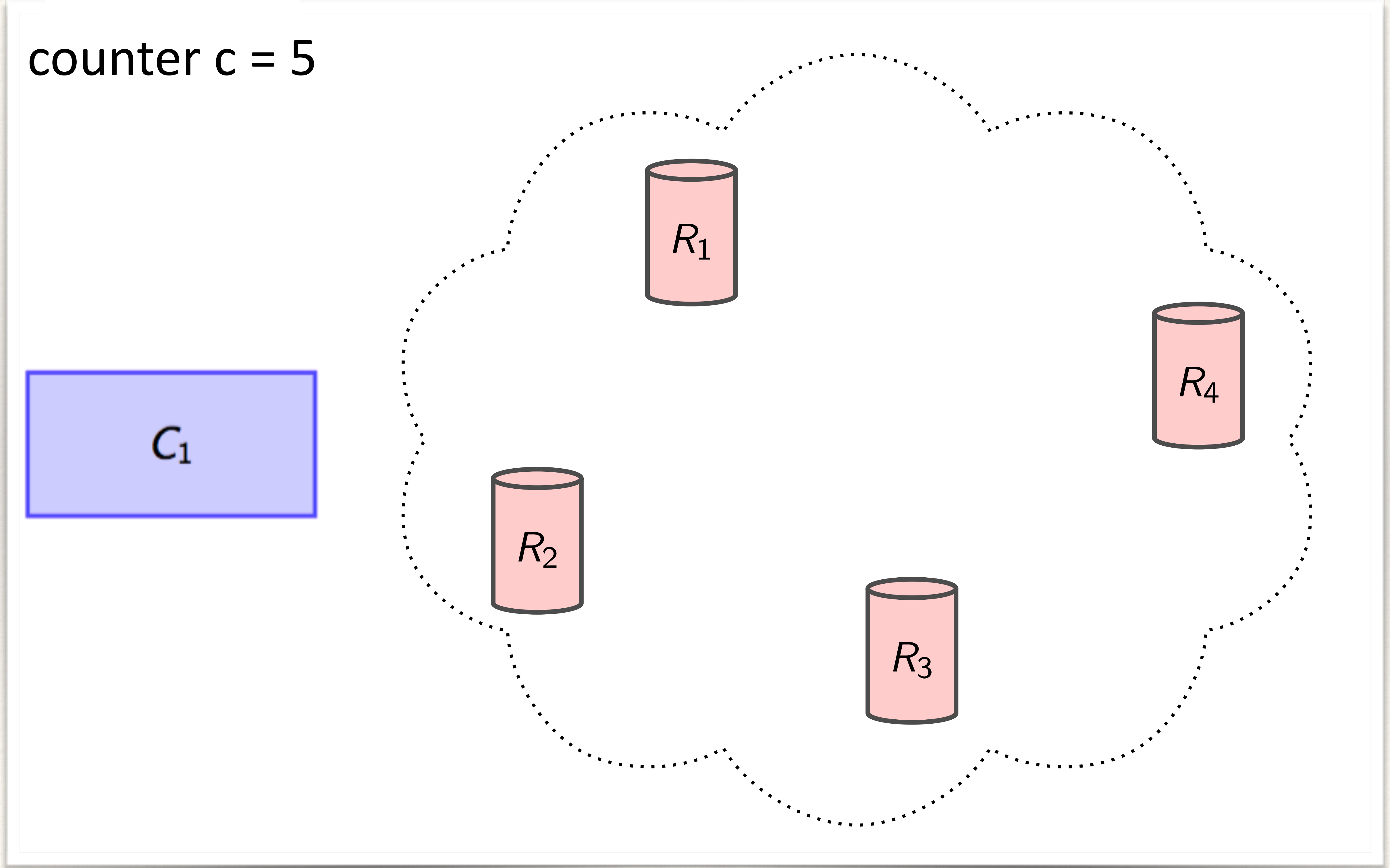
$$\frac{\langle \mathtt{G}_1, \mathtt{P}|_{\mathcal{E}_{\mathtt{G}_1}} \rangle \xrightarrow{\ell} \langle \mathtt{G}'_1, \mathtt{P}'_1 \rangle \qquad \mathtt{P}' \in \mathtt{P} \otimes \mathtt{P}'_1}{\langle \mathtt{G}_1 \sqcup \mathtt{G}_2, \mathtt{P} \rangle \xrightarrow{\ell} \langle \mathtt{G}'_1 \sqcup \mathtt{G}_2, \mathtt{P}' \rangle}$$

an abstract transition system against which
to compare (by asynchronous simulation)
those of actual implementations…

Implementing a specification

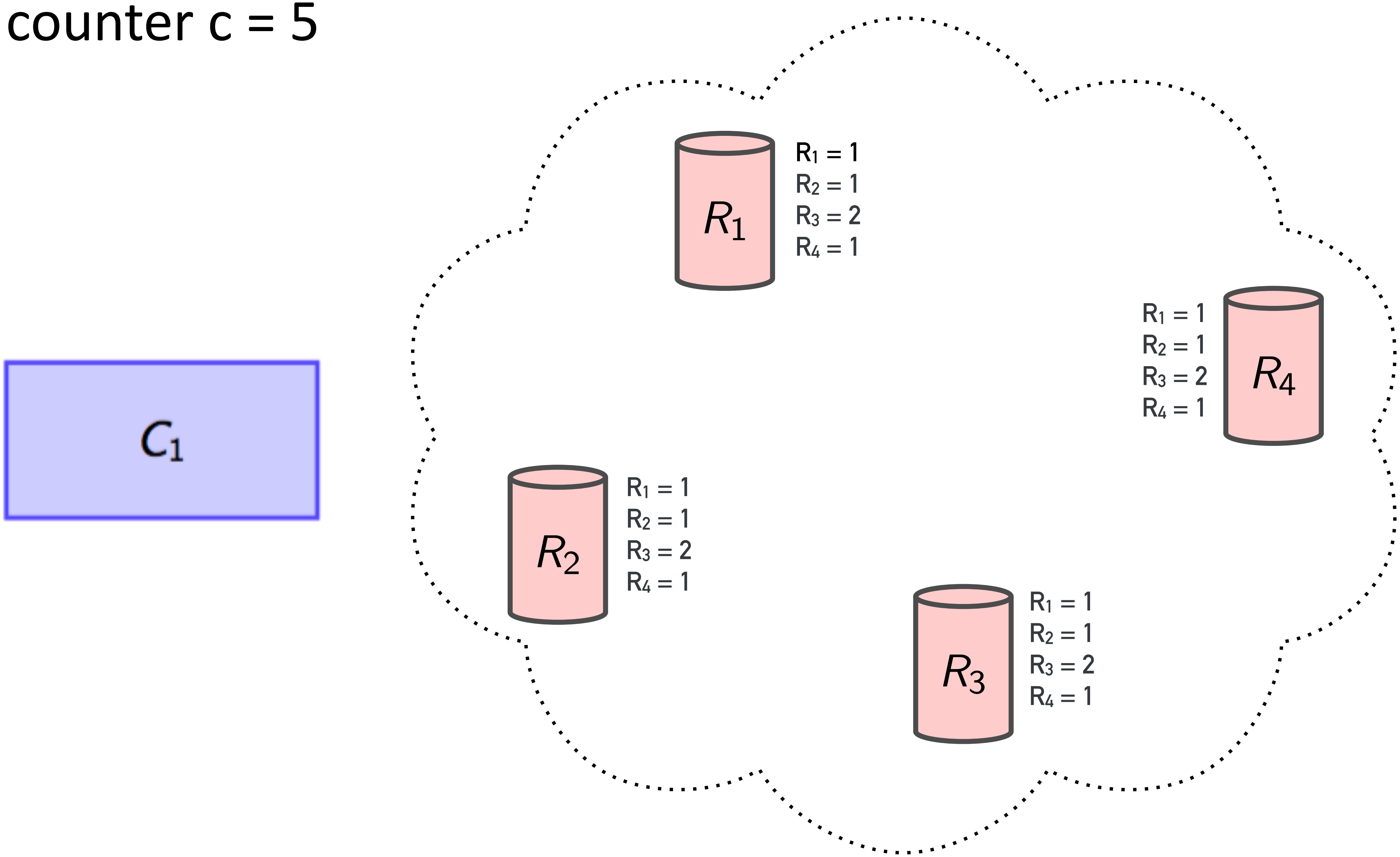
- Proposed implementation of RDTs [Burckhardt et al.].
  - Each replica propagates its state to the other replicas.
  - It is assumed that all replicas have the same behaviour.

A counter

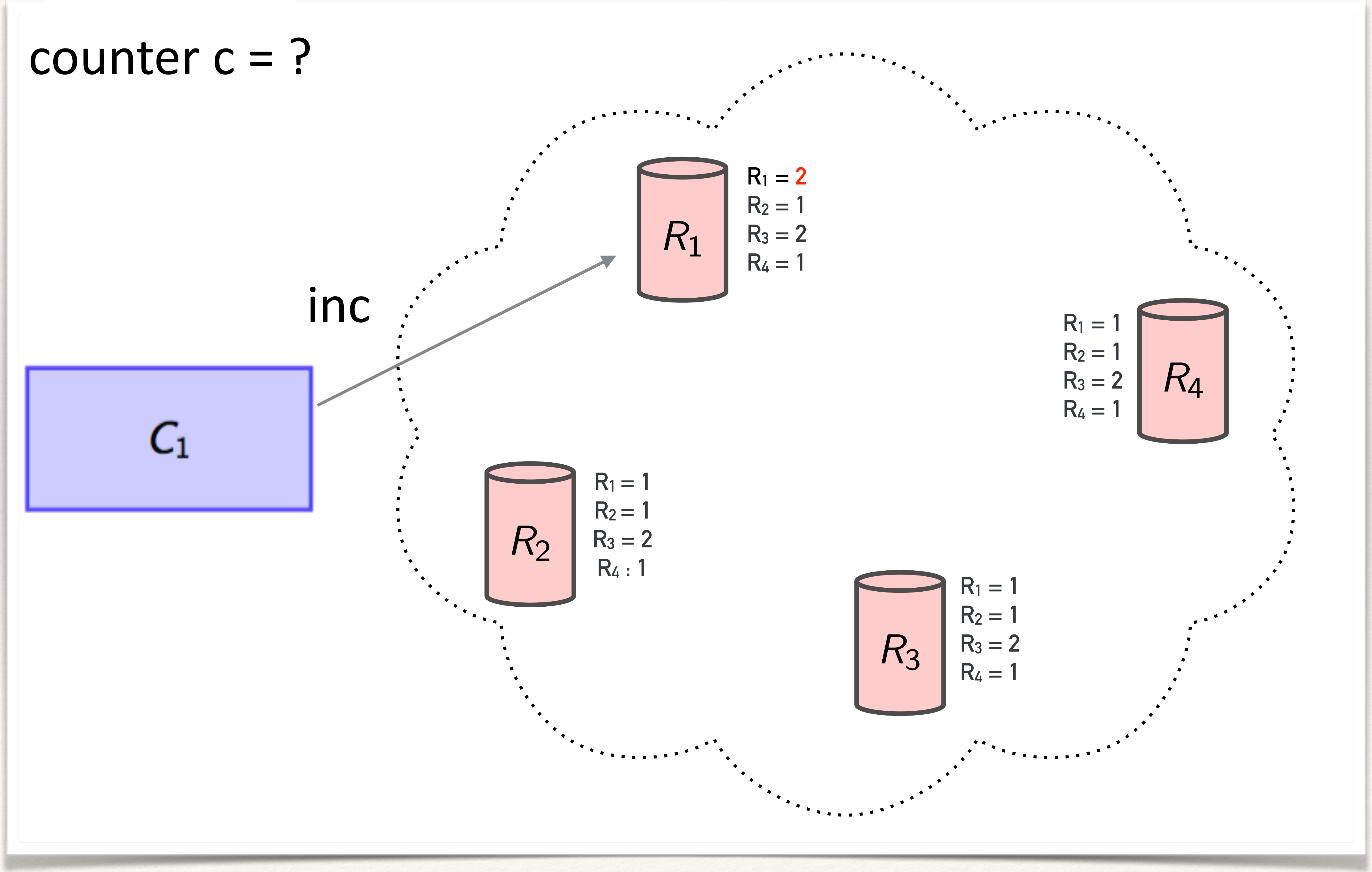


A counter

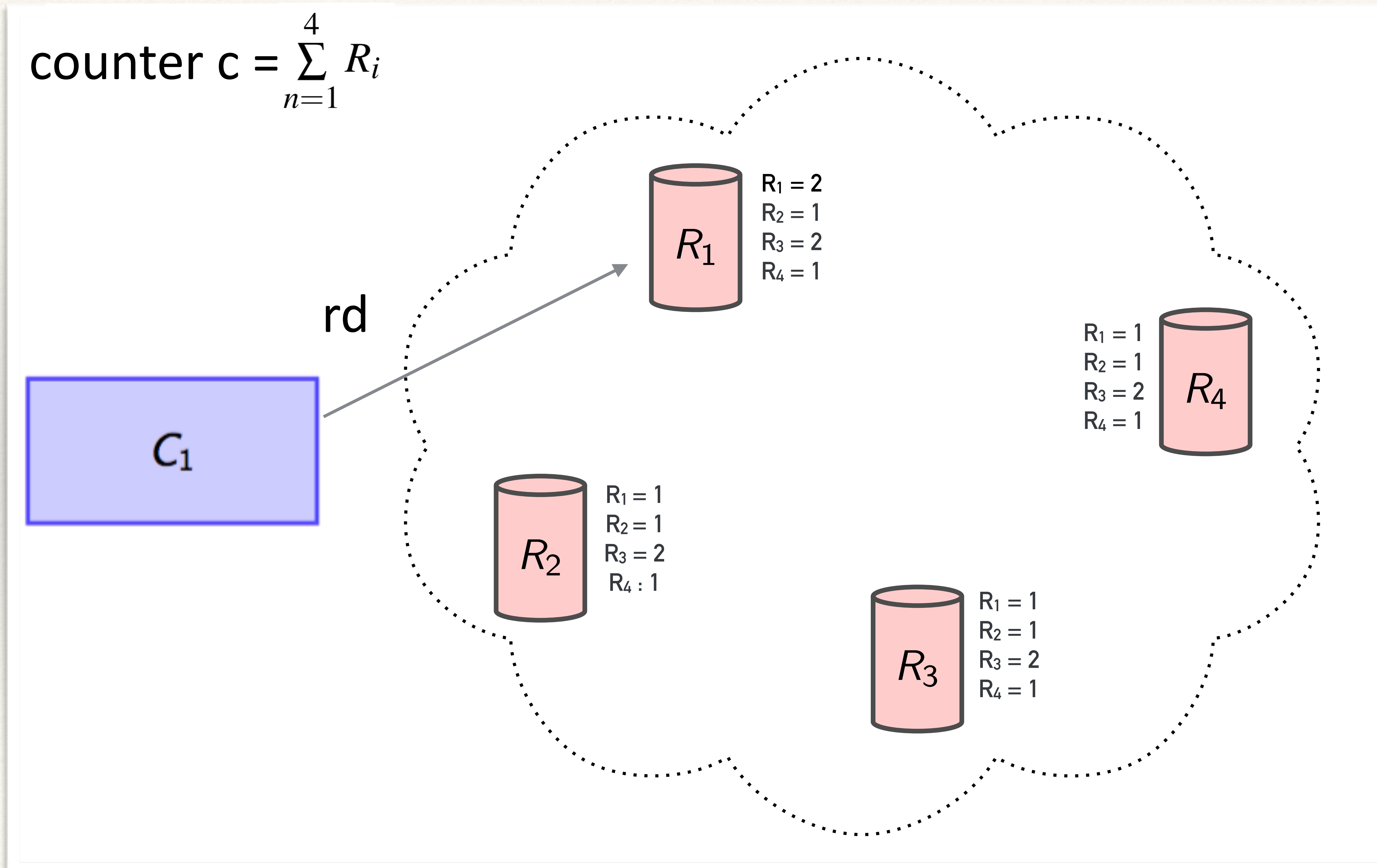
counter c = 5



A counter


counter c = ?
inc
C_1
R_1
R1 = 2
R2 = 1
R3 = 2
R4 = 1
R_4
R1 = 1
R2 = 1
R3 = 2
R4 = 1
R_2
R1 = 1
R2 = 1
R3 = 2
R4 : 1
R_3
R1 = 1
R2 = 1
R3 = 2
R4 = 1

A counter



counter c = $\sum_{n=1}^{4} R_i$
rd
C_1
R_1
$R_1 = 2$
$R_2 = 1$
$R_3 = 2$
$R_4 = 1$
R_2
$R_1 = 1$
$R_2 = 1$
$R_3 = 2$
$R_4 : 1$
R_3
$R_1 = 1$
$R_2 = 1$
$R_3 = 2$
$R_4 = 1$
R_4
$R_1 = 1$
$R_2 = 1$
$R_3 = 2$
$R_4 = 1$

an LTS for a counter

$$\Sigma = \mathcal{R} \times (\mathcal{R} \mapsto \mathbb{N})$$

$$\mathcal{L} = \{\langle \mathtt{inc}, \mathtt{ok} \rangle\} \cup (\{\mathtt{rd}\} \times \mathbb{N})$$

(READ)

$$\frac{\mathsf{k} = \Sigma_{\mathsf{s} \in dom(\mathsf{v})} \mathsf{v}(\mathsf{s})}{\langle \mathsf{r}, \mathsf{v} \rangle \xrightarrow{\mathtt{rd},\mathtt{k}} \langle \mathsf{r}, \mathsf{v} \rangle}$$

(INC)

$$\langle \mathsf{r}, \mathsf{v} \rangle \xrightarrow{\mathtt{inc},\mathtt{ok}} \langle \mathsf{r}, \mathsf{v}[\mathsf{r} \mapsto \mathsf{v}(\mathsf{r}) + 1] \rangle$$

an LTS for a counter

(SEND)

$$\langle \mathsf{r}, \mathsf{v} \rangle \xrightarrow{\mathtt{send}, \langle \mathsf{r}, \mathsf{v} \rangle} \langle \mathsf{r}, \mathsf{v} \rangle$$

(RCV)

$$\langle \mathsf{r}, \mathsf{v} \rangle \xrightarrow{\mathtt{rcv}, \langle \mathsf{r}_k, \mathsf{v}_k \rangle} \langle \mathsf{r}, \mathtt{max}\{\mathsf{v}, \mathsf{v}_k\} \rangle$$

$$\forall \mathsf{s}. \mathtt{max}\{\mathsf{v}, \mathsf{v}_k\}(\mathsf{s}) = \mathtt{max}\{\mathsf{v}(\mathsf{s}), \mathsf{v}_k(\mathsf{s})\}$$

the resulting LTS is correct wrt. the abstract LTS
(via a suitable simulation)

an LTS for multiple counters

(SEND)

$$\langle \mathsf{r}, \mathsf{v} \rangle \xrightarrow{\mathtt{send}, \langle \mathsf{r}, \mathsf{v} \rangle} \langle \mathsf{r}, \mathsf{v} \rangle$$

(RCV)

$$\langle \mathsf{r}, \mathsf{v} \rangle \xrightarrow{\mathtt{rcv}, \langle \mathsf{r}_k, \mathsf{v}_k \rangle} \langle \mathsf{r}, \mathtt{max}\{\mathsf{v}, \mathsf{v}_k\} \rangle$$

(PARL)

$$\frac{\sigma_1 \xrightarrow{\ell} \sigma_1'}{\sigma_1 \| \sigma_2 \xrightarrow{\ell} \sigma_1' \| \sigma_2}$$

(COMM)

$$\frac{\sigma_1 \xrightarrow{\mathtt{send}, \sigma} \sigma_1' \quad \sigma_2 \xrightarrow{\mathtt{rcv}, \sigma} \sigma_2'}{\sigma_1 \| \sigma_2 \xrightarrow{\tau} \sigma_1' \| \sigma_2'}$$

the resulting LTS is still correct wrt. the abstract LTS

Conclusions & future works

Conclusions & future works

- We provided a denotationally-flavoured characterisation of a well-accepted definition of replicated data types
    - thus making explicit some implicit assumptions

Conclusions & future works

- We provided a denotationally-flavoured characterisation of a well-accepted definition of replicated data types
  - thus making explicit some implicit assumptions
- …& a mechanism for proving correctness of implementations

Conclusions & future works

- We provided a denotationally-flavoured characterisation of a well-accepted definition of replicated data types
  - thus making explicit some implicit assumptions
- …& a mechanism for proving correctness of implementations
- We are looking for a categorical presentation
  - …in order to get operators for composing specifications

Conclusions & future works

- We provided a denotationally-flavoured characterisation of a well-accepted definition of replicated data types
  - thus making explicit some implicit assumptions
- …& a mechanism for proving correctness of implementations
- We are looking for a categorical presentation
  - …in order to get operators for composing specifications
- We plan to recast guarantee properties via the abstract LTS