

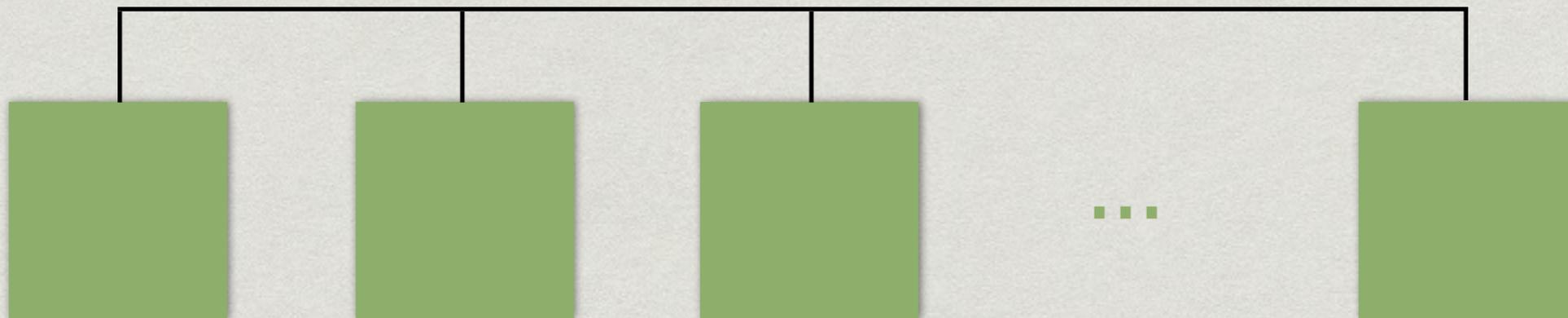
EFFICIENT VERIFICATION OF REPLICATED DATATYPES USING LATER APPEARANCE RECORDS (LAR)

**Madhavan Mukund, Gautham Shenoy R, S P Suresh
Chennai Mathematical Institute, Chennai, India**

ATVA 2015, Shanghai, China, 14 October 2015

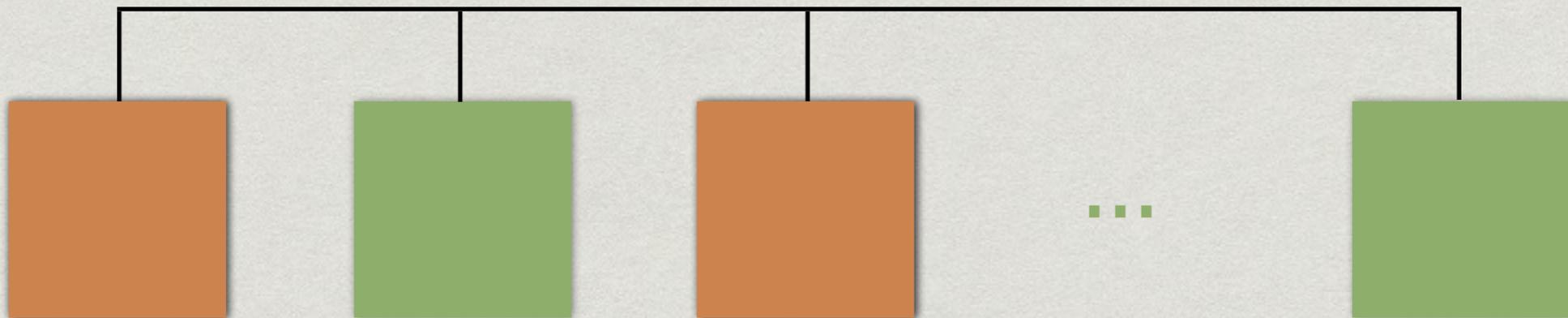
Distributed systems

- * N nodes connected by asynchronous network



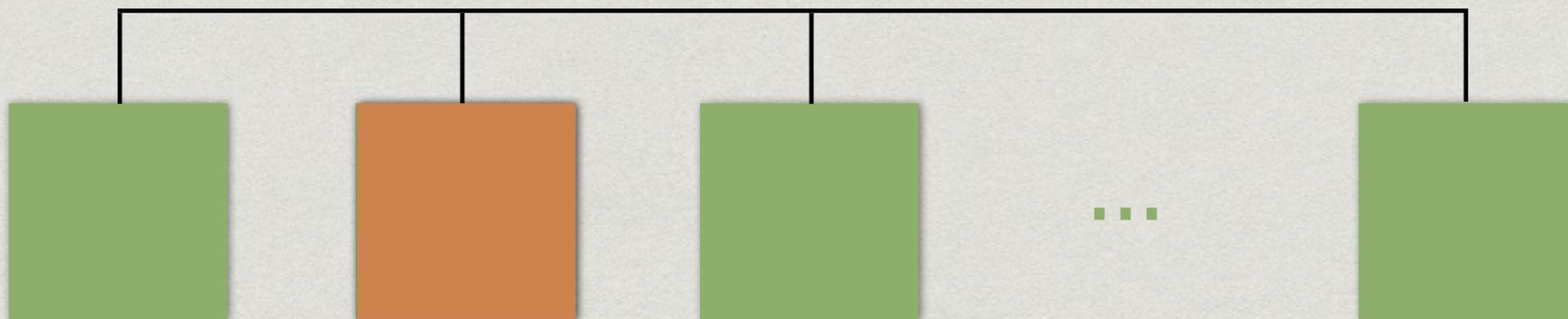
Distributed systems

- * N nodes connected by asynchronous network
- * Nodes may fail and recover infinitely often



Distributed systems

- * N nodes connected by asynchronous network
- * Nodes may fail and recover infinitely often
- * Nodes resume from safe state before failure



Replicated datatypes

- * Each node replicates the data structure



Replicated datatypes

- * Each node replicates the data structure
- * Queries / updates addressed to any replica
 - * Queries are side-effect free
 - * Updates change the state of the data structure



Replicated datatypes ...

- * Typical applications
 - * Amazon shopping carts
 - * Google docs
 - * Facebook “like” counters



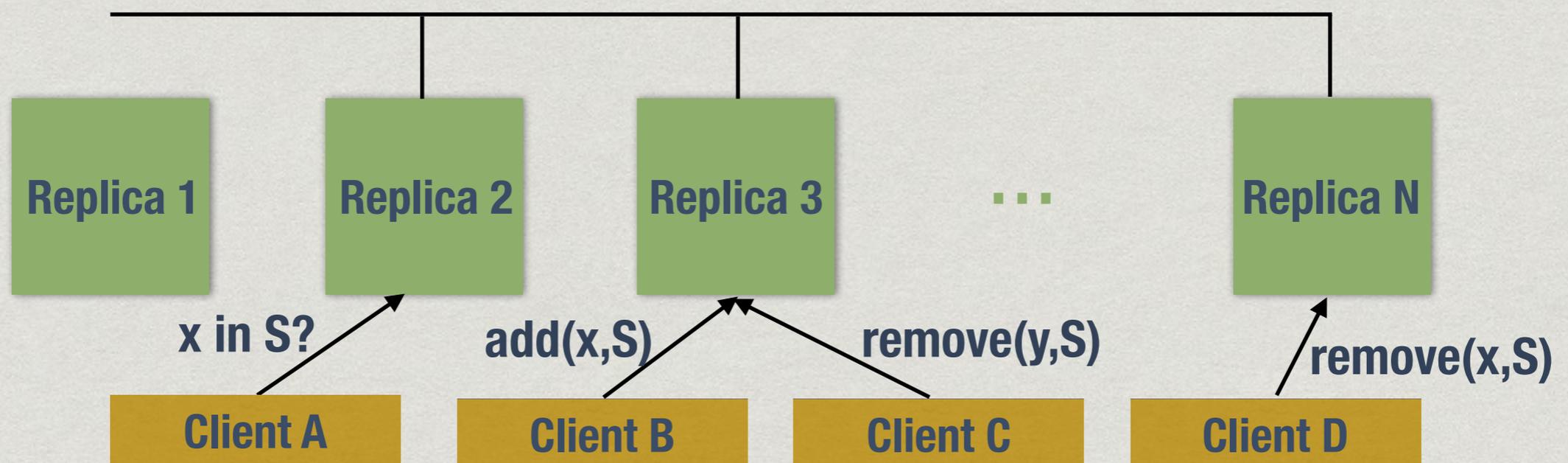
Replicated datatypes ...

- * Typical data structure — Sets
- * Query : is x a member of S ?
- * Updates : add x to S , remove x from S



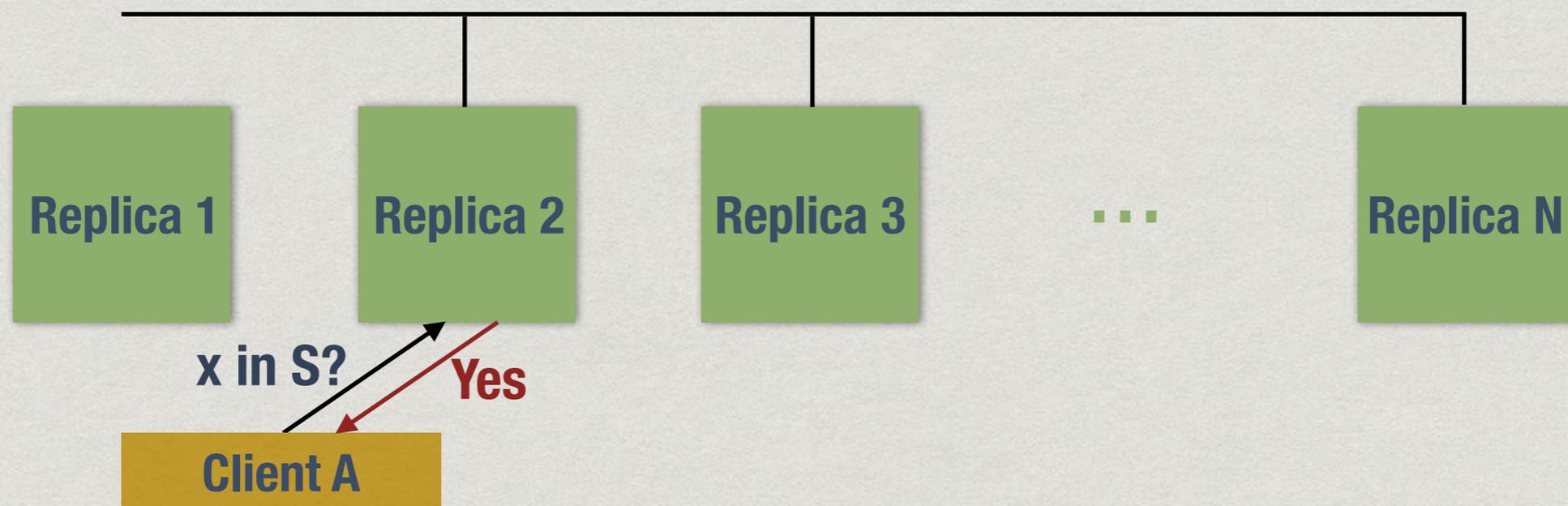
Clients and replicas

- * Clients issue query/update requests
- * Each request is fielded by an individual **source** replica

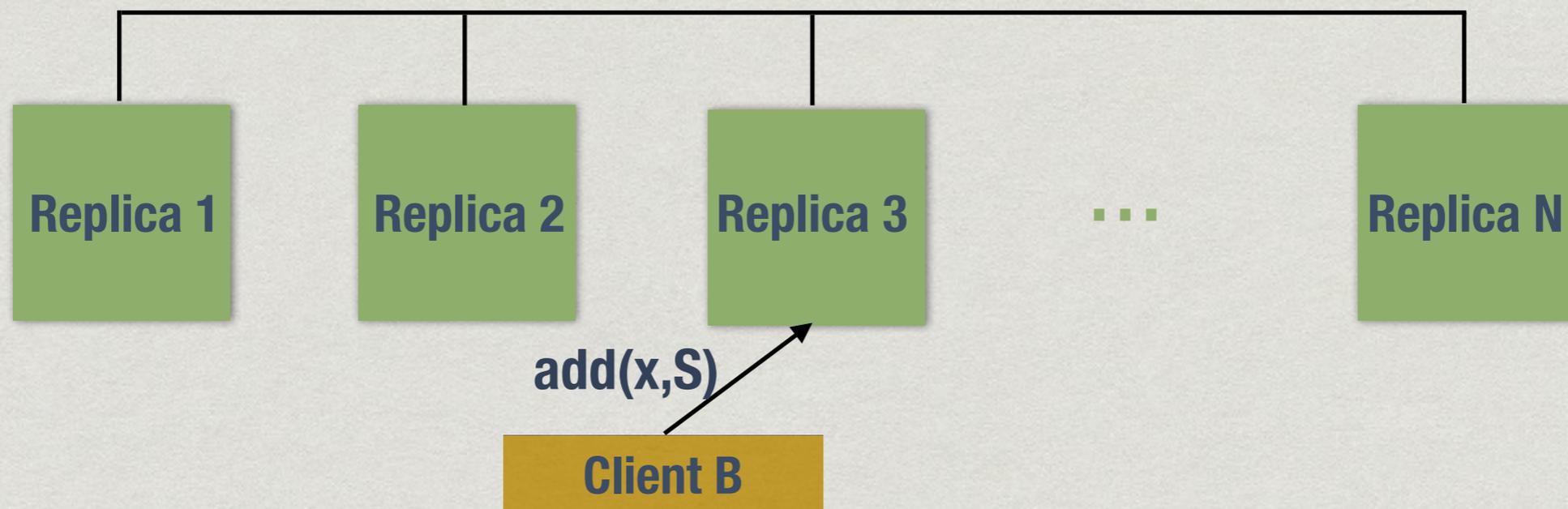


Processing query requests

- * Queries are answered directly by source replica, using local state

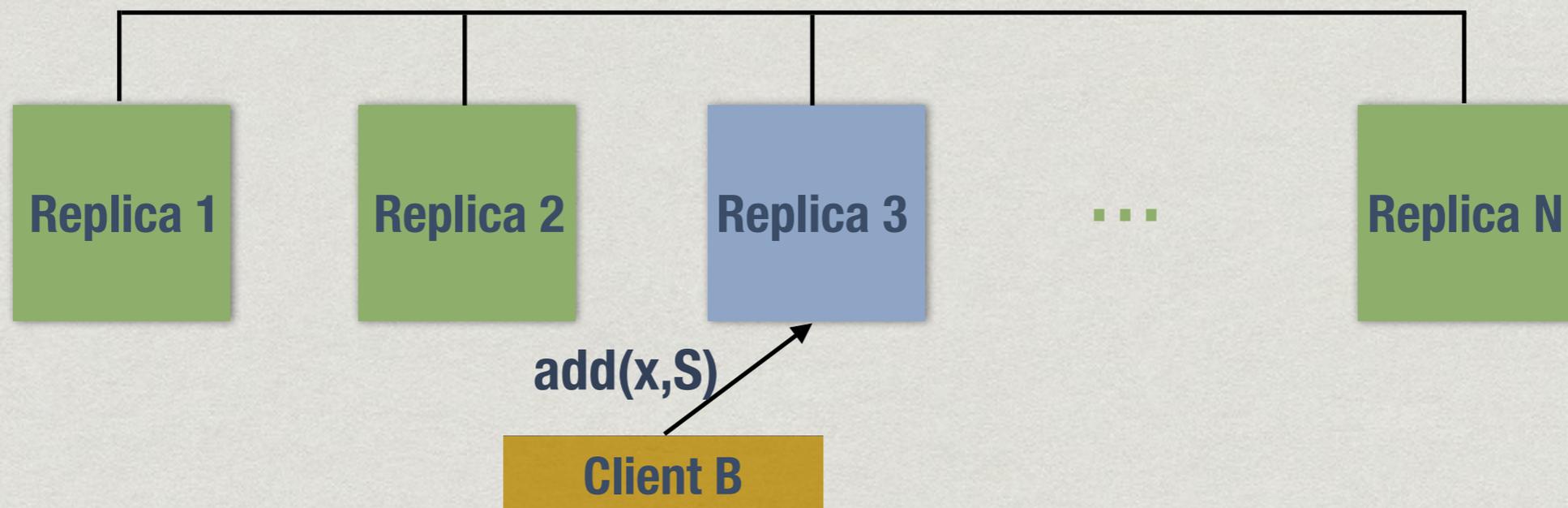


Processing updates



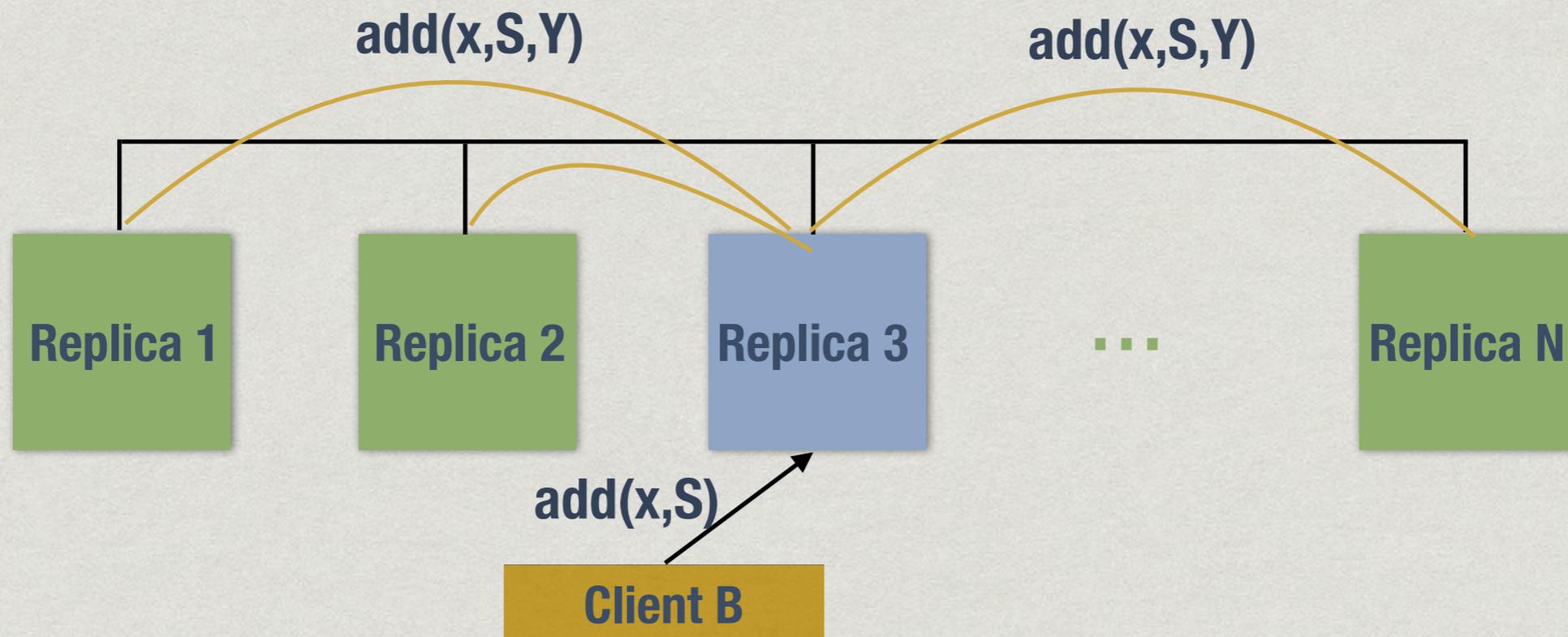
Processing updates

- * Source replica first updates its own state



Processing updates

- * Source replica first updates its own state
- * Propagates update message to other replicas
 - * With auxiliary metadata (timestamps etc)



Strong eventual consistency

- * Replicas may diverge while updates propagate
 - * All messages are reliably delivered
- * Replicas that receive the same set of updates must be query equivalent
- * After a period of quiescence, all replicas converge
- * Any stronger consistency requirement would negate availability or partition tolerance (Brewer's CAP theorem)

Facebook example (2012)

<http://markcathcart.com/2012/03/06/eventually-consistent/>

The image shows a screenshot of a Facebook post and its comments. The post is by Mark Cathcart, asking if anyone else has noticed that the Facebook locator is 'squiffy' (inaccurate) near Inkberrow. The comments are from Mike Gillespie, Mark Cathcart, and Martin Jenkins. The right side of the image shows the right-hand column of the Facebook interface, including a comment by Mike Gillespie on his own status, a comment by Jen Mathe, a birthday notification for Hugo Garza, a sponsored advertisement for Fab.com outdoor lanterns, and a political advertisement for Al Franken.

Mike Gillespie Has anyone else noticed that the FB locator is squiffy.... I am no where near Inkberrow....
Like · Comment · Share · 2 hours ago near Inkberrow, England · 🌐

Mark Cathcart are you on a wired network? They get it from the ISP based on the IP address...
2 hours ago · Like

Mike Gillespie I know.... Actually this might interest you... I didn't realise until today that there are actually two sets of recognised gps co ordinates used on the web - OSGB36 and WGS84... and depending on which set you use (given that we use post codes here and zip codes elsewhere) a post code can be as much as 100 metres out.....
about an hour ago · Like

Martin Jenkins and that matters because?
15 minutes ago · Like

Mark Cathcart well its of passing interest because Mike has a business that could benefit from being able to accurately locate properties based on the location of the people looking...
4 minutes ago · Like · 📌 1

Write a comment...

Mike Gillespie commented on his own status: "When you run a business that r..."

Jen Mathe commented on her own status: "Jenni Plane This is probably t..."

Hugo Garza's birthday is today
4 events this week

Sponsored Create an Ad

Outdoor Lanterns
Light up your outdoor space with these wireless lanterns. Join Fab.com and save 25%
159,998 people like Fab.com.

Join Al Franken
alfranken.com
Republicans think your employer should decide what health care you get. Sign here if you think they

Facebook example (2012)

<http://markcathcart.com/2012/03/06/eventually-consistent/>

The image shows a screenshot of a Facebook post and its comments. The post is by Mike Gillespie, asking if anyone else has noticed that the Facebook locator is 'squiffy'. The comments are from Mark Cathcart, Mike Gillespie, and Martin Jenkins. A red oval highlights a comment by Mike Gillespie on his own status, and a red arrow points from it to the comment by Mark Cathcart. The right side of the image shows a sidebar with a birthday notification for Hugo Garza, 4 events this week, a sponsored ad for Outdoor Lanterns, and a link to join Al Franken.

Mike Gillespie Has anyone else noticed that the FB locator is squiffy.... I am no where near Inkberrow....
Like · Comment · Share · 2 hours ago near Inkberrow, England · 🌐

Mark Cathcart are you on a wired network? They get it from the ISP based on the IP address...
2 hours ago · Like

Mike Gillespie I know.... Actually this might interest you... I didn't realise until today that there are actually two sets of recognised gps co ordinates used on the web - OSGB36 and WGS84... and depending on which set you use (given that we use post codes here and zip codes elsewhere) a post code can be as much as 100 metres out.....
about an hour ago · Like

Martin Jenkins and that matters because?
15 minutes ago · Like

Mark Cathcart well its of passing interest because Mike has a business that could benefit from being able to accurately locate properties based on the location of the people looking...
4 minutes ago · Like · 📍 1

Write a comment...

Mike Gillespie commented on his own status: "When you run a business that r..."

Jen Mathe commented on her own status: "Jenni Plane This is probably t..."

Hugo Garza's birthday is today

4 events this week

Sponsored Create an Ad

Outdoor Lanterns
Light up your outdoor space with these wireless lanterns. Join Fab.com and save 25%
159,998 people like Fab.com.

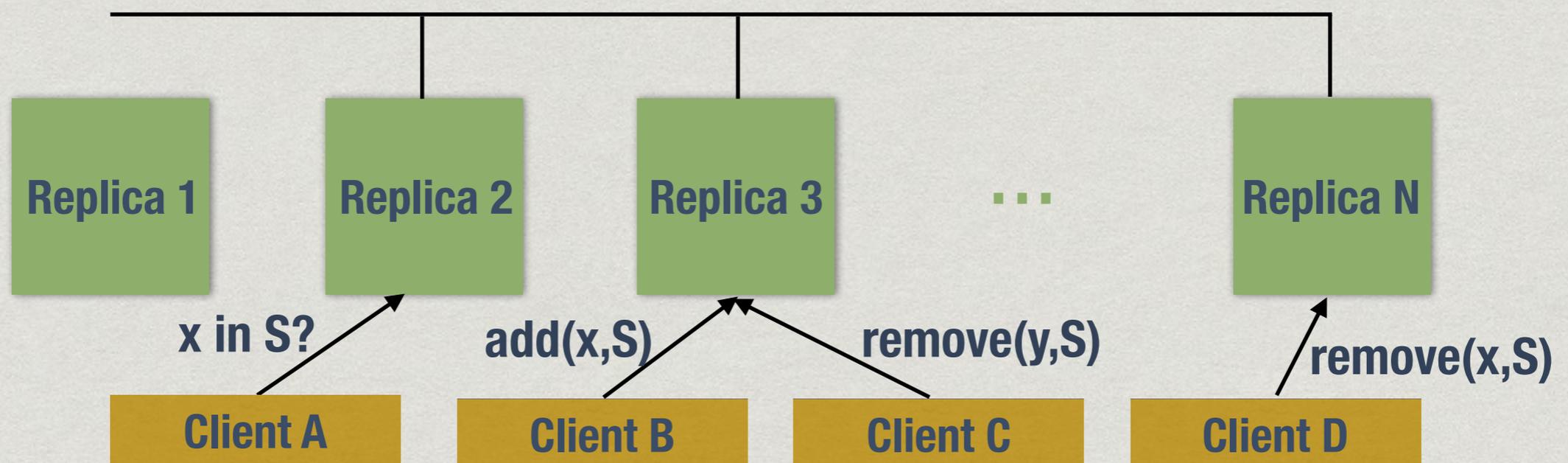
Join Al Franken
alfranken.com
Republicans think your employer should decide what health care you get. Sign here if you think they

CRDT: Conflict Free Data Types

- * Introduced by Shapiro et al 2011
 - * Implementations of counters, sets, graphs, ... that satisfy strong eventual consistency by design
 - * No independent specifications
 - * Correctness?
- * Formalisation by Burkhardt et al 2014
 - * Very detailed, difficult to use for verification

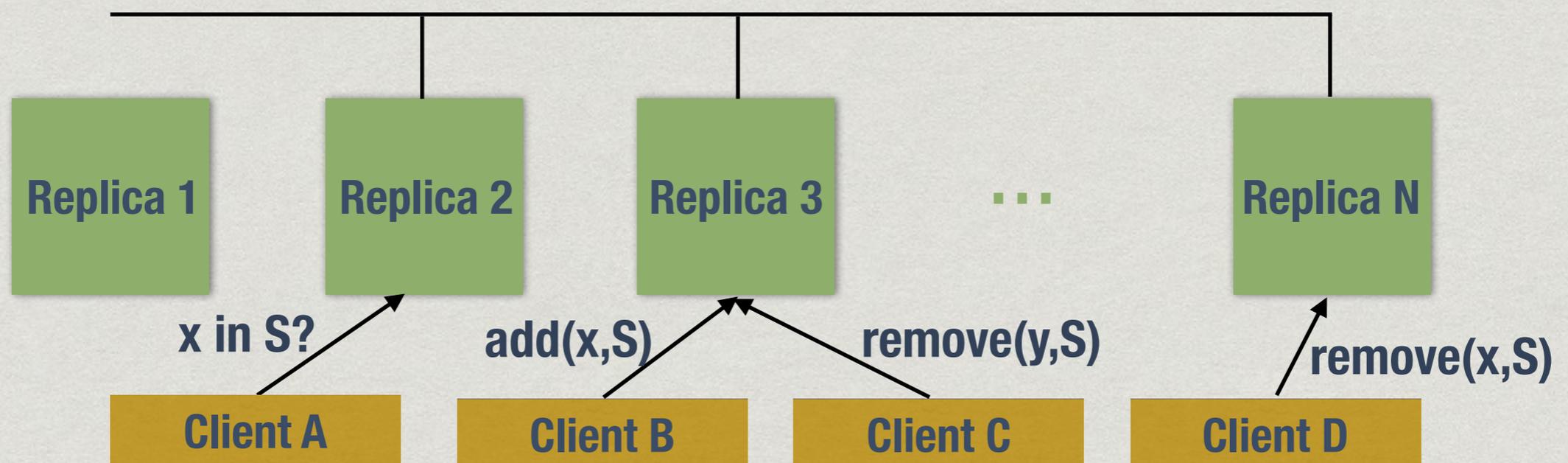
Need for specifications

- * How to resolve conflicts?
- * What does it mean to concurrently apply $\text{add}(x,S)$ and $\text{remove}(x,S)$ to a set S ?
 - * Different replicas see these updates in different orders
- * Observed-Remove (OR) sets: add wins



“Operational” specifications

- * My implementation uses timestamps, ... to detect causality and concurrency
- * If my replica received $\langle \text{add}(x,S), t \rangle$ and $\langle \text{remove}(x,S), t' \rangle$ and t and t' are related by ..., then answer Yes to “ x in S ?”, otherwise No



Declarative specification

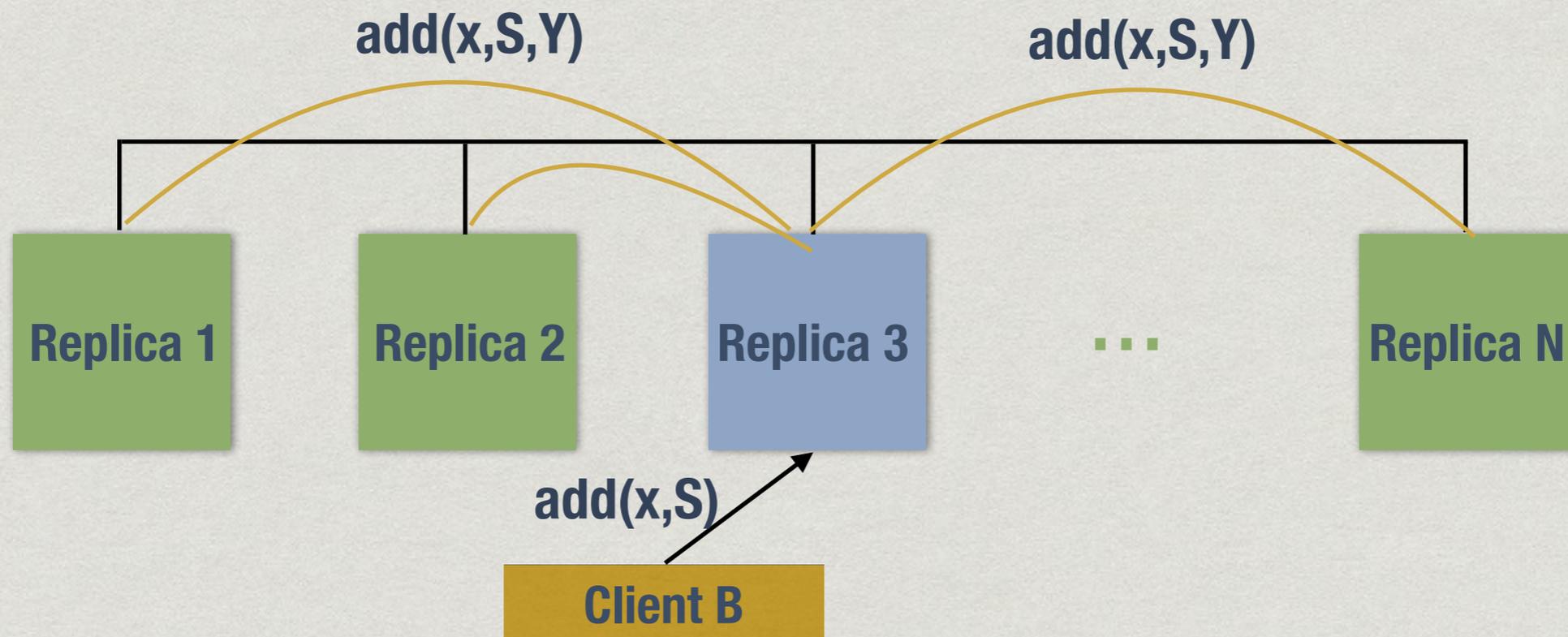
- * Represent a concurrent computation canonically
 - * Say a labelled partial order
- * Describe effect of a query based on partial order
 - * Reordering of concurrent updates does not matter
 - * Strong eventual consistency is guaranteed

CRDTs

- * Conflict-free Replicated Data Type: $D = (V, Q, U)$
 - * V — underlying universe of values
 - * Q — query operations
 - * U — update operations
- * For instance, for OR-sets,
 $Q = \{\text{member-of}\}$, $U = \{\text{add, remove}\}$

Runs of CRDTs

- * Recall that each update is
 - * locally applied at source replica,
 - * followed by N-1 messages to other replicas



Runs of CRDTs ...

- * Sequence of query, update and receive operations



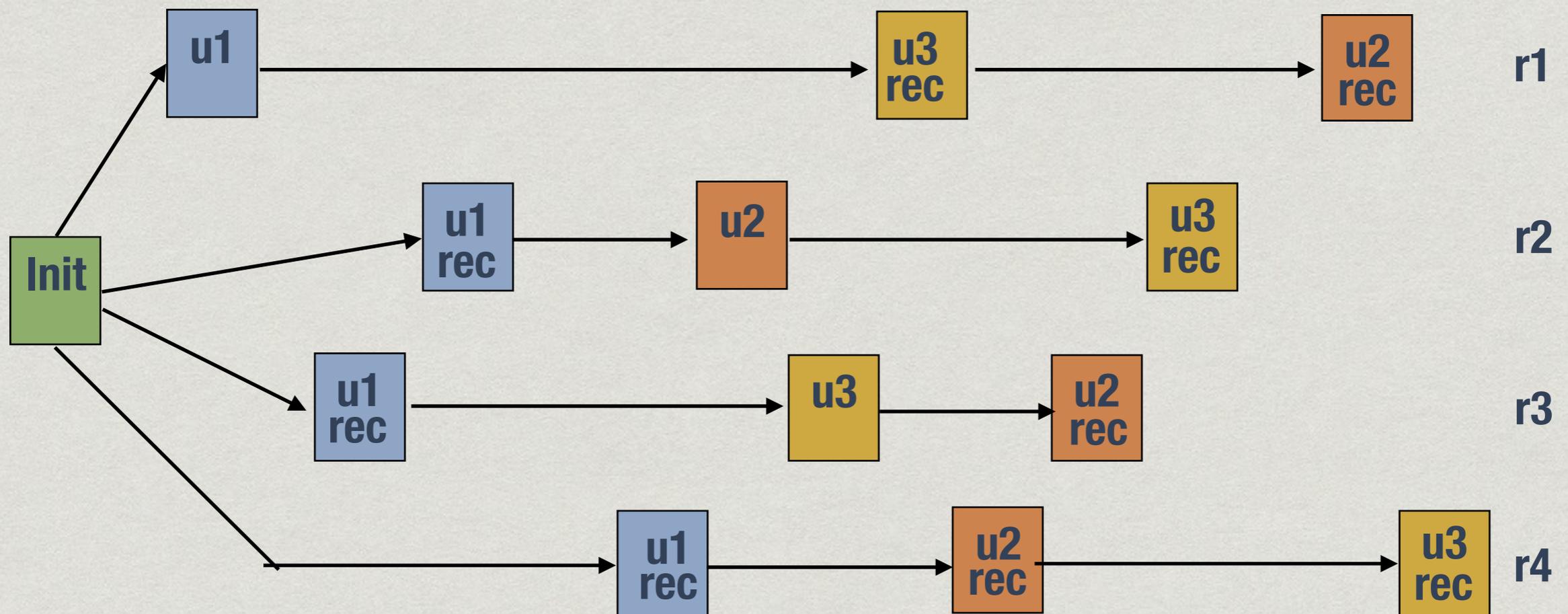
Runs of CRDTs ...

- * Ignore query operations
- * Associate a unique event with each update and receive operation



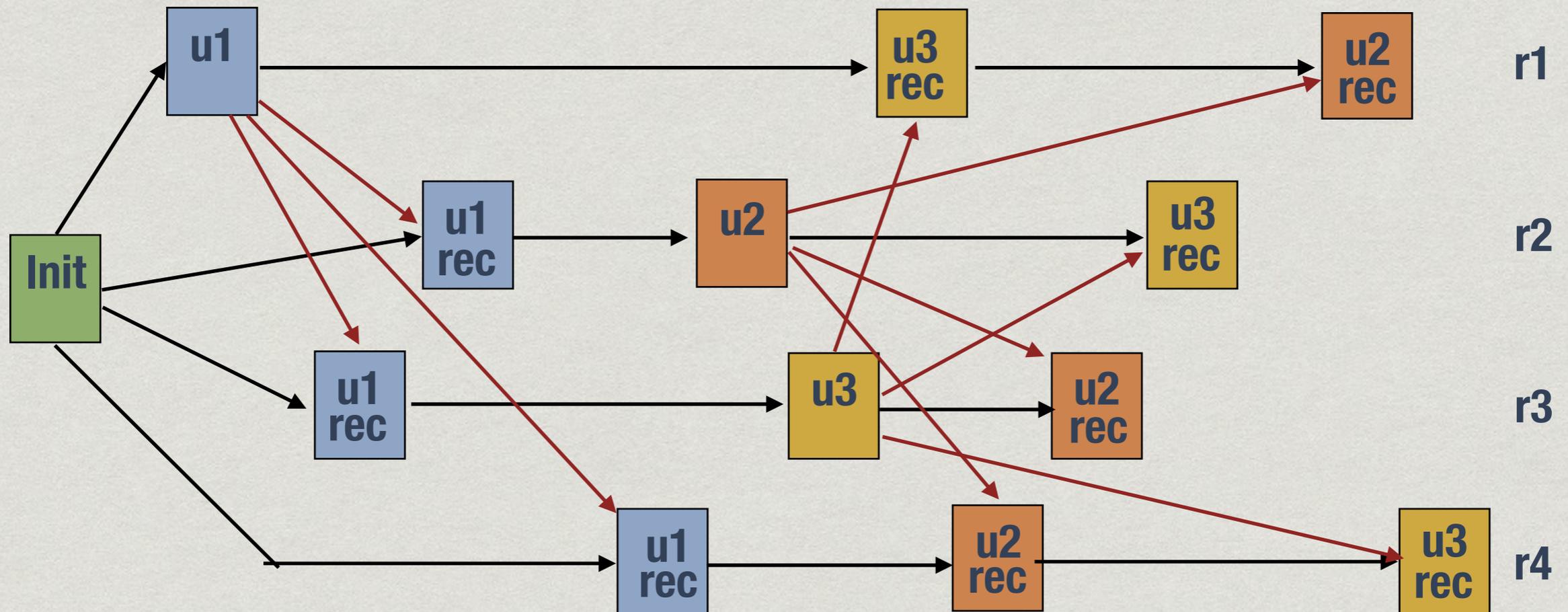
Runs of CRDTs ...

- * Replica order: total order of each replica's events



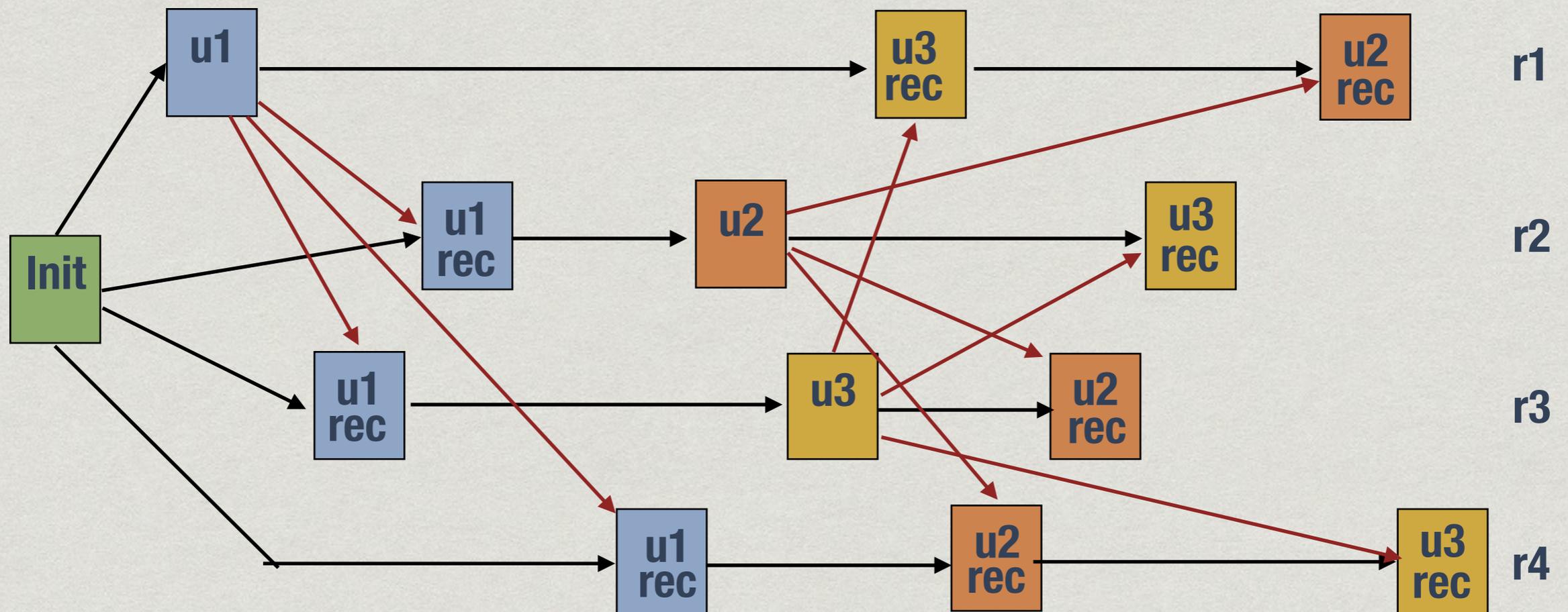
Runs of CRDTs ...

- * Delivery order: match receives to updates



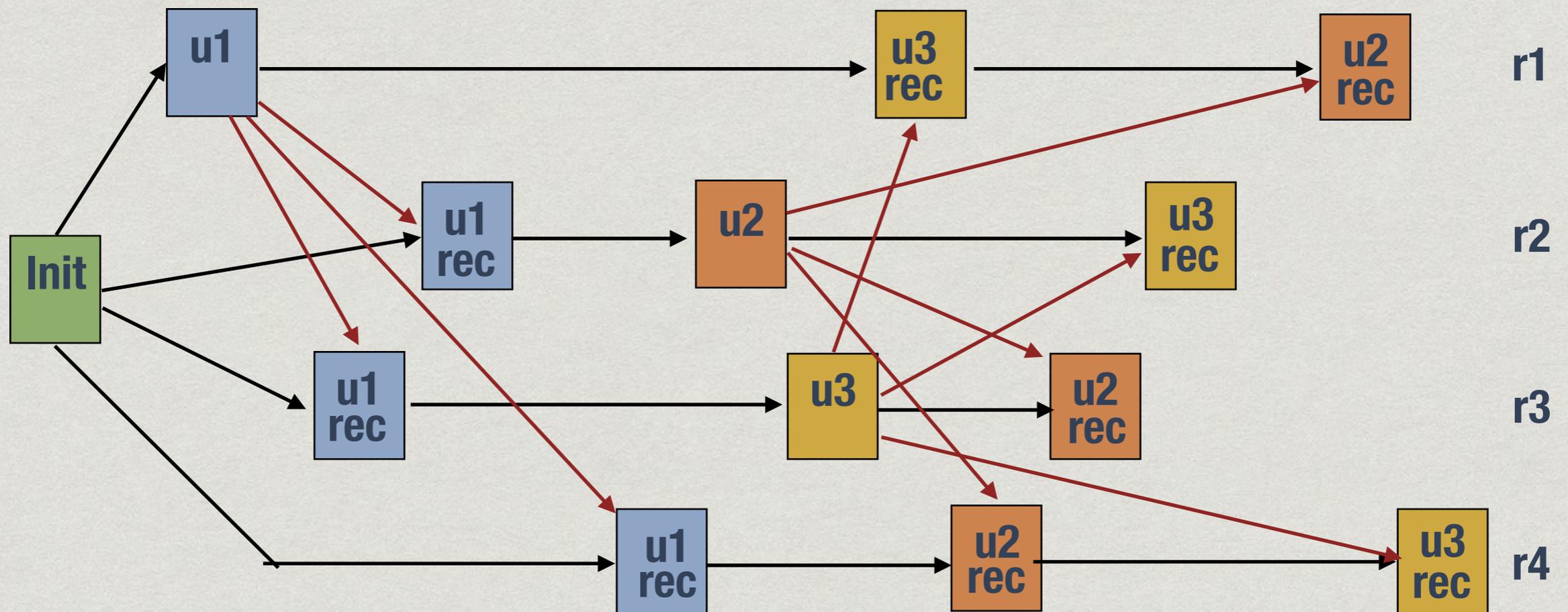
Runs of CRDTs ...

- * Happened before order on updates: Replica + Delivery
 - * Need not be transitive
 - * Causal delivery of messages makes it transitive



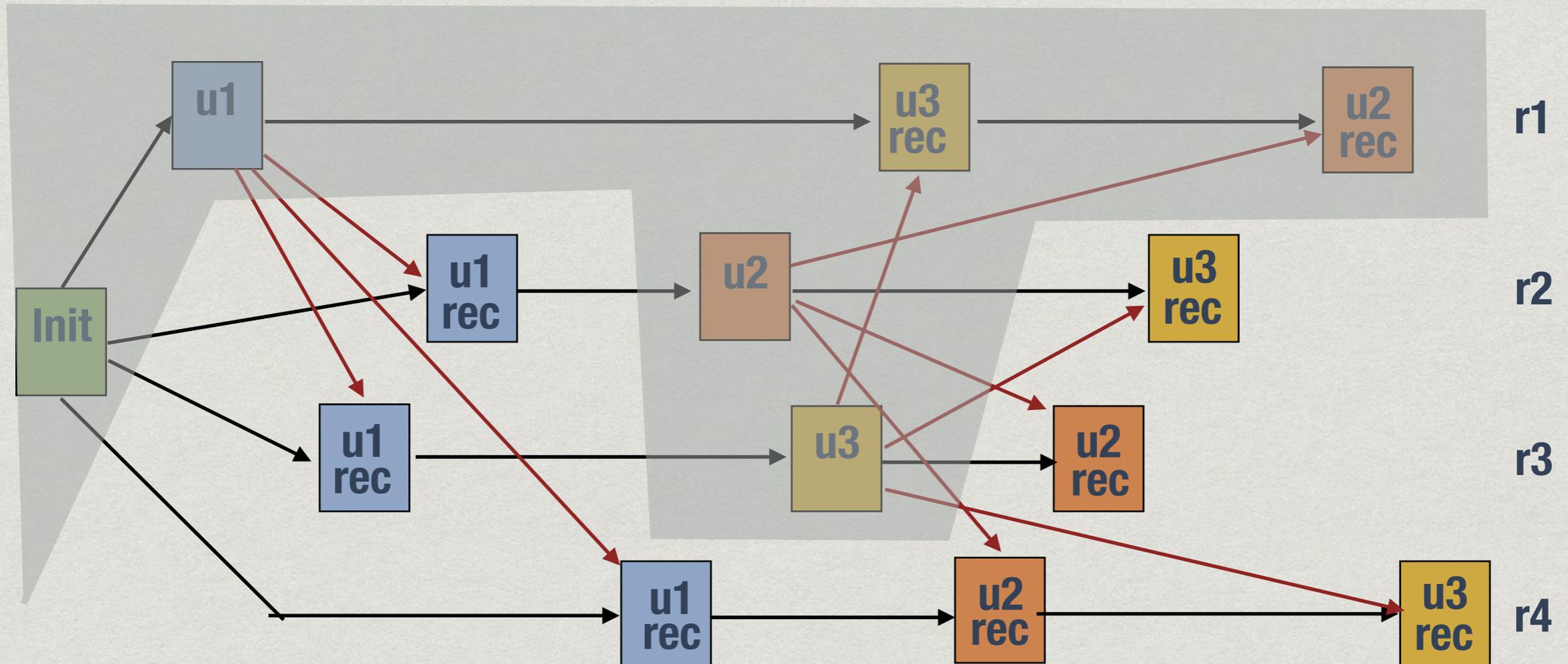
Runs of CRDTs ...

- * Local view of a replica
- * Whatever is visible below its maximal event



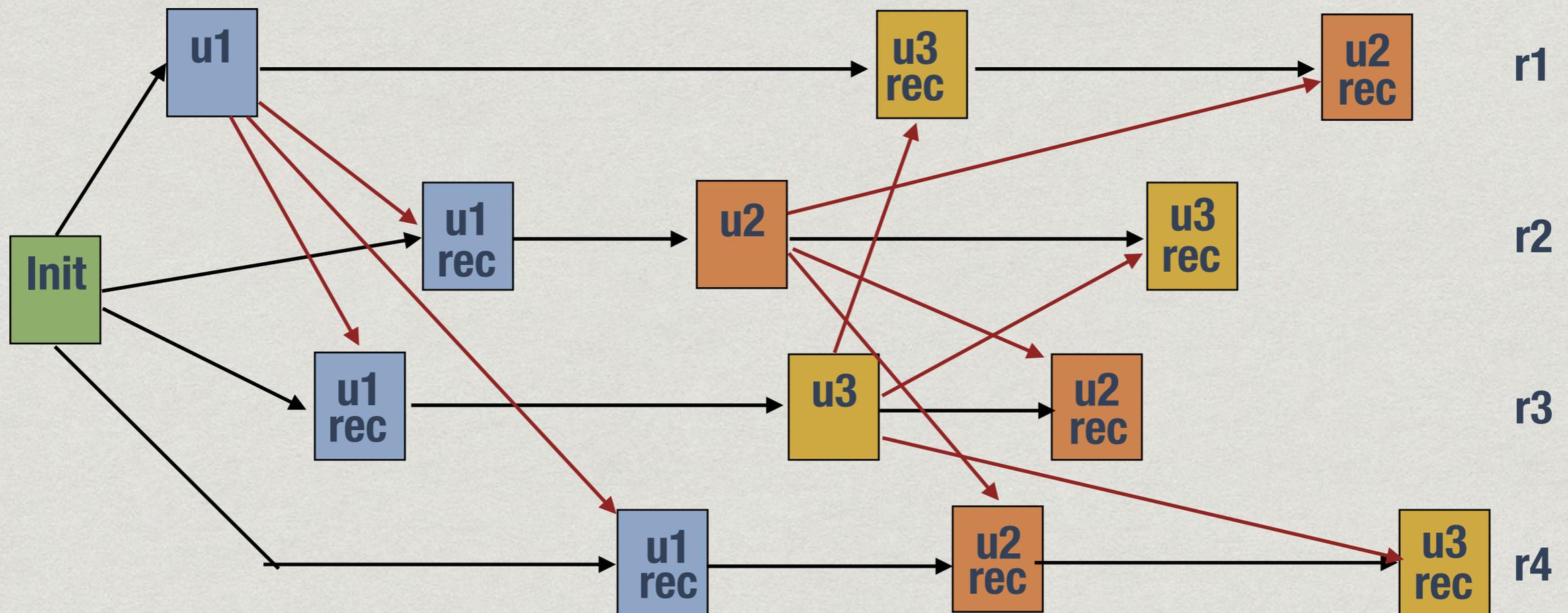
Runs of CRDTs ...

- * Local view of a replica
- * Whatever is visible below its maximal event



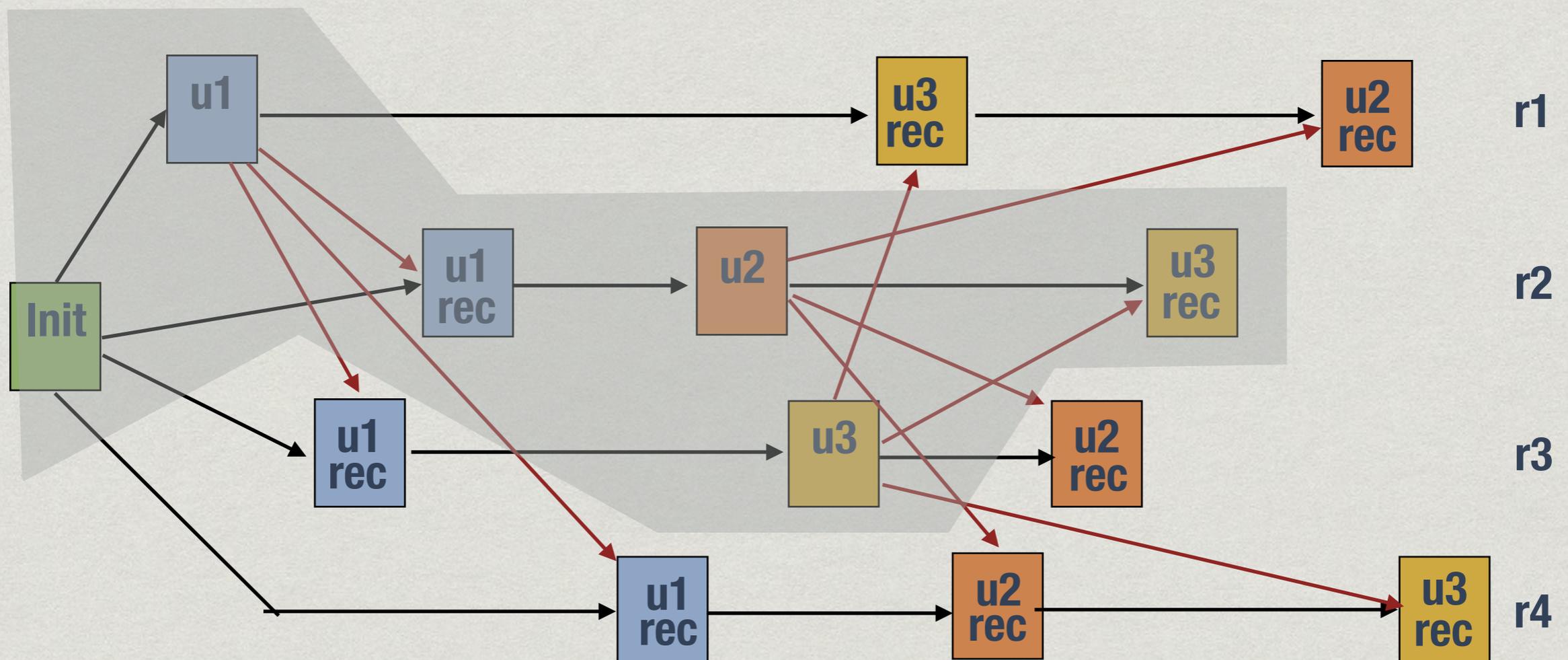
Runs of CRDTs ...

- * Local view of a replica
- * Whatever is visible below its maximal event



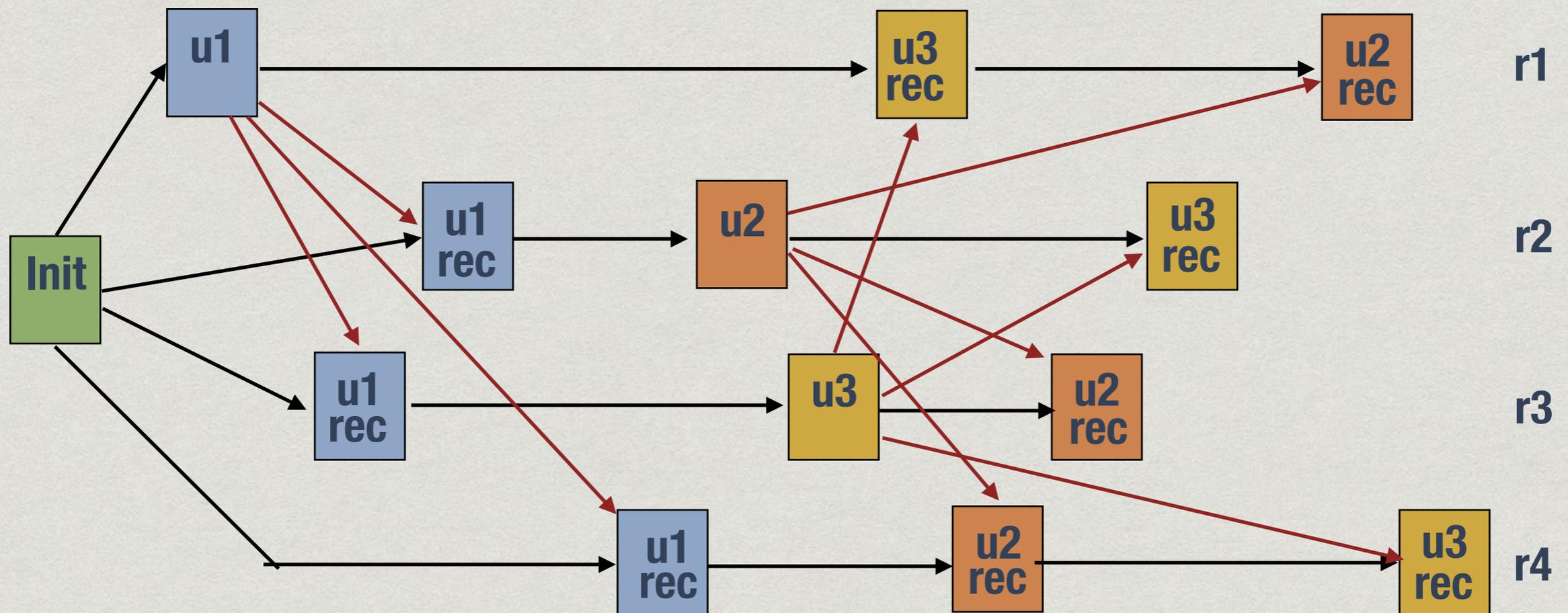
Runs of CRDTs ...

- * Local view of a replica
- * Whatever is visible below its maximal event



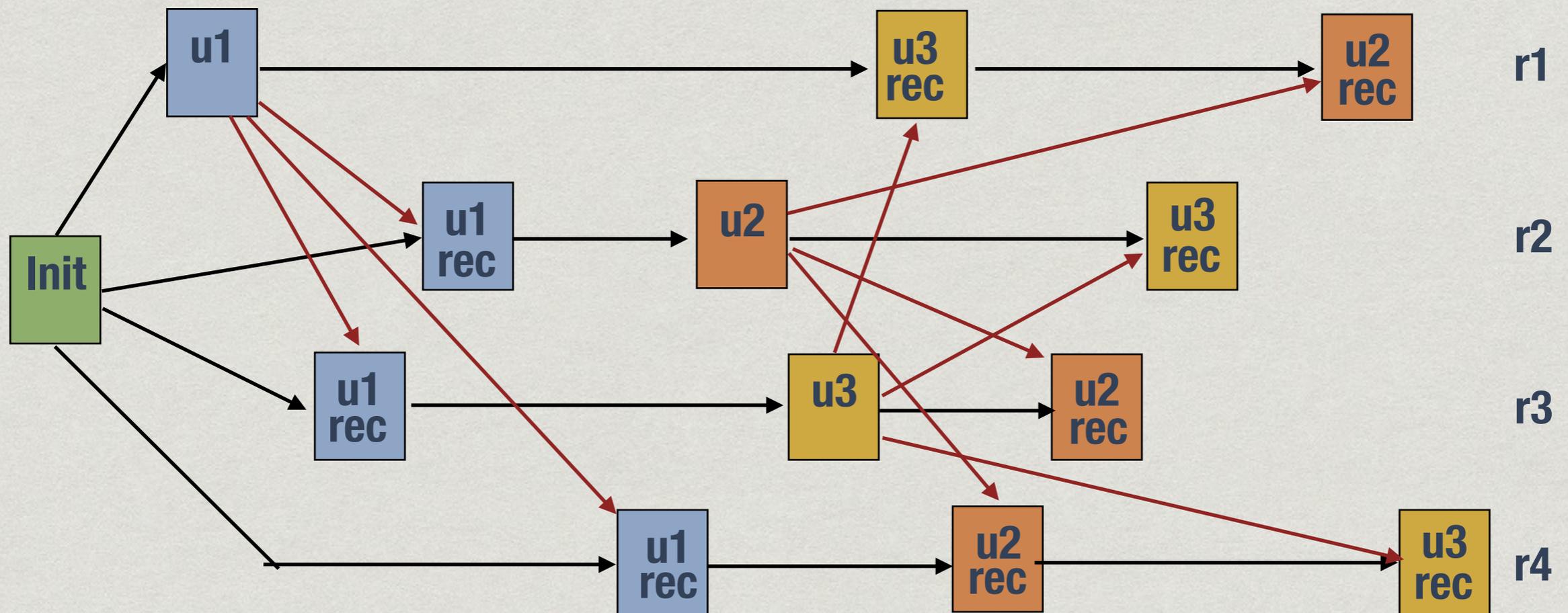
Runs of CRDTs ...

- * Local view of a replica
- * Whatever is visible below its maximal event



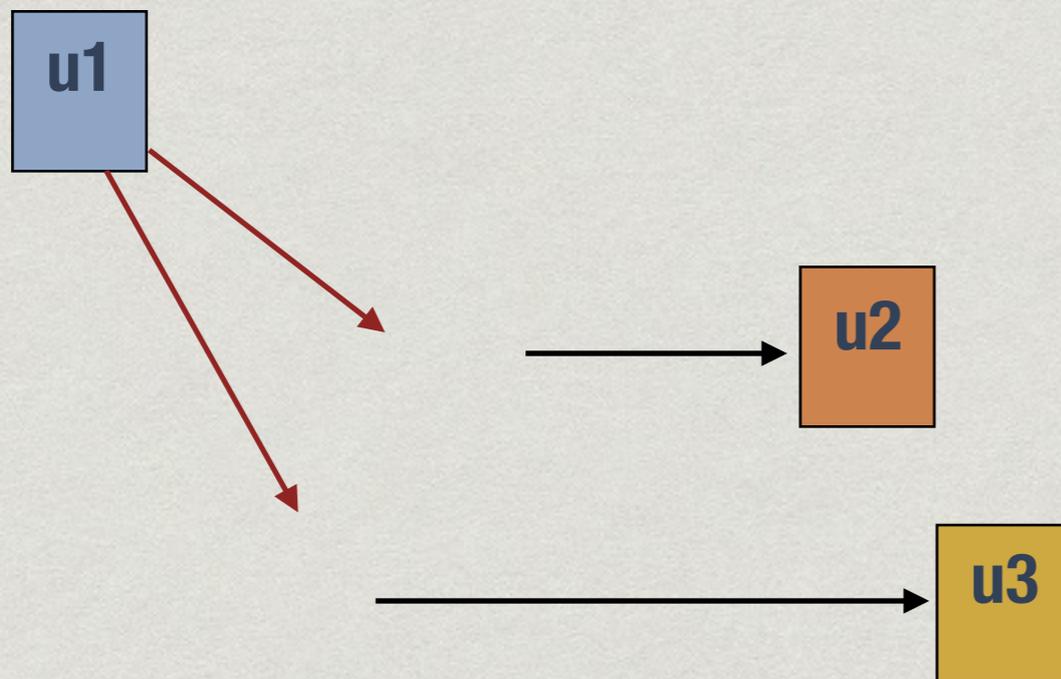
Runs of CRDTs ...

- * Even if updates are received locally in different orders, “happened before” on updates is the same



Runs of CRDTs ...

- * Even if updates are received locally in different orders, “happened before” on updates is the same



Declarative specification

- * Define queries in terms of partial order of updates in local view
- * For example: add wins in an OR-set
 - * Report “x in S” to be true if some maximal update is $\text{add}(x,S)$
 - * Concurrent $\text{add}(x,S)$, $\text{remove}(x,S)$ will both be maximal

Bounded past

- * Typically do not need entire local view to answer a query
- * Membership in OR-sets requires only maximal update for each element
 - * N events per element

Verification

- * Given a CRDT $D = (V, Q, U)$, does every run of D agree with the declarative specification?
- * Strategy
 - * Build a reference implementation from declarative specification
 - * Compare the behaviour of D with reference implementation

Finite-state implementations

- * Assume universe is bounded
- * Can use distributed timestamping to build a sophisticated distributed reference implementation [VMCAI 2015]
- * Asynchronous automata theory
- * Requires bounded concurrency for timestamps to be bounded

Global implementation

- * A simpler global implementation suffices for verification
- * Each update event is labelled by the source replica with an integer (will be bounded later)
- * Maintain sequence of updates applied at each replica
 - * either local update from client
 - * or remote update received from another replica

Later Appearance Record

- * Each replica's history is an LAR of updates
 - * $(u_1, l_1) (u_2, l_2) \dots (u_k, l_k)$
 - * u_j has details about update: source replica, arguments
 - * l_j is label tagged to u_j by source replica
- * Labels are consistent across LARs — (u_i, l) in r_1 and (u_j, l) in r_2 denote same update event
- * Maintain LAR for each replica

Causality and concurrency

- * Suppose r_3 receives (u, l) from r_1 and (u', l') from r_2
 - * If (u, l) is causally before (u', l') , (u, l) must appear in r_2 's LAR before (u', l')
 - * If (u, l) is not causally before (u', l') and (u', l') is not causally before (u, l) , they must have been concurrent
- * Can recover partial order and answer queries according to declarative specification

Pruning LARs

- * Only need to keep latest updates in each local view
- * If (u,l) generated by r is not latest for any other replica, remove all copies of (u,l)
- * To prune LARs, maintain a global table keeping track of which updates are pending (not yet delivered to all replicas)
- * Labels of pruned events can be safely reused

Outcome

- * Simple global reference implementation that conforms to declarative specification of CRDT
- * Reference implementation is bounded if we make suitable assumptions about operating environment
 - * Bounded universe
 - * Bounded message delivery delays

Verification strategy

- * Counter Example Guided Abstraction Refinement (CEGAR)
- * Build a finite-state abstraction of given CRDT
- * Compute synchronous product with reference implementation
- * If an incompatible state is reached, trace out corresponding bad run in CRDT
 - * If we find a bad run, we have found a bug
 - * If not, refine abstraction and repeat

Future work

- * Build a tool!
- * Extend formalisation of CRDTs to wider classes
 - * Composite CRDTs : Hash maps, graphs
 - * Multiple CRDTs with internal consistency constraints
- * Partially replicated data — local sync in Dropbox, Google Drive