# Specification and Implementation of Replicated List
## — The Jupiter Protocol Revisited

(Brief Announcement at PODC'2018)

**Hengfeng Wei**, Yu Huang, Jian Lu

Nanjing University

July 24, 2018

# Background

# Collaborative Text Editing Systems

(a) Google Docs

(b) Apache Wave

(c) Wikipedia

(d) LaTeX Editor

# Replication (for availability)

# Replication (for availability)


- Replicas respond to user operations immediately
  - Updates are propagated asynchronously

# List

$\textsc{Ins}(a, p)$ : Insert $a$ at position $p$.

$\textsc{Del}(p)$ : Delete an element at position $p$.

$\textsc{Read}$ : Return the list.

## List

$\textsc{Ins}(a, p)$ : Insert $a$ at position $p$.

$\textsc{Del}(p)$ : Delete an element at position $p$.

$\textsc{Read}$ : Return the list.

To implement a highly available replicated list object.

### Definition (Eventual Convergence (EC) [])

The lists at all replicas are identical *at quiescence*.

### Definition (Strong Eventual Consistency (SEC) [])

The lists at the replicas that *have executed the same set of user operations* are identical.

**Definition (Eventual Convergence (EC) [])**

The lists at all replicas are identical *at quiescence*.

**Definition (Strong Eventual Consistency (SEC) [])**

The lists at the replicas that *have executed the same set of user operations* are identical.

Specify little on *intermediate states* going through by replicas.

**Specification and Complexity of Collaborative Text Editing**

Hagit Attiya
Technion

Sebastian Burckhardt
Microsoft Research

Alexey Gotsman
IMDEA Software Institute

Adam Morrison
Technion

Hongseok Yang
University of Oxford

Marek Zawirski*
Inria & Sorbonne Universités,
UPMC Univ Paris 06, LIP6

## Strong/Weak List Specification []

Specify global properties on all (intermediate) states at all replicas.

**Specification and Complexity of Collaborative Text Editing**

Hagit Attiya
Technion

Sebastian Burckhardt
Microsoft Research

Alexey Gotsman
IMDEA Software Institute

Adam Morrison
Technion

Hongseok Yang
University of Oxford

Marek Zawirski*
Inria & Sorbonne Universités,
UPMC Univ Paris 06, LIP6

## Strong/Weak List Specification []

Specify global properties on all (intermediate) states at all replicas.

Proved: RGA [?] satisfies the strong list spec.

Conjecture: *Jupiter* [?] satisfies the weak list spec.

Does Jupiter satisfy the weak list specification?


Yes, it does.

# Weak List Specification

Definition (Weak List Specification $\mathcal{A}_{\text{weak}}$ [?])

Informally, $\mathcal{A}_{\text{weak}}$ requires the ordering between elements that are not deleted to be consistent across the system.

**Definition (Weak List Specification $\mathcal{A}_{\text{weak}}$ [?])**

Informally, $\mathcal{A}_{\text{weak}}$ requires the ordering between elements that are not deleted to be consistent across the system.

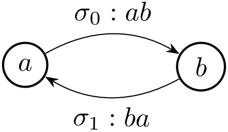
Pairwise state compatibility property:

$$\forall \sigma, \sigma' : a, b \in \sigma \cap \sigma' \implies (a \prec_{\sigma} b \iff a \prec_{\sigma'} b)$$

$$(\sigma, \sigma' : \text{ list}; \quad a, b : \text{ element}; \quad \prec_{\sigma}: \text{ precedes})$$

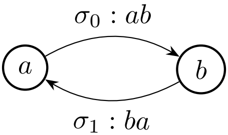
$$\forall \sigma, \sigma' : a, b \in \sigma \cap \sigma' \implies (a \prec_{\sigma} b \iff a \prec_{\sigma'} b)$$

$\boxed{\sigma_0 : ab}$ $\boxed{\sigma_1 : ba}$

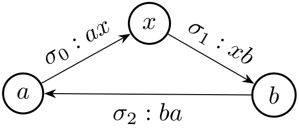



$$\forall \sigma, \sigma' : a, b \in \sigma \cap \sigma' \implies (a \prec_{\sigma} b \iff a \prec_{\sigma'} b)$$

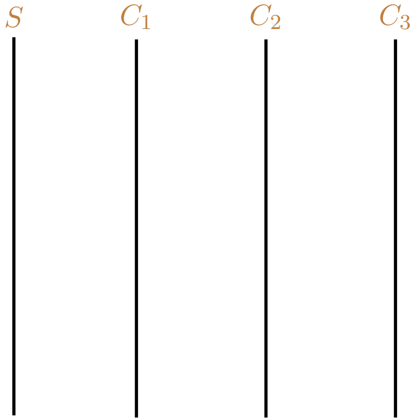
$\boxed{\sigma_0 : ab}$ $\boxed{\sigma_1 : ba}$




$\boxed{\sigma_0 : ax}$ $\boxed{\sigma_1 : xb}$ $\boxed{\sigma_2 : ba}$

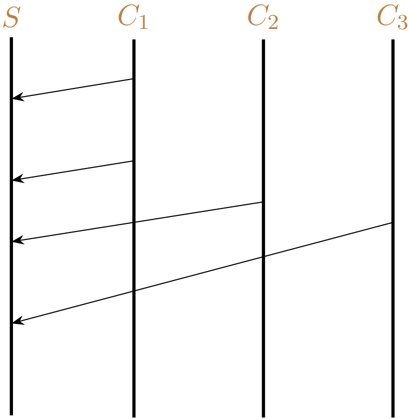


# Jupiter



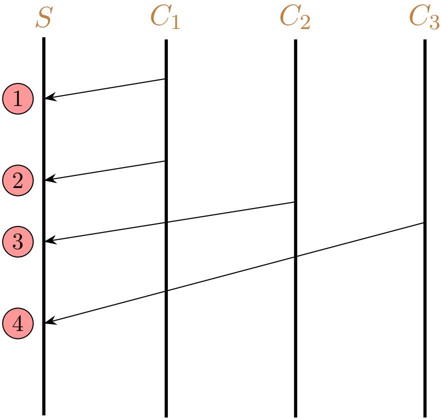

System model of Jupiter:

- client-server architecture



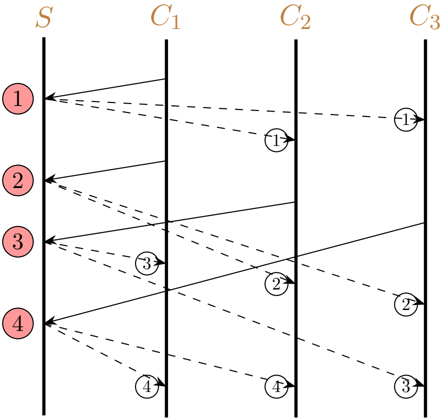

System model of Jupiter:

- client-server architecture
- client $\xrightarrow{\text{FIFO}}$ server




System model of Jupiter:

- client-server architecture
- client $\xrightarrow{\text{FIFO}}$ server
- totally ordered at the server

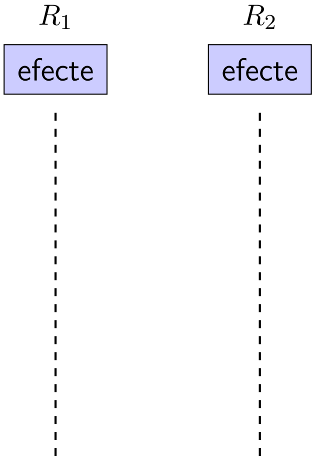



System model of Jupiter:

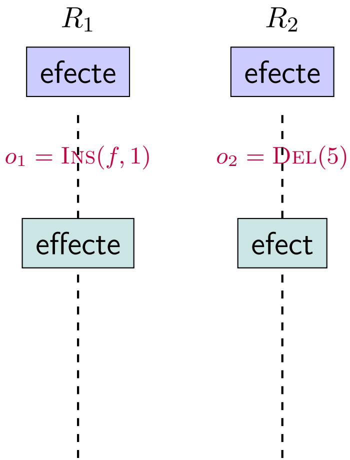
- client-server architecture
- client $\xrightarrow{\text{FIFO}}$ server
- totally ordered at the server
- server $\xrightarrow{\text{FIFO}}$ client

# OT (Operational Transformation) []

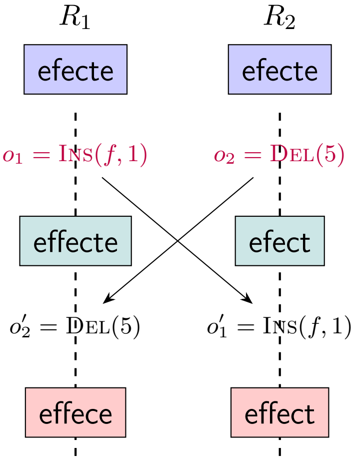
# OT (Operational Transformation) []



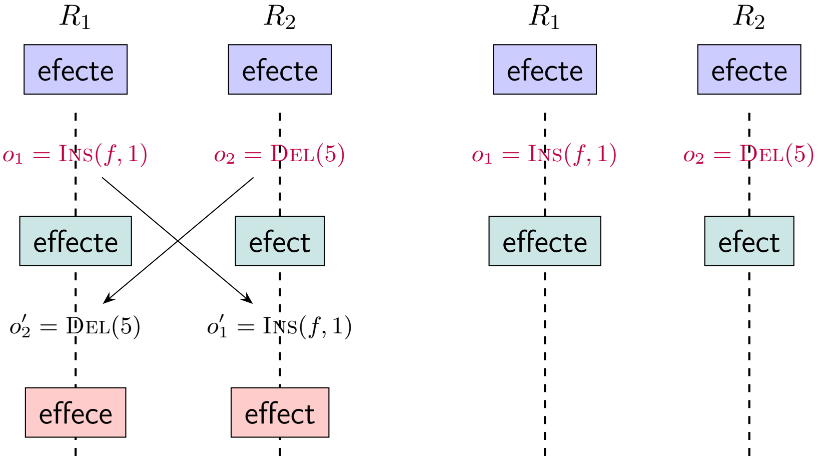
# OT (Operational Transformation) []



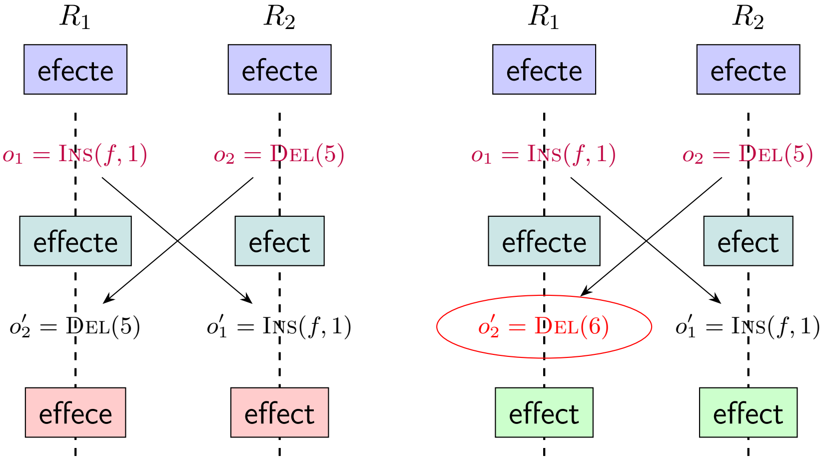
# OT (Operational Transformation) []

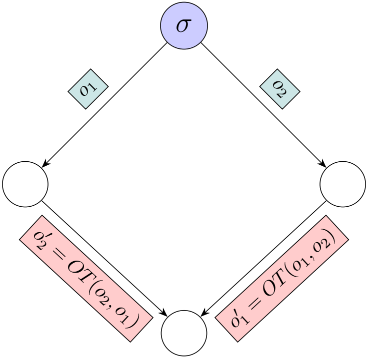


# OT (Operational Transformation) []



# OT (Operational Transformation) []



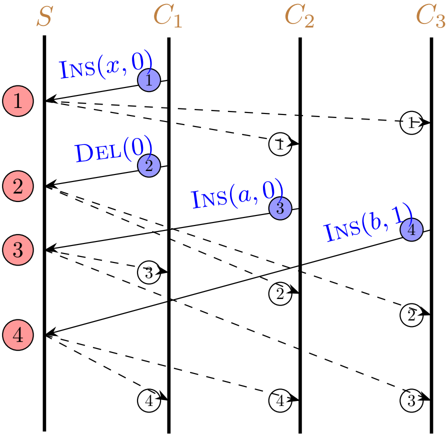



$$\sigma; o_1; o'_2 \equiv \sigma; o_2; o'_1 \ []$$

Jupiter uses $2D$ state spaces []
to manage how and when to perform OTs.

fig here

Consider a replicated system with $n$ clients.



# Each client maintains a $2D$ state space.

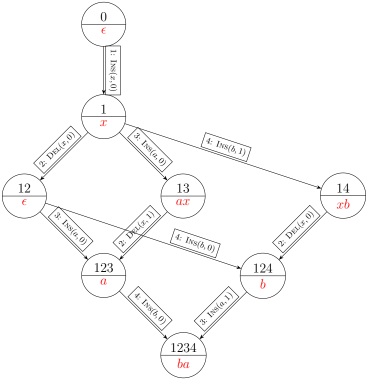
# The server maintains $n$ $2D$ state spaces, one for each client.

Global property on all replica states
specified by the weak list specification


Local view each replica maintains in Jupiter

# CJupiter (Compact Jupiter)

CJupiter maintains an $n$-ary ordered state space for each replica.



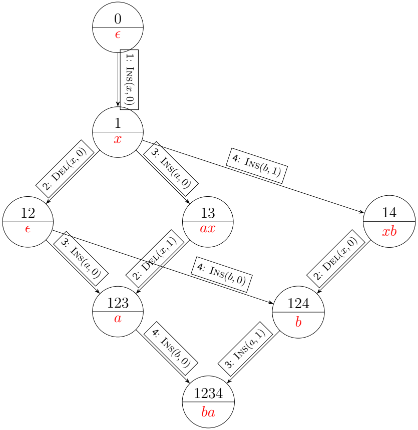

Nodes represent states

Edges are labeled with operations.

Edges from the same node are totally ordered by associated operations.

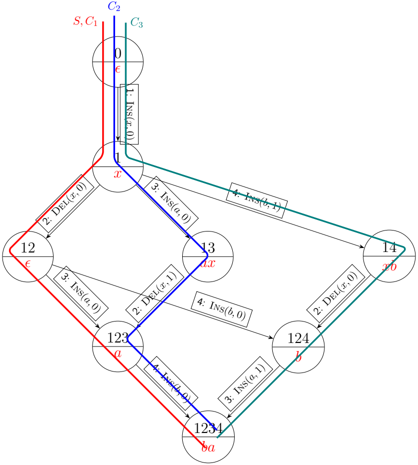
## Proposition ($n + 1 \to 1$ (Informal))

*At a high level, CJupiter maintains only one $n$-ary ordered state space.*



Proposition ($n + 1 \rightarrow 1$ (Informal))

*At a high level, CJupiter maintains only one $n$-ary ordered state space.*




Each replica behavior corresponds to a path going through this state space.

**Theorem (Equivalence)**

*Under the same schedule, the behaviors of corresponding replicas in CJupiter and Jupiter are the same.*

**Theorem (Equivalence)**

*Under the same schedule, the behaviors of corresponding replicas in CJupiter and Jupiter are the same.*

At the server side:

**Proposition ($n \leftrightarrow 1$ (Informal))**

*The single $n$-ary ordered state space at the server side in CJupiter is a compact representation of $n$ $2D$ state spaces at the server side in Jupiter.*

**Theorem (Equivalence)**

*Under the same schedule, the behaviors of corresponding replicas in CJupiter and Jupiter are the same.*

At the server side:

**Proposition ($n \leftrightarrow 1$ (Informal))**

*The single $n$-ary ordered state space at the server side in CJupiter is a compact representation of $n$ $2D$ state spaces at the server side in Jupiter.*

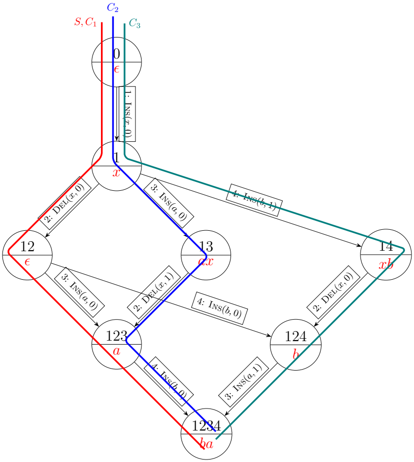
At the client side:

**Proposition ($1 \leftrightarrow 1$ (Informal))**

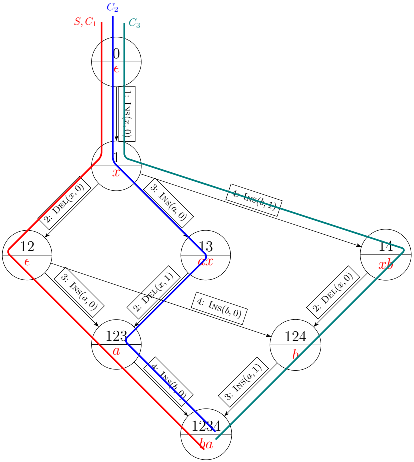
*Jupiter is slightly optimized in implementation at clients by eliminating redundant OTs than CJupiter.*

CJupiter satisfies the weak list specification.

We study a single $n$-ary ordered state space $\mathsf{CSS}_s$ at the server which provides a global view of all possible replica states.



We study a single $n$-ary ordered state space $\mathsf{CSS}_s$ at the server which provides a global view of all possible replica states.

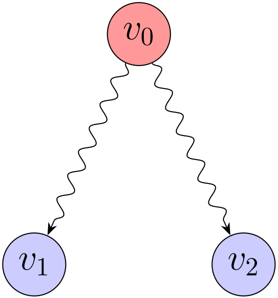



To show the pairwise state compatibility property in three steps.

1 Take any two nodes/states $v_1$ and $v_2$.

**Lemma (LCA (Lowest Common Ancestor))**

*Each pair of states in the $n$-ary ordered state space has a unique LCA.*

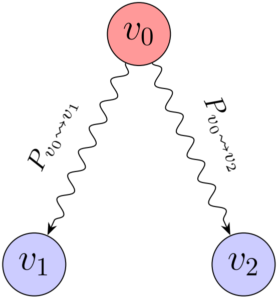



$$v_0 = \mathsf{LCA}(v_1, v_2)$$

2 Consider the paths to $v_1$ and $v_2$ from their LCA $v_0$.

**Lemma (Disjoint Paths)**

*The set of operations $O_{v_0 \rightsquigarrow v_1}$ along $P_{v_0 \rightsquigarrow v_1}$ is disjoint from the set of operations $O_{v_0 \rightsquigarrow v_2}$ along $P_{v_0 \rightsquigarrow v_2}$.*

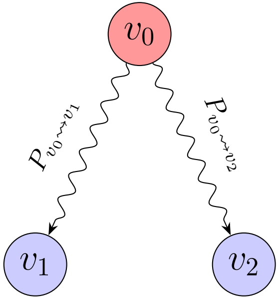



$v_0 = \mathsf{LCA}(v_1, v_2)$

③ Consider the states in these two paths.

**Lemma (Compatible Paths)**

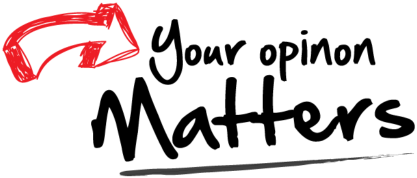
*Each pair of states consisting of one state in $P_{v_0 \rightsquigarrow v_1}$ and the other in $P_{v_0 \rightsquigarrow v_2}$ are compatible.*




$v_0 = \mathsf{LCA}(v_1, v_2)$

In particular,
$v_1$ and $v_2$ are compatible.


Thank
You!




Office 302

Mailbox: H016

hfwei@nju.edu.cn