Specification and Implementation of Replicated List
— The Jupiter Protocol Revisited

(OPODIS'2018)

Hengfeng Wei, Yu Huang, Jian Lu

Nanjing University

December 17, 2018

The Main Contribution

The Jupiter protocol [Nichols et al., 1995][a] for replicated list satisfies the weak list specification [Attiya et al., 2016][b].

[a] David A. Nichols et al. (1995). "High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System". In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology.* UIST '95. ACM, pp. 111–120.

[b] Hagit Attiya et al. (2016). "Specification and complexity of collaborative text editing". In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing.* PODC '16. ACM, pp. 259–268.

This was proposed as a *conjecture* in a PODC paper [Attiya et al., 2016].

Outline

1. Why do we care about replicated list?
2. What is the weak list specification $\mathcal{A}_{\text{weak}}$?
3. How does the Jupiter protocol work?
4. How to prove that Jupiter satisfies $\mathcal{A}_{\text{weak}}$?

Replicated List

Replicated Collaborative Text Editing Systems

(a) Google Docs

(b) Apache Wave

(c) Wikipedia

(d) LaTeX Editor

Replicas are required to respond to user operations immediately.

Updates are propagated to other replicas asynchronously.

Replicated list object: to model the core functionality

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

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

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

Weak List Specification

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

Definition (Weak List Specification $\mathcal{A}_{\text{weak}}$ [Attiya et al., 2016])

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

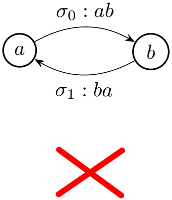
Specify a global property *on all states* across the system.

We show that $\mathcal{A}_{\text{weak}}$ can be rephrased as

Definition (Pairwise State Compatibility Property)

For any pair of list states, there cannot be two elements a and b such that a precedes b in one state but b precedes a in the other.

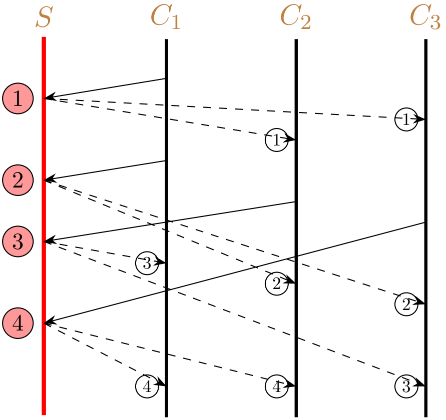
$\boxed{\sigma_0 : ab}$ $\boxed{\sigma_1 : ba}$



Jupiter

Jupiter adopts the client-server architecture [Nichols et al., 1995]:

$(n+1)$ replicas $\triangleq$ (n) Client + (1) Server

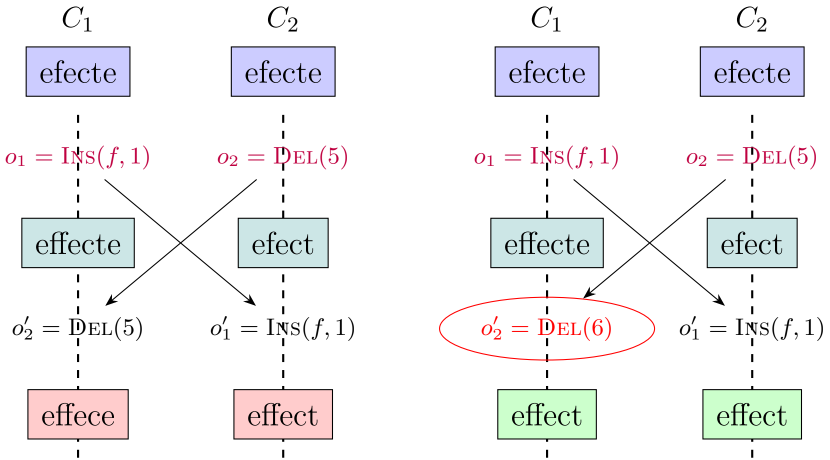


User $\xrightarrow{\text{Issue Ops}}$ Client $\xrightarrow[\text{Updates}]{\text{FIFO}}$ Server $\xrightarrow[\text{Updates}]{\text{FIFO}}$ Other Clients

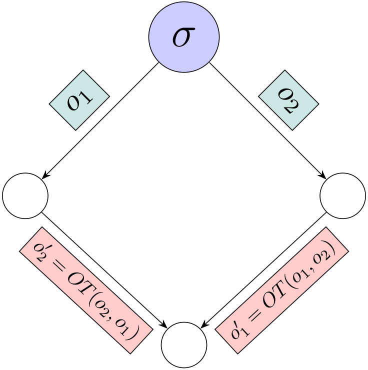
Respond Immed.

Challenge: Conflicts caused by concurrent operations

(The server is not drawn.)



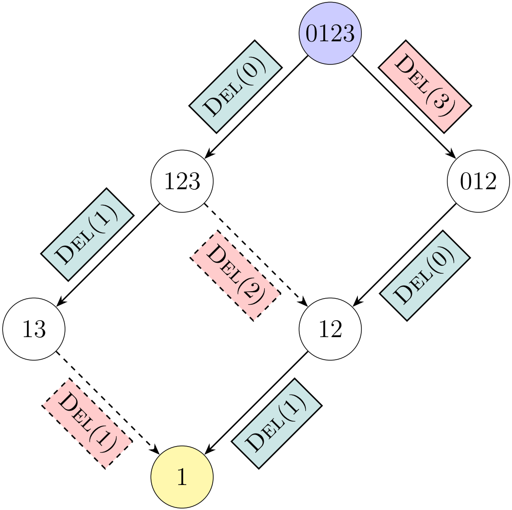
Solution: Operational Transformation (OT) [Ellis and Gibbs, 1989]



Commutative: $\sigma; o_1; o_2' \equiv \sigma; o_2; o_1'$

[Ellis and Gibbs, 1989]

Q : What if replicas diverge by ≥ 2 steps?



Key Challenge to Solve:

When a replica r receives an operation o from another replica redirected by the server, how should o be transformed?

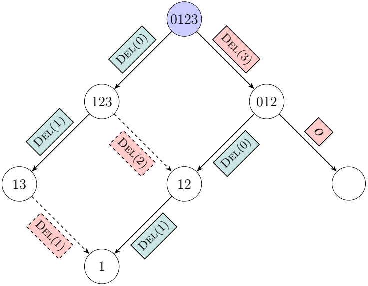
Transformed with which operations and in what order?

Key Ideas:

1. With concurrent operations previously executed at r
2. In the serialization order of operations established at the server

Jupiter uses 2D state spaces [Xu, Sun, and Li, 2014]

to manage the procedure of performing OTs [Ellis and Gibbs, 1989].

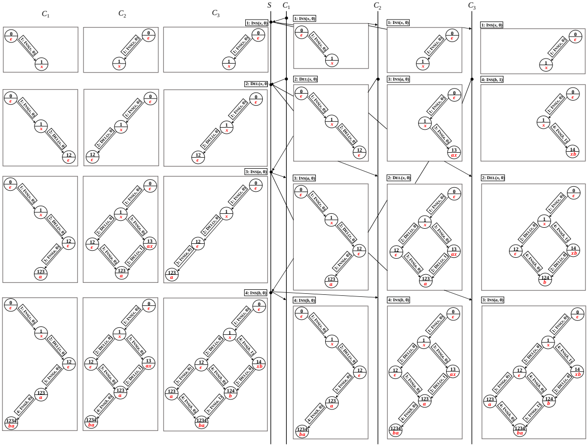


Edges are labeled with operations.

Local Dimension: For operations generated by the client

Global Dimension: For operations generated by others

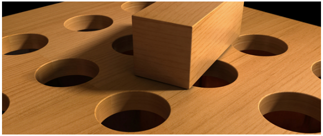
Each client maintains a $2D$ state space.



The server maintains n $(= 3)$ $2D$ state spaces, one for each client.

Mismatch!

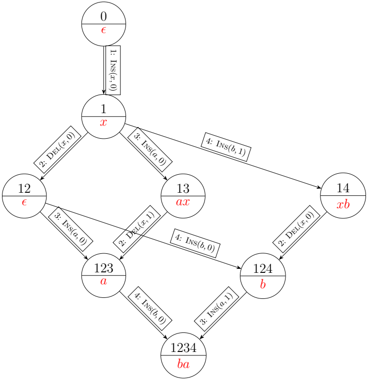
Global property on all replica states specified by $\mathcal{A}_{\text{weak}}$



Local view each replica maintains in Jupiter

CJupiter (Compact Jupiter)

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



There can be more than two edges coming from the same node.

Edges from the same node are totally ordered according to the serialization order of associated operations.

Theorem (Equivalence of CJupiter and Jupiter)

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

Schedule: ISSUE, SEND, and RECEIVE of operations

Behavior: A sequence of replica states

Equivalence from the perspectives of both the server and clients.

At the server side:

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

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

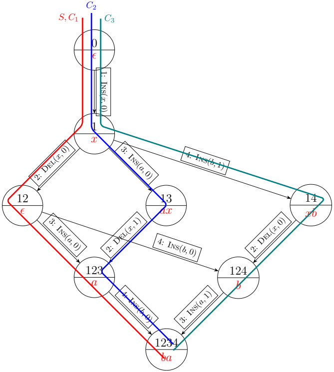
At the client side:

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

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

Proposition (Compactness of CJupiter (Informal))

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

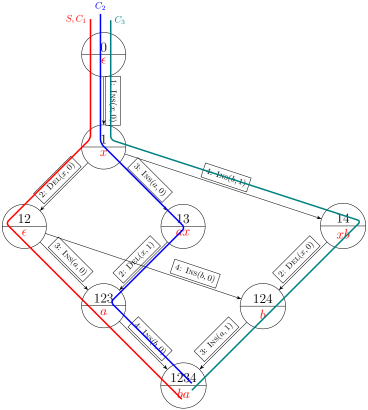


All replica states are represented in a single data structure.

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

CJupiter Satisfies the Weak List Specification

We focus on a single n-ary ordered state space.



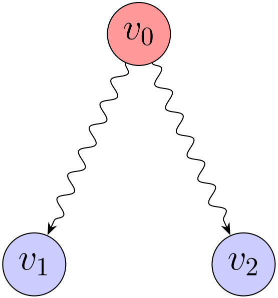
We show the pairwise state compatibility property in three steps.

By Contradiction, By Induction, and By Case Analysis.

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.

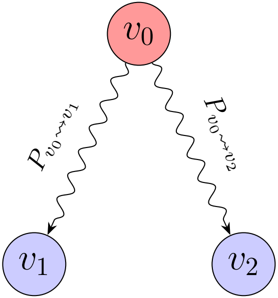


$$v_0 = \mathrm{LCA}(v_1, v_2)$$

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 $P_{v_0 \rightsquigarrow v_2}$.

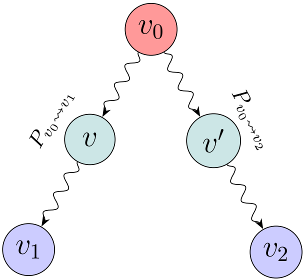


$$v_0 = \mathrm{LCA}(v_1, v_2)$$

③ Consider the states in these two paths.

Lemma (Compatible Paths)

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



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

In particular,
v_1 and v_2 are compatible.

The Main Contribution

The Jupiter protocol [Nichols et al., 1995][a] for replicated list satisfies the weak list specification [Attiya et al., 2016][b].

[a] David A. Nichols et al. (1995). "High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System". In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST '95. ACM, pp. 111–120.

[b] Hagit Attiya et al. (2016). "Specification and complexity of collaborative text editing". In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. PODC '16. ACM, pp. 259–268.

This was proposed as a *conjecture* in a PODC paper [Attiya et al., 2016].

Model checking/verifying a family of Jupiter protocols using TLA+/TLAPS

AbsJupiter (Wei@TR'18)

↑

CJupiter (Wei@OPODIS'18)

↑

XJupiter (Xu@CSCW'14)

↑

AJupiter (Attiya@PersonalComm'17)

Refinement


Thank
You!

Attiya, Hagit et al. (2016). “Specification and complexity of collaborative text editing”. In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. PODC ’16. ACM, pp. 259–268.

Ellis, C. A. and S. J. Gibbs (1989). “Concurrency Control in Groupware Systems”. In: *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’89. ACM, pp. 399–407.

Nichols, David A. et al. (1995). “High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System”. In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST ’95. ACM, pp. 111–120.

Roh, Hyun-Gul et al. (Mar. 2011). “Replicated Abstract Data Types: Building Blocks for Collaborative Applications”. In: *J. Parallel Distrib. Comput.* 71.3, pp. 354–368.

Shapiro, Marc et al. (2011). “Conflict-free Replicated Data Types”. In: *Proceedings of the 13th International Conference on Stabilization, Safety, and Security of Distributed Systems*. SSS’11. Springer-Verlag, pp. 386–400.

Xu, Yi, Chengzheng Sun, and Mo Li (2014). “Achieving Convergence in Operational Transformation: Conditions, Mechanisms and Systems”. In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work*. CSCW ’14. ACM, pp. 505–518.

Backup

Does Jupiter satisfy the weak list specification?

Yes, it does.

Replication (for availability)

Replicas respond to user operations immediately

Updates are propagated asynchronously

Definition (Eventual Convergence [Ellis and Gibbs, 1989])

The lists are identical at all replicas at quiescence, i.e., all update operations have been executed at all replicas.

Definition (Strong Eventual Consistency [Shapiro et al., 2011])

The lists are identical at all replicas whenever after executing the same set of update operations.

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

Strong/weak list specification [Attiya et al., 2016]

Specify global properties on all states across the system.

Specification and Complexity of Collaborative Text Editing

Hagit Attiya
Technion

Sebastian Burckhardt
Microsoft Research

Alexey Gotsman
IMDEA Software Institute

Adam Morrison
Technion

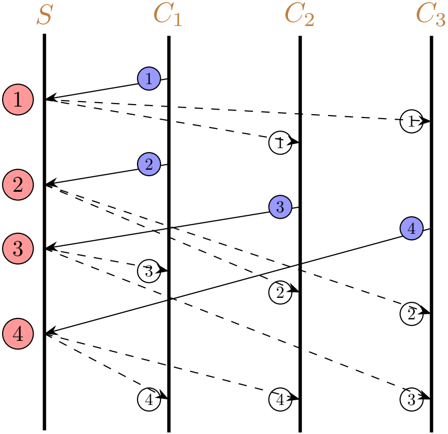
Hongseok Yang
University of Oxford

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

Proved: RGA [Roh et al., 2011] satisfies the strong list specification.

Conjecture: Jupiter [Nichols et al., 1995] satisfies the weak list specification.

It is still challenging to achieve convergence despite the server.



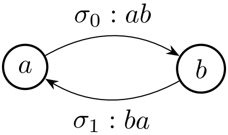
Serializability may not be desirable.

It does not imply that clients process operations in the same order.

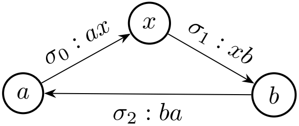
$$\forall \sigma, \sigma' : a, b \in \sigma \cap \sigma' \implies (a \prec_{\sigma} b \iff a \prec_{\sigma'} b)$$

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

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



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



OT functions for a replicated list object [Ellis and Gibbs, 1989]

$$OT\Big(\textsc{Ins}(a_1, p_1, pr_1), \textsc{Ins}(a_2, p_2, pr_2)\Big) = \begin{cases} \textsc{Ins}(a_1, p_1, pr_1) & p_1 < p_2 \\ \textsc{Ins}(a_1, p_1 + 1, pr_1) & p_1 > p_2 \\ \text{NOP} & p_1 = p_2 \land a_1 = a_2 \\ \textsc{Ins}(a_1, p_1 + 1, pr_1) & p_1 = p_2 \land a_1 \neq a_2 \land pr_1 > pr_2 \\ \textsc{Ins}(a_1, p_1, pr_1) & p_1 = p_2 \land a_1 \neq a_2 \land pr_1 \leq pr_2 \end{cases}$$

$$OT\Big(\textsc{Ins}(a_1, p_1, pr_1), \textsc{Del}(\_, p_2, pr_2)\Big) = \begin{cases} \textsc{Ins}(a_1, p_1, pr_1) & p_1 \leq p_2 \\ \textsc{Ins}(a_1, p_1 - 1, pr_1) & p_1 > p_2 \end{cases}$$

$$OT\Big(\textsc{Del}(\_, p_1, pr_1), \textsc{Ins}(a_2, p_2, pr_2)\Big) = \begin{cases} \textsc{Del}(\_, p_1, pr_1) & p_1 < p_2 \\ \textsc{Del}(\_, p_1 + 1, pr_1) & p_1 \geq p_2 \end{cases}$$

$$OT\Big(\textsc{Del}(\_, p_1, pr_1), \textsc{Del}(\_, p_2, pr_2)\Big) = \begin{cases} \textsc{Del}(\_, p_1, pr_1) & p_1 < p_2 \\ \textsc{Del}(\_, p_1 - 1, pr_1) & p_1 > p_2 \\ \text{NOP} & p_1 = p_2 \end{cases}$$

Consider a replicated system with n $(= 3)$ clients.