Setup: 3 computers and a website to keep up to date
Editing is nicer to do on a personal machine
Web publishing tools may not know how to do synchronization
Push and pull changes, may use heterogeneous concrete formats
Time to go home, synchronize the laptop
Urgent update needed (contact email needs to be changed), do it directly from home using the notebook and up-to-date local version of the website.
Update done, time to synchronize
and Synchronization fails... We need a better tool for this. Will not go into details why this failed, this shows the setting we’re interested in
The Setting

- Optimistic replication
- State-based synchronization
- Edit often, synchronize rarely
- Many conflicts

Not well studied in the literature
We propose a better tool for this setting
Joke about many different affiliations and synchronization?
Conflict repair (reconciliation) may be automatic require human supervision (our concern)
Genuine conflicts: unrelated different values
False conflicts: relation undetected

For optimistic replication with many devices that are seldom synchronized, many real conflicts
Dealing with Conflicts

Ignore them
  No convergence
Repair them: choose a better value
  Create a new value dominating the conflict
Convergence?
real conflict
same decision: choose v2’s value
causal history approach does not detect the agreement
Our Approach

- Record agreement explicitly with equivalence
- Equivalence enables optimistic reconciliation
- Bounded representation in $O(n^4)$ of the history graph with equivalence edges
Equivalence
\( v_a \) and \( v_b \) conflict
new edges created only from newly introduced values
no edge created between previously existing values
$v_a$ and $v_b$ conflict
Choose $v_b$
Equivalence Class Merging

Explicit equivalence edge allow distributed reconciliation
Replicas independently decide events are equivalent

No need for a central decision replica

→ distributed reconciliation
several events, all equivalent from a human point of view (but not yet from the system)
at least 3 synchronizations needed, in many systems the order might matter
in our approach, the order does not matter
Work machine
Laptop
Web Server
Home machine
w₀
s₀
l₀
h₀
Conflict

\( l_0 \quad s_0 \quad \text{Web Server} \)

\( w_0 \quad l_0 \quad s_0 \quad h_0 \)

Work machine
Laptop
Home machine
Conflict repaired by equivalence at laptop
Conflict repaired by equivalence at web server
Conflict repaired by equivalence at both sites
Conflict

![Diagram showing conflict between work machine, laptop, and home machine]
Conflict repaired by equivalence at both sites
Equivalence reached “globally”; conflict?
No conflict, already equivalent
Optimistic Editing
Create a new dominating event before the system has converged does not necessarily preclude convergence.

Not only distributed reconciliation but optimistic reconciliation: one may edit before conflict resolution has reached everyone.
Conflict repaired by equivalence
Conflict repaired by equivalence
(Optimistic) edit on laptop
Any further synchronization will propagate v2l safely, even though neither work nor web talked to laptop since synchronizing (optimistic reconciliation)
Interesting new kind of conflicts
Conflicts may be solved in incompatible ways

- choose new incompatible events
- choose a different event as the better one
Work machine
Laptop
Web Server

$w_0$

$L_0$

Work machine
Laptop
Conflict: work machine value chosen

Web Server

Work machine

Laptop

$w_0 \quad l_0 \quad l_0 \quad w_0$
Conflict: laptop value chosen

- Work machine
- Laptop
- Web Server

$w_0 l_0 w_1 l_1 w_0$
No dominant (maximal) event
Latest events
Notion of maximal event
classes, components, latest events, maximal events
new edges only created to maximal events
A Bounded Representation
Slight generalization to simplify the examples and consistent with the paper
One simple way to restrict communication, otherwise unbounded storage is needed
n: number of replicas
Reciprocal Communication

A

Cannot talk to A

B

Cannot talk to A, C

C

Cannot talk to C

Cannot talk to A, C
Open Events

New edges only created to maximal events

Event ‘v’ considered open when
  sent a as latest event to replica A
  no communication from A since
  ⇒ A may have created edges to ‘v’

Events that matter: equivalent to open events
Sparse Representation

1. Events considered open
2. Most recent events taken into account by latest events
3. Oldest events equivalent to or dominating open events
4. Equivalence class of open events
Most recent events taken into account

Oldest events dominating or equivalent

Most recent events taken into account
At most $n^2$ events from each replica considered open
This is a tight limit
Goal: create $n(n-1)/2$ open events from laptop, $n=4$

Events considered open shown
Green event: reason it is open at laptop

Green: events that have been latest and not communicated back to Laptop
Work machine
Laptop
Web Server
Home machine
First round: 3 events open at Laptop because of Home

1, 2, and 3 considered open because they have been latest events at the Home machine and the laptop has not (transitively) heard back from it.
Second round
Cannot send v4 to Home because we have not heard from it
Second round: 2 new open events because of Web
Last round: can only send event to Work
Last round: 3 + 2 + 1 open events at Laptop

n*(n−1)/2 events considered open at Laptop
Simulations: how many open events?

Implementation and integration with Harmony

http://www.seas.upenn.edu/~harmony/
Take home points

Very optimistic replication: many conflicts
Equivalence enables optimistic reconciliation
Correct bounded representation in $O(n^4)$ (as worst case)