

Agenda

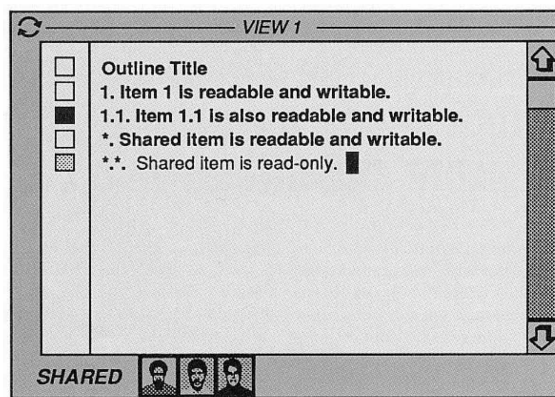
- Optimistic replication
 - CVS, Subversion
 - Duplicated databases
 - Collaborative editing requirements
 - Operational transformation: properties for convergence, transformation functions, algorithms

Collaborative editing: from users to community of users



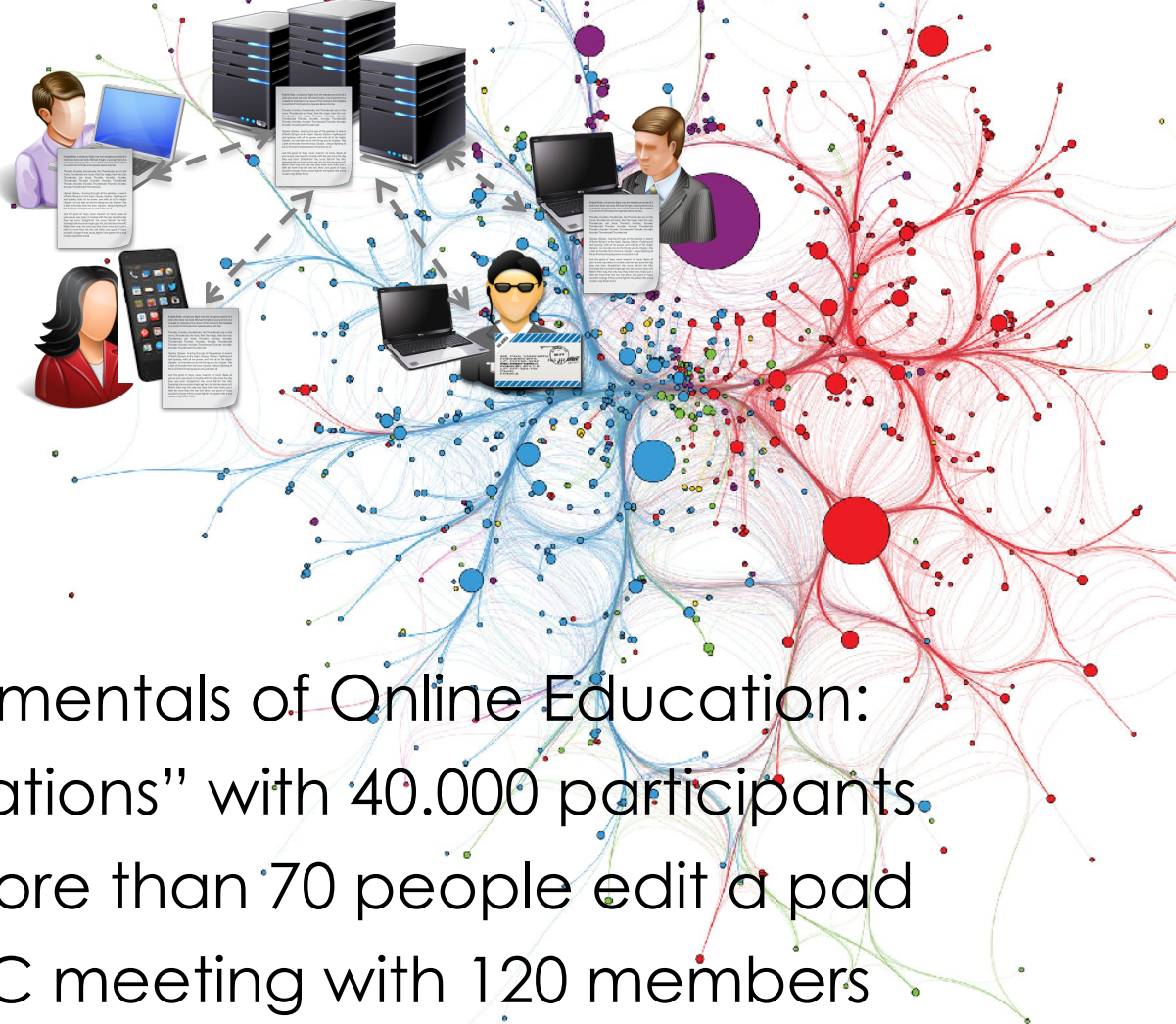
“Isn't it chaotic to all edit in the same document, even the same paragraph, at the same time?”

“Why would a group ever want to edit in the same line of text at the same time?”



GROVE, 1989

Collaborative editing: from users to community of users



2013: MOOC “Fundamentals of Online Education: Planning and Applications” with 40.000 participants.

2016: Nuit debout, more than 70 people edit a pad

2018: online CSCW PC meeting with 120 members

Collaborative editing: from users to community of users

Real-time Wikipedia

The screenshot shows the Wikipedia revision history page for the article 'Ponte Morandi'. The page title is 'Ponte Morandi: Revision history'. Below the title, there is a search bar for revisions, with filters for 'From year (and earlier): 2018', 'From month (and earlier): all', and 'Tag filter:'. The page lists 24 revisions, each with a date, time, and user name. The revisions are sorted by newest first. The first revision is from 12:22, 14 August 2018, by Pigsonthewing, with a size of 4,619 bytes and a change of -1. The last revision is from 12:12, 14 August 2018, by Prioryman, with a size of 3,766 bytes and a change of +354. The page also includes a sidebar with links to various Wikipedia pages and a footer with the text 'R plication et coh rence de donn es'.

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Atom
Upload file
Special pages
Page information
Wikidata item

Languages

Ponte Morandi: Revision history

View logs for this page (view filter log)

Search for revisions

From year (and earlier): 2018 From month (and earlier): all Tag filter: Show

For any version listed below, click on its date to view it. For more help, see [Help:Page history](#) and [Help:Edit summary](#).
External tools: [Revision history statistics](#) · [Revision history search](#) · [Edits by user](#) · [Number of watchers](#) · [Page view statistics](#) · [Fix dead links](#)

(cur) = difference from current version, (prev) = difference from preceding version, m = minor edit, → = section edit, ← = automatic edit summary
(newest | oldest) View (newer 24 | older 24) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- [\(cur | prev\)](#) 12:22, 14 August 2018 Pigsonthewing (talk | contribs) m . . (4,619 bytes) (-1) . . (→top: ce) (undo)
- [\(cur | prev\)](#) 12:22, 14 August 2018 Pigsonthewing (talk | contribs) m . . (4,620 bytes) (-4) . . (→References: ce) (undo)
- [\(cur | prev\)](#) 12:22, 14 August 2018 Pigsonthewing (talk | contribs) . . (4,624 bytes) (-44) . . (→Collapse of the bridge: redirected here) (undo)
- [\(cur | prev\)](#) 12:21, 14 August 2018 Pigsonthewing (talk | contribs) . . (4,668 bytes) (+29) . . (→top: fmt) (undo)
- [\(cur | prev\)](#) 12:21, 14 August 2018 Pigsonthewing (talk | contribs) . . (4,639 bytes) (+12) . . {{{current}}} (undo)
- [\(cur | prev\)](#) 12:20, 14 August 2018 Pigsonthewing (talk | contribs) . . (4,627 bytes) (-29) . . (ce) (undo)
- [\(cur | prev\)](#) 12:20, 14 August 2018 37.159.90.23 (talk) . . (4,656 bytes) (0) . . (undo) (Tags: Mobile edit, Mobile web edit)
- [\(cur | prev\)](#) 12:19, 14 August 2018 Themandrak (talk | contribs) m . . (4,656 bytes) (+269) . . (Added reference) (undo)
- [\(cur | prev\)](#) 12:19, 14 August 2018 Avaya1 (talk | contribs) . . (4,387 bytes) (+17) . . (undo)
- [\(cur | prev\)](#) 12:19, 14 August 2018 Prioryman (talk | contribs) . . (4,370 bytes) (+46) . . (→top: - bold name) (undo)
- [\(cur | prev\)](#) 12:19, 14 August 2018 Gianluigi02 (talk | contribs) . . (4,324 bytes) (-44) . . (undo) (Tags: Mobile edit, Mobile web edit)
- [\(cur | prev\)](#) 12:18, 14 August 2018 Denimadept (talk | contribs) . . (4,368 bytes) (-2) . . (replaced "closed" with "collapsed" in infobox) (undo)
- [\(cur | prev\)](#) 12:17, 14 August 2018 37.74.150.97 (talk) . . (4,370 bytes) (+167) . . (cats) (undo)
- [\(cur | prev\)](#) 12:16, 14 August 2018 Pigsonthewing (talk | contribs) . . (4,203 bytes) (-42) . . (→top: redirected ehre) (undo)
- [\(cur | prev\)](#) 12:15, 14 August 2018 Avaya1 (talk | contribs) . . (4,245 bytes) (+4) . . (undo)
- [\(cur | prev\)](#) 12:14, 14 August 2018 Avaya1 (talk | contribs) . . (4,241 bytes) (+75) . . (undo)
- [\(cur | prev\)](#) 12:14, 14 August 2018 Prioryman (talk | contribs) . . (4,166 bytes) (+17) . . (→top: - add closed, it'll certainly never be used again now) (undo)
- [\(cur | prev\)](#) 12:13, 14 August 2018 Prioryman (talk | contribs) . . (4,149 bytes) (-1) . . (→top: - English spelling) (undo)
- [\(cur | prev\)](#) 12:13, 14 August 2018 Urbourbo (talk | contribs) . . (4,150 bytes) (+44) . . (→Collapse of the bridge) (undo)
- [\(cur | prev\)](#) 12:13, 14 August 2018 Avaya1 (talk | contribs) . . (4,106 bytes) (+327) . . (undo)
- [\(cur | prev\)](#) 12:12, 14 August 2018 Pieceofmetalwork (talk | contribs) m . . (3,779 bytes) (+2) . . (undo)
- [\(cur | prev\)](#) 12:12, 14 August 2018 Prioryman (talk | contribs) m . . (3,777 bytes) (-29) . . (fix) (undo)
- [\(cur | prev\)](#) 12:12, 14 August 2018 Pieceofmetalwork (talk | contribs) . . (3,806 bytes) (+40) . . (undo)
- [\(cur | prev\)](#) 12:12, 14 August 2018 Prioryman (talk | contribs) . . (3,766 bytes) (+354) . . (→Collapse of the bridge: - more) (undo)

Compare selected revisions

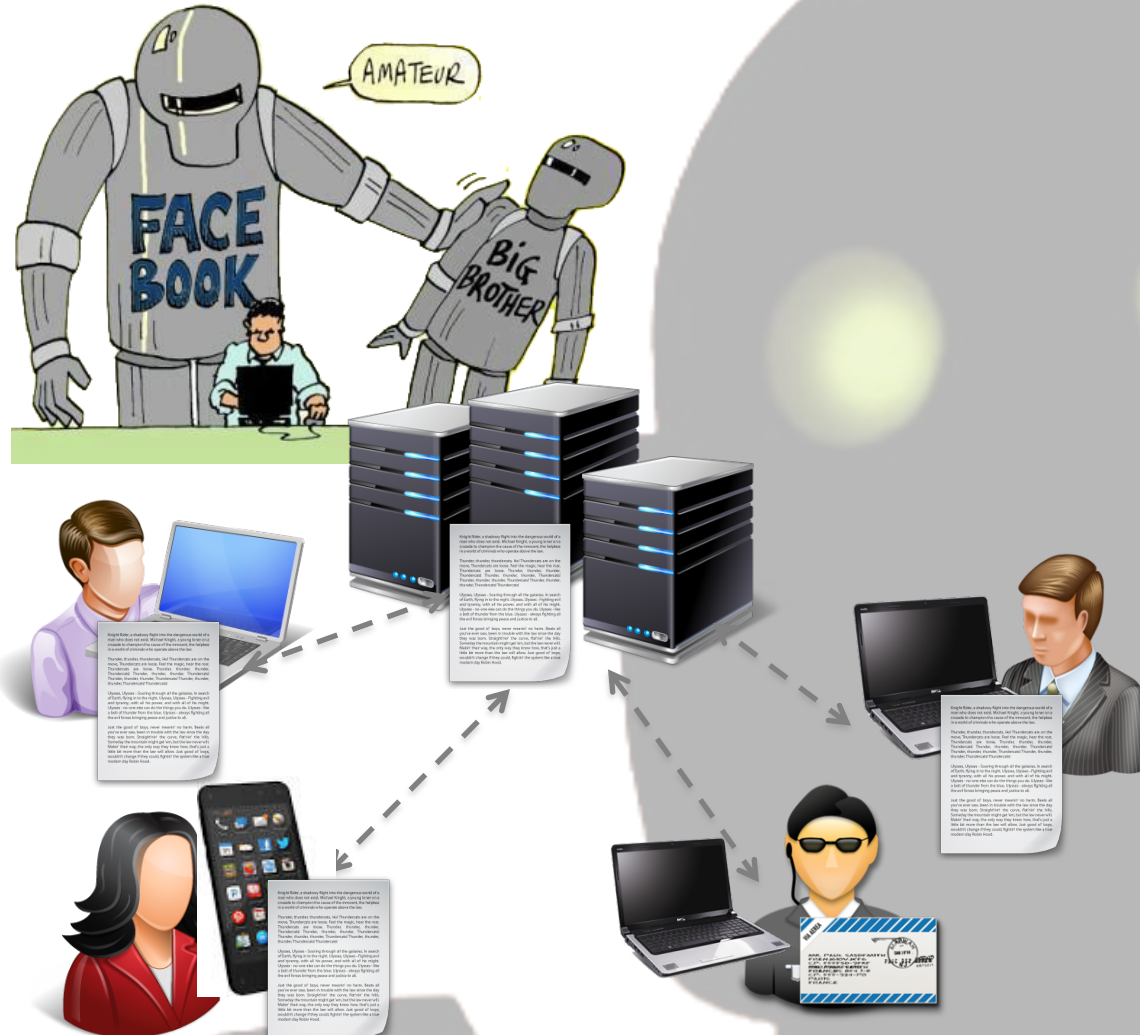
(newest | oldest) View (newer 24 | older 24) (20 | 50 | 100 | 250 | 500)

Limitations of Central Authority Systems

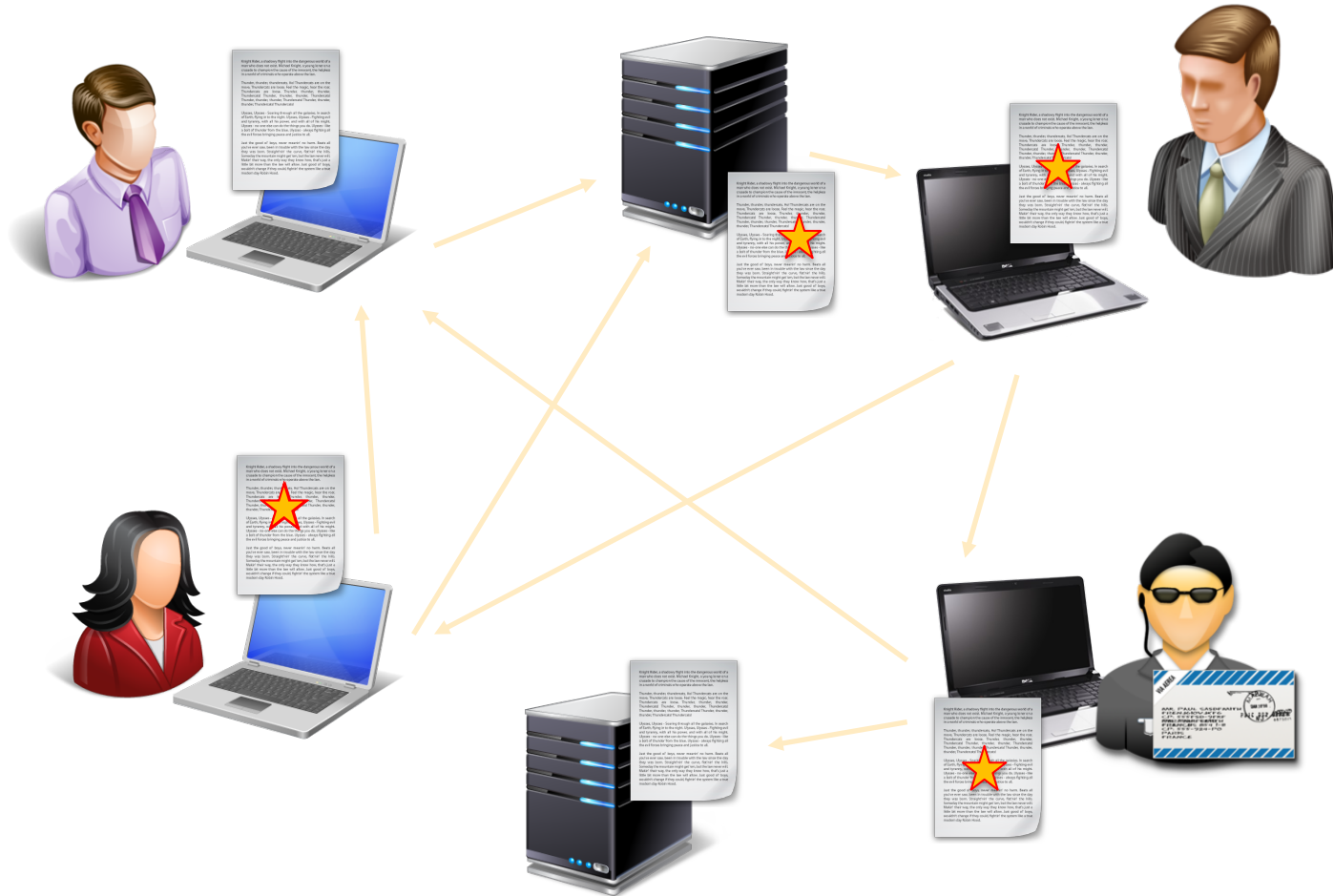


SCALABILITY

PRIVACY

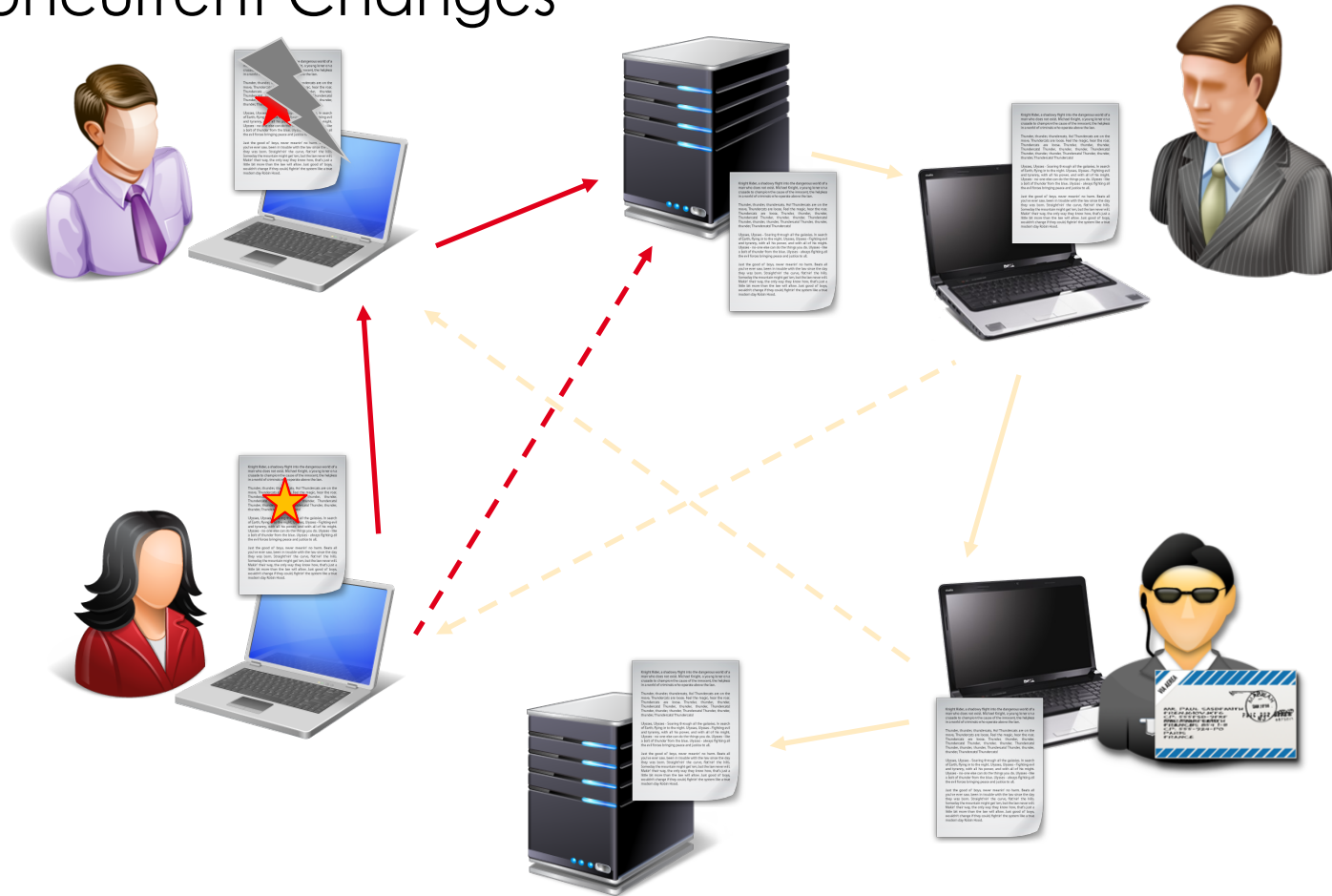


Peer-to-Peer Collaborative Systems



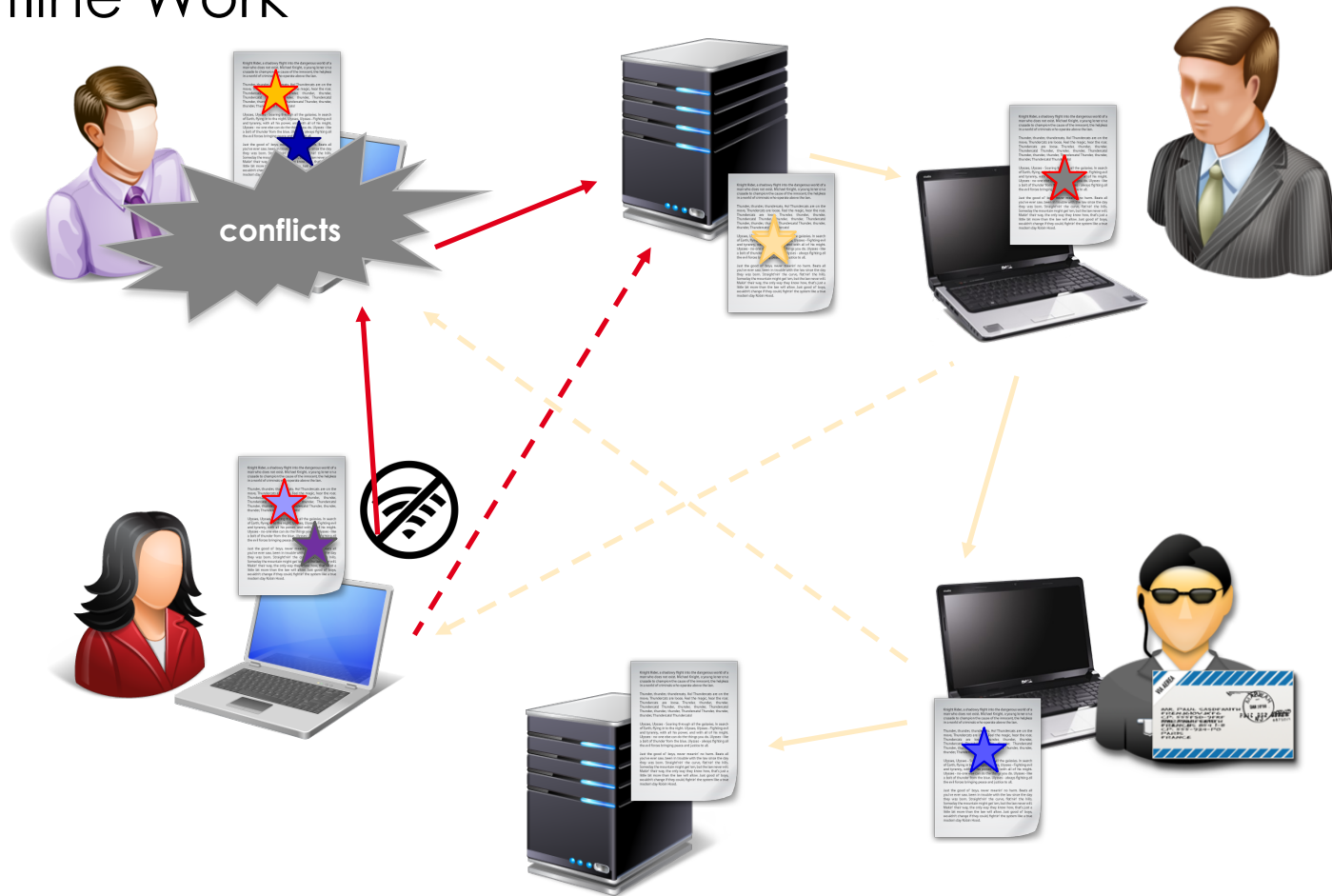
Collaboration Modes

Concurrent Changes



Collaboration Modes

Offline Work

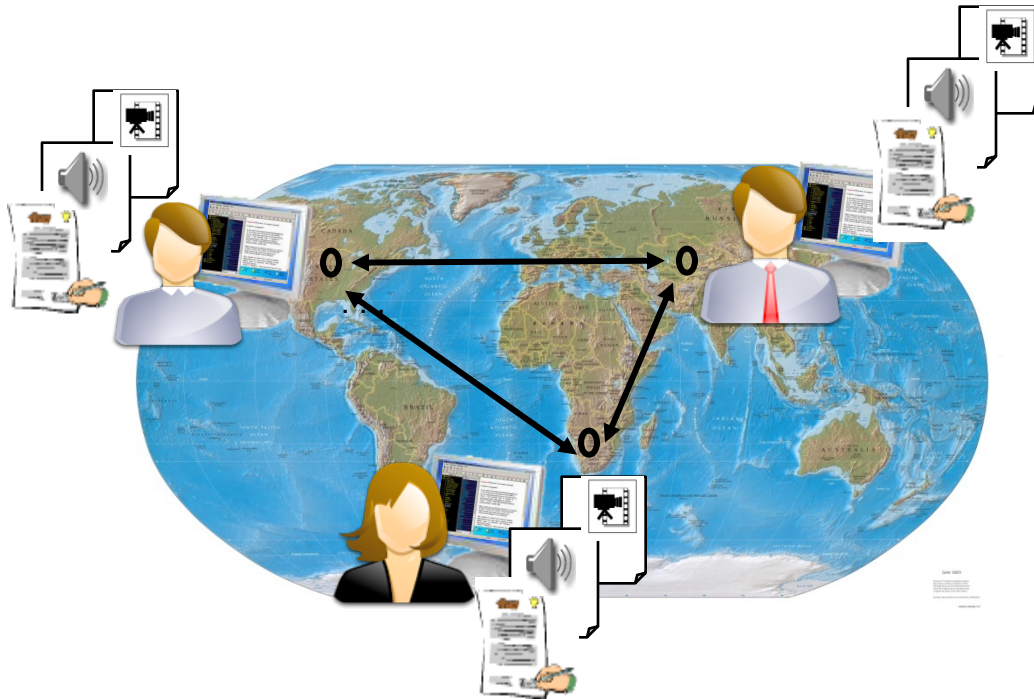


Collaboration Modes

Ad-hoc Collaboration



Operational transformation



- Domain of application: collaborative editing
- Document replication
 - Disconnected work
 - Better response time for real-time collaboration

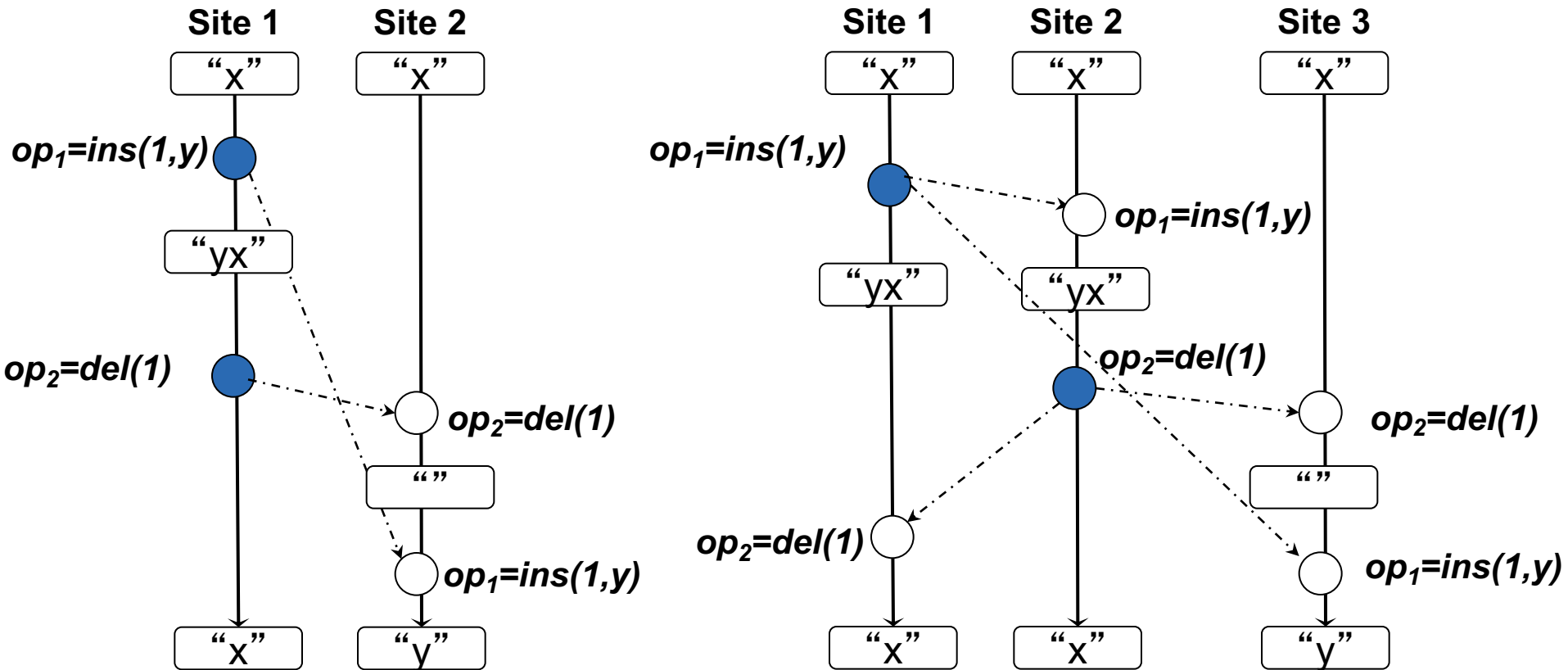
Operational transformation

- Optimistic replication model
 - An operation is :
 - Locally executed,
 - Sent to other sites,
 - Received by a site,
 - Transformed according to concurrent operations,
 - Executed on local copy
- 2 components :
 - An integration algorithm : diffusion, integration
 - Some transformation functions

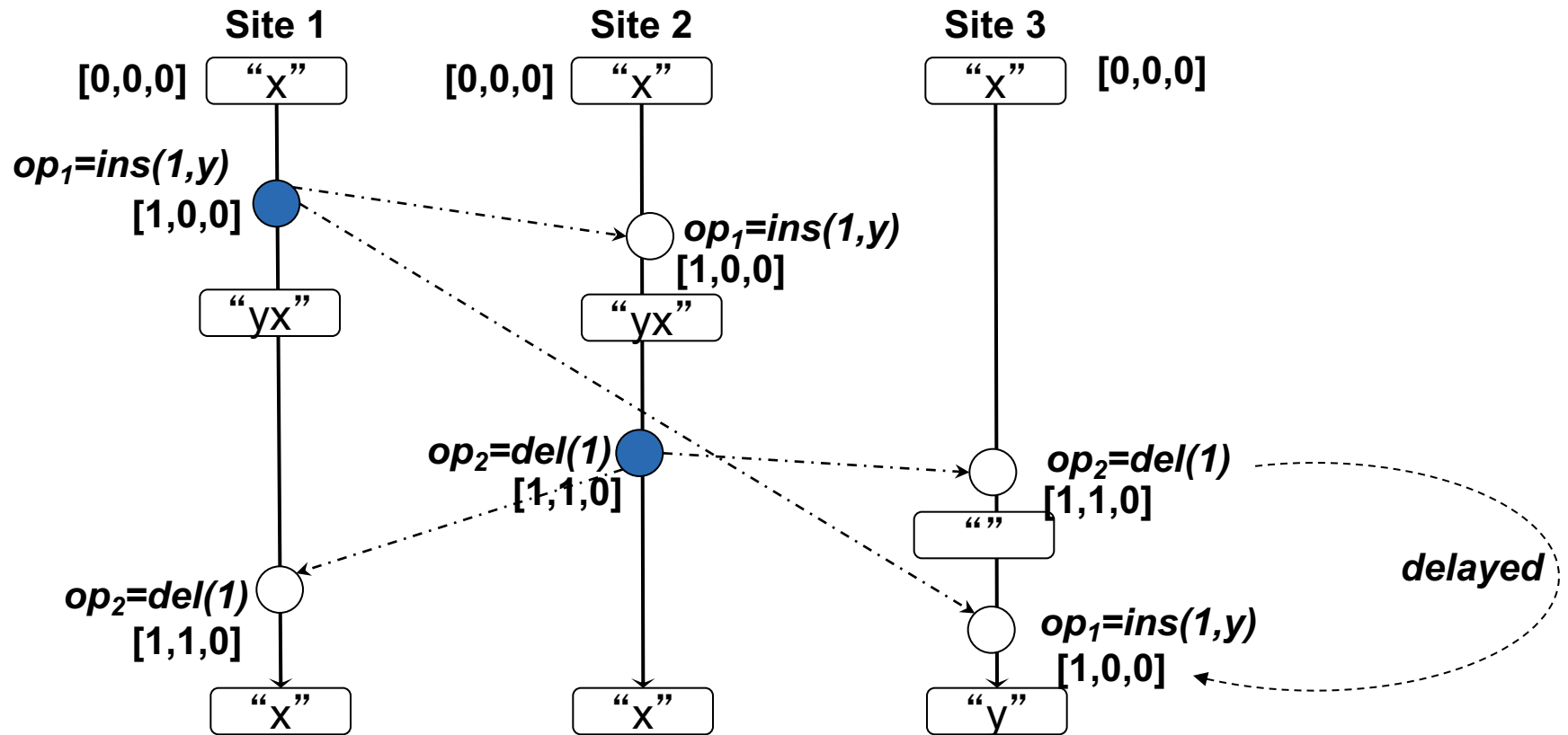
Operational transformation

- Textual documents seen as a sequence of characters
- Operations
 - $\text{ins}(p,c)$
 - $\text{del}(p)$
- Three main issues
 - Causality preservation
 - Intention preservation
 - Convergence

Causality



Causality



Intention

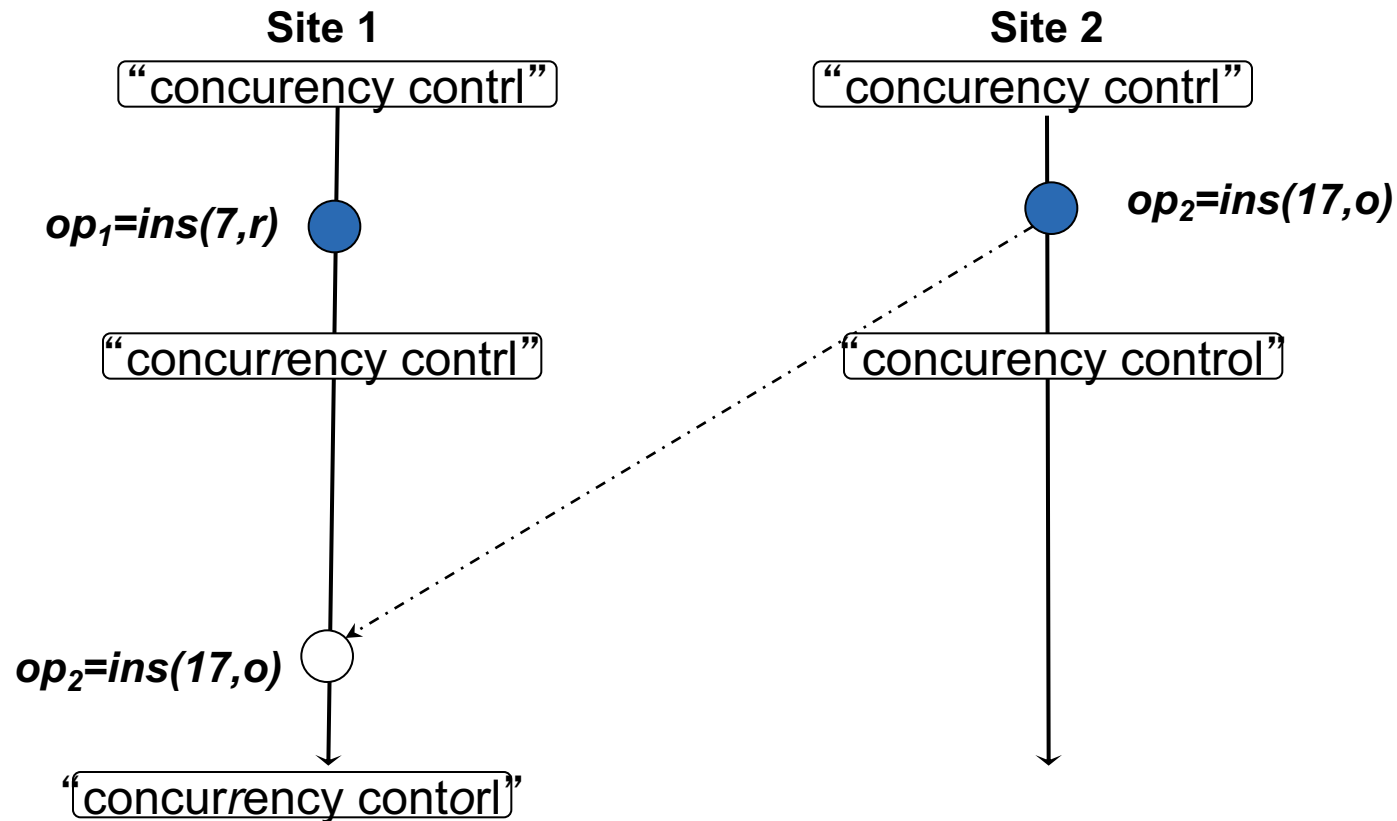
- Intention of an operation is the observed effect as result of its execution on its generation state
- Passing from initial state “ab” to final state “aXb” we can observe:
 - $\text{ins}(2, X)$
 - $\text{ins}(a < X < b)$
 - $\text{ins}(a < X)$
 - $\text{ins}(X < b)$

Preserving user intention (*)

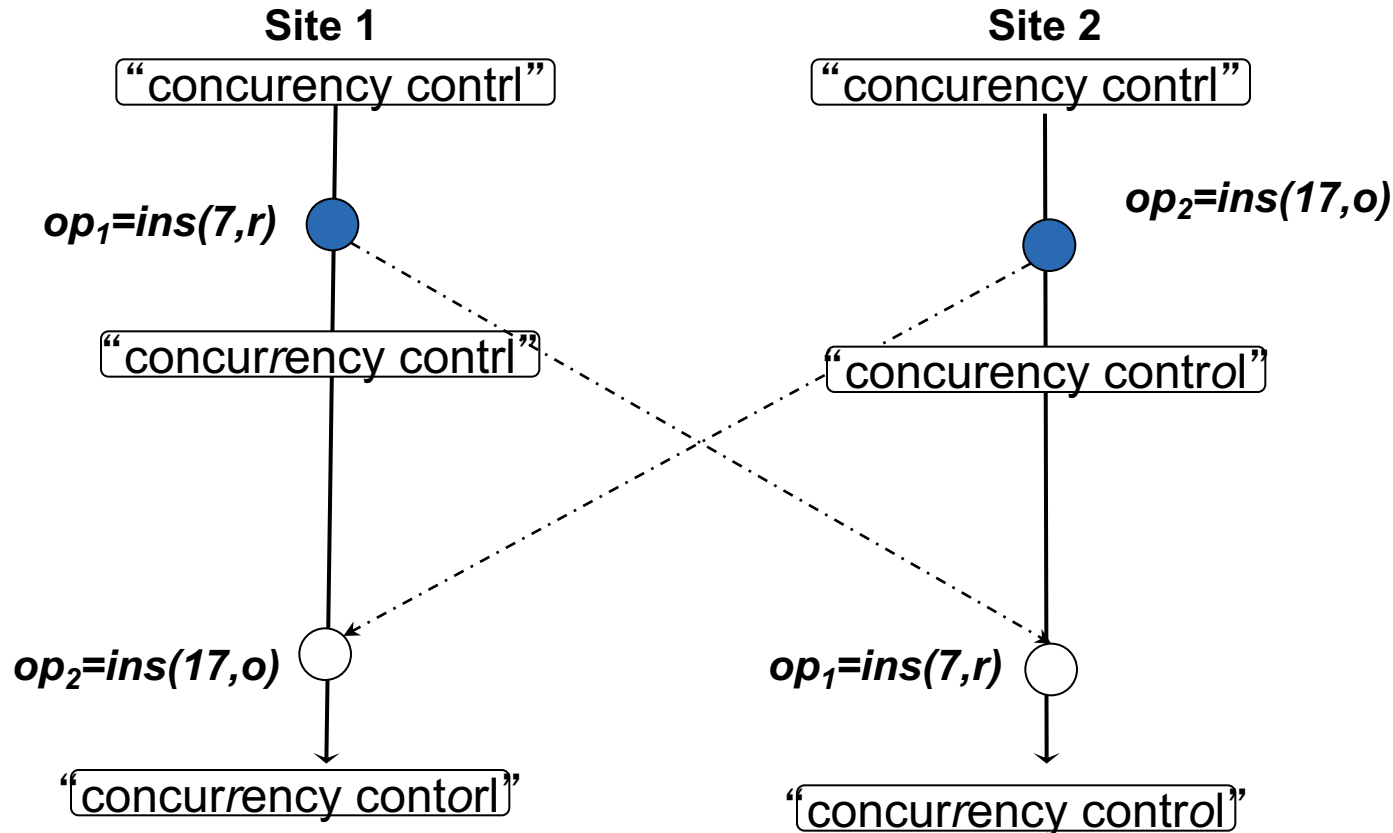
- For any operation op , the effects of executing op at all sites should be the same as the intention of op
- The effect of executing O does not change the effects of independent operations.

(*) Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang, and David Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Transactions on Computer-Human Interaction*, 5(1):63–108, March 1998.

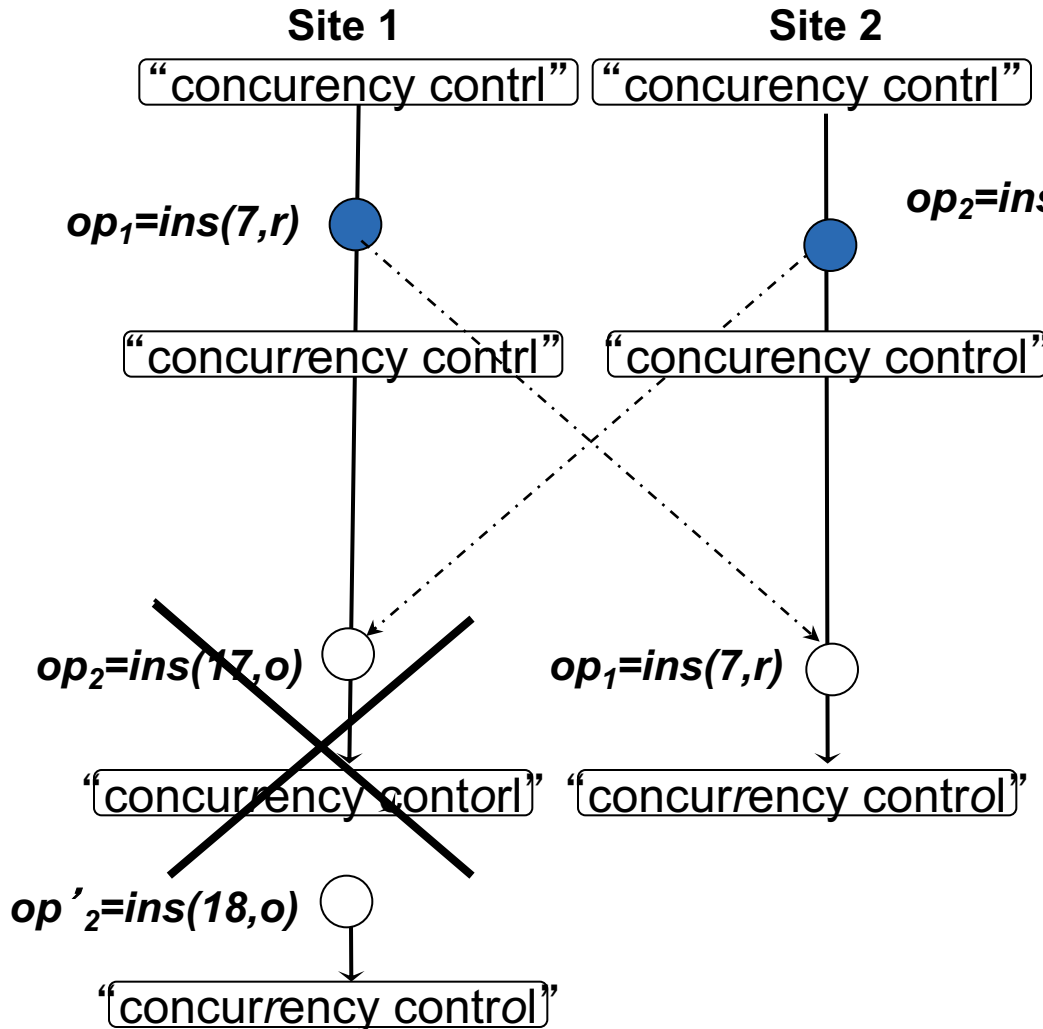
Intention violation



Intention violation + divergence



Intention preservation



$T(ins(p_1, c_1), ins(p_2, c_2)) :-$
 if $(p_1 < p_2)$
 return $ins(p_1, c_1)$
 else
 return $ins(p_1 + 1, c_1)$
 endif

Example transformation functions

$T(ins(p_1, c_1), ins(p_2, c_2)) :-$
 if $(p_1 < p_2)$ **return** $ins(p_1, c_1)$
 else return $ins(p_1 + 1, c_1)$

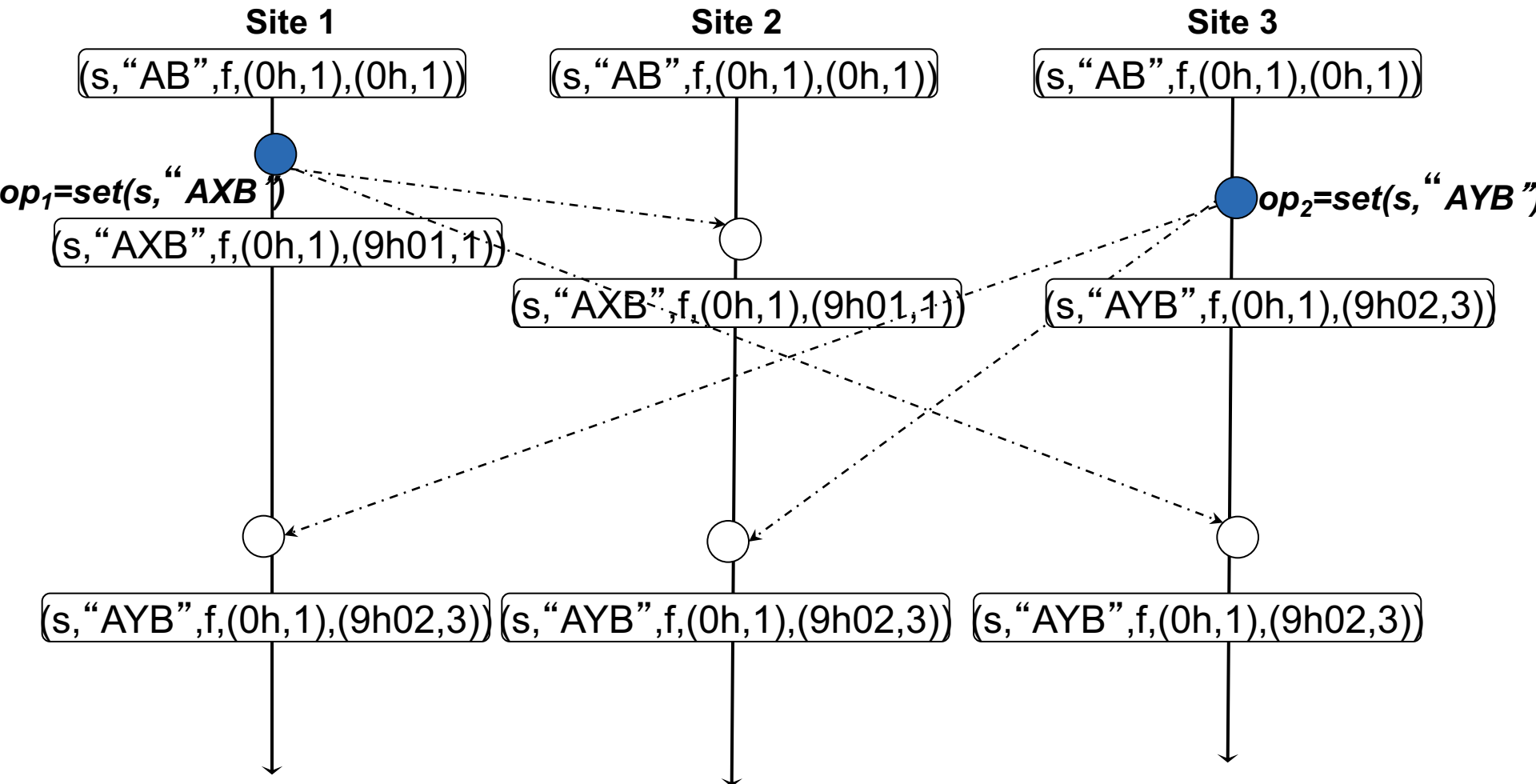
$T(ins(p_1, c_1), del(p_2)) :-$
 if $(p_1 \leq p_2)$ **return** $ins(p_1, c_1)$
 else return $ins(p_1 - 1, c_1)$
 endif

$T(del(p_1), ins(p_2, c_2)) :-$
 if $(p_1 < p_2)$ **return** $del(p_1)$
 else return $del(p_1 + 1)$

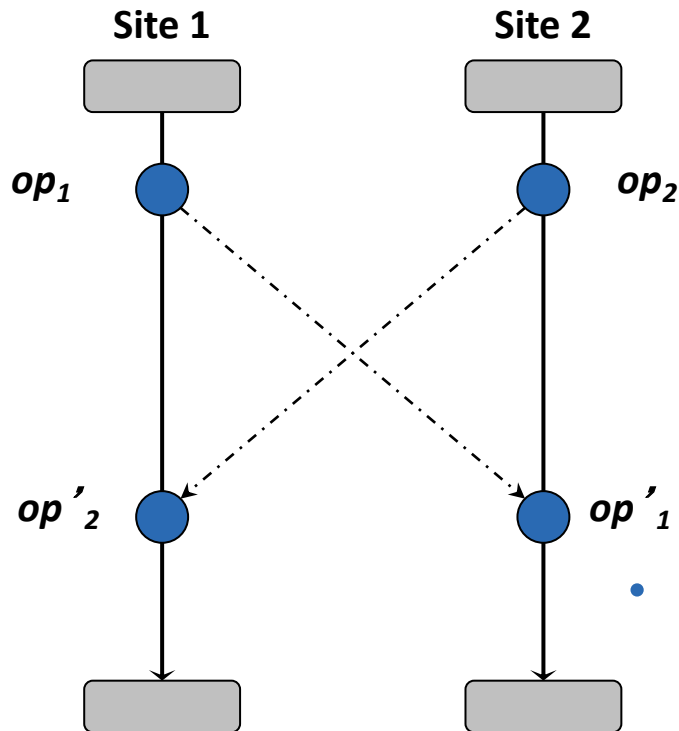
$T(del(p_1), del(p_2)) :-$
 if $(p_1 < p_2)$ **return** $del(p_1)$
 else if $(p_1 > p_2)$ **return** $del(p_1 - 1)$
 else return $id()$

Convergence but no intention preservation

Thomas Write Rule



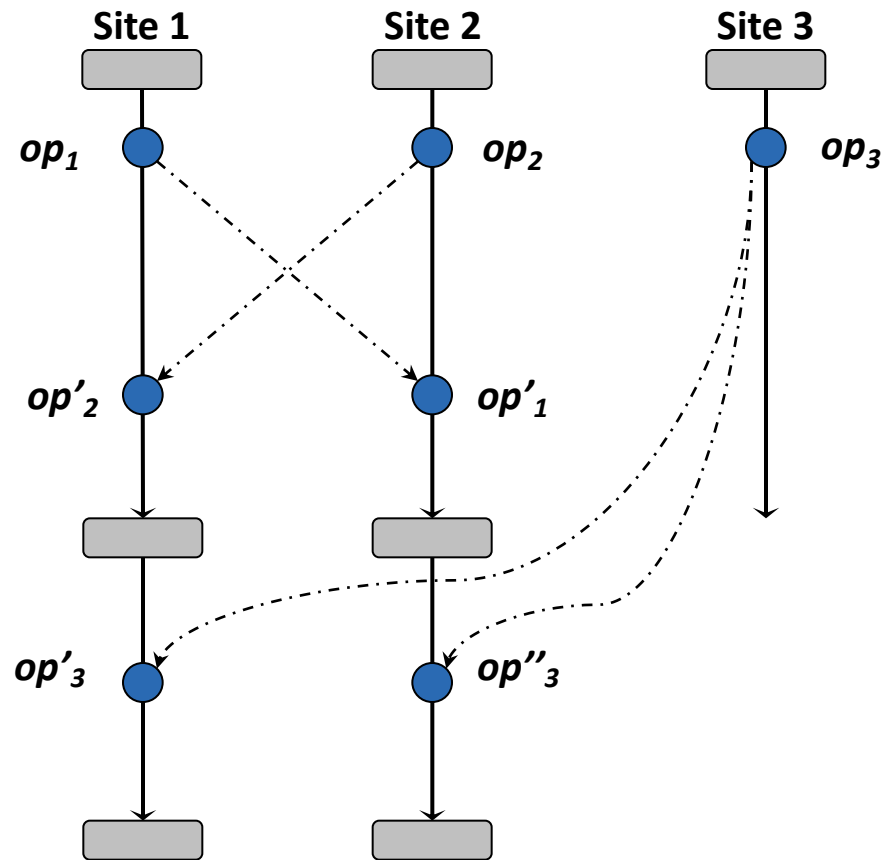
Convergence – TP1 property



- $T(op_2: \text{operation}, op_1: \text{operation}) = op'_2$
 - op_1 and op_2 concurrent, defined on a state S
 - op'_2 same effects as op_2 , defined on $S.op_1$

$$[TP1] \quad op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)$$

Convergence – TP2 property

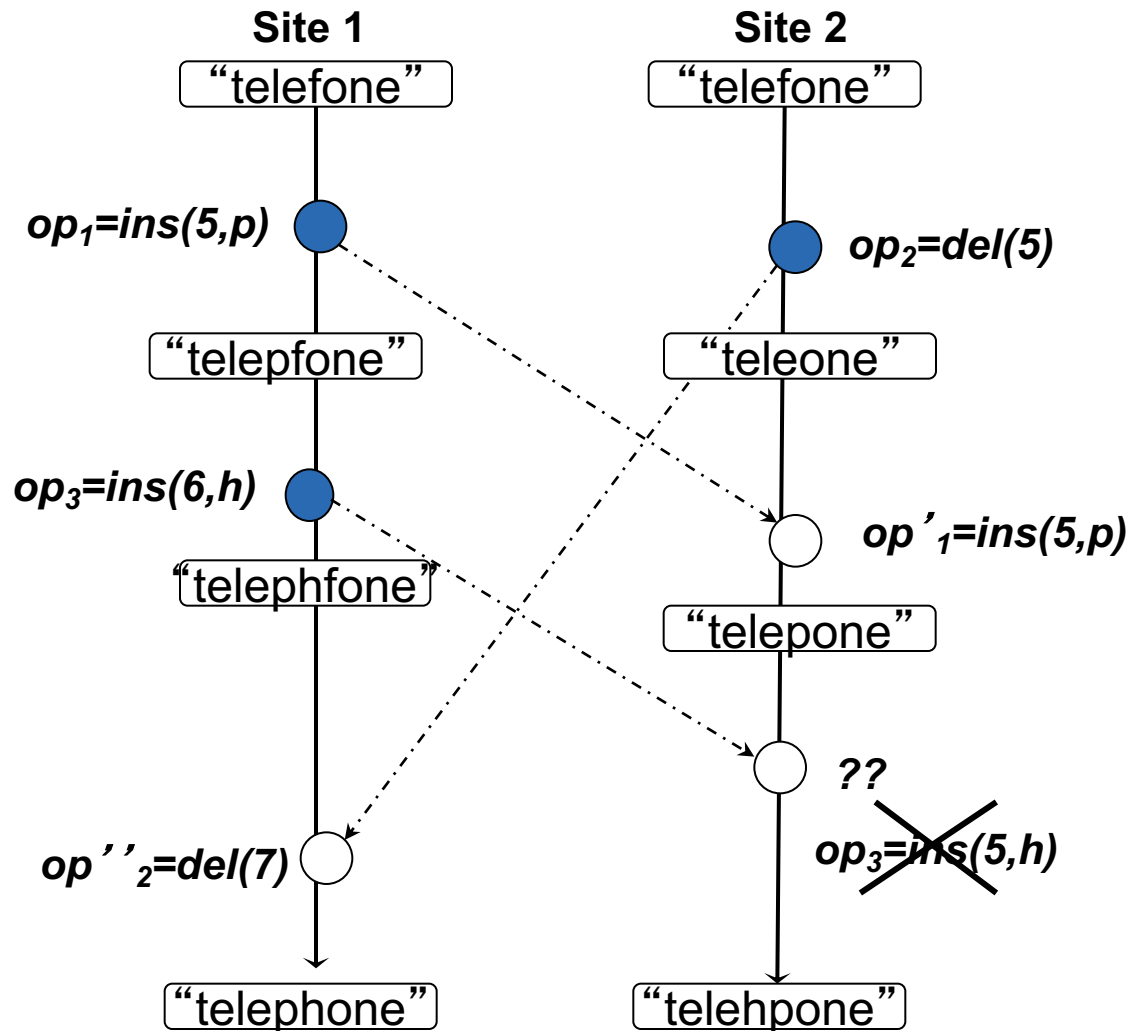


$$[TP2] \quad T(op_3, op_1 \circ T(op_2, op_1)) = T(op_3, op_2 \circ T(op_1, op_2))$$

OT Problems

- Design and verify Transformation functions T
- T also known as `transpose_fd`
- Verification of conditions TP1 and TP2
 - Combinatorial explosion (>100 cases for a string)
 - Iterative process
 - Repetitive and error prone task

Partial concurrency



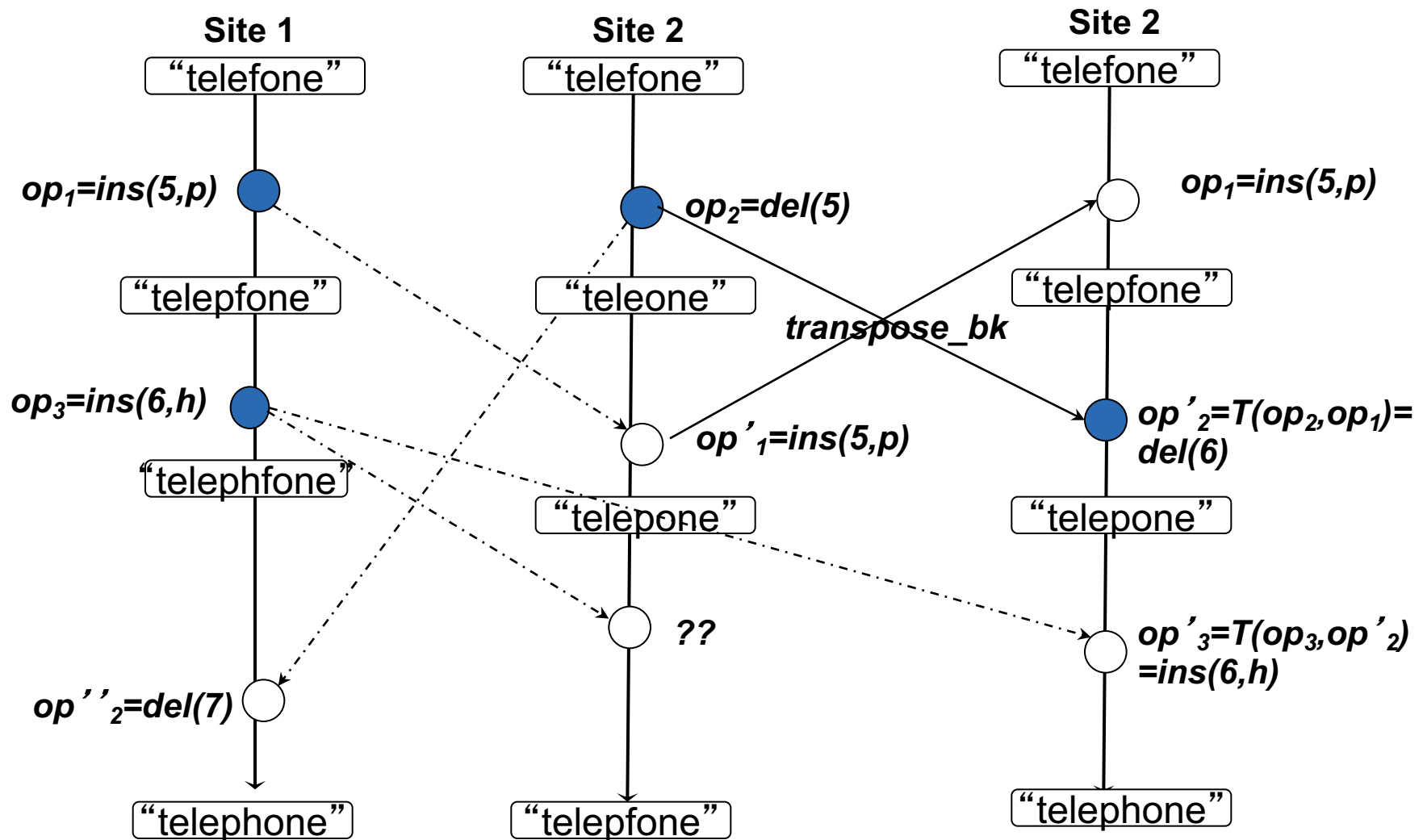
$op'_2 = T(op_2, op_1) = del(6)$

$op''_2 = T(op'_2, op_3) = del(7)$

$op'_1 = T(op_1, op_2) = ins(5)$

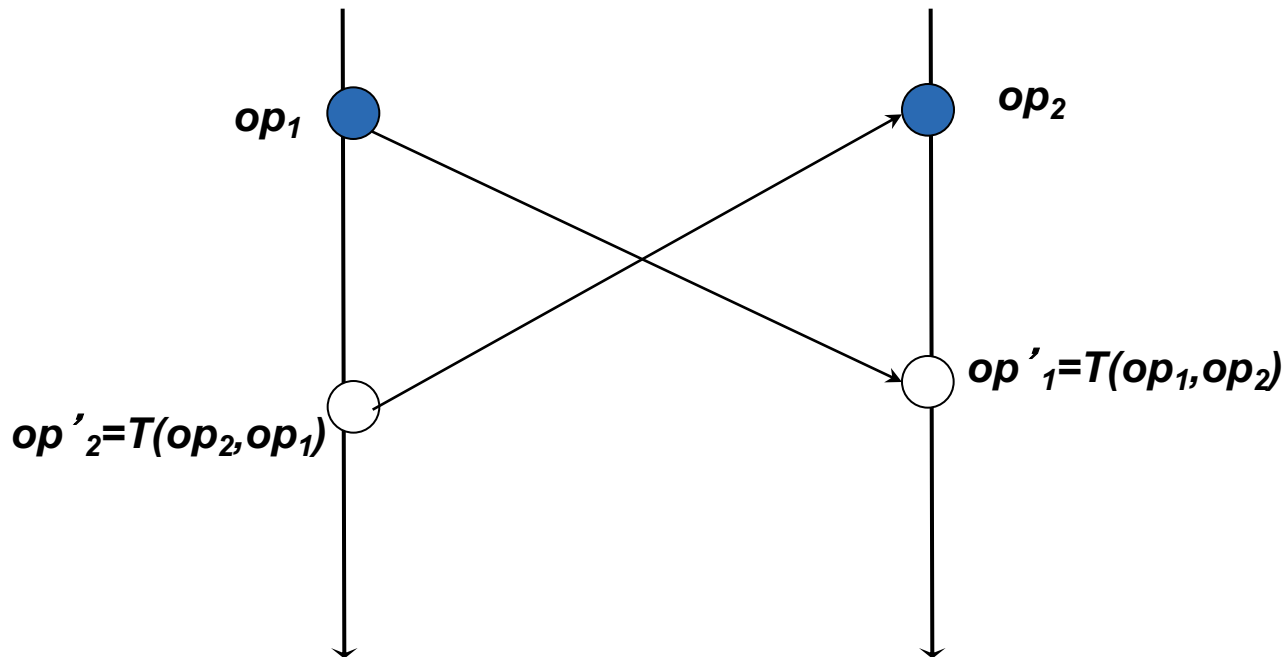
$T(op_3, op_2)$ not allowed to be performed !!!

Partial concurrency



Partial concurrency

- $\text{transpose_bk}(op_1, op'_2) = (op_2, op'_1)$
 - $op'_2 = T(op_2, op_1)$
Therefore $op_2 = T^{-1}(op'_2, op_1)$
 - $op'_1 = T(op_1, op_2)$



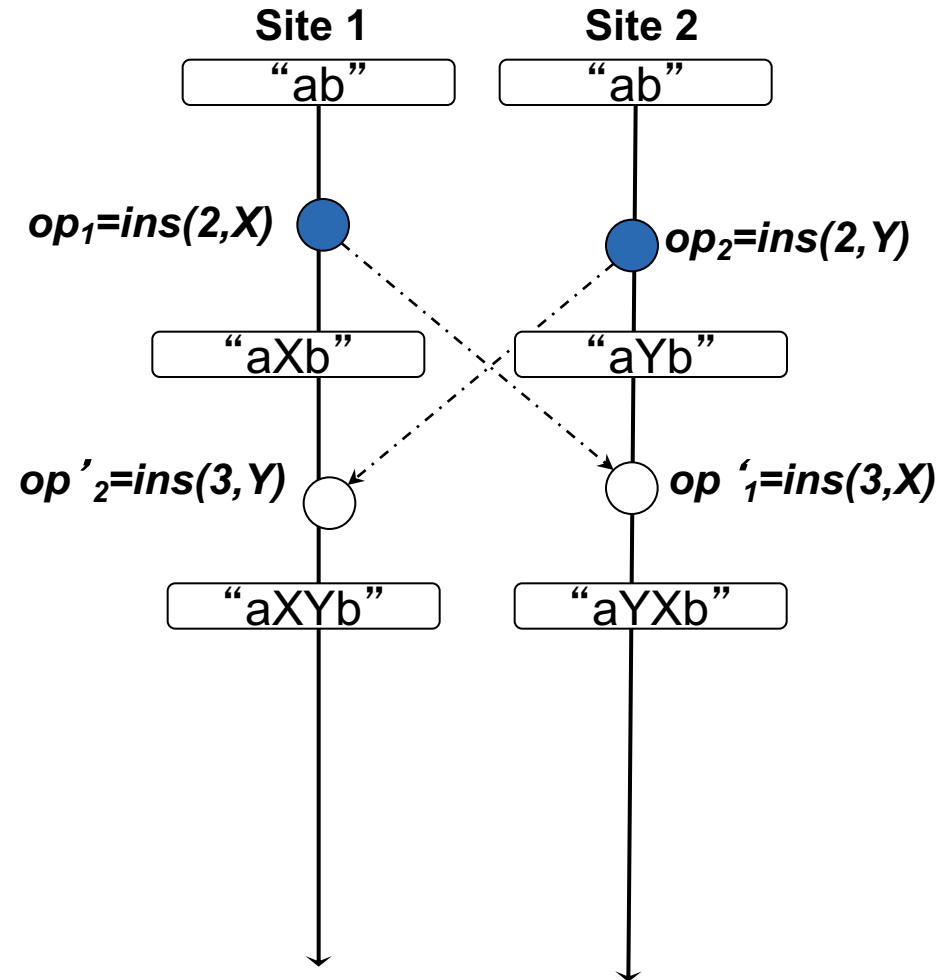
Example transformation functions

$T(ins(p_1, c_1), ins(p_2, c_2)) :-$
 if ($p_1 < p_2$) **return** $ins(p_1, c_1)$
 else return $ins(p_1 + 1, c_1)$

$T(ins(p_1, c_1), del(p_2)) :-$
 if ($p_1 \leq p_2$) **return** $ins(p_1, c_1)$
 else return $ins(p_1 - 1, c_1)$
 endif

$T(del(p_1), ins(p_2, c_2)) :-$
 if ($p_1 < p_2$) **return** $del(p_1)$
 else return $del(p_1 + 1)$

$T(del(p_1), del(p_2)) :-$
 if ($p_1 < p_2$) **return** $del(p_1)$
 else if ($p_1 > p_2$) **return** $del(p_1 - 1)$
 else return $id()$



Ressel transformation functions (*)

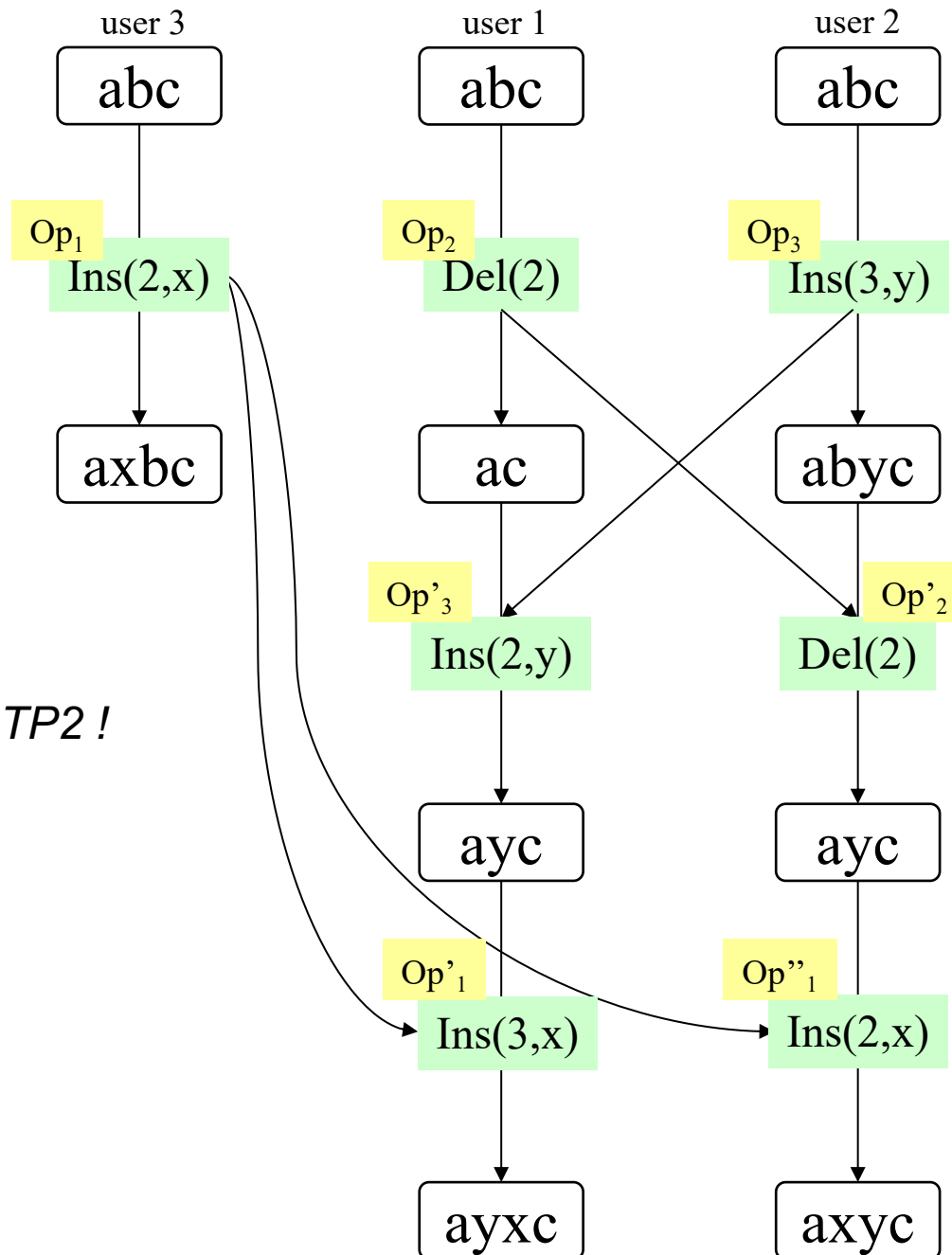
$T(ins(p_1, c_1, u_1), ins(p_2, c_2, u_2)) :-$
 if $((p_1 < p_2) \text{ or } (p_1 = p_2 \text{ and } u_1 < u_2))$ **return** $ins(p_1, c_1, u_1)$
 else return $ins(p_1 + 1, c_1, u_1)$

$T(ins(p_1, c_1, u_1), del(p_2, u_2)) :-$
 if $(p_1 \leq p_2)$ **return** $ins(p_1, c_1, u_1)$
 else return $ins(p_1 - 1, c_1, u_1)$
 endif

$T(del(p_1, u_1), ins(p_2, c_2, u_2)) :-$
 if $(p_1 < p_2)$ **return** $del(p_1, u_1)$
 else return $del(p_1 + 1, u_1)$

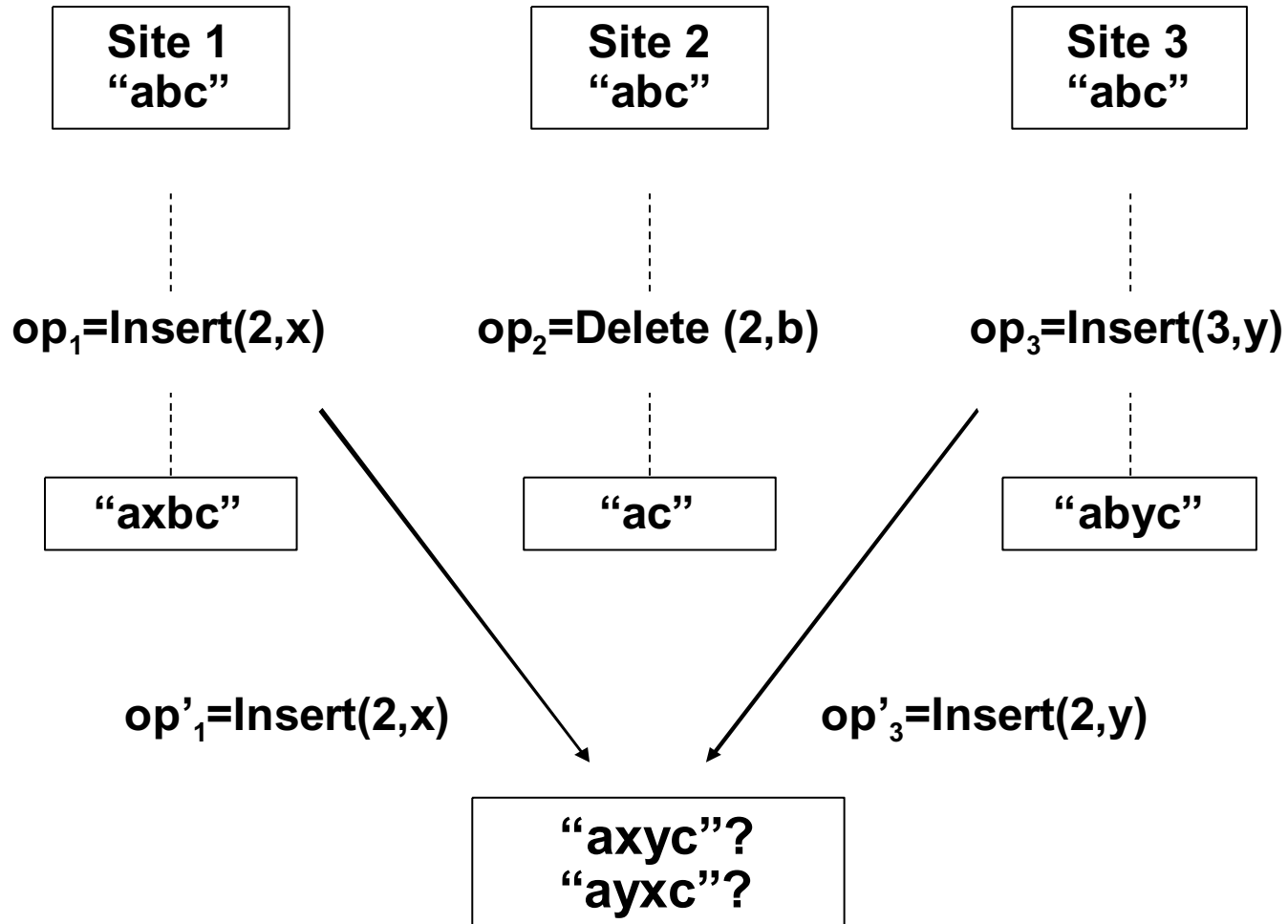
$T(del(p_1, u_1), del(p_2, u_2)) :-$
 if $(p_1 < p_2)$ **return** $del(p_1, u_1)$
 else if $(p_1 > p_2)$ **return** $del(p_1 - 1, u_1)$
 else return $id()$

(*) Ressel, M., Nitsche-Ruhland, D. & Gunzenhauser, R. (1996), An integrating, transformation oriented approach to concurrency control and undo in group editors, Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW' 96), Boston, Massachusetts, USA, pp. 288–297.



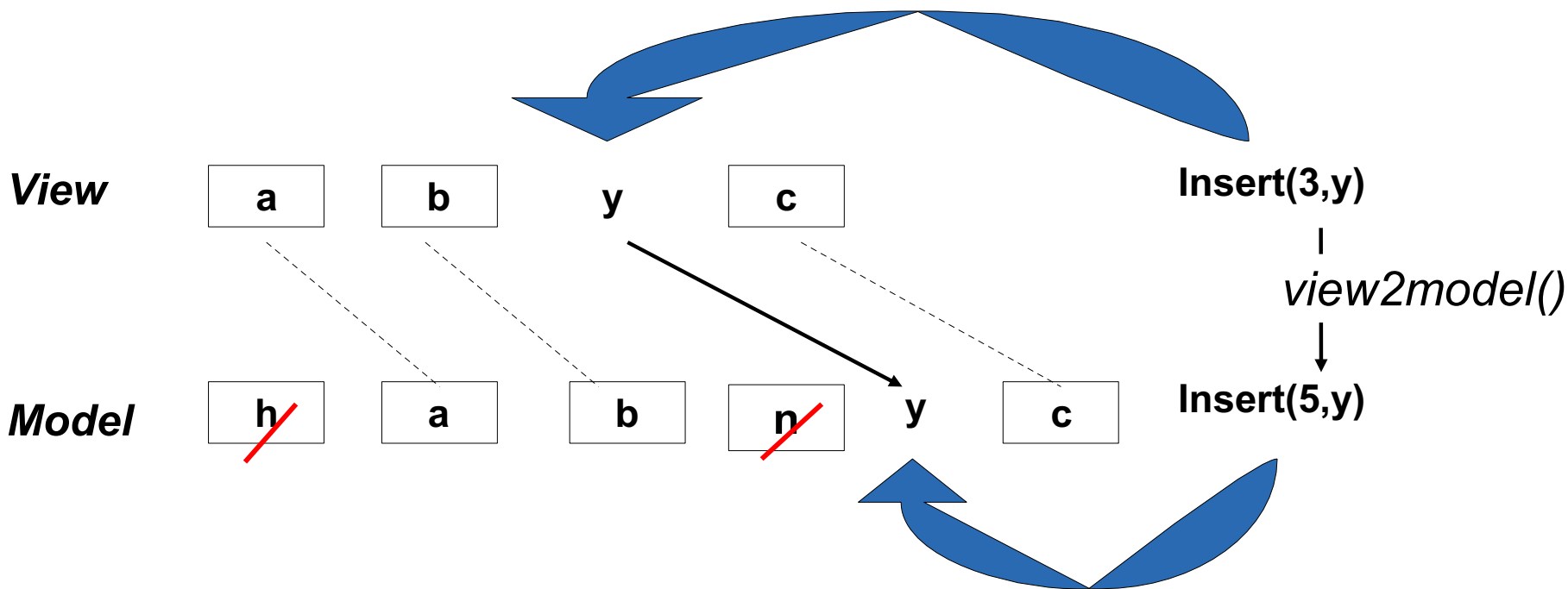
TP1 ok, but not TP2 !

False-tie problem



TTF (Tombstone Transformation Functions) Approach (*)

- Keep “tombstones” of deleted elements

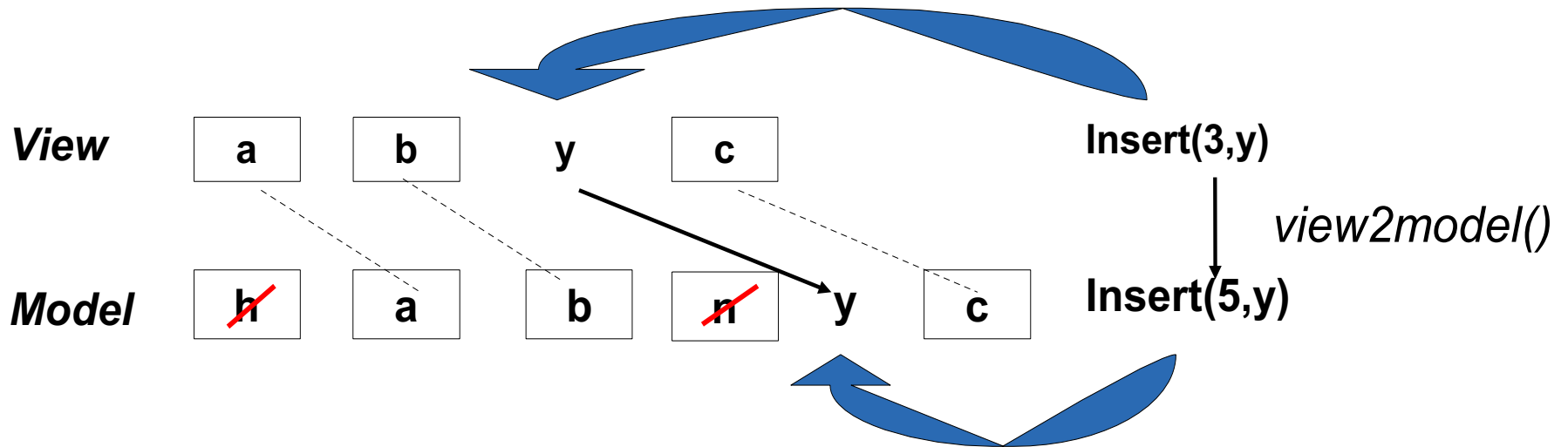


(*) G. Oster, P. Urso, P. Molli, and A. Imine. Tombstone transformation functions for ensuring consistency in collaborative editing systems. In The Second International Conference on Collaborative Computing : Networking, Applications and Worksharing (CollaborateCom 2006), Atlanta, Georgia, USA, November 2006. IEEE Press.

Tombstone Transformation Functions

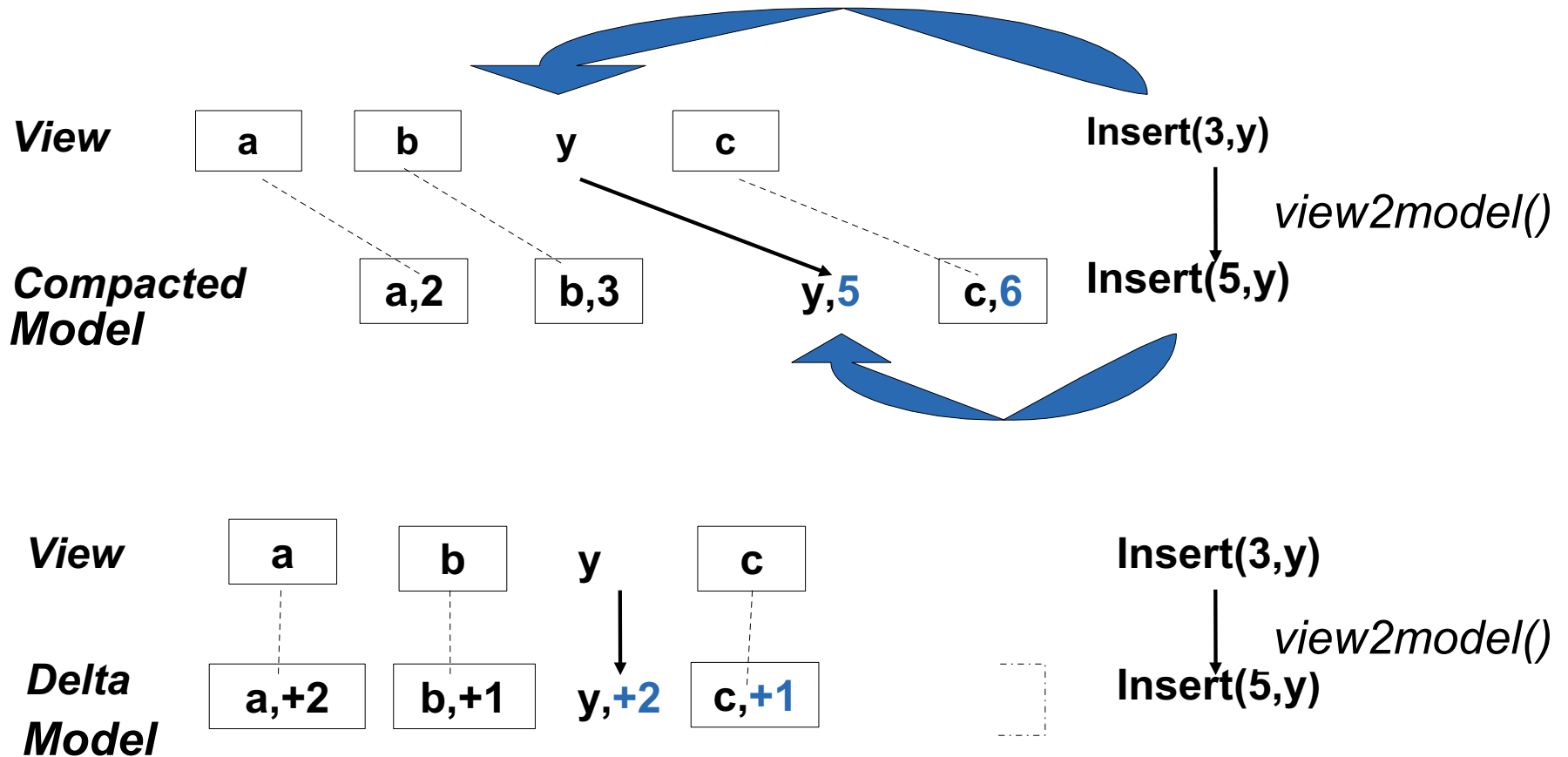
- $T(\text{insert}(p_1, el_1, sid_1), \text{insert}(p_2, el_2, sid_2))\{$
 if($p_1 < p_2$) return $\text{insert}(p_1, el_1, sid_1)$
 else if($p_1 = p_2$ and $sid_1 < sid_2$) return $\text{insert}(p_1, el_1, sid_1)$
 else return $\text{insert}(p_1 + 1, el_1, sid_1)$
}
- $T(\text{insert}(p_1, el_1, sid_1), \text{delete}(p_2, el_2, sid_2))\{$
 return $\text{insert}(p_1, el_1, sid_1)$
}
- $T(\text{delete}(p_1, sid_1), \text{insert}(p_2, sid_2))\{$
 if($p_1 < p_2$) return $\text{delete}(p_1, sid_1)$
 else return $\text{delete}(p_1 + 1, sid_1)$
}
- $T(\text{delete}(p_1, sid_1), \text{delete}(p_2, sid_2))\{$
 return $\text{delete}(p_1, sid_1)$
}

Compacted storage model



- Compacted model = sequence of (character, abs_pos)

Delta storage model



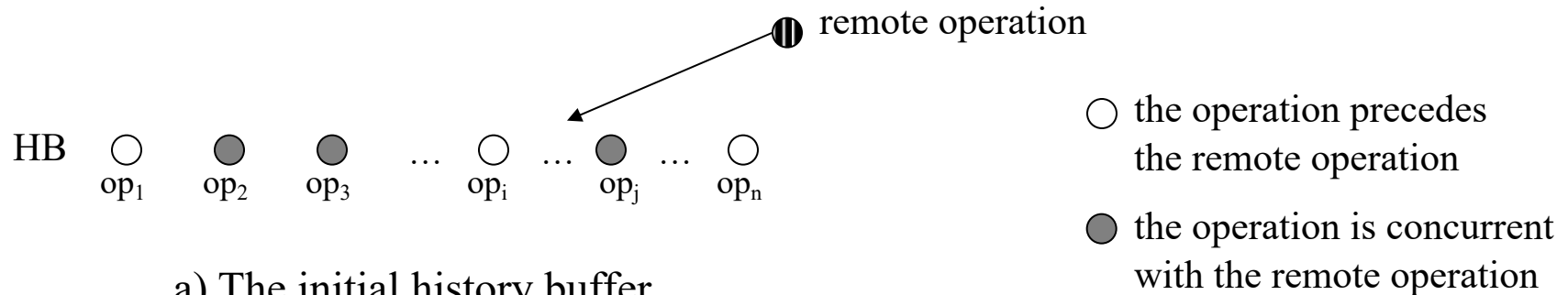
- Delta model = sequence of (character, offset)

Models comparison

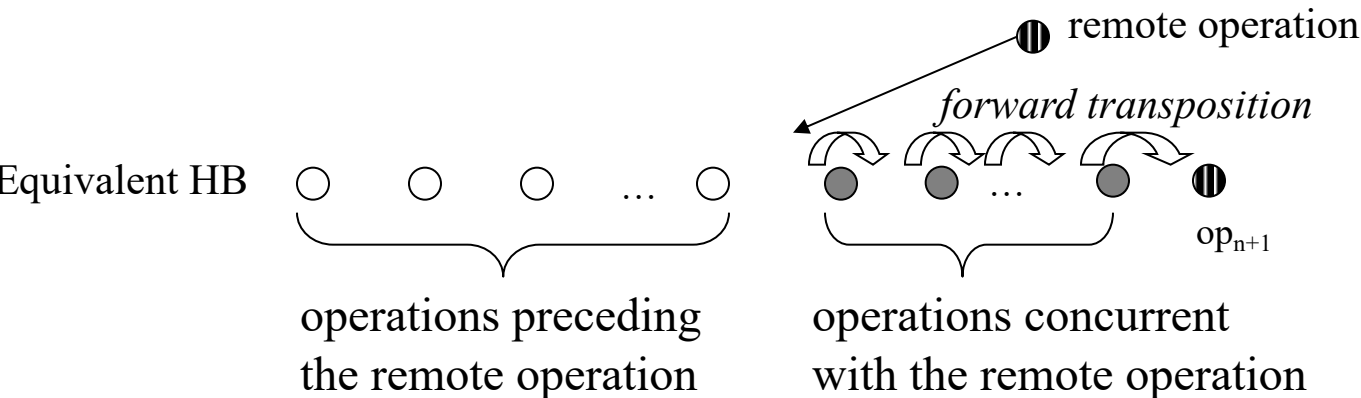
- Basic Model
 - Deleted characters are kept
 - Size of the model is growing infinitely
- Compacted Model
 - Update absolute position of all characters located after the effect position
- Delta Model
 - Update the offset of next character
- Our observations
 - View2model can be optimised (caret position)
 - Overhead of view2model is not significant

SOCT2 algorithm(*)

General control algorithm



a) The initial history buffer



b) Principle of integration

(*) M. Suleiman, M. Cart, and J. Ferrié. Serialization of concurrent operations in a distributed collaborative environment. In Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work : (GROUP'97), pages 435.445, Phoenix, Arizona, United States, November 1997.

GOT algorithm(*)

- Does not need to satisfy TP1 and TP2
- Requires a global serialisation order
 - Sum of state vector components
 - If equality, then priority on sites
- Requires undo/redo mechanism
- Undo/redo very costly

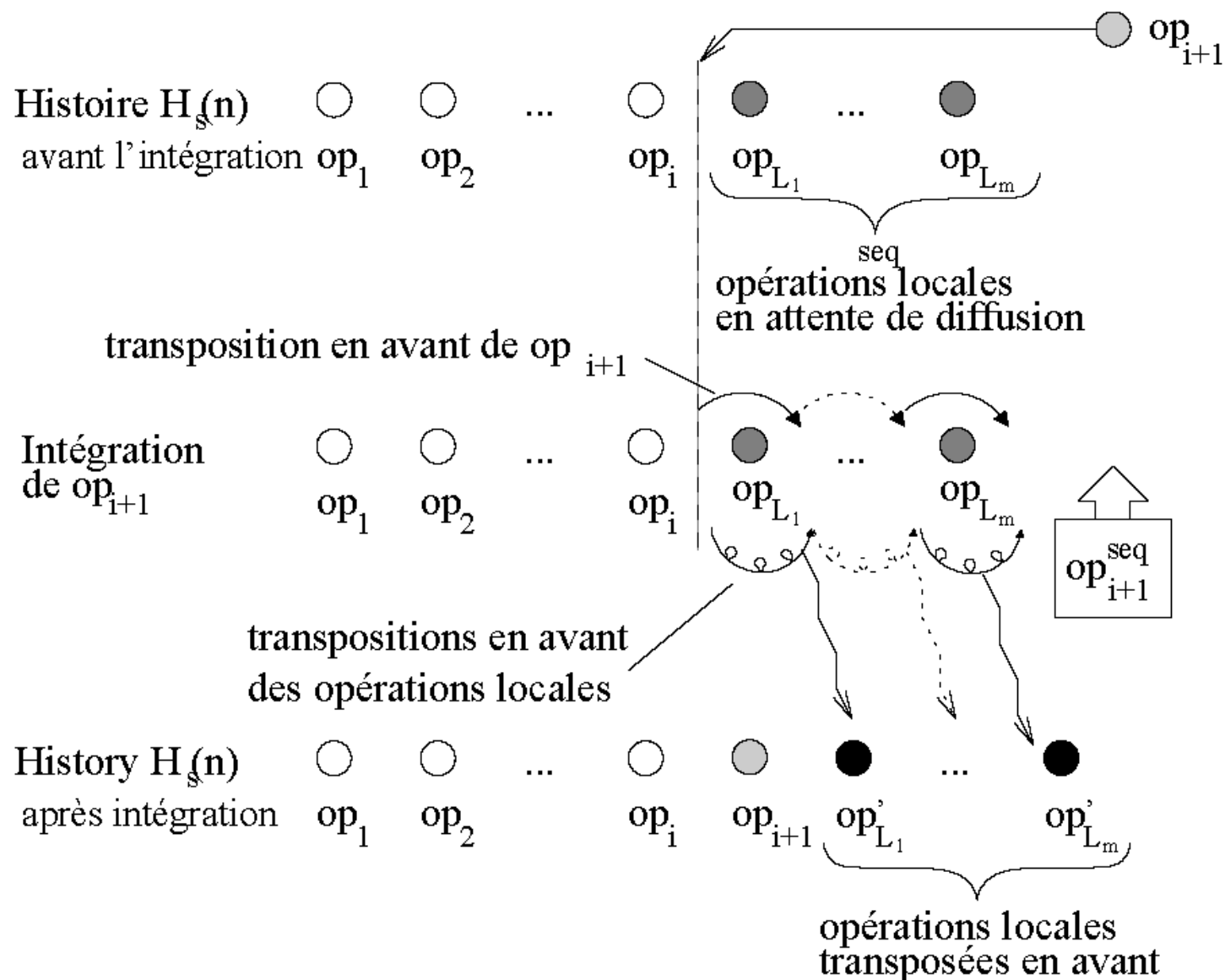
(*) Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang, and David Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Transactions on Computer-Human Interaction*, 5(1):63–108, March 1998.

SOCT4 algorithm(*)

- Does not use undo/redo mechanism
- Eliminates TP2, but requires TP1
- Does not need state vectors
- Global order of operations according to timestamps generated by a sequencer
- Local operations executed immediately
- Assigns a timestamp to the operation and transmits it to the other sites
- Defers broadcast until all preceding operations were executed
- Transformations performed by each site

(*) Nicolas Vidot, Michèle Cart, Jean Ferrié, and Maher Suleiman. Copies convergence in a distributed real-time collaborative environment. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'00)*, page 171–180, Philadelphia, Pennsylvania, USA, December 2000.

SOCT4 algorithm



So6- variation of SOCT4 algorithm(*)

```
Sync(log, Ns) :-  
  while ((opr = getOp(Ns+1)) != ∅) 2  
    for (i=0; i < log.size(); i++)  
      opl = log[i]; 4  
      log[i] = T(opl, opr)  
      opr = T(opr, opl); 6  
    endfor  
    execute opr 8  
    Ns = Ns + 1  
  endwhile 10  
  
  for (i=0; i < log.size(); i++) 12  
    op'l = log[i];  
    if send(op'l, Ns+1) then 14  
      Ns = Ns + 1  
    else 16  
      error 'need to synchronize'  
    endif 18  
  endfor
```

getOp(ticket) retrieves operation identified by timestamp *ticket*

send(op, ticket) sends local operation with timestamp *ticket*. If ticket already exists (a concurrent synchronization is in progress), returns false

(*) Pascal Molli, Gérald Oster, Hala Skaf-Molli and Abdessamad Imine. *Using the Transformational Approach to Build a Safe and Generic Data Synchronizer*. In Proceedings of the ACM SIGGROUP Conference on Supporting Group Work - GROUP 2003, pages 212-220, Sanibel Island, Florida, USA, nov 2003

So6 algorithm

Site1, Ns=0	Site2, Ns=0
op1	op3
op2	op4
s1=synchronize	
	s2=synchronize
s3=synchronize	

- At s3
 - $\text{sync}([], 2)$
 - $\text{op}'' 3$ and $\text{op}'' 4$ are executed
 - $Ns=4$
- At s1
 - $\text{sync}([\text{op}1, \text{op}2], 0)$
 - Send op1, op2
 - $Ns=2$
- At s2
 - $\text{sync}([\text{op}3, \text{op}4], 0)$
 - $\text{op}' 1 = T(\text{op}1, \text{op}3)$
 - $\text{op}' 3 = T(\text{op}3, \text{op}1)$
 - $\text{op}'' 1 = T(\text{op}' 1, \text{op}4)$
 - $\text{op}' 4 = T(\text{op}4, \text{op}' 1)$
 - $\text{op}' 2 = T(\text{op}2, \text{op}' 3)$
 - $\text{op}'' 3 = T(\text{op}' 3, \text{op}2)$
 - $\text{op}'' 2 = T(\text{op}' 2, \text{op}' 4)$
 - $\text{op}'' 4 = T(\text{op}' 4, \text{op}' 2)$
 - $\text{op}'' 1, \text{op}'' 2$ are executed
 - $\text{op}'' 3, \text{op}'' 4$ are sent
 - $Ns=4$

SO6 algorithm

Site1, Ns=0	Site2, Ns=0
op1	op3
op2	op4
s1=synchronize	
	s2=synchronize
s3=synchronize	

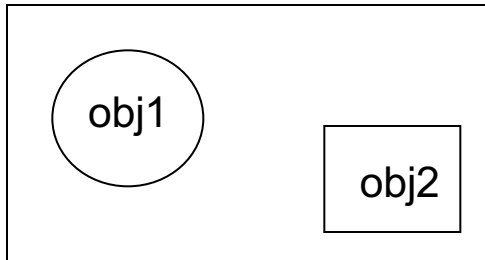
Site 1

op1
op2
$op''3 = T(T(op3, op1), op2)$
$op''4 = T(T(op4, op'1), op'2)$

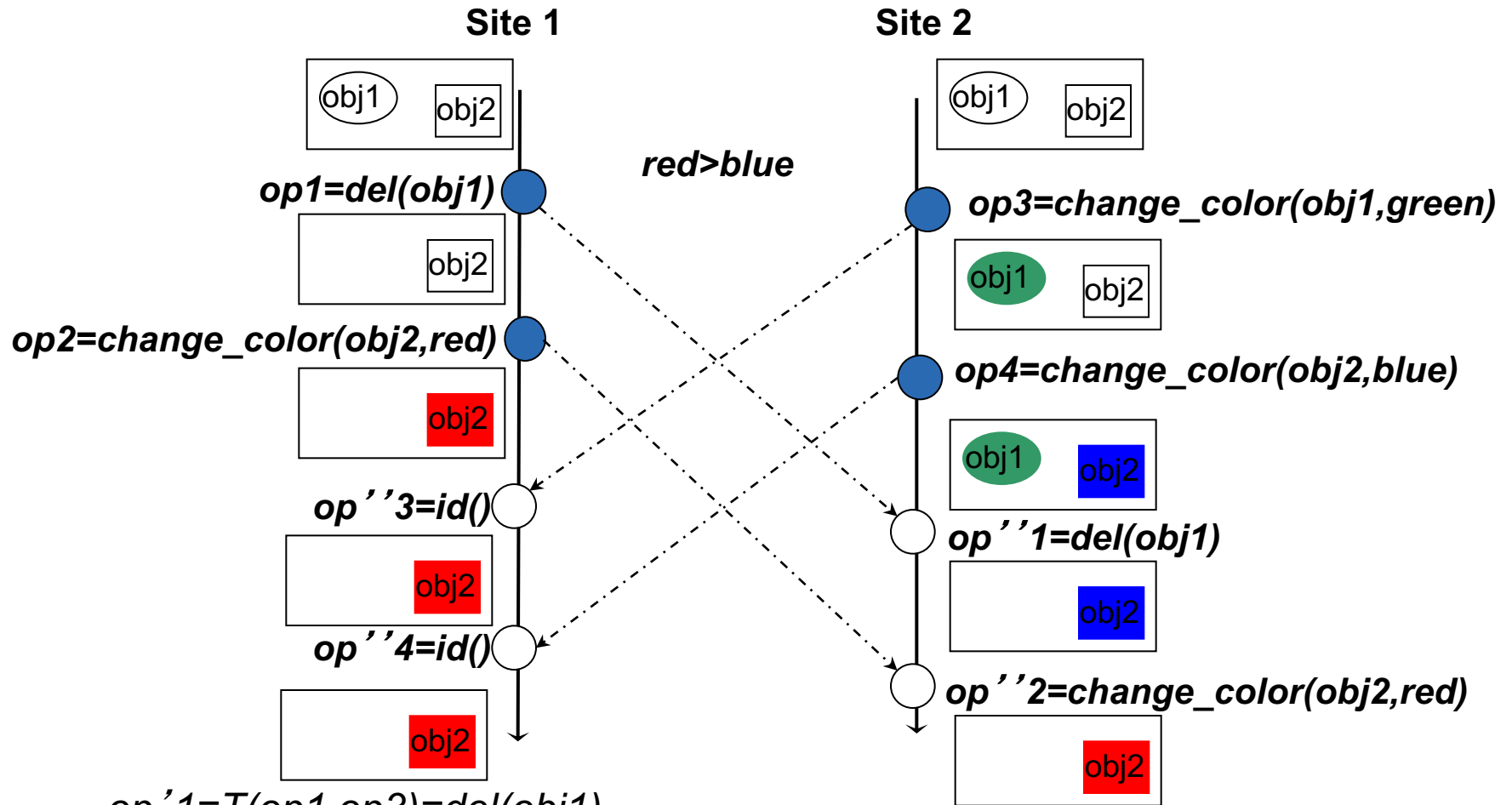
Site 2

op3
op4
$op''1 = T(T(op1, op3), op4)$
$op''2 = T(T(op2, op'3), op'4)$

SO6 algorithm



- `create(obj)`
 - `change_color(obj,color)`
 - `del(obj)`
- `T(create(obj),_-`
 `return create(obj)`
 - `T(change_color(obj1,c), del(obj2)):-`
 `if (obj1==obj2) then return id()`
 `else return change_color(obj1,c)`
 - `T(del(obj1), del(obj2)):-`
 `if (obj1==obj2) then return id()`
 `else return del(obj1)`
 - `T(del(obj1),change_color(obj2,c)):-`
 `return del(obj1)`
 - `T(change_color(obj1,c1),change(obj2,c2)):-`
 `if (obj1==obj2) then`
 `if(c1>c2) then return change_color(obj1,c1)`
 `else return id()`
 `else return change_color(obj1,c1)`
 - `T(id(),_-` `return id()`
 - `T(op,id()):-` `return op`



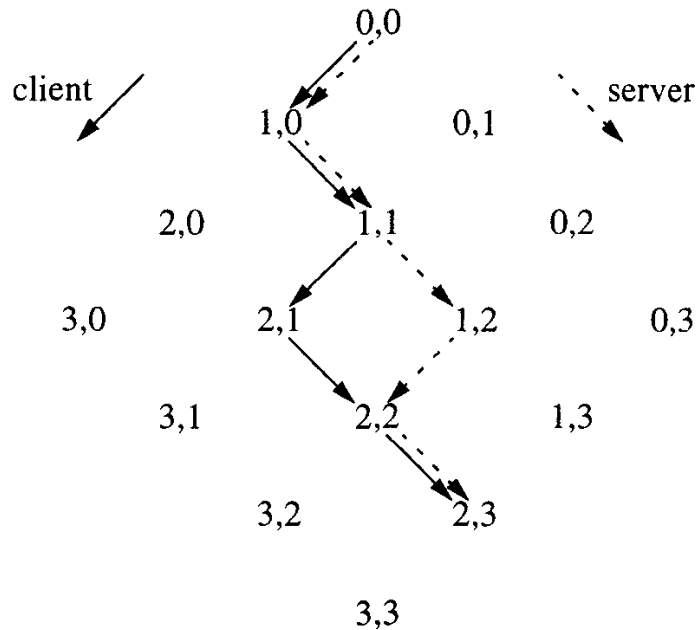
- $op'1 = T(op1, op3) = del(obj1)$
- $op'3 = T(op3, op1) = id()$
- $op''1 = T(op'1, op4) = del(obj1)$
- $op'4 = T(op4, op'1) = change_color(obj2, blue)$
- $op'2 = T(op2, op'3) = change_color(obj2, red)$
- $op''3 = T(op'3, op2) = id()$
- $op''2 = T(op'2, op'4) = change_color(obj2, red)$
- $op''4 = T(op'4, op'2) = id()$

Jupiter algorithm(*)

- Used in Google Drive
- Requires a central server
- Eliminates TP2, but requires TP1
- Does not need state vectors
- Transformations done on the server + client side

(*) David A. Nichols, Pavel Curtis, Michael Dixon, and John Lamping. 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)*, page 111–120, Pittsburgh, Pennsylvania, USA, 1995.

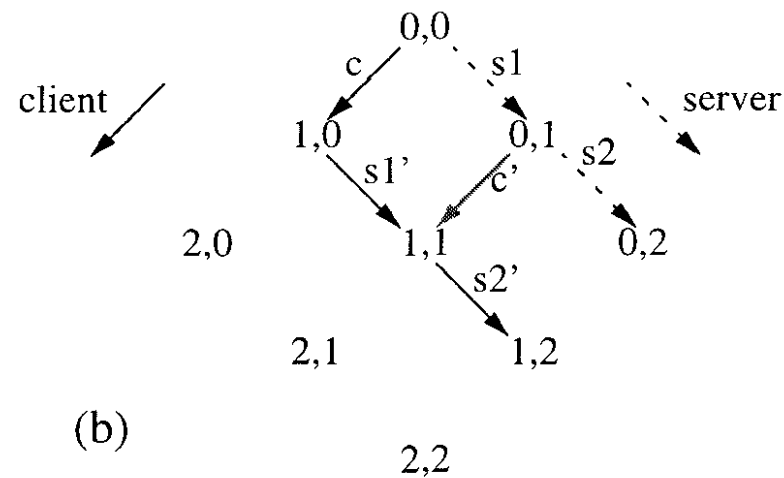
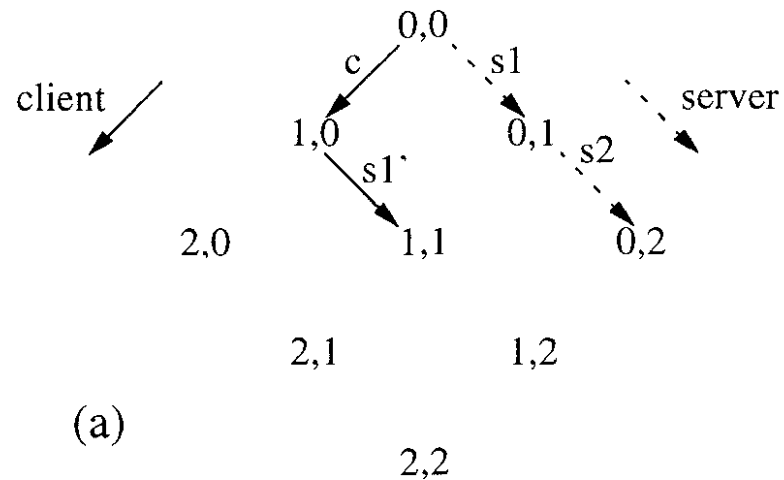
Jupiter algorithm



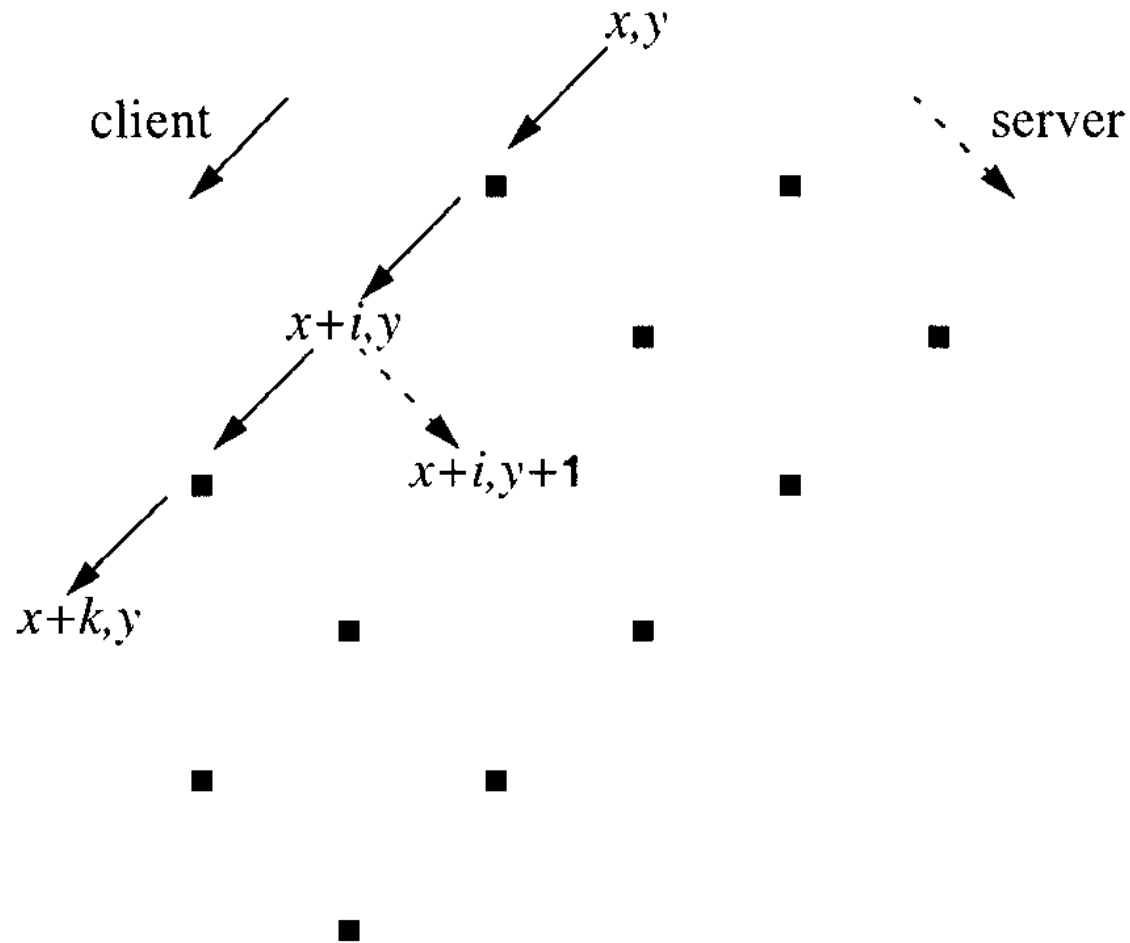
- $xform(c,s)=\{c',s'\}$
- $xform(del\ x, del\ y)=$
 - $\{del\ x-1, del\ y\}$ if $x>y$
 - $\{del\ x, del\ y-1\}$ if $x<y$
 - $\{no-op, no-op\}$ if $x=y$

...

Jupiter algorithm



Jupiter algorithm



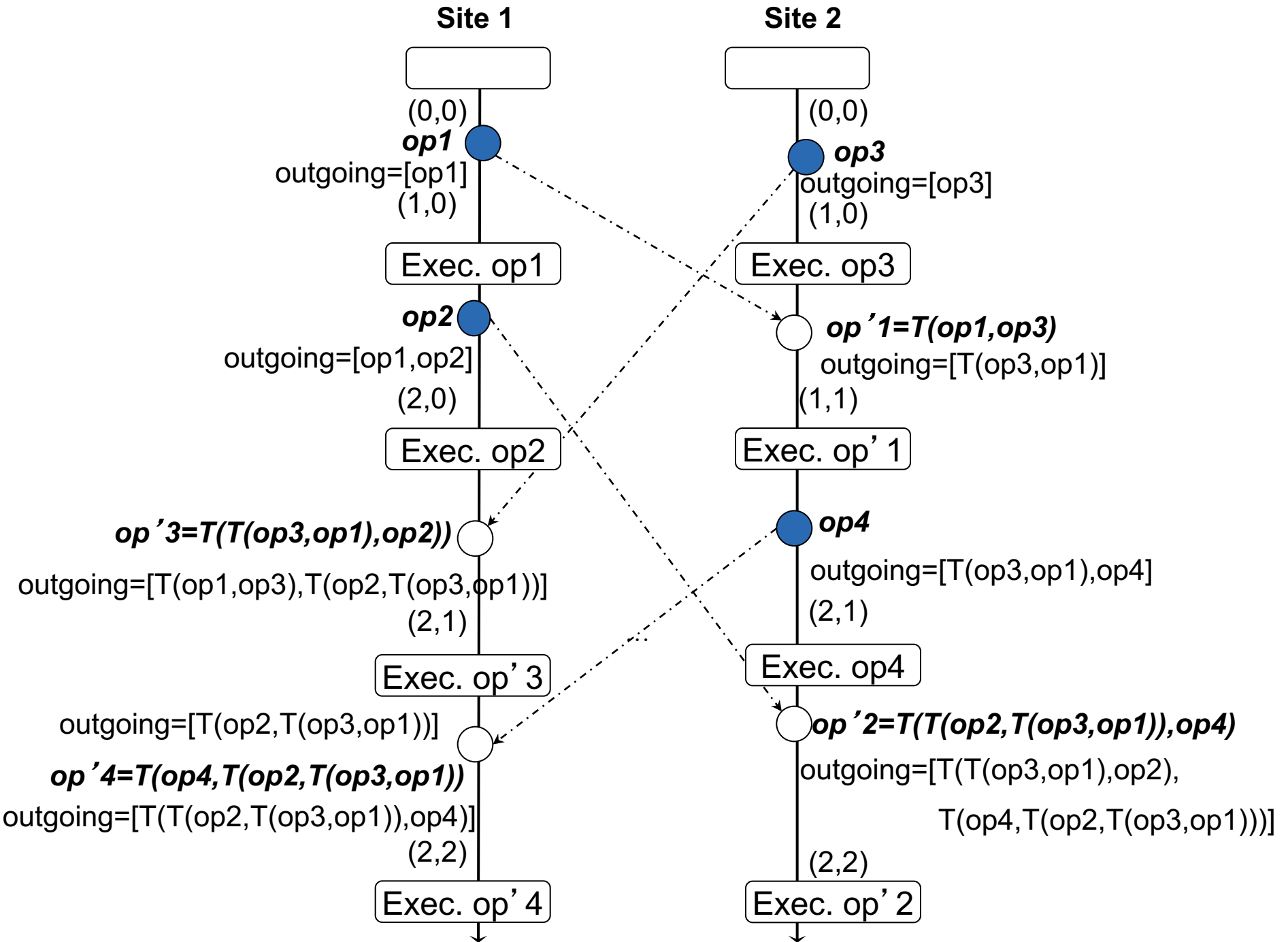
Jupiter algorithm

2 sites

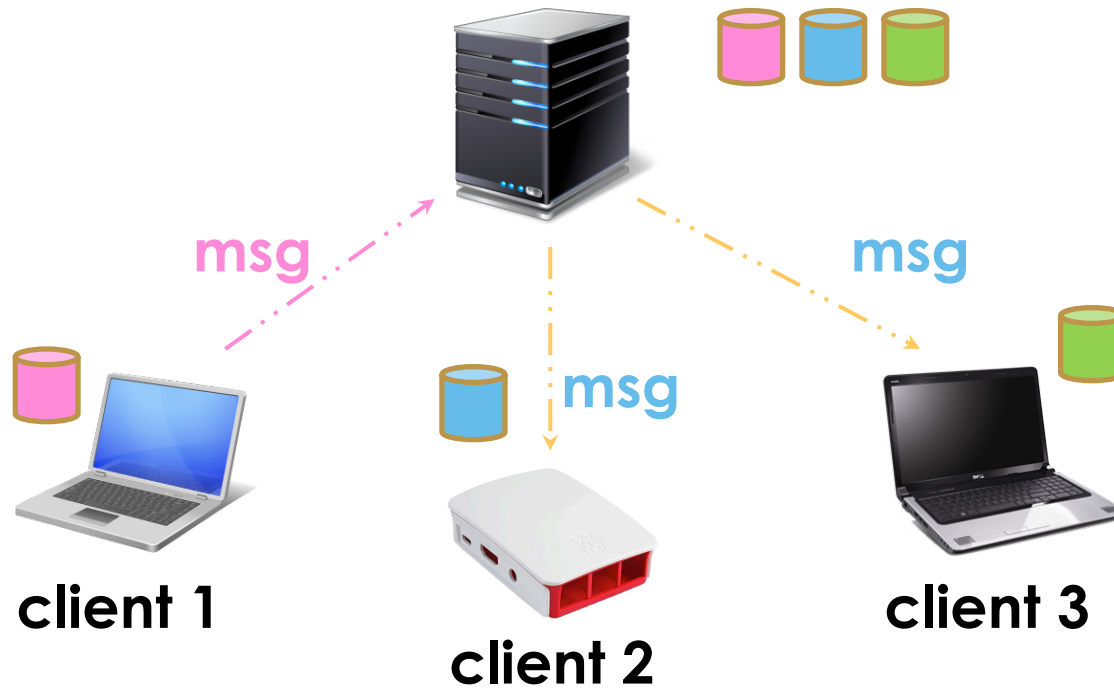
```
int myMsgs = 0; /* number of messages generated */
int otherMsgs = 0; /* number of messages received */
queue outgoing = {};
```

```
Generate(op) {
    apply op locally;
    send(op, myMsgs, otherMsgs);
    add (op, myMsgs) to outgoing;
    myMsgs = myMsgs + 1;
}
```

```
Receive(msg) {
    /* Discard acknowledged messages. */
    for m in (outgoing) {
        if (m.myMsgs < msg.otherMsgs)
            remove m from outgoing
    }
    /* ASSERT msg.myMsgs == otherMsgs. */
    for i in [1..length(outgoing)] {
        /* Transform new message and the ones in
           the queue. */
        {msg, outgoing[i]} = xform(msg, outgoing[i]);
    }
    apply msg.op locally;
    otherMsgs = otherMsgs + 1;
}
```



Jupiter algorithm – generalisation n Clients



apply msg.op locally;



**Algorithm changes
at server side**

apply msg.op locally;

```
for (c in client list) {  
    if (c != client)  
        send(c, msg);  
}
```

Jupiter algorithm

- Requires a server that performs transformations
- Not suitable for P2P environments
- False tie scenario gives different results according to integration order

