

# TIW 8

## Technologies Web synchrones et multi-dispositifs

---

CM5 - Édition partagée et algorithmes de synchronisation

Basé sur le cours de Michel Beaudouin-Lafon

<https://aurelient.github.io/react-iga/2020/>

Version automne à Lyon : <https://aurelient.github.io/tiw8/2020/>

# Deux approches

---

## Collaboration-transparent system

- ▶ Partage d'écran/fenêtre d'application mono-utilisateur
- ▶ Tour par tour
- ▶ Exemple: VNC

## **Collaboration-aware system**

- ▶ Conçu pour le travail collaboratif
- ▶ Gestion de la cohérence du contenu
- ▶ Robustesse, un problème de réseau n'affecte pas l'utilisation locale
- ▶ Exemple : Google Docs

# Plan

---

## Édition partagée

- ▶ Exemples
- ▶ Principes
- ▶ Défis de la synchronisation

## Algorithmes de synchronisation

- ▶ Operational Transform
- ▶ CRDT

## Groupware

# Aujourd'hui

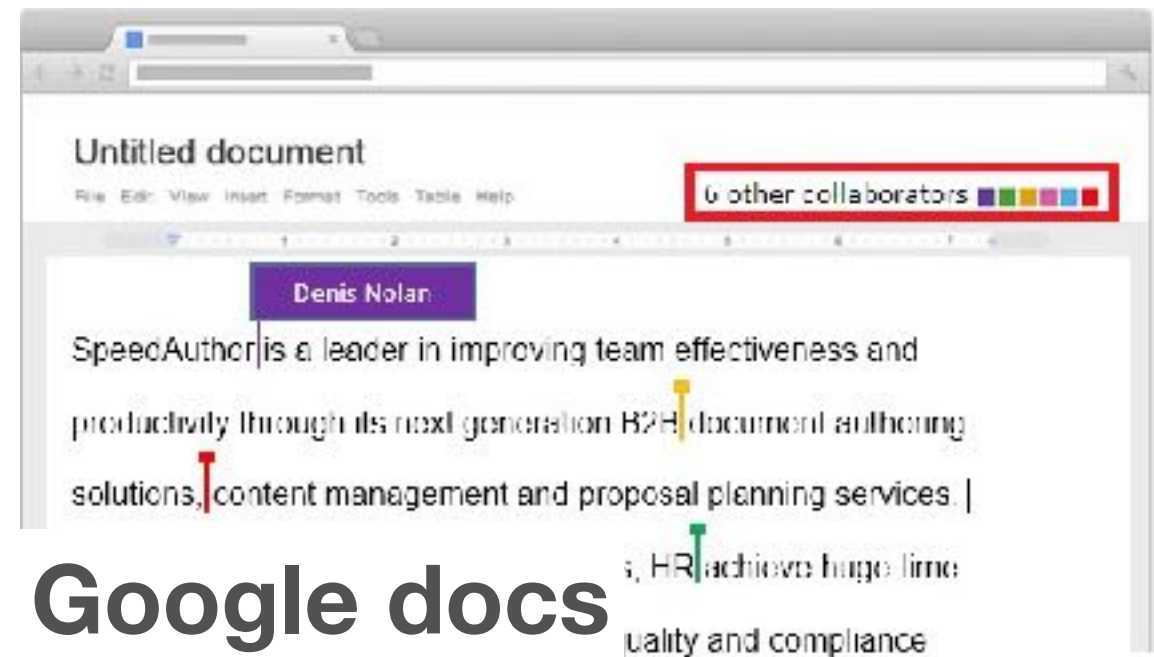
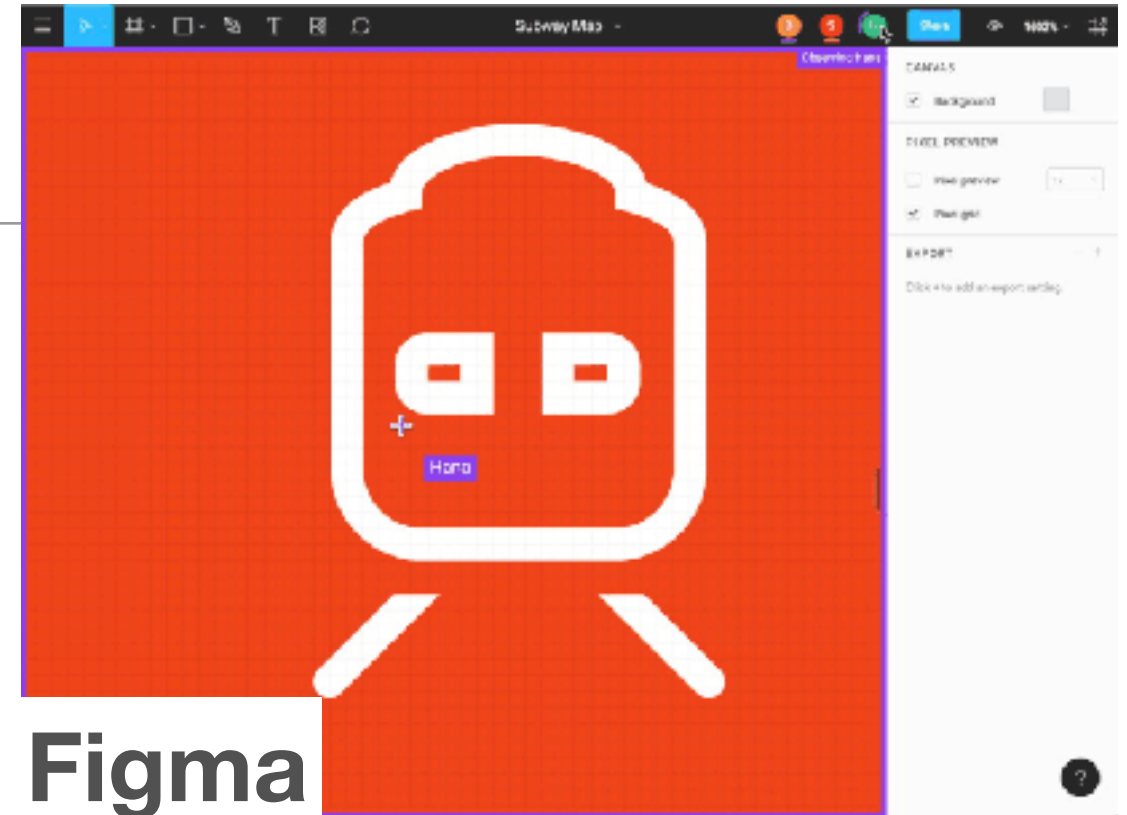
Pads: texte simple

Logiciels de :

- ▶ Traitement de texte
- ▶ Présentation
- ▶ Tableur
- ▶ ...

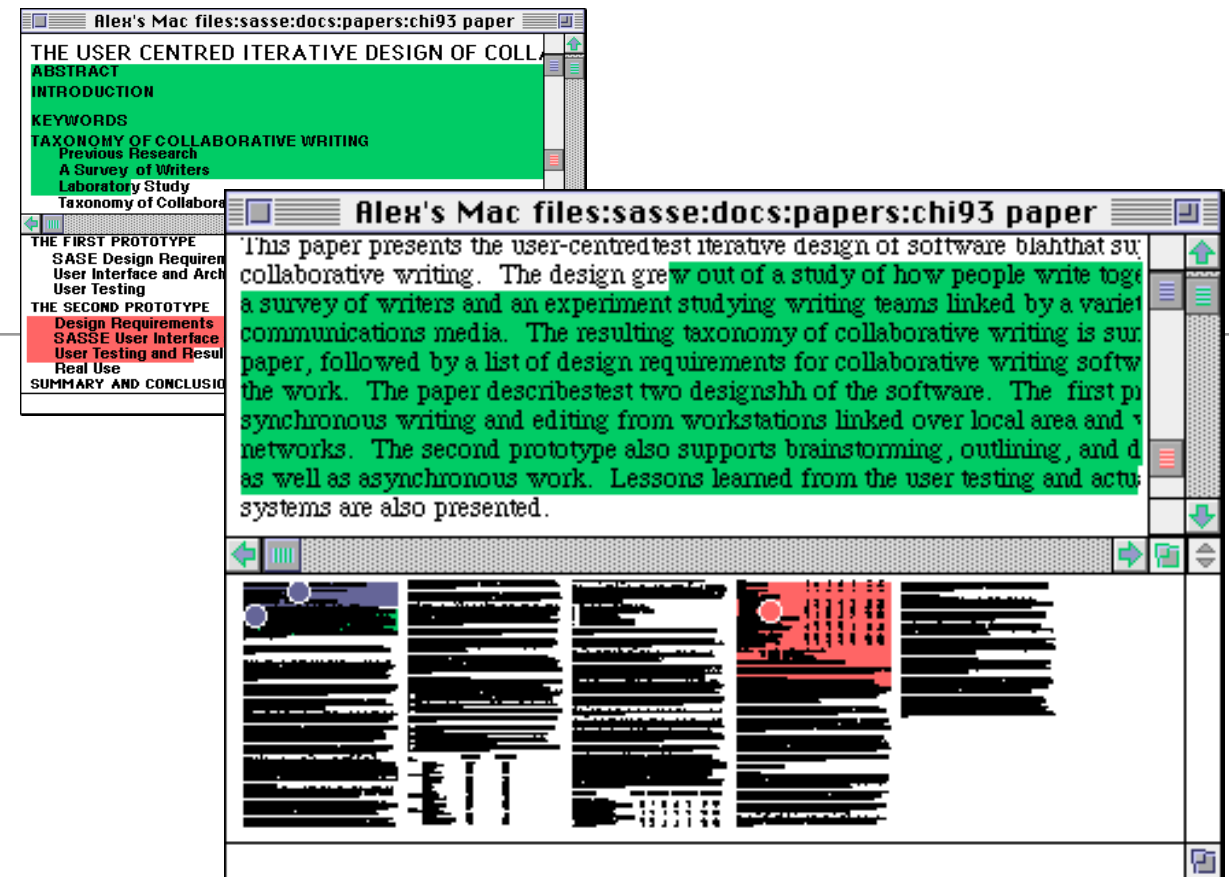
Logiciels de dessin

Logiciels 3D



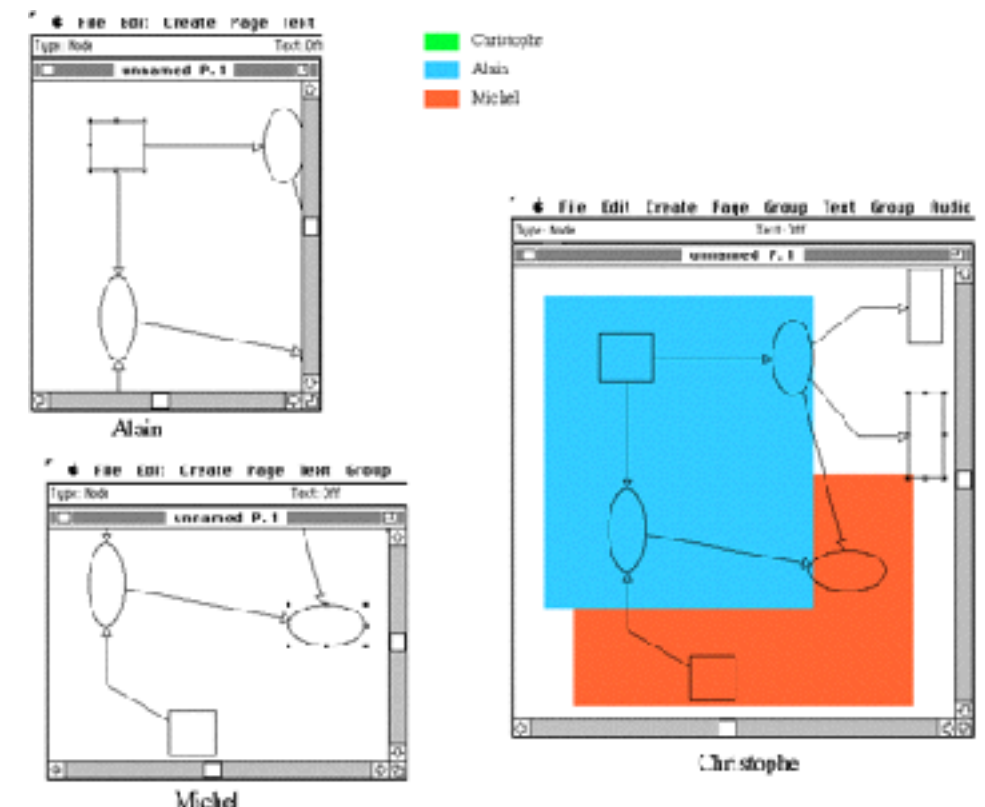
# Au début

## Éditeurs de texte



Sasse (Baecker et al., 1993)

## Logiciels de dessin



GroupDesign (Karsenty, 1992)

# Plan

---

## Édition partagée

- ▶ Exemples
- ▶ Principes
- ▶ Défis de la synchronisation

## Algorithmes de synchronisation

- ▶ Operational Transform
- ▶ CRDT

## Groupware

# Édition partagée

---

Création et édition collaborative de documents partagés.

Coordination : les utilisateurs travaillent dans le même but, avec une coordination implicite ou explicite de leurs actions

- ▶ “*awareness*” : on peut voir, être conscient de ce que font les collaborateurs.
- ▶ Régulation : les collaborateurs, planifient, suivent et évaluent l’activité en cours et ajustent leur comportement

# Différents types d'éditeurs

---

- ▶ Synchrone : changements visibles immédiatement
- ▶ Asynchrone : changements visibles plus tard
- ▶ Homogènes : les utilisateurs utilisent le même logiciel
- ▶ Hétérogènes : les utilisateurs peuvent utiliser des logiciels différents
- ▶ Collaboration-aware: offre des fonctionnalités d'awareness
- ▶ Collaboration-transparent : pas de fonctionnalités d'awareness.



# La congruence

---

Stefik et al., 1987

## Congruence de vues

- ▶ Partie du document en train d'être regardé

## Congruence de l'espace d'affichage

- ▶ Organisation de l'espace de travail

## Temporalité de la congruence

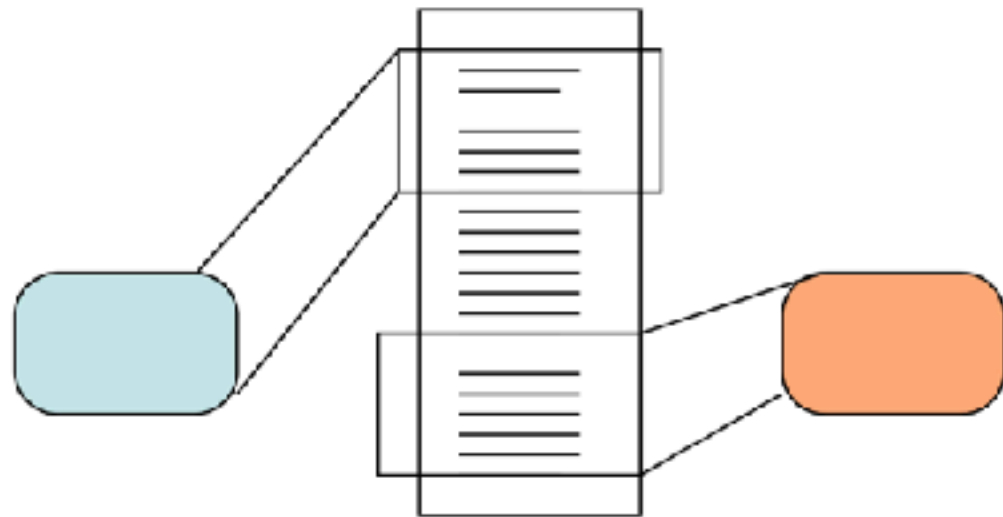
- ▶ Quand les changements sont vus par d'autres

# WYSIWIS / WYSIAWIS

---



WYSIWIS  
Congruence stricte des vues



WYSIAWIS  
Congruence relaxée des vues

# Un peu de vocabulaire

---

- ▶ Participant
- ▶ Session
- ▶ Invitation: Donner accès à un utilisateur à une session
- ▶ Prise de tour (turn-taking) : Quand un utilisateur peut éditer à la fois.
- ▶ Télé-pointeur : représentation du curseur des autres utilisateurs
  
- ▶ Couplage : comment les actions locales sont liées aux actions distantes
- ▶ Temps de réponse : Temps pour qu'une action soit exécutée localement
- ▶ Temps de notification : Temps pour qu'une action soit exécutée à distance
- ▶ Réplication: Gestion transparente des multiples copies d'un document
- ▶ Robustesse : sensibilité aux fautes

# Similarités et différences avec les BDs

---

## Des similarités

- ▶ Plusieurs utilisateurs, localement et à distance, accès concurrents, réplication, tolérance aux fautes...

## Des différences : L'utilisateur est en 1<sup>e</sup> ligne

- ▶ Il est conscient de ce qui se passe, peut résoudre les conflits...
- ▶ Congruence -> travaille t'on au même endroit ou pas
- ▶ Feedthrough -> on peut communiquer sur ce qu'on va faire
- ▶ Latecomers -> des personnes peuvent arriver en cours de route et devoir être mise à jour

# Quelles architecture ?

---

## Centralisé :

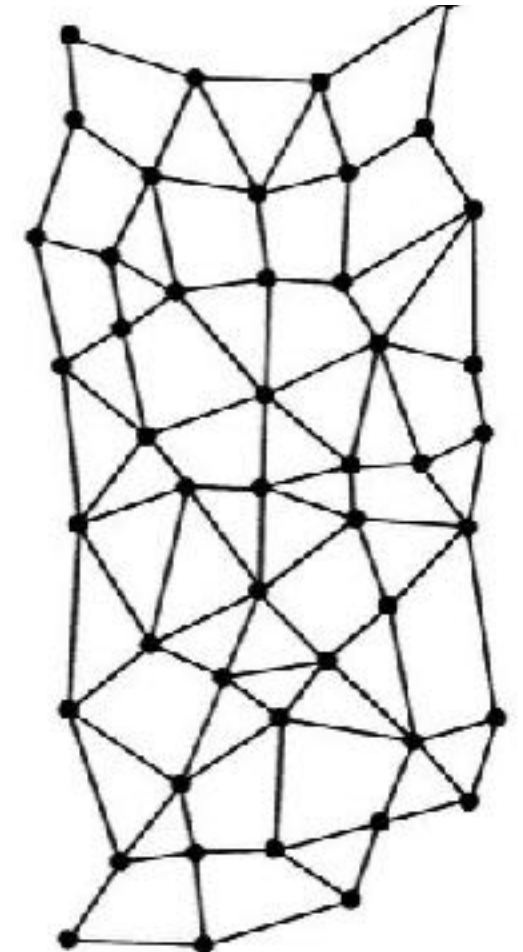
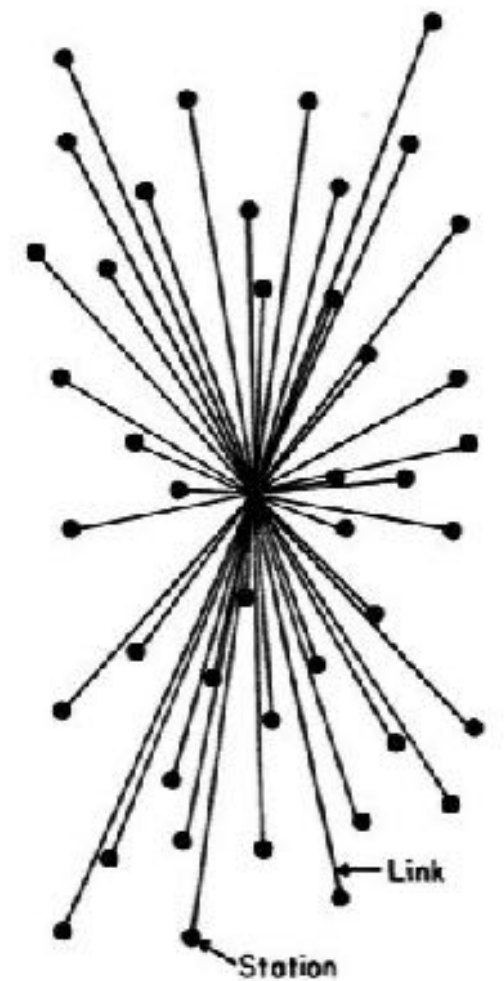
- ▶ Simple, temps de réponse moyen,
- ▶ manque robustesse

## Distribué/répliqué :

- ▶ Bon temps de réponse/notification, robuste,
- ▶ complexe à mettre en place (gestion répliquation, conflits)

## Hybride :

- ▶ répliqué avec quelques fonctions centralisées



# Plan

---

## Édition partagée

- ▶ Exemples
- ▶ Principes
- ▶ Défis de la synchronisation

## Algorithmes de synchronisation

- ▶ Operational Transform
- ▶ CRDT

## Groupware

# Opérations de base

---

HAT



CHAT



CAT

insert("C", 0)

delete(1)

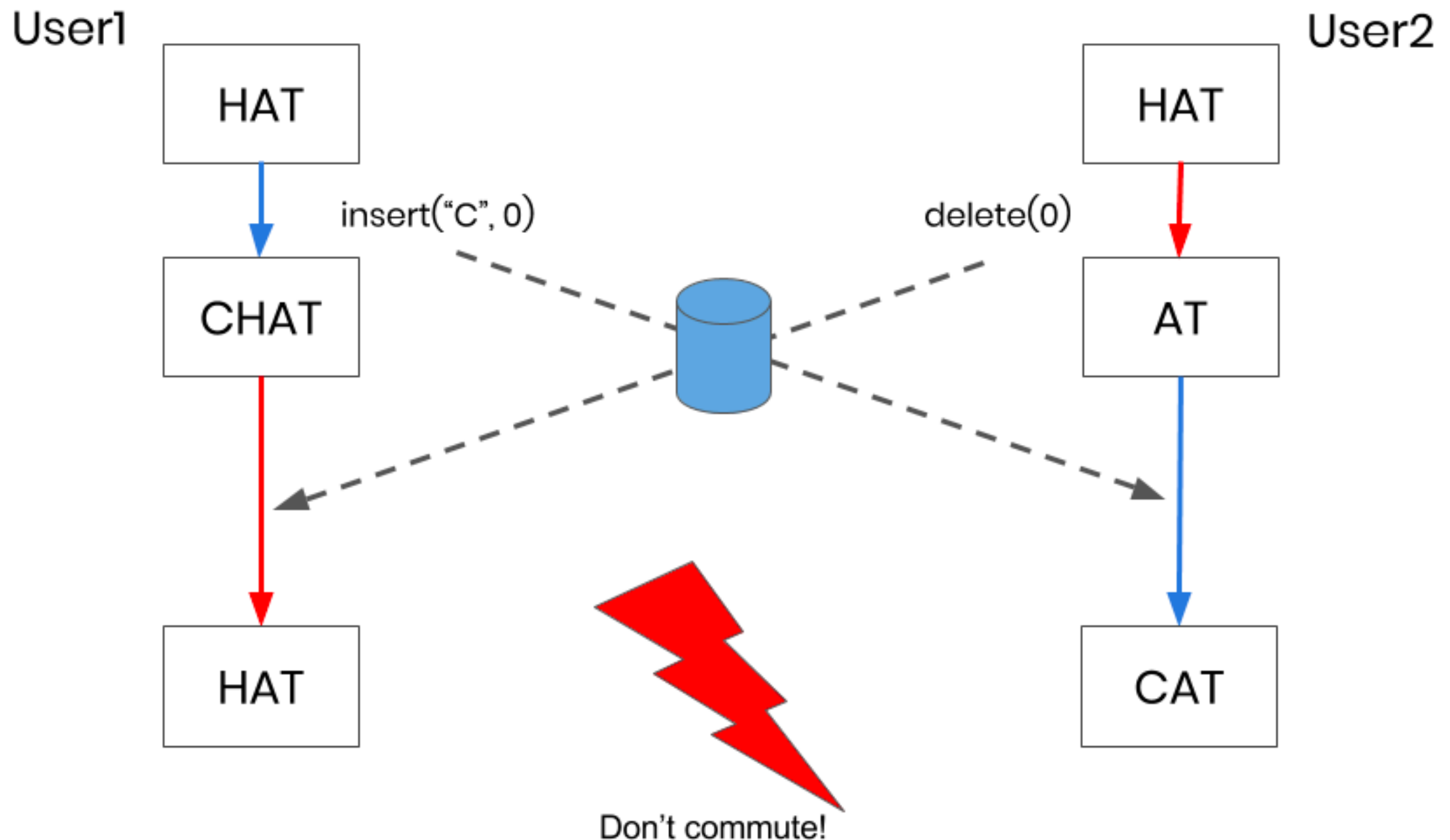
H	A	T
0	1	2

C	H	A	T
0	1	2	3

C	A	T
0	1	2

# Commutativité

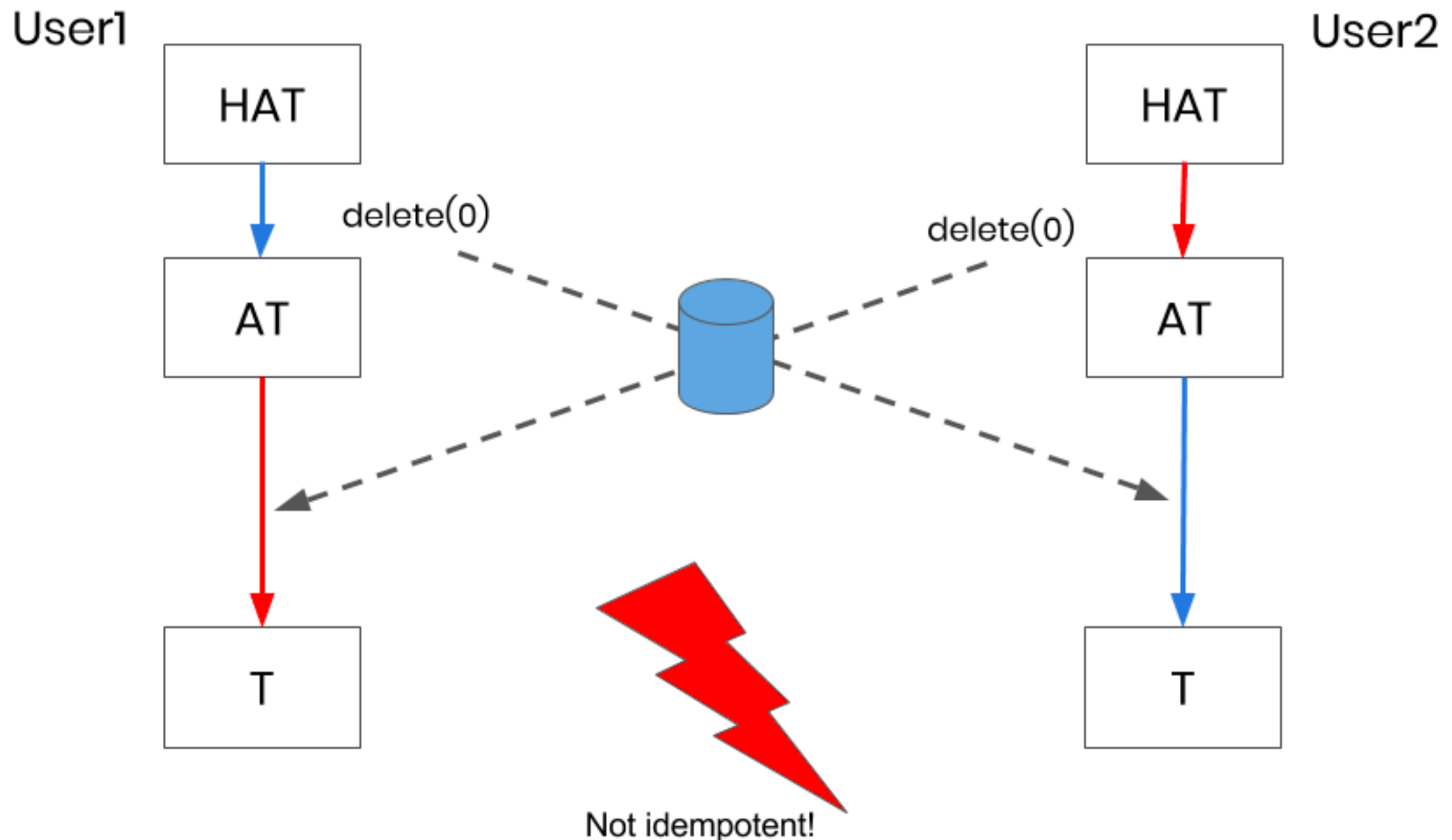
On peut changer l'ordre des opérations sans changer le résultat





# Idempotence

Une opération a le même effet qu'on l'applique une ou plusieurs fois



# Problèmes

---

Comment maintenir la cohérence de données distribuées ?

- ▶ Commutativité : les sites envoient des opérations qui doivent converger vers le même résultat quelque soit l'ordre d'application
- ▶ Idempotence : les opérations répétées produisent le même résultat

Deux classes d'algorithmes

- ▶ Pessimistes (locks)
- ▶ Optimistes (events + undo)

# Quelques algorithmes optimistes

---

- ▶ Operational transformation, e.g. dOpt (GROVE)
- ▶ Optimized undo/redo, e.g. ORESTE (GroupDesign)
- ▶ Conflict-Free Replicated Data Type (CRDT)

# Plan

---

## Édition partagée

- ▶ Exemples
- ▶ Principes
- ▶ Défis de la synchronisation

## Algorithmes de synchronisation

- ▶ Operational Transform
- ▶ CRDT

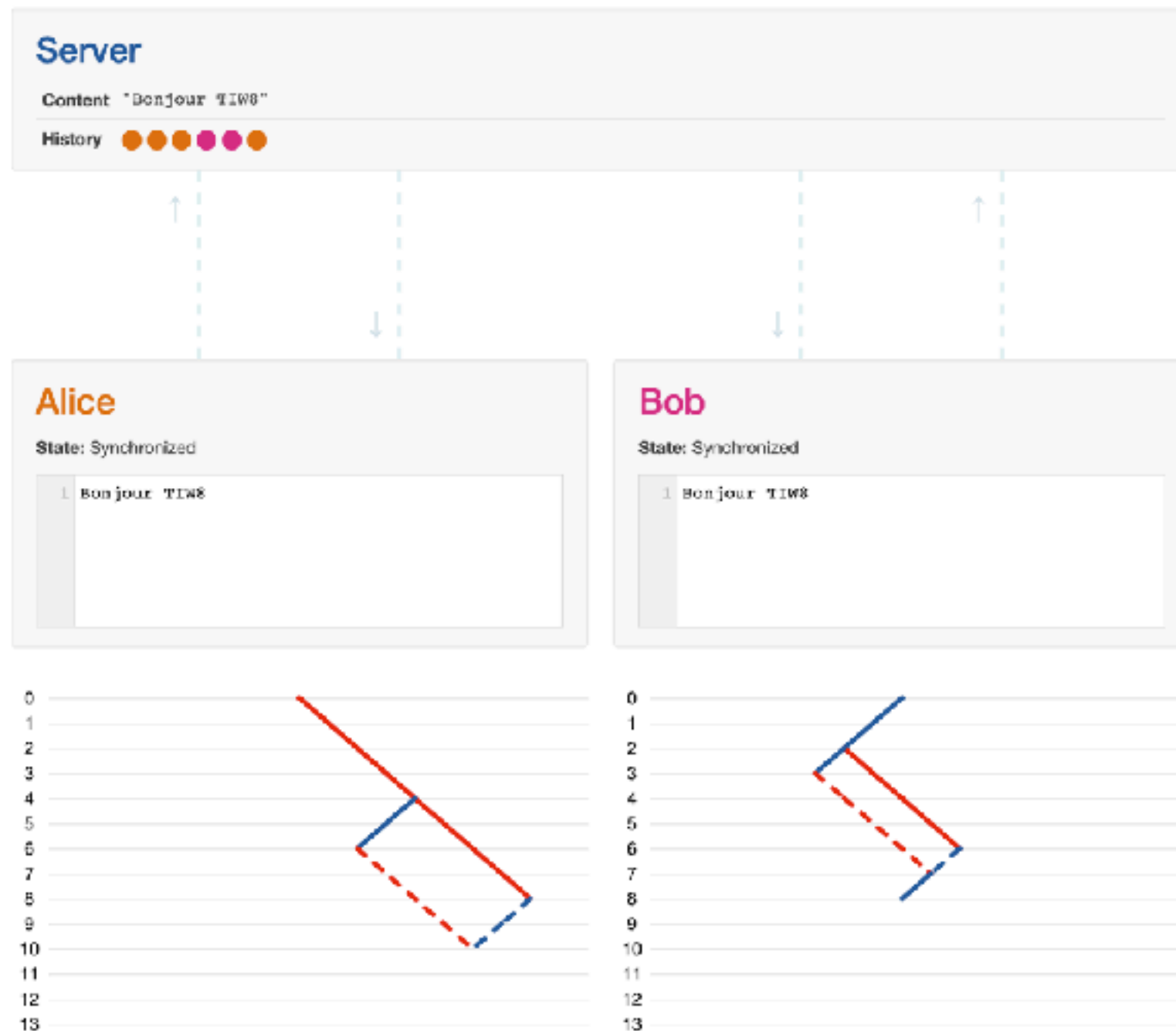
## Groupware

# Operational Transform : Principes

---

- ▶ On ordonne les opérations (lamport timestamps)  
-> maintient de la causalité
- ▶ Quand une opération n'arrive pas dans l'ordre elle est **transformée** pour prendre en compte les effets des opérations qui ont eu lieu avant.
- ▶ Pour toute paire d'opérations op1, op2,
  - ▶ Quand op2 arrive après op1 (qui a eu lieu avant)
  - ▶ On effectue une transformation  $T(op1, op2) = op'2$ , telle que
  - ▶  $op'2(op1(text)) = op1(op2(text))$
- ▶ Quand une opération arrive elle est transformée par celle qui ont eu lieu avant
- ▶ Peut nécessiter le maintien d'un historique infini

# Démo interactive (avec serveur)



<https://operational-transformation.github.io/visualization.html>

# Operational Transform

---

- ▶ Créer les transformations est difficile
- ▶ Peu/pas de preuve formelle
  - ▶ De nombreuses propositions alternatives
- ▶ Propriétés
  - ▶ Préservation de la causalité : les opérations qui dépendent les unes des autres sont exécutée dans le même ordre sur la même page.
  - ▶ Convergence : le même état sur toutes les pages qui ont traité tous les messages
  - ▶ Conservation des intentions : fait ce que l'utilisateur veut
- ▶ Bibliothèque : share.js
- ▶ Retour d'expérience sur le développement de Google Wave

# Plan

---

## Édition partagée

- ▶ Exemples
- ▶ Principes
- ▶ Défis de la synchronisation

## Algorithmes de synchronisation

- ▶ Operational Transform
- ▶ CRDT

## Groupware



# Conflict-Free Replicated Data Type (CRDT)

<https://hal.inria.fr/inria-00609399/document>

---

- ▶ Décentralisé
- ▶ Un type de données répliqué + interface qui fait que :
  - ▶ Les répliques puissent être modifiées sans coordinations avec les autres répliques
  - ▶ Quand deux répliques reçoivent les mêmes mises à jour, elle atteigne le même état, de manière déterministe
- ▶ Propriétés
  - ▶ Grow-only set :
  - ▶ Last-writer-wins register
- ▶ Plus simple à appréhender que Operational Transform
- ▶ Comme OT, de nombreuses variantes.

# Chaque élément devient un objet

---

A



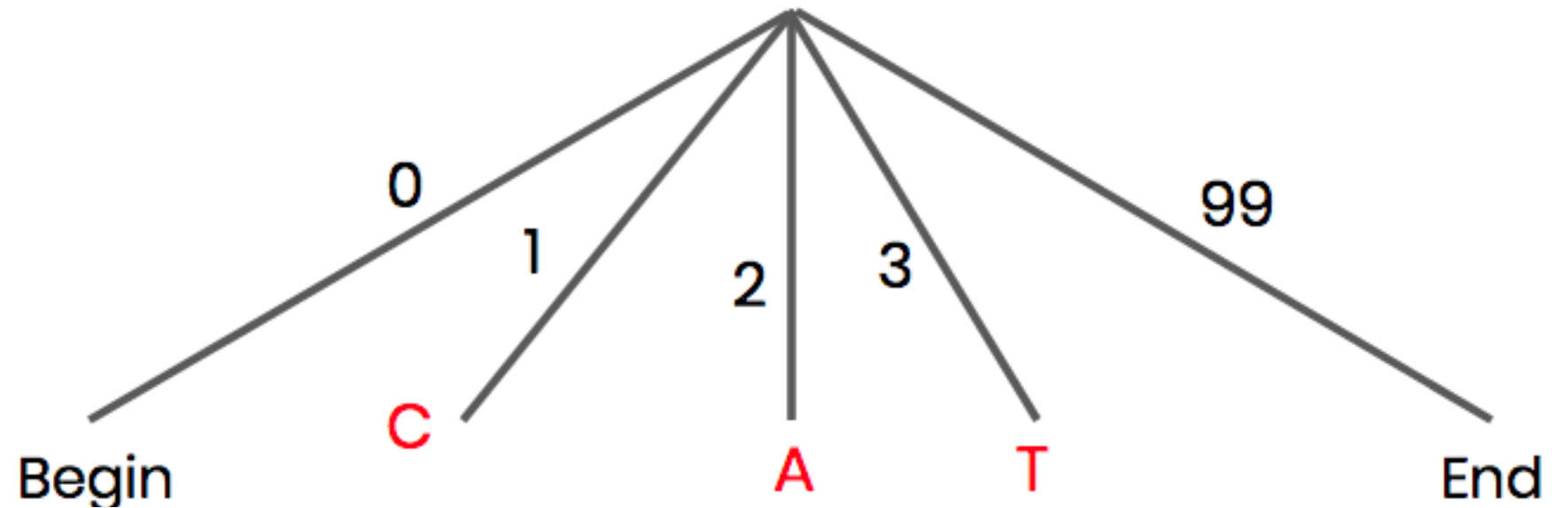
Character Object	
siteld	"skj7-329d"
value	"A"
position	[8, 9]

- ▶ siteld: un identifiant de l'utilisateur
- ▶ value: la lettre que l'objet représente
- ▶ position: une liste d'entier représentant la position du caractère dans le document, cette position est relative aux caractères autour.

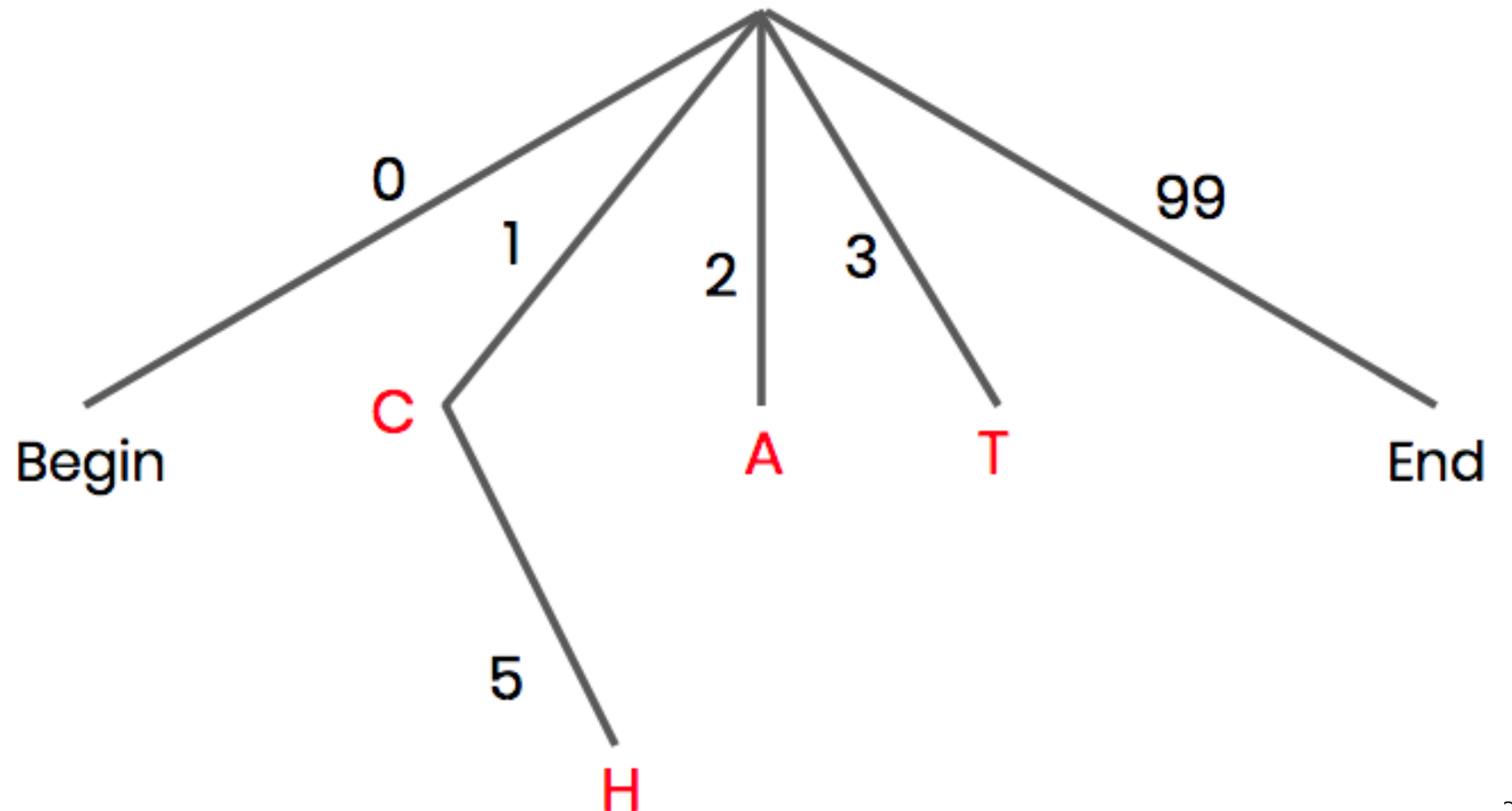
# Structure d'arbre

<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>

1. CAT



2. CHAT

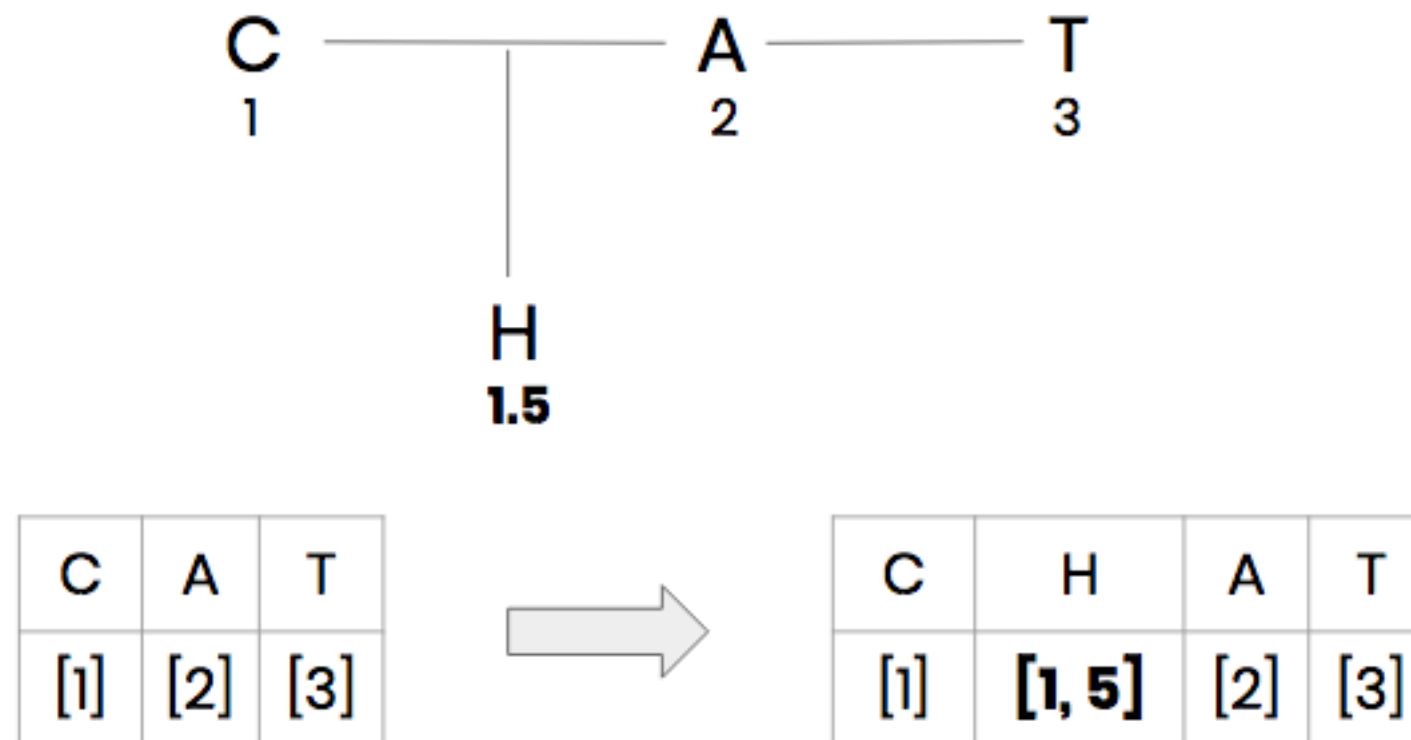


# Données

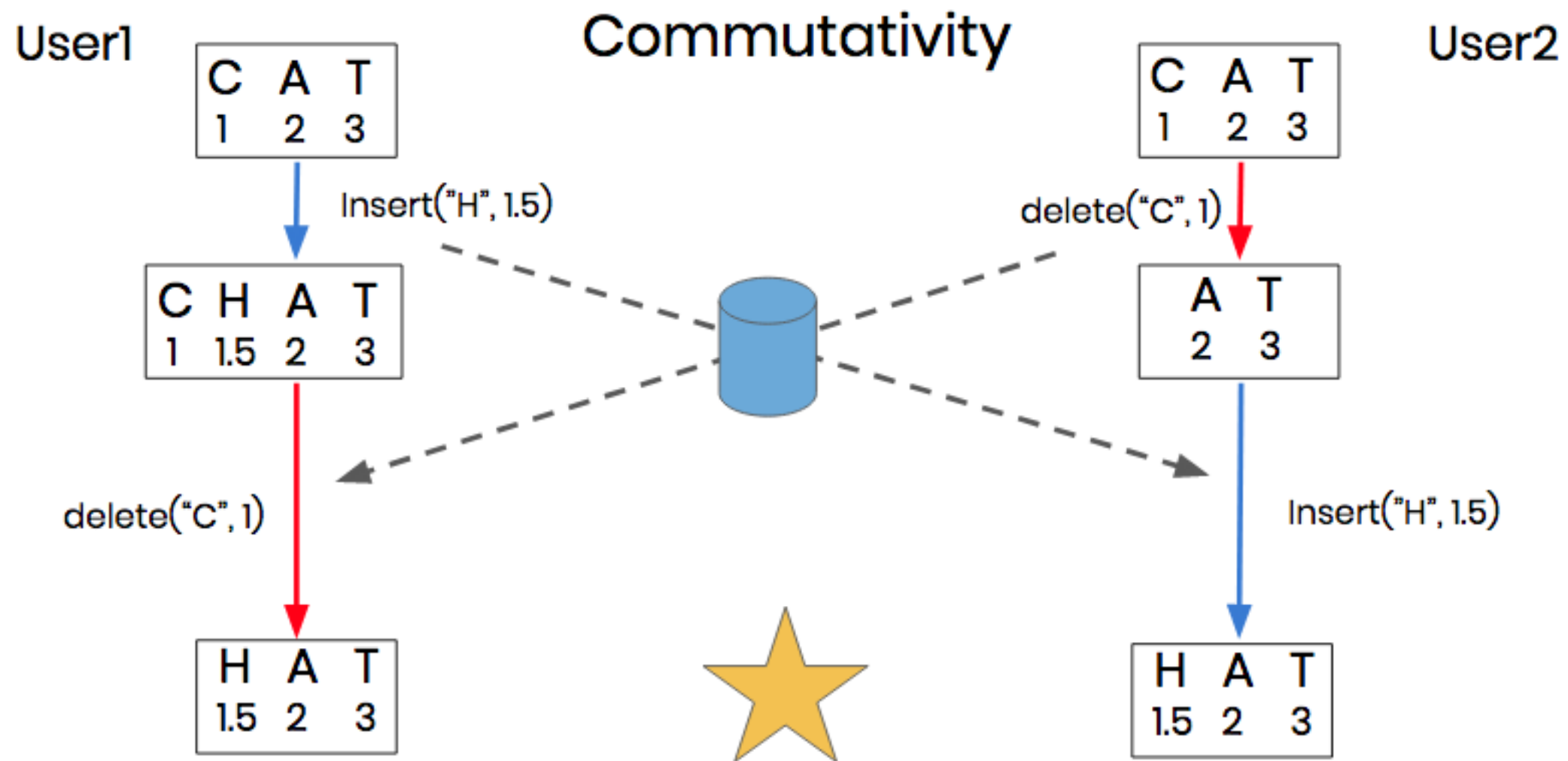
<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>

## Structure de données correspondante

### ► Positions relatives

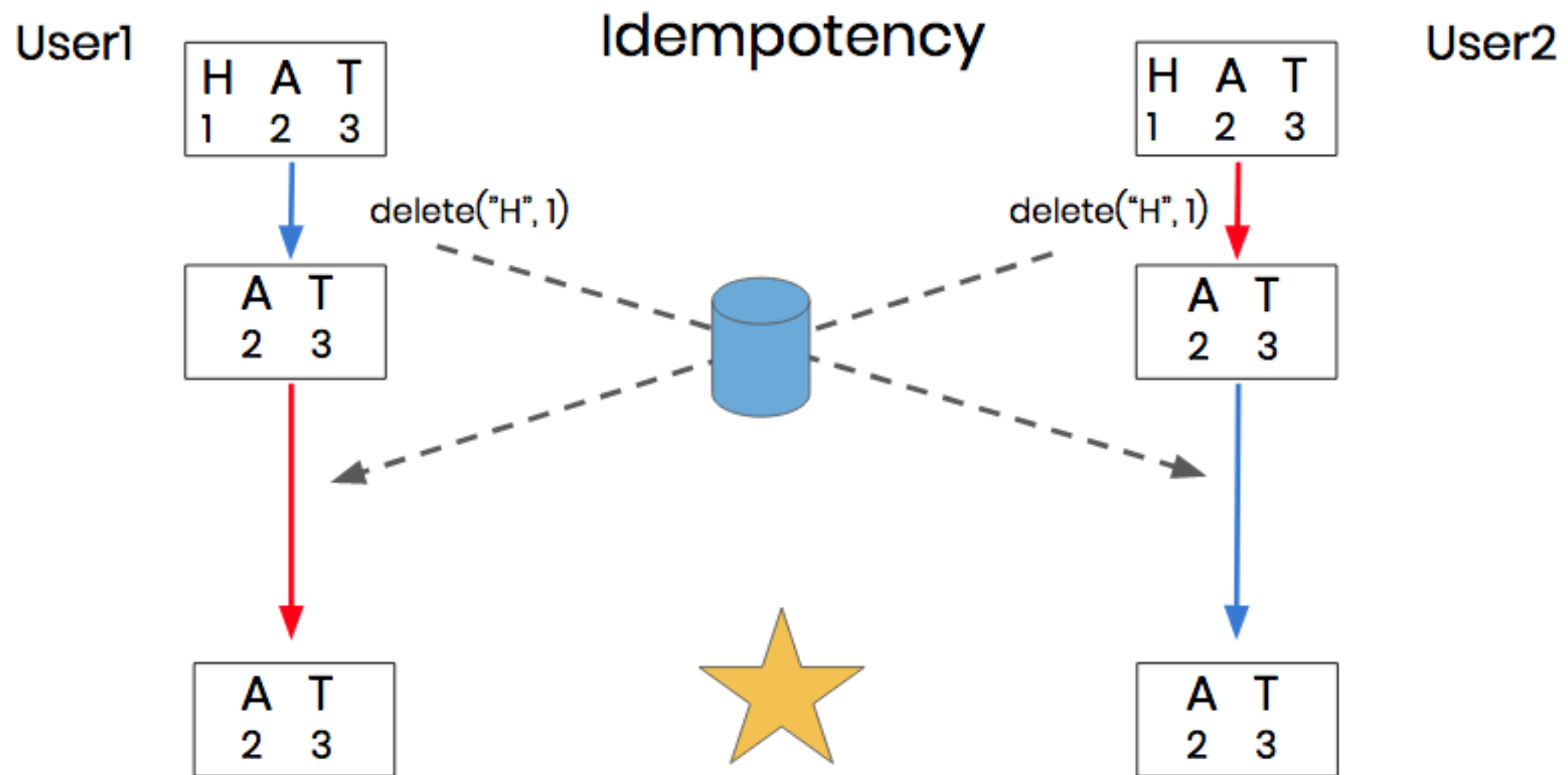


# Commutativité



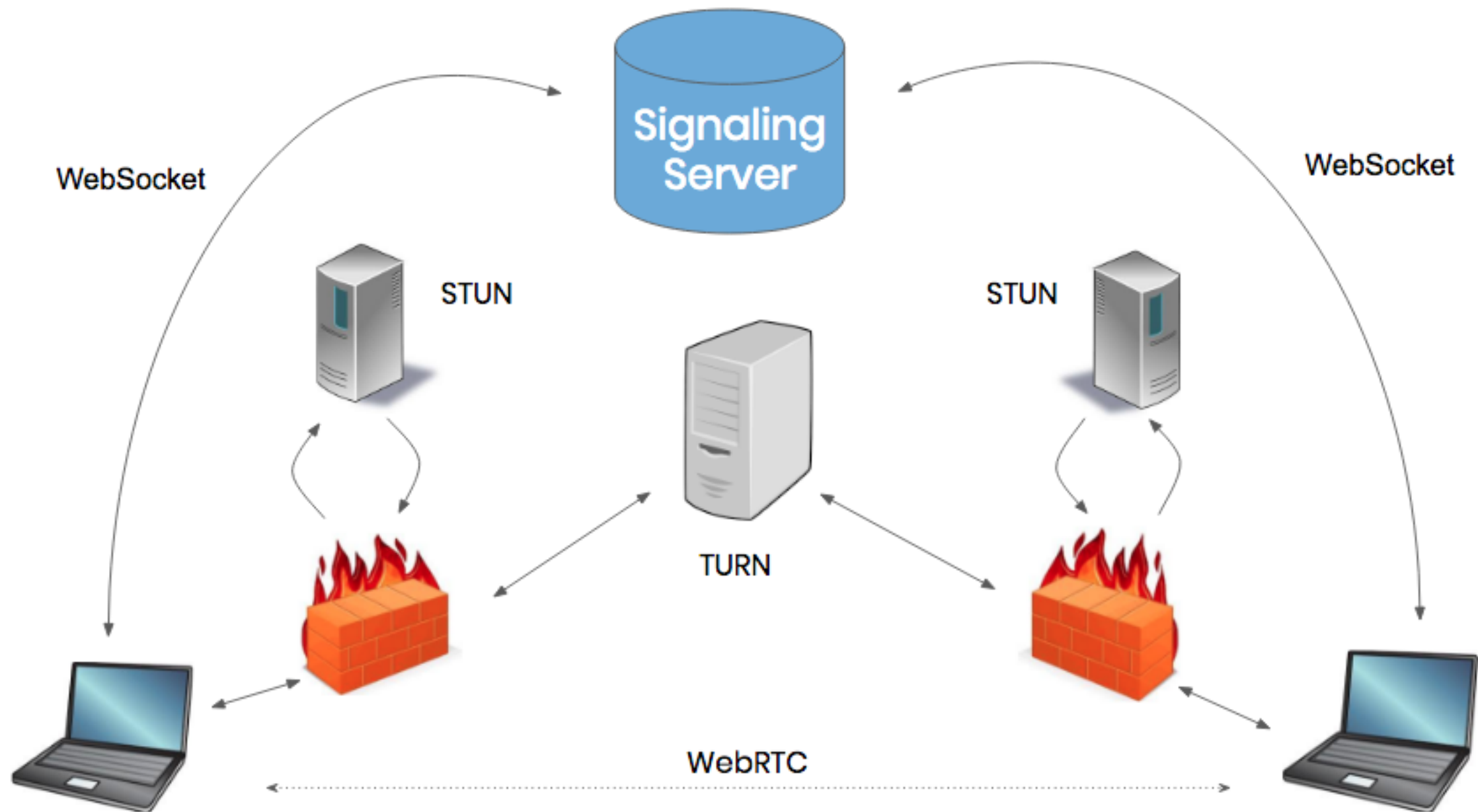
<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>

# Idempotence



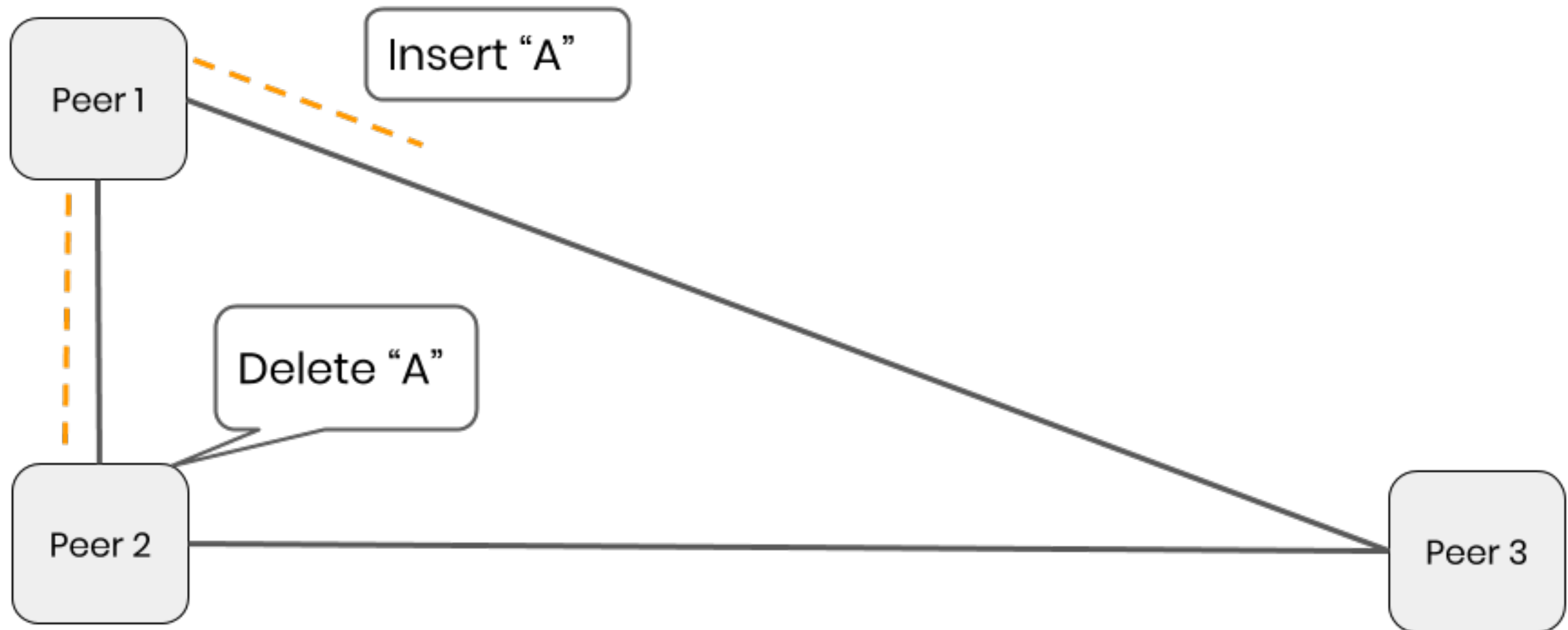
<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>

# Décentralisation -> WebRTC



# Causalité en distribué

---

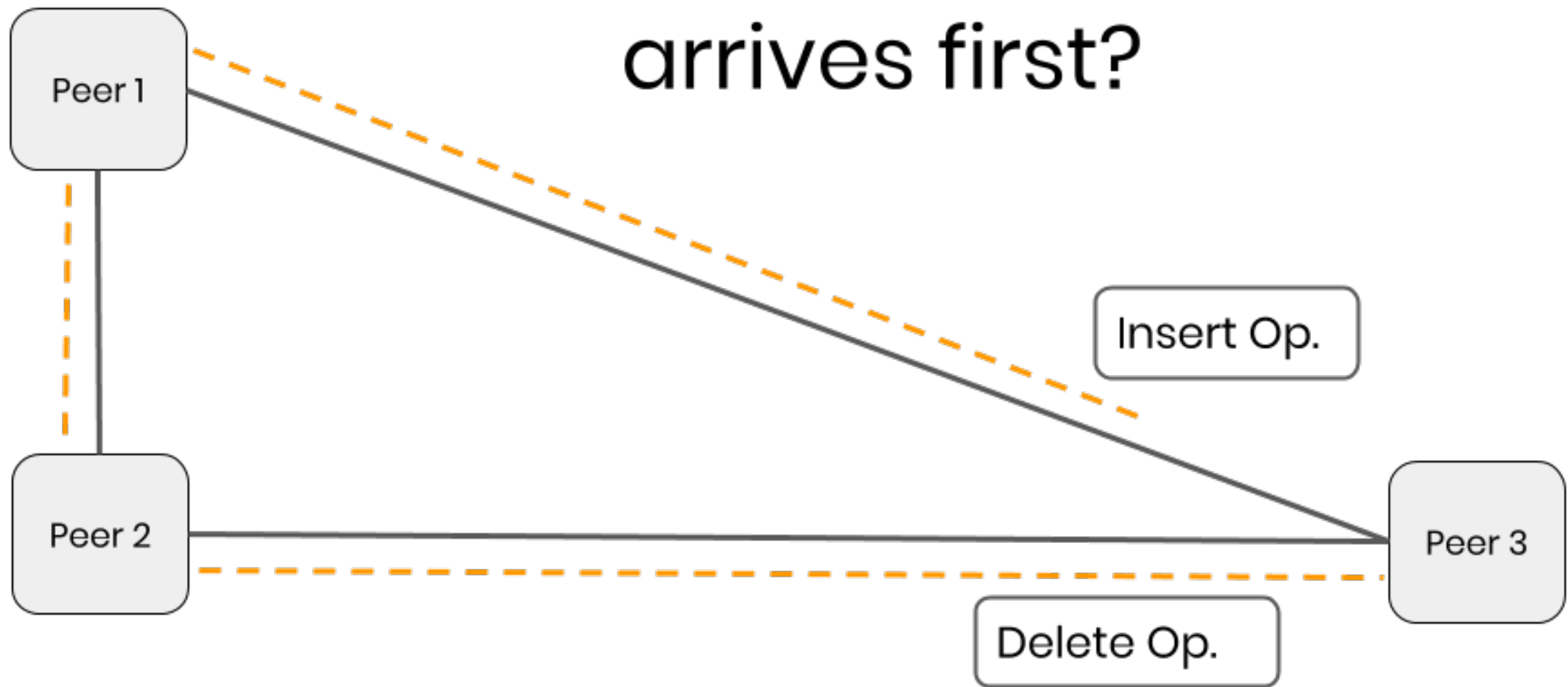


<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>



# Causalité en distribué

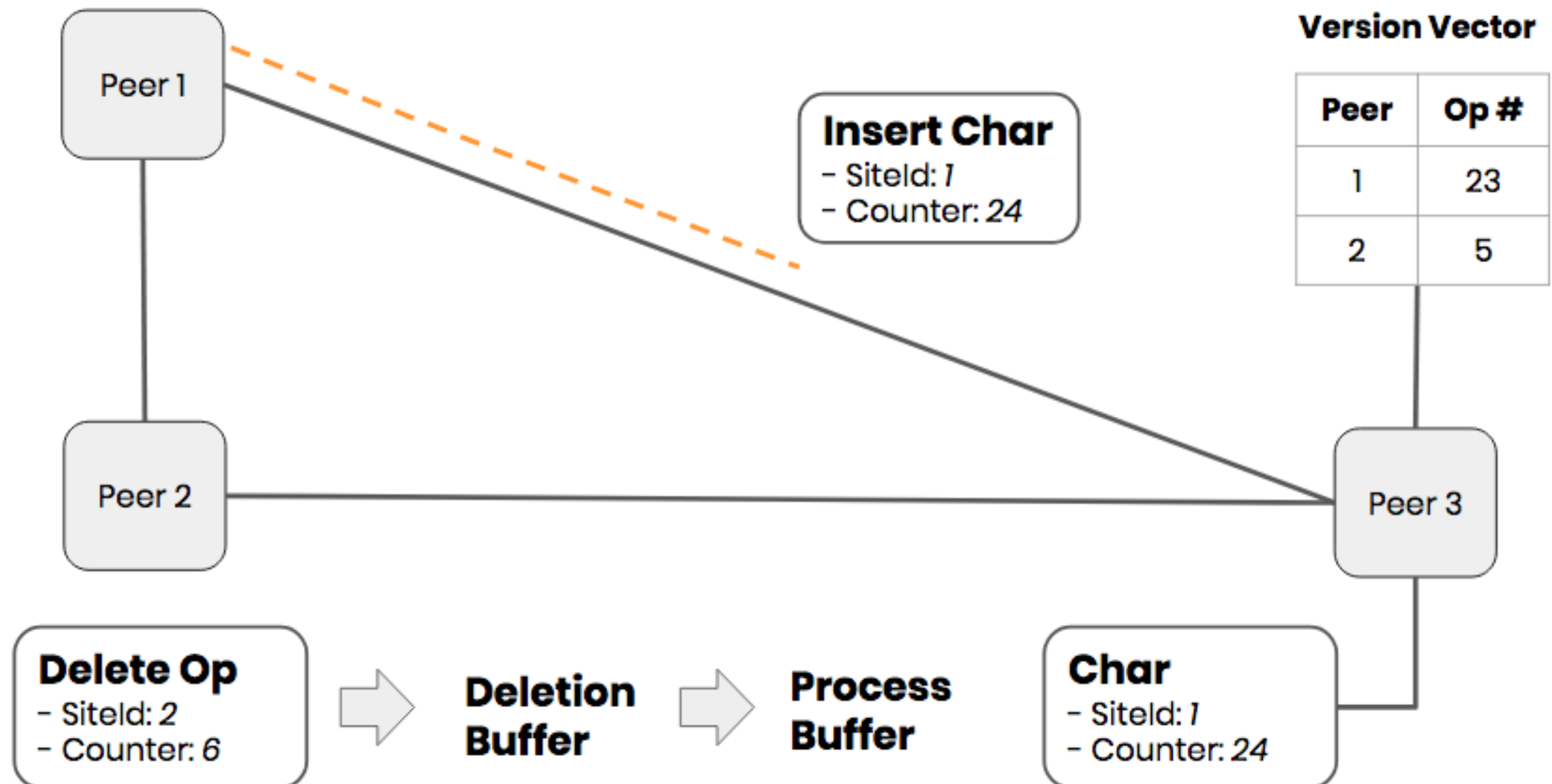
What if the delete arrives first?



<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>

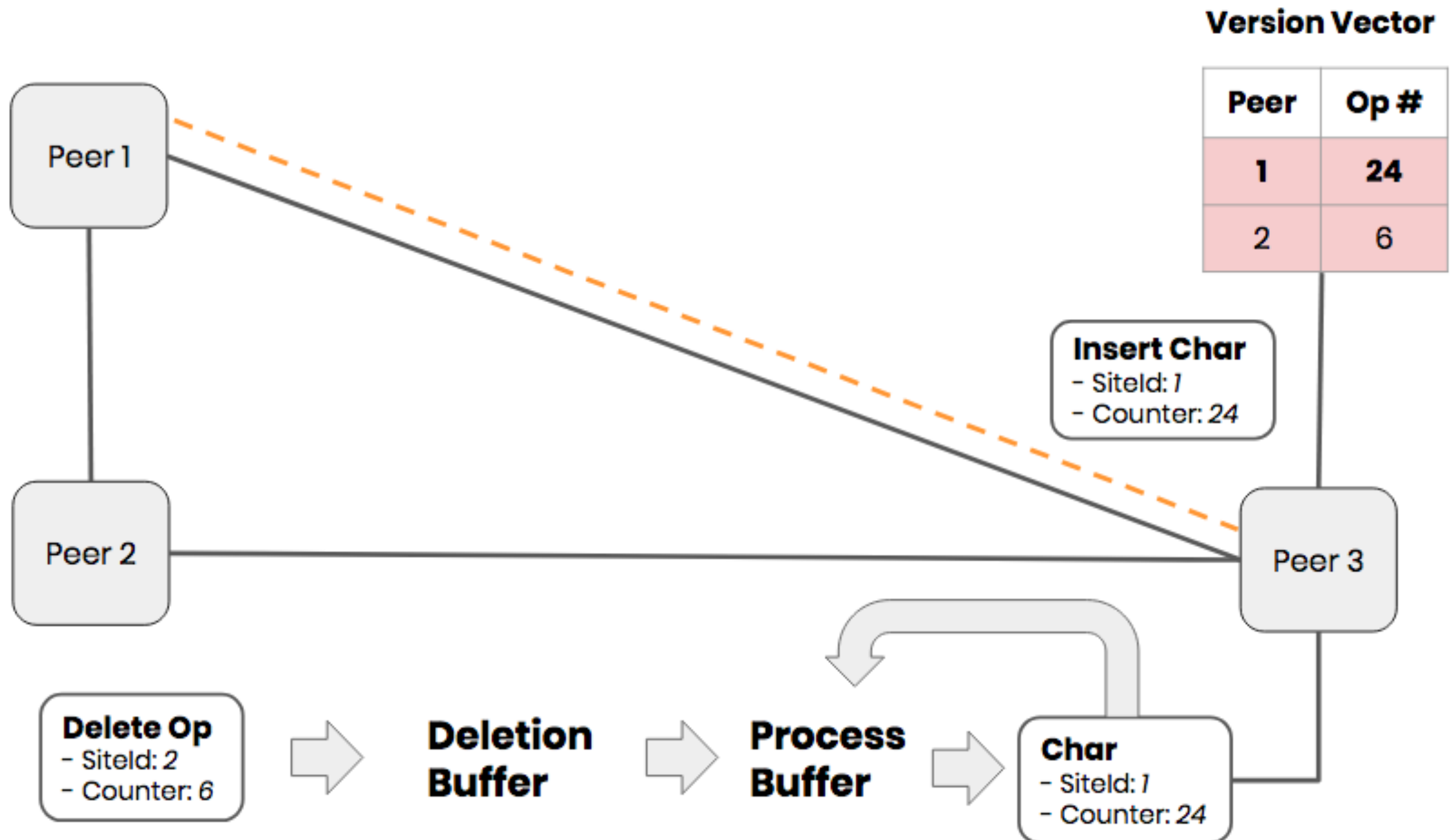
# Causalité en distribué

<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>



# Causalité en distribué

<https://hackernoon.com/building-conclave-a-decentralized-real-time-collaborative-text-editor-a6ab438fe79f>



# CRDT

---

Si les objets sont simples (ex : lettre) alors peu d'intérêt

Des cas pratiques d'application :

- ▶ Voir les optimisations de Conclave sur la structure d'objet :  
<https://conclave-team.github.io/conclave-site/#optimizations>
- ▶ Chez Figma, principes de CRDT mais centralisé :  
<https://www.figma.com/blog/how-figmas-multiplayer-technology-works/>

Des pistes pour intégrer CRDT et Operational Transform

- ▶ Operational Replicated Data Types

# Aller plus loin

---



<https://github.com/automerge/automerge>



<https://github.com/yjs/yjs>



Martin Kleppmann, University of Cambridge  
@martinkl mk428@cst.cam.ac.uk

## CRDT the hard parts



**Ink & Switch**

LEVERHULME  
TRUST



Quelles similarités, quelles différences  
avec les jeux vidéos ?

# Plan

---

## Édition partagée

- ▶ Exemples
- ▶ Principes
- ▶ Défis de la synchronisation

## Algorithmes de synchronisation

- ▶ Operational Transform
- ▶ CRDT

## Groupware

# Quels outils pour développer cela ?

---

Peu encore aujourd'hui

Des prototypes de recherche :

- ▶ Groupkit (Univ. de Calgary)
- ▶ ReticularSpaces (IT University of Copenhagen)



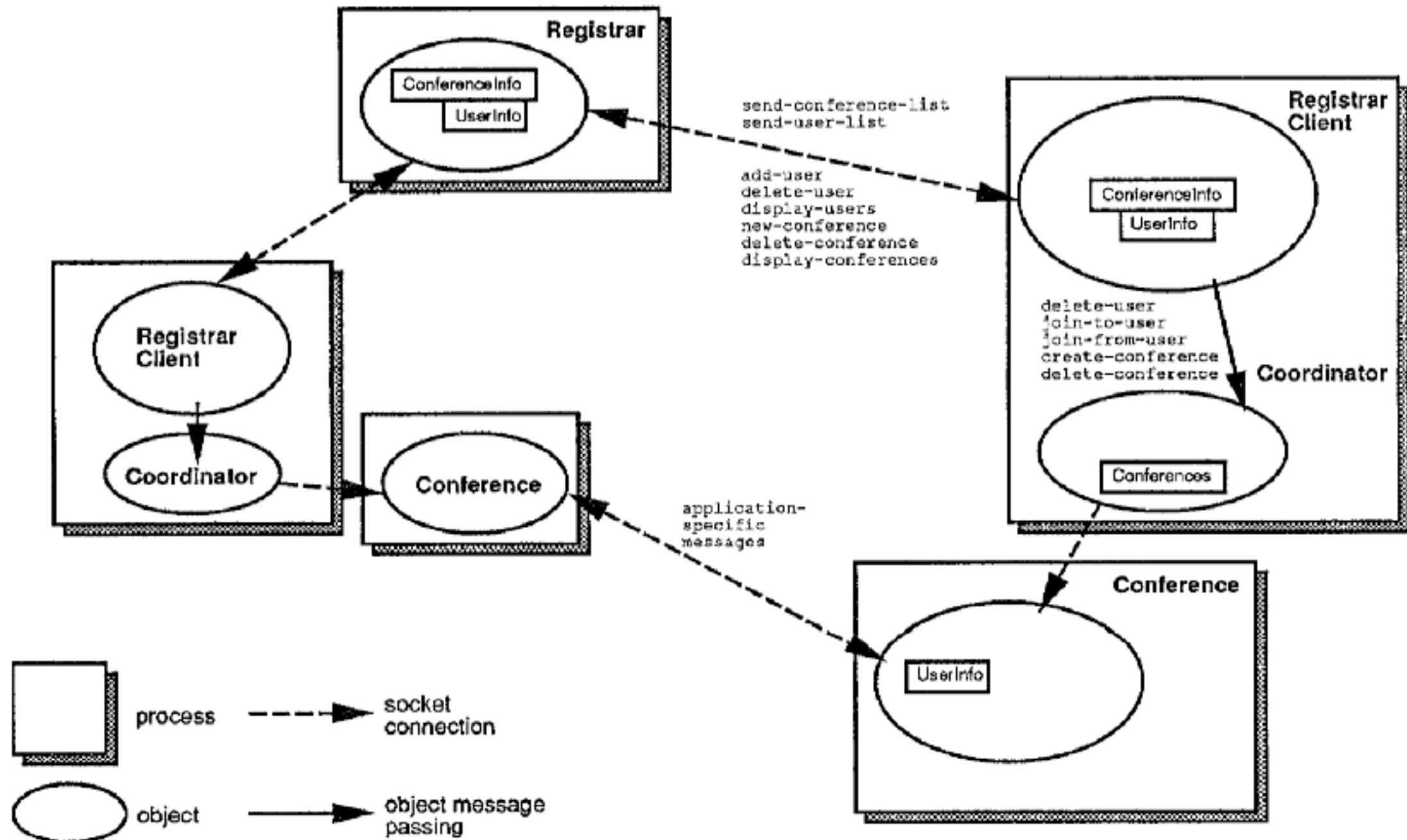
# Groupkit

---

- ▶ Gestion de sessions utilisateurs (arrivées, départs)
- ▶ Gestion de la distribution de données (1:1, 1:n)
- ▶ Widgets spécifiques pour l'interaction collaborative

# GroupKit : architecture

[https://www.researchgate.net/publication/220878991\\_GROUPKIT\\_A\\_Groupware\\_Toolkit\\_for\\_Building\\_Real-Time\\_Conferencing\\_Applications](https://www.researchgate.net/publication/220878991_GROUPKIT_A_Groupware_Toolkit_for_Building_Real-Time_Conferencing_Applications)

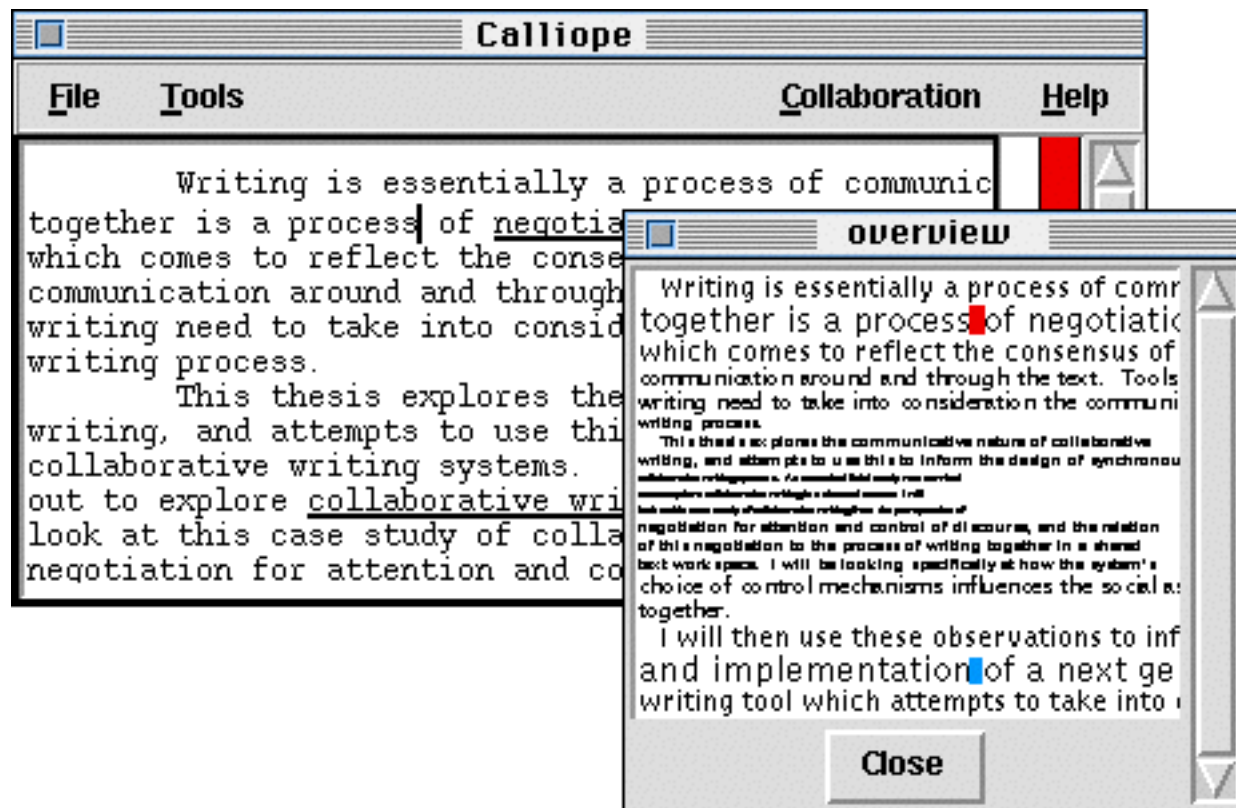


# Faciliter l'awareness

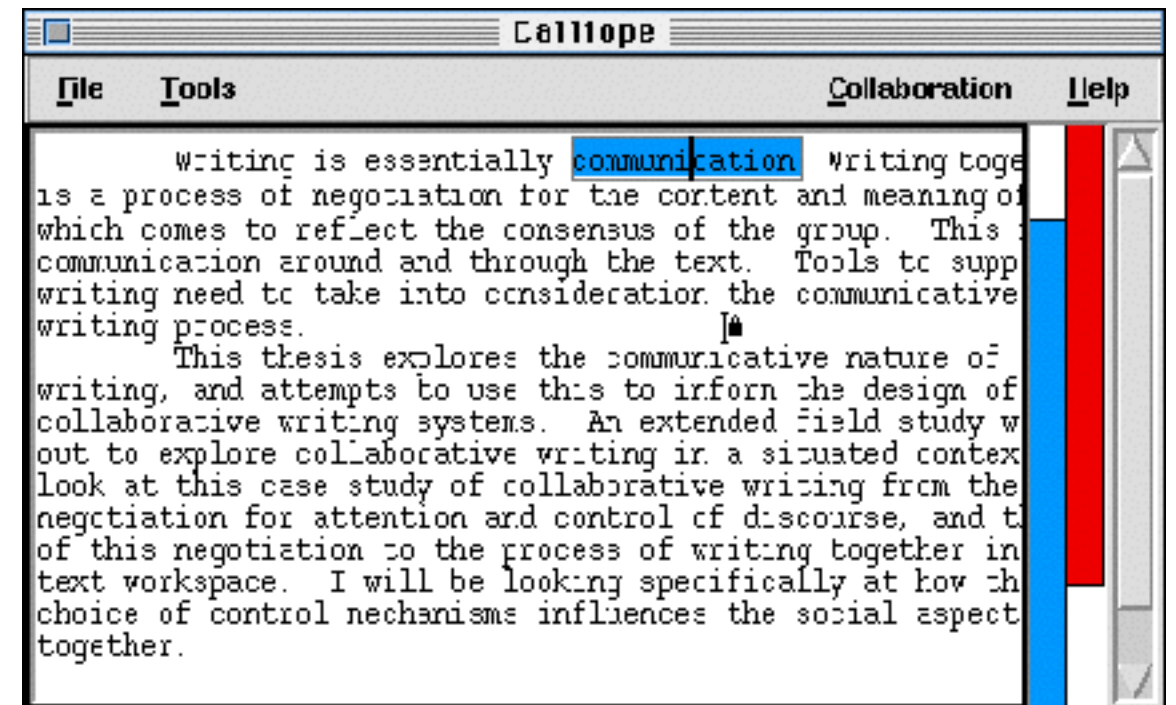
---

- ▶ Qui participe à l'activité ?
- ▶ Où sont-ils ?
- ▶ Que voient-ils ?
  
- ▶ Quel est leur niveau d'activité ?
- ▶ Que font-ils, qu'utilisent-ils ?
- ▶ De quoi ont-ils besoin ?
  
- ▶ Que vont-ils faire ?
- ▶ Que peuvent-ils faire ?

# Awareness widgets

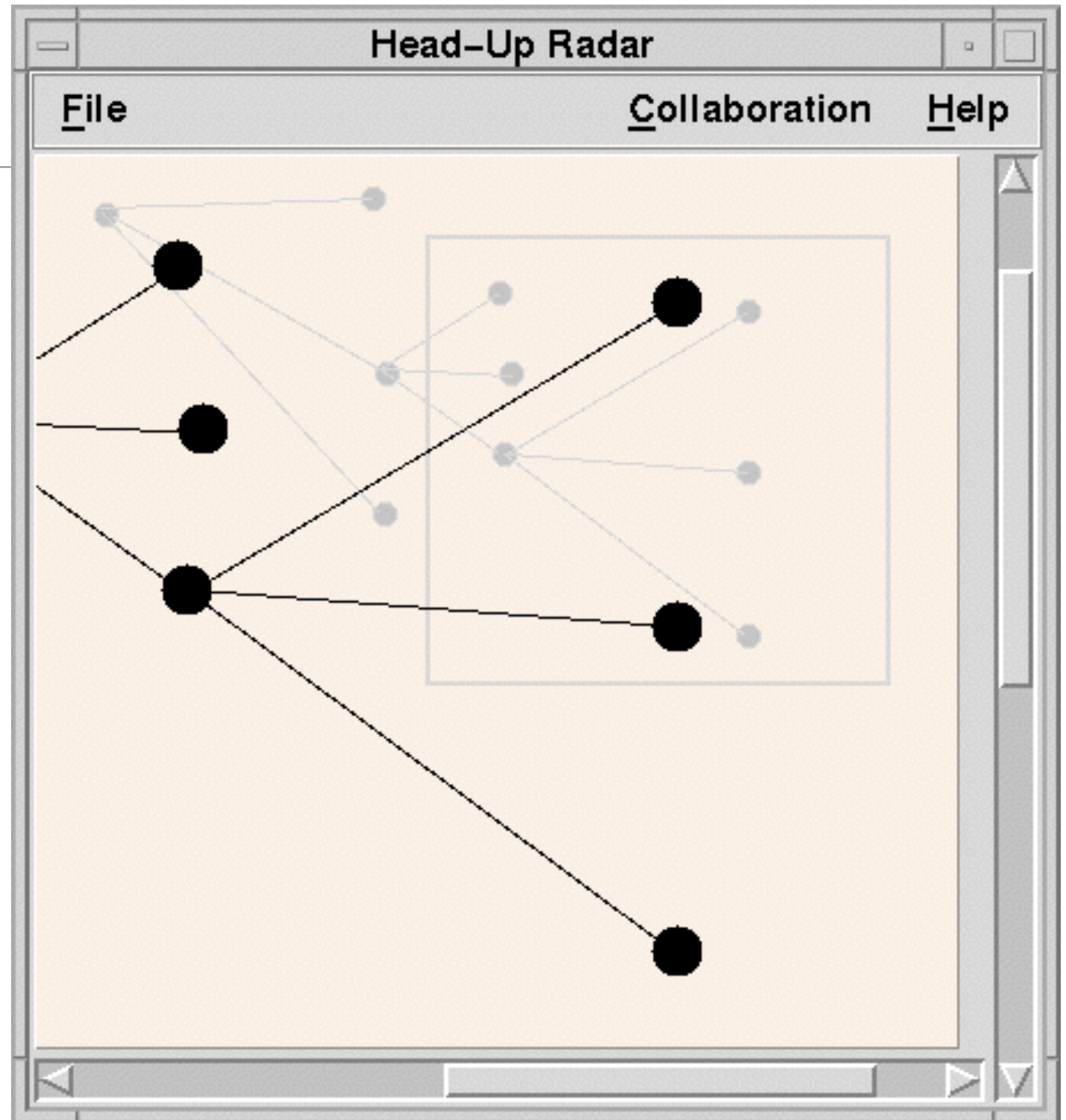


Télé-pointeurs

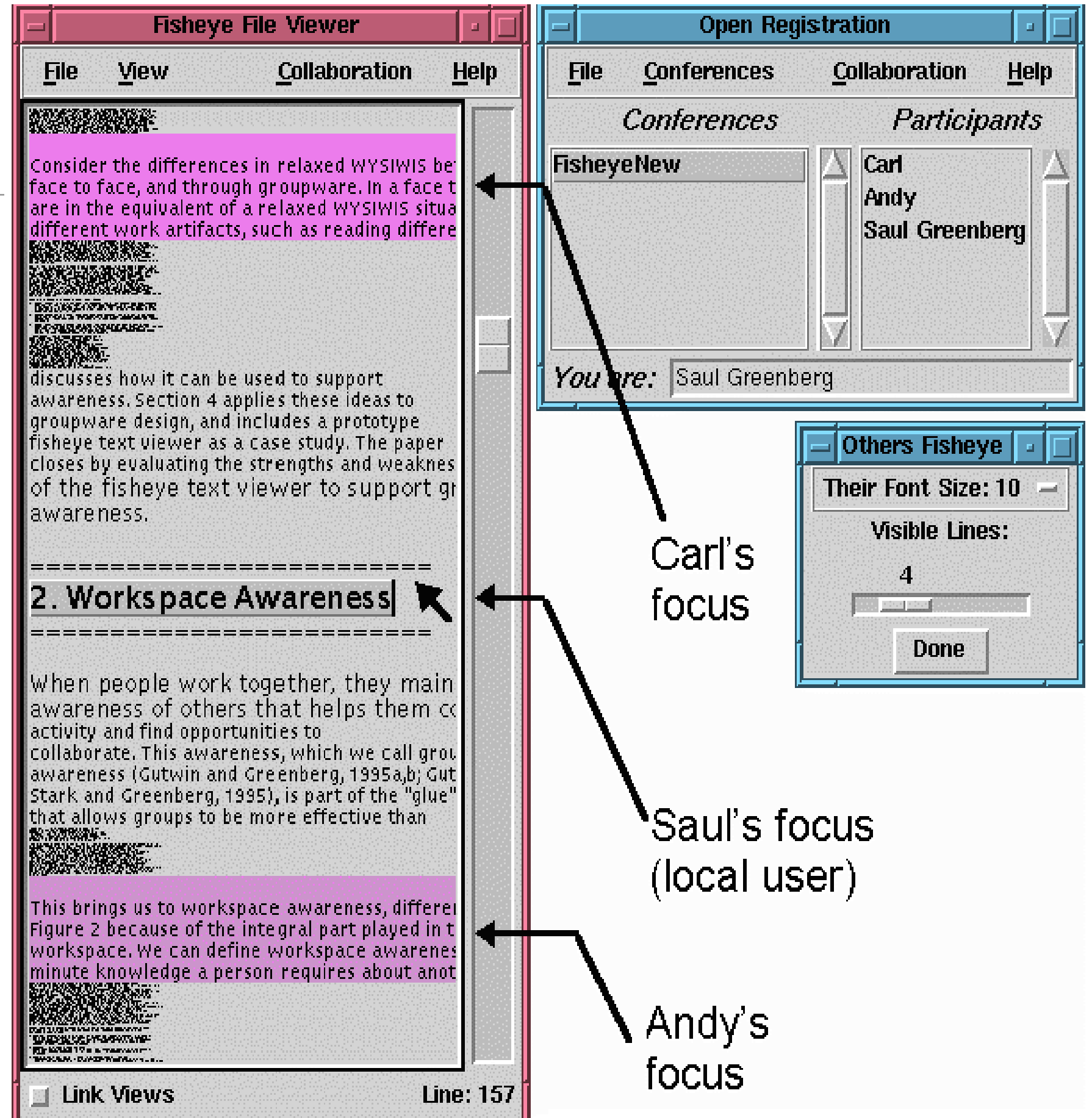


Barres de défilement multiples

# Vue radar



# Vue Fisheye

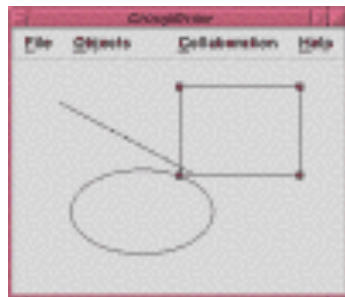


# GroupKit : applications

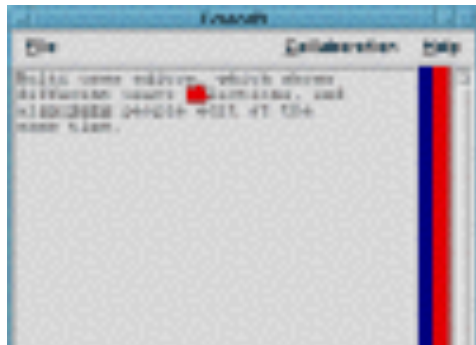
---



- ▶ Brainstorming
- ▶ Text chat (talk à plusieurs)



- ▶ Dessin (bitmap ou vectoriel)
- ▶ Edition de graphes



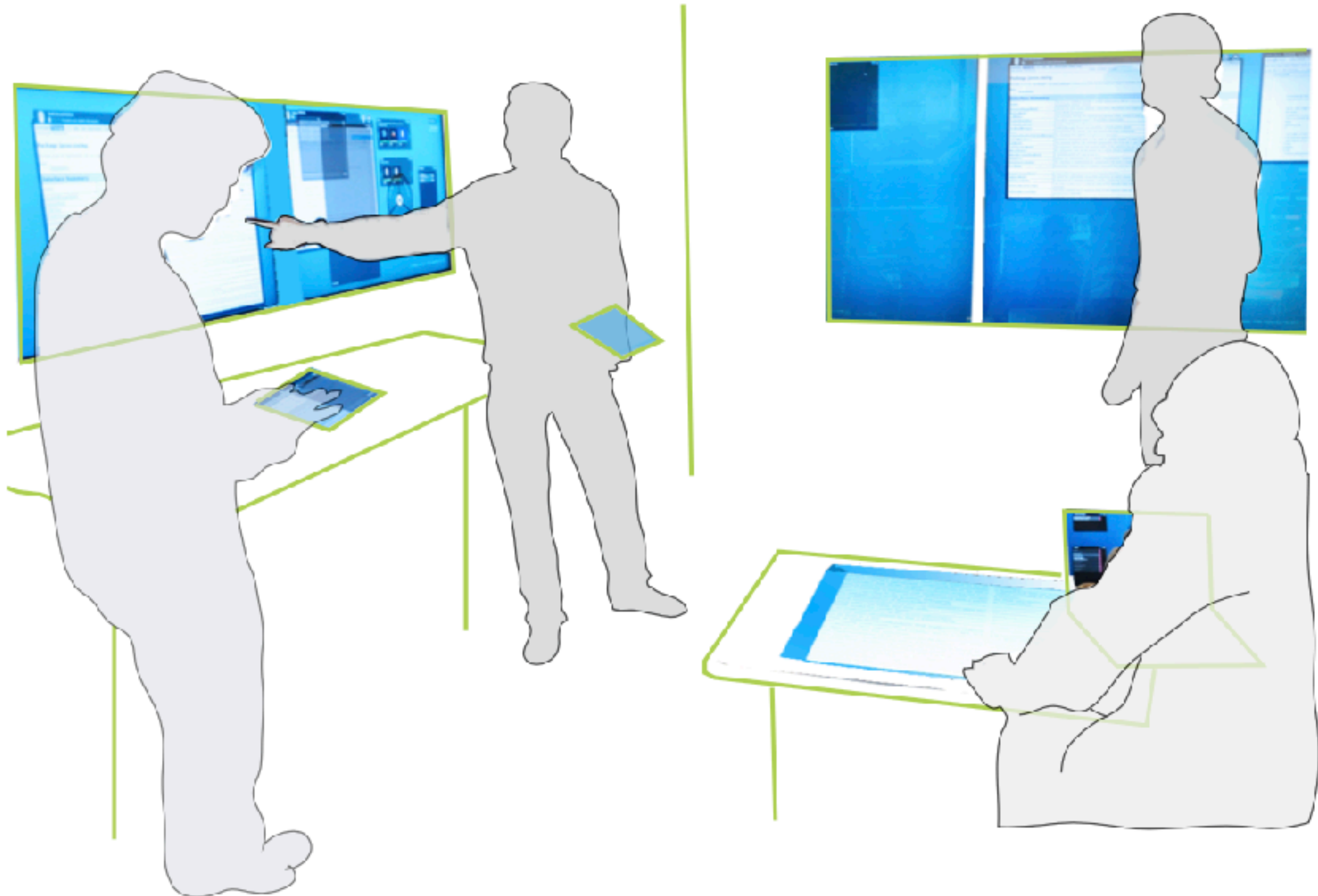
- ▶ Consultation de fichier
- ▶ Editeurs de texte



- ▶ Jeux (morpion, cartes, tetrominoes)

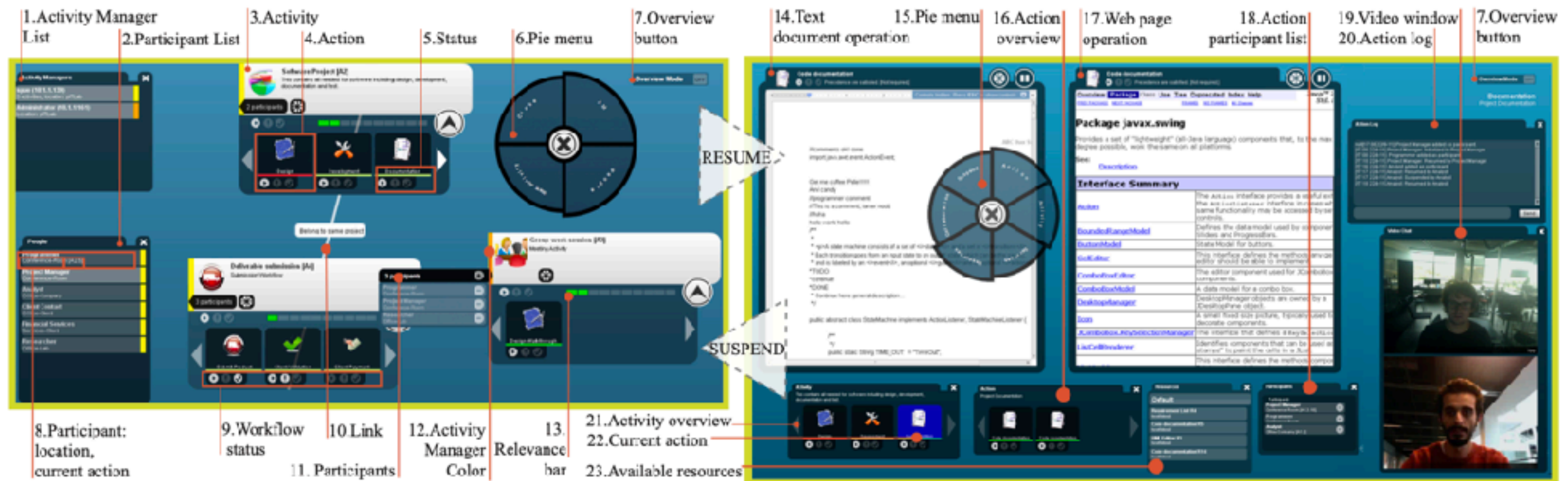
# Activity Based Computing : Reticular Spaces

---

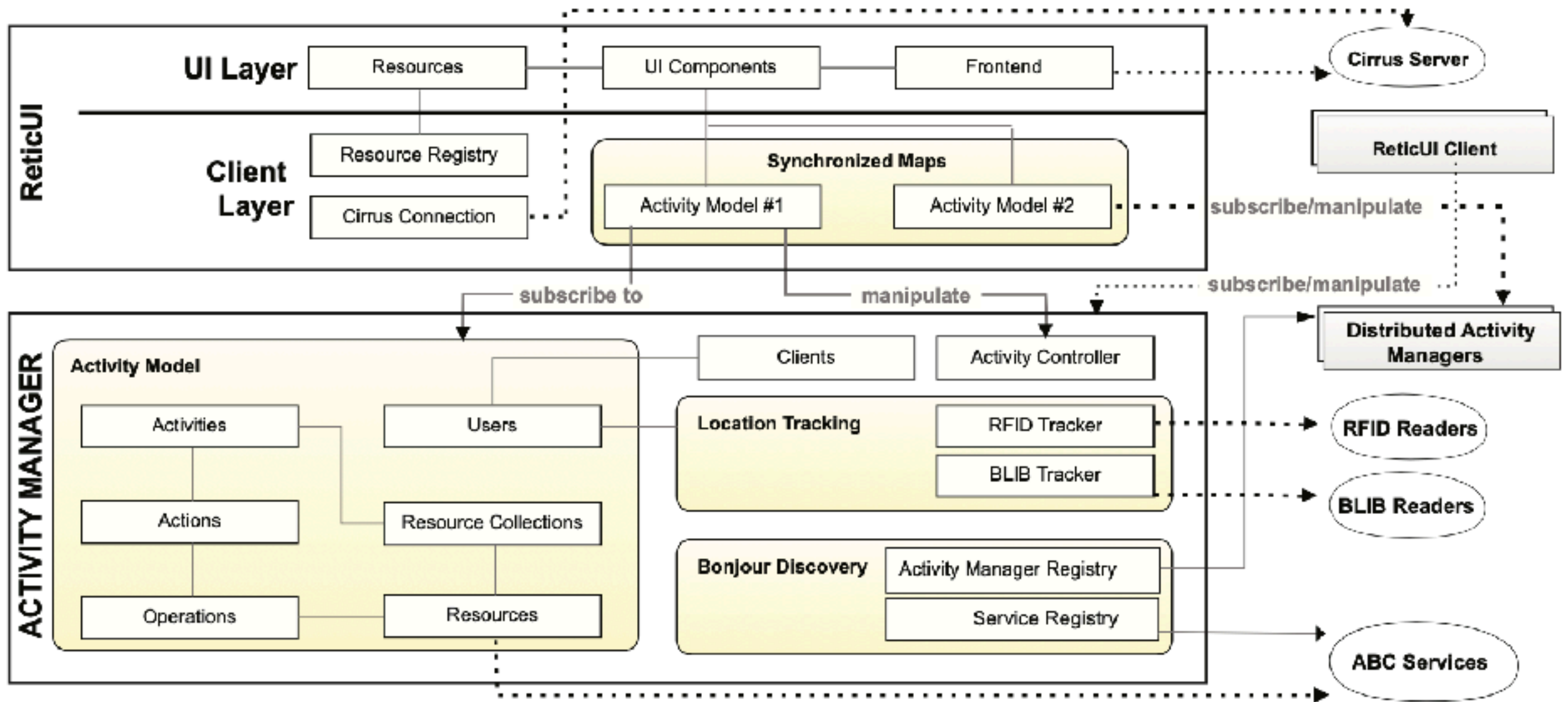




# Activity Based Computing : Reticular Spaces



# Architecture



# Plan

---

## Édition partagée

- ▶ Exemples
- ▶ Principes
- ▶ Défis de la synchronisation

## Algorithmes de synchronisation

- ▶ Operational Transform
- ▶ CRDT

## Groupware